package logic;

public class CombineData {

private byte[] data;

public CombineData() {

// TODO Auto-generated constructor stub

data=null;

}

void addData(byte data){

byte[] a=new byte[1];

a[0]=data;

addData(a);

}

void addData(byte[] data){

if(this.data==null){

this.data=data;

}else{

byte[] a=new byte[this.data.length+data.length];

for(int i=0;i<this.data.length;i++){

a[i]=this.data[i];

}

for(int i=this.data.length;i<this.data.length+data.length;i++){

a[i]=data[i-this.data.length];

}

this.data=a;

}

}

void addData(int data){

byte[] a=new byte[4];

for(int i=0;i<4;i++){

a[i]=(byte) (data%256);

data=data/256;

}

addData(a);

}

void addData(int[] data){

if(data==null){

addData(0);

return;

}

addData(data.length);

for(int i:data){

addData(i);

}

}

void addData(long data){

byte[] a=new byte[8];

for(int i=0;i<8;i++){

a[i]=(byte) (data%256);

data=data/256;

}

addData(a);

}

byte[] getData(){

return data;

}

}

package logic;

import java.io.UnsupportedEncodingException;

class DivideData {

private byte[] data;

private int loc;

DivideData(byte[] data) {

// TODO Auto-generated constructor stub

this.data=data;

loc=0;

}

void resert(byte[] data){

this.data=data;

loc=0;

}

byte[] getBytes(int num){

byte[] a=new byte[num];

for(int i=0;i<num;i++){

a[i]=data[loc+i];

}

loc+=num;

return a;

}

int getInt() {

int num=0;

for(int i=3;i>=0;i--){

num\*=256;

num+=0xff&data[loc+i];

}

loc+=4;

return num;

}

long getLong(){

long num=0;

for(int i=7;i>=0;i--){

num\*=256;

num+=0xff&data[loc+i];

}

loc+=8;

return num;

}

byte getByte(){

byte d;

d=data[loc];

loc+=1;

return d;

}

String getString(byte size){

byte[] s=new byte[2\*size];

for(int i=0;i<2\*size;i++){

s[i]=data[loc+i];

}

loc+=2\*size;

try {

return new String(s,"utf-8");

} catch (UnsupportedEncodingException e) {

// TODO Auto-generated catch block

e.printStackTrace();

return null;

}

}

}

package logic;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.security.InvalidAlgorithmParameterException;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import javax.crypto.Cipher;

import javax.crypto.CipherInputStream;

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.spec.SecretKeySpec;

import javax.crypto.CipherOutputStream;

import javax.crypto.NoSuchPaddingException;

import ui.Task;

eOperation {

private CipherInputStream in;

private CipherOutputStream out;

private DataInputStream sin;

private DataOutputStream sout;

//要加密的部分文件的大小

private long num;

private Cipher cipher;

private byte[] IV;

{

try {

cipher=Cipher.getInstance("AES/CBC/PKCS5Padding");

} catch (NoSuchAlgorithmException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (NoSuchPaddingException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

FileOperation(File file,byte[] key,byte[] IV,int left,int right,DataOutputStream sout) {

// TODO Auto-generated constructor stub

//Cipher cipher = null;

this.sout=sout;

num=right-left+1;

//System.out.println(num);

num\*=16;

this.IV=IV;

try {

byte[] key16=new byte[16];

for(int i=0;i<16;i++){

key16[i]=key[i];

}

cipher.init(Cipher.ENCRYPT\_MODE, new SecretKeySpec(key16, "AES"),new IvParameterSpec(IV));

} catch (InvalidKeyException | InvalidAlgorithmParameterException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

FileInputStream input = null;

try {

input = new FileInputStream(file);

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

long a=(left-1)\*16;

if(left==0)

a=0;

while(a!=0)

{

try {

a-=input.skip(a);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

in = new CipherInputStream(input, cipher);

}

void sendFile(Task task) throws IOException{

//设立512的缓冲区，加密一点，传输一点

byte[] buffer=new byte[512];

int nSend=0;

int sendnum=0;

long allNum=num;

sout.write(IV);

num-=16;

while(num>0){

nSend=in.read(buffer);

//System.out.println(nSend);

if(nSend<num){

sout.write(buffer,0,nSend);

}else{

sout.write(buffer,0,(int)num);

}

num-=nSend;

sendnum+=nSend;

if(sendnum\*100>allNum){

task.getOnePoint(sendnum);

sendnum=0;

}

}

task.finish();

in.close();

sout.close();

}

FileOperation(File file,byte[] key,byte[] IV,DataInputStream sin,long size) {

// TODO Auto-generated constructor stub

//此处要注意的注释与上面的一个构造函数相同

//Cipher cipher = null;

this.sin=sin;

num=size;

try {

byte[] key16=new byte[16];

for(int i=0;i<16;i++){

key16[i]=key[i];

}

cipher.init(Cipher.DECRYPT\_MODE, new SecretKeySpec(key16, "AES"),new IvParameterSpec(IV));

} catch (InvalidKeyException | InvalidAlgorithmParameterException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

FileOutputStream output = null;

try {

output = new FileOutputStream(file);

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

out = new CipherOutputStream(output, cipher);

}

void receiveFile(Task task) throws IOException{

//此处注意点与上面的接收文件的相同

byte[] buffer=new byte[512];

int nRecv=0;

int recnum=0;

long allNum=num;

//JOptionPane.showMessageDialog(null,"allnum"+allNum);

while(num>0){

//JOptionPane.showMessageDialog(null,"num"+num);

nRecv=sin.read(buffer);

if(nRecv<num){

out.write(buffer,0,nRecv);

}else{

out.write(buffer,0,(int)num);

}

num-=nRecv;

recnum+=nRecv;

if(recnum\*100>allNum){

task.getOnePoint(recnum);

recnum=0;

}

}

task.finish();

out.flush();

sin.close();

out.close();

}

// public static void main(String[] args){

// FileOperation a=new FileOperation(new File("C:\\2.txt"), new byte[32], new byte[16]);

// }

}

package logic;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.UnsupportedEncodingException;

import java.net.Socket;

import java.net.UnknownHostException;

import java.security.KeyStore;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import javax.net.ssl.SSLContext;

import javax.net.ssl.SSLSocket;

import javax.net.ssl.SSLSocketFactory;

import javax.net.ssl.TrustManager;

import javax.net.ssl.TrustManagerFactory;

import javax.swing.JOptionPane;

import ui.PanFile;

import ui.PanFriend;

import ui.Task;

public class MyConnection {

private DataInputStream socketIn;

private DataOutputStream socketOut;

private static String host="192.168.79.1";

private static int port=3456;

private static byte[] cookies=null;

//private int type;

private MySecret mySecret;

private Socket socket;

private String userName;

private static SSLContext context;

private static SSLSocketFactory ssf;

{

if(context == null){

try {

KeyStore ts = KeyStore.getInstance("JKS");

ts.load(this.getClass().getResourceAsStream("/resourse/clientTrust.jks"), "123456".toCharArray());

System.out.println(this.getClass().getResourceAsStream("/resourse/clientTrust.jks").available());

TrustManagerFactory tmf = TrustManagerFactory.getInstance("SunX509");

tmf.init(ts);

TrustManager [] tm = tmf.getTrustManagers();

context = SSLContext.getInstance("SSL");

context.init(null, tm, null);

} catch (Exception e) { //省略捕获的异常信息

e.printStackTrace();

}

ssf = context.getSocketFactory();

}

}

public static final byte SIGN\_IN=1;

public static final byte SIGN\_OUT=2;

public static final byte SIGN\_UP=3;

public static final byte REQUIRE\_FILES=4;

public static final byte REQUIRE\_FRIENDS=5;

public static final byte UPLOAD\_FILE=7;

public static final byte DOWNLOAD\_FILE=8;

public static final byte ADD\_FRIEND=9;

public static final byte SHARE\_FILE=10;

public static final byte CREATE\_DIRECTORY=11;

public static final byte DELETE\_FILE=12;

public static final byte SUCCESS=101;

public static final byte FAILED\_ACTION=100;

//public static final byte SUCCESS\_LOGN=101;

public static final byte GIVE\_FILES=102;

public static final byte GIVE\_FRIENDS=103;

public static final byte DEMMAND\_FILE=104;

public static final byte GIVE\_FILE=105;

public static final byte GIVE\_FRIEND=106;

public static final byte UPLOAD\_FINISH=107;

public static final byte SUCCESS\_SHARE=108;

public static final byte SUCCESS\_CREATE=109;

public static final byte SUCCESS\_DELETE=110;

public MyConnection(byte type) throws NumberFormatException, UnknownHostException, IOException, MyTimeOutException{

//ssl模式

socket = (SSLSocket) ssf.createSocket(host,port);

//socket = new Socket(host,port);

socketIn=new DataInputStream(socket.getInputStream());

socketOut=new DataOutputStream(socket.getOutputStream());

//this.type=type;

if(type==UPLOAD\_FILE||type==DOWNLOAD\_FILE){

socketOut.write(type);

socketOut.write(cookies);

// byte[] successFlag=new byte[1];

// socketIn.read(successFlag);

// if(successFlag[0]!=(byte)SUCCESS){

// throw new MyTimeOutException();

// }

}

}

/\*\*

\* 登录

\* @param name 用户名

\* @param password 用户密码

\* @return 是否登录成功

\*/

public boolean signIn(char[] name,char[] password){

MessageDigest sha256 = null;

byte[] nameHash = null;

byte[] passwordHash;

try {

sha256 = MessageDigest.getInstance("SHA-256");

} catch (NoSuchAlgorithmException e) {

// TODO Auto-generated catch block

//不可能发生

e.printStackTrace();

}

userName=new String(name);

//以hash(hash(name)+hash(password))作为公私钥产生的种子

try {

sha256.update(new String(name).getBytes("utf-8"));

nameHash=sha256.digest();

sha256.update(new String(password).getBytes("utf-8"));

} catch (UnsupportedEncodingException e2) {

// TODO Auto-generated catch block

e2.printStackTrace();

}

passwordHash=sha256.digest();

sha256.update(nameHash);

sha256.update(passwordHash);

mySecret=new MySecret(sha256.digest());

sha256.update(passwordHash);

sha256.update(sha256.digest());

sha256.update(nameHash);

CombineData data=new CombineData();

+hash(name)+hash(hash(hash(password))+hash(name))

data.addData(SIGN\_IN);

data.addData(nameHash);

data.addData(sha256.digest());

try {

socketOut.write(data.getData());

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

byte[] f=new byte[1];

try {

socketIn.read(f);

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

if(f[0]==SUCCESS){

cookies=new byte[32];

int readNum=0;

while(readNum!=32){

try {

readNum+=socketIn.read(cookies,readNum,32-readNum);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

return true;

}else{

return false;

}

}

public int signOut() {

byte[] b=new byte[1];

b[0]=SIGN\_OUT;

try {

socketOut.write(b);

socket.close();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return 0;

}

public boolean signUp(char[] name,char[] password) {

MessageDigest sha256 = null;

byte[] nameHash = null;

byte[] passwordHash;

userName=new String(name);

try {

sha256 = MessageDigest.getInstance("SHA-256");

} catch (NoSuchAlgorithmException e) {

// TODO Auto-generated catch block

//不可能发生

e.printStackTrace();

}

try {

sha256.update(new String(name).getBytes("utf-8"));

nameHash=sha256.digest();

sha256.update(new String(password).getBytes("utf-8"));

} catch (UnsupportedEncodingException e2) {

// TODO Auto-generated catch block

e2.printStackTrace();

}

passwordHash=sha256.digest();

sha256.update(nameHash);

sha256.update(passwordHash);

mySecret=new MySecret(sha256.digest());

sha256.update(passwordHash);

sha256.update(sha256.digest());

sha256.update(nameHash);

CombineData data=new CombineData();

data.addData(SIGN\_UP);

data.addData(nameHash);

data.addData(sha256.digest());

try {

if(new String(name).getBytes("utf-8").length>255){

JOptionPane.showMessageDialog(null,"用户名过长，无法注册！");

return false;

}

data.addData((byte)new String(name).getBytes("utf-8").length);

data.addData(new String(name).getBytes("utf-8"));

} catch (UnsupportedEncodingException e2) {

// TODO Auto-generated catch block

e2.printStackTrace();

}

data.addData(mySecret.getPublicKey());

try {

socketOut.write(data.getData());

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

byte[] f=new byte[1];

try {

socketIn.read(f);

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

if(f[0]==SUCCESS){

cookies=new byte[32];

int readNum=0;

while(readNum!=32){

try {

readNum+=socketIn.read(cookies,readNum,32-readNum);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

return true;

}else{

return false;

}

}

public PanFile[] requireFiles(int loc,int deep) throws IOException {

PanFile[] panFiles;

CombineData data=new CombineData();

data.addData(REQUIRE\_FILES);

//这个方法里会连着数组的大小一起添加

data.addData(loc);

data.addData(deep);

socketOut.write(data.getData());

byte[] f=new byte[5];

int readNum=0;

while(readNum!=5){

readNum+=socketIn.read(f,readNum,5-readNum);

}

DivideData dataDiv=new DivideData(f);

if(dataDiv.getByte()!=GIVE\_FILES){

return null;

}else{

int t=dataDiv.getInt();

// if(t==0){

// return null;

// }

panFiles=new PanFile[t];

for (int i = 0; i < t; i++) {

int id;

byte[] fileID=new byte[32];

byte[] tempKey=new byte[129];

byte[] fileKey=new byte[32];

byte[] recdata=new byte[166];

byte type;

int size;

readNum=0;

while(readNum!=36){

readNum+=socketIn.read(recdata,readNum,36-readNum);

}

dataDiv.resert(recdata);

id=dataDiv.getInt();

fileID=dataDiv.getBytes(32);

String name=readString();

//System.out.println(name);

readNum=0;

while(readNum!=134){

readNum+=socketIn.read(recdata,readNum,134-readNum);

}

dataDiv.resert(recdata);

size=dataDiv.getInt();

tempKey=dataDiv.getBytes(129);

type=dataDiv.getByte();

if(type==1)

fileKey=mySecret.decrypt(tempKey);

else {

fileKey=null;

}

panFiles[i]=new PanFile(name, fileID, fileKey, type, id,deep, (size)\*16, 0);

}

return panFiles;

}

}

public boolean shareFiles(PanFile[] files,PanFriend[] friends){

boolean ret=true;

for(PanFriend friend:friends){

CombineData data=new CombineData();

data.addData(SHARE\_FILE);

data.addData(friend.getId());

data.addData((int)files.length);

for(PanFile file:files){

if(file.getType()==PanFile.DIRECTORY){

PanFile[] nextFiles = null;

try {

nextFiles=requireFiles(file.getLoc(), file.getDeep()+1);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

data.addData(file.getFileNameSize());

try {

data.addData(file.getName().getBytes("utf-8"));

} catch (UnsupportedEncodingException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

data.addData((int)0);

data.addData((byte)PanFile.DIRECTORY);

if(nextFiles==null)

data.addData((int)0);

else{

data.addData(nextFiles.length);

if(!getDeepFiles(nextFiles, data,friend.getKey())){

return false;

}

}

}else{

data.addData(file.getFileNameSize());

try {

data.addData(file.getName().getBytes("utf-8"));

} catch (UnsupportedEncodingException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

data.addData((int)(file.getFileSize()/16));

data.addData((byte)PanFile.FILE);

data.addData(file.getId());

data.addData(mySecret.encryptWithOthers(file.getKey(), friend.getKey()));

}

}

try {

socketOut.write(data.getData());

byte[] a=new byte[1];

socketIn.read(a);

if(a[0]!=SUCCESS\_SHARE){

ret&=false;

}

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

return false;

}

}

return ret;

}

boolean getDeepFiles(PanFile[] files,CombineData data,byte[] friendKey){

boolean ret=true;

for(PanFile file:files){

if(file.getType()==PanFile.DIRECTORY){

PanFile[] nextFiles = null;

try {

nextFiles=requireFiles(file.getLoc(), file.getDeep()+1);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

return false;

}

data.addData(file.getFileNameSize());

try {

data.addData(file.getName().getBytes("utf-8"));

} catch (UnsupportedEncodingException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

data.addData((int)0);

data.addData((byte)PanFile.DIRECTORY);

if(nextFiles==null)

data.addData((int)0);

else{

data.addData(nextFiles.length);

ret&=getDeepFiles(nextFiles, data,friendKey);

}

}else{

data.addData(file.getFileNameSize());

try {

data.addData(file.getName().getBytes("utf-8"));

} catch (UnsupportedEncodingException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

data.addData((int)(file.getFileSize()/16));

data.addData((byte)PanFile.FILE);

data.addData(file.getId());

data.addData(mySecret.encryptWithOthers(file.getKey(), friendKey));

}

}

return ret;

}

public PanFriend[] requireFriends() throws IOException {

byte[] a=new byte[1];

a[0]=REQUIRE\_FRIENDS;

try {

socketOut.write(a);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

byte[] f=new byte[5];

int readNum=0;

while(readNum!=5){

readNum+=socketIn.read(f,readNum,5-readNum);

}

DivideData dataDiv=new DivideData(f);

PanFriend[] friends=null;

if(dataDiv.getByte()!=GIVE\_FRIENDS){

return null;

}else{

int t=dataDiv.getInt();

// if(t==0){

// return null;

// }

friends=new PanFriend[t];

for (int i = 0; i < t; i++) {

byte[] friendID=new byte[32];

byte[] friendKey=new byte[65];

readNum=0;

while(readNum!=32){

readNum+=socketIn.read(friendID,readNum,32-readNum);

}

String name=readString();

readNum=0;

while(readNum!=65){

readNum+=socketIn.read(friendKey,readNum,65-readNum);

}

friends[i]=new PanFriend(name, friendID, 0, null, friendKey);

}

}

return friends;

}

public PanFriend addFriend(char[] friendName){

MessageDigest sha256 = null;

byte[] nameHash = null;

try {

sha256 = MessageDigest.getInstance("SHA-256");

} catch (NoSuchAlgorithmException e) {

// TODO Auto-generated catch block

//不可能发生

e.printStackTrace();

}

try {

sha256.update(new String(friendName).getBytes("utf-8"));

nameHash=sha256.digest();

} catch (UnsupportedEncodingException e2) {

// TODO Auto-generated catch block

e2.printStackTrace();

}

CombineData data=new CombineData();

data.addData(ADD\_FRIEND);

data.addData(nameHash);

try {

socketOut.write(data.getData());

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

byte[] f=new byte[1];

try {

socketIn.read(f);

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

byte[] friendKey=new byte[65];

if(f[0]==GIVE\_FRIEND){

int readNum=0;

while(readNum!=65){

try {

readNum+=socketIn.read(friendKey,readNum,65-readNum);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

return new PanFriend(new String(friendName), nameHash, 0, null, friendKey);

}else{

return null;

}

}

public boolean uploadFile(File file,int loc,int deep,Task task) {

//上传和下载都是一个独立的线程，建立一个新的连接用于传输数据

MyConnection upConnection=null;

//先把程序主要部分填起来，然后再去补时间超时异常的坑

try {

upConnection=new MyConnection(MyConnection.UPLOAD\_FILE);

} catch (NumberFormatException | IOException | MyTimeOutException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

byte[] fileName = null;

byte[] hashOfFile;

byte[] fileID;

byte[] fileKey;

int fileSize;

CombineData data=new CombineData();

try {

fileName=file.getName().getBytes("utf-8");

} catch (UnsupportedEncodingException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

fileSize=(int)(file.length()/16+2);

data.addData((byte)fileName.length);

if(fileName.length>=256){

JOptionPane.showMessageDialog(null,"文件名过长，文件将无法上传！");

return false;

}

if(file.length()/16+2>Integer.MAX\_VALUE){

JOptionPane.showMessageDialog(null,"文件过大，文件将无法上传！");

return false;

}

data.addData(fileName);

data.addData(fileSize);

MessageDigest sha256 = null;

try {

sha256 = MessageDigest.getInstance("SHA-256");

} catch (NoSuchAlgorithmException e) {

// TODO Auto-generated catch block

//不可能发生

e.printStackTrace();

}

FileInputStream fileStream=null;

try {

fileStream = new FileInputStream(file);

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

byte[] fileBytes = new byte[1024\*4];

int nread = 0;

try {

while ((nread = fileStream.read(fileBytes)) != -1) {

sha256.update(fileBytes, 0, nread);

}

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

};

try {

fileStream.close();

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

hashOfFile = sha256.digest();

sha256.update(hashOfFile, 0, hashOfFile.length);

fileID=sha256.digest();

fileKey=mySecret.encrypt(hashOfFile);

data.addData(fileID);

data.addData(fileKey);

data.addData(deep);

data.addData(loc);

try {

upConnection.socketOut.write(data.getData());

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

byte[] receiveData=new byte[25];

int receiveNum=0;

while(receiveNum!=25){

try {

receiveNum+=upConnection.socketIn.read(receiveData,receiveNum,25-receiveNum);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

try {

upConnection.socket.close();

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

return false;

}

}

DivideData recData=new DivideData(receiveData);

if(recData.getByte()!=GIVE\_FILE){

try {

upConnection.socket.close();

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

return false;

}

//通过另一个类来操作文件

FileOperation fileOperation=new FileOperation(file, hashOfFile, recData.getBytes(16), recData.getInt(), recData.getInt(),upConnection.socketOut);

byte[] f=new byte[1];

f[0]=UPLOAD\_FILE;

try {

upConnection.socketOut.write(f);

fileOperation.sendFile(task);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

try {

upConnection.socket.close();

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

return false;

}

try {

upConnection.socket.close();

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

return true;

}

public boolean downloadFile(byte[] id,byte[] key,Long size,String fileLoc,Task task) {

File file=new File(fileLoc);

MyConnection doConnection=null;

try {

doConnection=new MyConnection(MyConnection.DOWNLOAD\_FILE);

} catch (NumberFormatException | IOException | MyTimeOutException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

CombineData data=new CombineData();

//data.addData(DOWNLOAD\_FILE);

data.addData(id);

try {

doConnection.socketOut.write(data.getData());

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

byte[] f=new byte[1];

try {

doConnection.socketIn.read(f);

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

if(f[0]!=GIVE\_FILE){

try {

doConnection.socket.close();

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

return false;

}

byte[] iv=new byte[16];

int receiveNum=0;

while(receiveNum!=16){

try {

receiveNum+=doConnection.socketIn.read(iv,receiveNum,16-receiveNum);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

FileOperation fileOperation=new FileOperation(file, key, iv,doConnection.socketIn,size-16);

try {

fileOperation.receiveFile(task);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

try {

doConnection.socket.close();

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

return false;

}

try {

doConnection.socket.close();

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

return true;

}

public String getName(){

return userName;

}

public boolean deleteFile(int loc,int deep){

CombineData data=new CombineData();

data.addData(DELETE\_FILE);

data.addData(loc);

data.addData(deep);

byte[] r=new byte[1];

try {

socketOut.write(data.getData());

socketIn.read(r);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

if(r[0]==SUCCESS\_DELETE){

return true;

}

return false;

}

public int createDirectory(byte[] name,int loc,int deep){

CombineData data=new CombineData();

data.addData(CREATE\_DIRECTORY);

data.addData((byte)name.length);

data.addData(name);

data.addData(loc);

data.addData(deep);

try {

socketOut.write(data.getData());

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

byte[] d=new byte[5];

int readNum=0;

while(readNum!=5){

try {

readNum+=socketIn.read(d,readNum,5-readNum);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

DivideData getData=new DivideData(d);

if(getData.getByte()==SUCCESS\_CREATE){

int w=getData.getInt();

return w;

}else{

return 0;

}

}

public static void setHost(String host) {

MyConnection.host = host;

}

public static void setPort(int port) {

MyConnection.port = port;

}

/\*\*

\* 从输入流中读取一个String

\* @return

\* @throws IOException

\*/

String readString() throws IOException{

int readNum=0;

//先读取这个字符串的长度，然后根据长度读取字符串

byte[] nameSize=new byte[1];

socketIn.read(nameSize);

byte[] name=new byte[(int)0xff&nameSize[0]];

readNum=0;

int a=0xff&nameSize[0];

while(readNum!=a){

readNum+=socketIn.read(name,readNum,0xff&nameSize[0]-readNum);

}

return new String(name,"utf-8");

}

int[] readInts() throws IOException{

byte[] a=new byte[4];

int readNum=0;

while(readNum!=4){

readNum+=socketIn.read(a,readNum,4-readNum);

}

int length=0;

for(int i=3;i>=0;i--){

length\*=256;

length+=0xff&a[i];

}

readNum=0;

byte[] data=new byte[length\*4];

while(readNum!=length\*4){

readNum+=socketIn.read(data,readNum,length\*4-readNum);

}

DivideData intData=new DivideData(data);

int[] ints=new int[length];

for(int i=0;i<length;i++){

ints[i]=intData.getInt();

}

return ints;

}

}

package logic;

import java.math.BigInteger;

import org.bouncycastle.crypto.AsymmetricCipherKeyPair;

import org.bouncycastle.crypto.params.ECPrivateKeyParameters;

import org.bouncycastle.crypto.params.ECPublicKeyParameters;

import org.bouncycastle.math.ec.ECPoint;

import Cipher.Cipher;

import Cipher.SM2;

import Cipher.Util;

class MySecret {

private byte[] publicKey;

private byte[] privateKey;

private SM2 sm2;

@SuppressWarnings("deprecation")

MySecret(byte[] userSecert) {

sm2=new SM2(userSecert);

AsymmetricCipherKeyPair key = sm2.ecc\_key\_pair\_generator.generateKeyPair();

ECPrivateKeyParameters ecpriv = (ECPrivateKeyParameters) key.getPrivate();

ECPublicKeyParameters ecpub = (ECPublicKeyParameters) key.getPublic();

//getD得到的是BigInteger

privateKey = ecpriv.getD().toByteArray();

//getQ得到的是ECPoint---来自新引入的包

publicKey=ecpub.getQ().getEncoded();

}

byte[] getPublicKey(){

return publicKey;

}

byte[] encrypt(byte[] message){

return encryptWithOthers(message, publicKey);

}

byte[] c1Bytes = Util.hexToByte(dataHexString.substring(0,130));

int c2Len = data.length - 97;

byte[] c2 = Util.hexToByte(dataHexString.substring(130,130 + 2 \* c2Len));

byte[] c3 = Util.hexToByte(dataHexString.substring(130 + 2 \* c2Len,194 + 2 \* c2Len));

BigInteger userD = new BigInteger(1, privateKey);

//通过C1实体字节来生成ECPoint

ECPoint c1 = sm2.ecc\_curve.decodePoint(c1Bytes);

Cipher cipher = new Cipher();

cipher.Init\_dec(userD, c1);

cipher.Decrypt(c2);

cipher.Dofinal(c3);

//返回解密结果

return c2;

}

byte[] encryptWithOthers(byte[] message,byte[] otherKey){

byte[] source = new byte[message.length];

System.arraycopy(message, 0, source, 0, message.length);

Cipher cipher = new Cipher();

ECPoint userKey = sm2.ecc\_curve.decodePoint(otherKey);

ECPoint c1 = cipher.Init\_enc(sm2, userKey);

cipher.Encrypt(source);

byte[] c3 = new byte[32];

cipher.Dofinal(c3);

byte[] temp=new byte[129];

int i=0;

@SuppressWarnings("deprecation")

byte[] tc1=c1.getEncoded();

for(;i<tc1.length;i++)

temp[i]=tc1[i];

for(;i<tc1.length+source.length;i++)

temp[i]=source[i-tc1.length];

for(;i<tc1.length+source.length+c3.length;i++)

temp[i]=c3[i-tc1.length-source.length];

return temp;

}

}

package logic;

public class MyTimeOutException extends Exception{

private static final long serialVersionUID = 1L;

String reason;

MyTimeOutException(){

super();

reason="time out";

}

}

package Cipher;

import java.math.BigInteger;

import org.bouncycastle.crypto.AsymmetricCipherKeyPair;

import org.bouncycastle.crypto.params.ECPrivateKeyParameters;

import org.bouncycastle.crypto.params.ECPublicKeyParameters;

import org.bouncycastle.math.ec.ECPoint;

@SuppressWarnings("deprecation")

public class Cipher

{

private int ct;

private ECPoint p2;

private SM3Digest sm3keybase;

private SM3Digest sm3c3;

private byte key[];

private byte keyOff;

public Cipher()

{

this.ct = 1;

this.key = new byte[32];

this.keyOff = 0;

}

private void Reset()

{

this.sm3keybase = new SM3Digest();

this.sm3c3 = new SM3Digest();

byte p[] = Util.byteConvert32Bytes(p2.getX().toBigInteger());

this.sm3keybase.update(p, 0, p.length);

this.sm3c3.update(p, 0, p.length);

p = Util.byteConvert32Bytes(p2.getY().toBigInteger());

this.sm3keybase.update(p, 0, p.length);

this.ct = 1;

NextKey();

}

private void NextKey()

{

SM3Digest sm3keycur = new SM3Digest(this.sm3keybase);

sm3keycur.update((byte) (ct >> 24 & 0xff));

sm3keycur.update((byte) (ct >> 16 & 0xff));

sm3keycur.update((byte) (ct >> 8 & 0xff));

sm3keycur.update((byte) (ct & 0xff));

sm3keycur.doFinal(key, 0);

this.keyOff = 0;

this.ct++;

}

public ECPoint Init\_enc(SM2 sm2, ECPoint userKey)

{

AsymmetricCipherKeyPair key = sm2.ecc\_key\_pair\_generator.generateKeyPair();

ECPrivateKeyParameters ecpriv = (ECPrivateKeyParameters) key.getPrivate();

ECPublicKeyParameters ecpub = (ECPublicKeyParameters) key.getPublic();

BigInteger k = ecpriv.getD();

ECPoint c1 = ecpub.getQ();

this.p2 = userKey.multiply(k);

Reset();

return c1;

}

public void Encrypt(byte data[])

{

this.sm3c3.update(data, 0, data.length);

for (int i = 0; i < data.length; i++)

{

if (keyOff == key.length)

{

NextKey();

}

data[i] ^= key[keyOff++];

}

}

public void Init\_dec(BigInteger userD, ECPoint c1)

{

this.p2 = c1.multiply(userD);

Reset();

}

public void Decrypt(byte data[])

{

for (int i = 0; i < data.length; i++)

{

if (keyOff == key.length)

{

NextKey();

}

data[i] ^= key[keyOff++];

}

this.sm3c3.update(data, 0, data.length);

}

public void Dofinal(byte c3[])

{

byte p[] = Util.byteConvert32Bytes(p2.getY().toBigInteger());

this.sm3c3.update(p, 0, p.length);

this.sm3c3.doFinal(c3, 0);

Reset();

}

}

package ui;

import java.io.File;

import javax.swing.filechooser.FileFilter;

class FileFilterOfDocument extends FileFilter {

@Override

public boolean accept(File f) {

// TODO Auto-generated method stub

return f.getName().endsWith("txt")||f.getName().endsWith("doc")||f.getName().endsWith("docx")||f.getName().endsWith("xls")||

f.getName().endsWith("xlsx")||f.getName().endsWith("pdf")||f.getName().endsWith("ppt")||f.getName().endsWith("pptx")||

f.isDirectory();

}

@Override

public String getDescription() {

// TODO Auto-generated method stub

return "文档(txt,doc,docx,xls,xlsx,pdf,ppt,pptx)";

}

}

class FileFilterOfMovie extends FileFilter {

@Override

public boolean accept(File f) {

// TODO Auto-generated method stub

return f.getName().endsWith("mp4")||f.getName().endsWith("avi")||f.getName().endsWith("flv")||

f.getName().endsWith("mkv")||f.getName().endsWith("rmvb")||f.isDirectory();

}

@Override

public String getDescription() {

// TODO Auto-generated method stub

return "视频(mp4,mkv,rmvb,avi,flv)";

}

}

class FileFilterOfMusic extends FileFilter {

@Override

public boolean accept(File f) {

// TODO Auto-generated method stub

return f.getName().endsWith("mp3")||f.getName().endsWith("wav")

||f.getName().endsWith("m4a")||f.isDirectory();

}

@Override

public String getDescription() {

// TODO Auto-generated method stub

return "音频(mp3,wav,m4a)";

}

}

class FileFilterOfPicture extends FileFilter {

@Override

public boolean accept(File f) {

// TODO Auto-generated method stub

return f.getName().endsWith("jpg")||f.getName().endsWith("png")||f.getName().endsWith("ico")||

f.getName().endsWith("bmp")||f.getName().endsWith("gif")||f.isDirectory();

}

@Override

public String getDescription() {

// TODO Auto-generated method stub

return "图片(jpg,png,ico,bmp,gif)";

}

}

package ui;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.GradientPaint;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.Image;

import java.awt.Paint;

import java.awt.Point;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.MouseMotionAdapter;

import java.awt.geom.RoundRectangle2D;

import java.io.IOException;

import java.util.regex.Pattern;

import javax.swing.ImageIcon;

import javax.swing.JCheckBox;

import javax.swing.JEditorPane;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.UIManager;

import logic.MyConnection;

import logic.MyTimeOutException;

public class LoginFrame extends JFrame{

private static final long serialVersionUID = 1L;

private MyConnection connection;

private JPanel onPanel;

private Point mouse;

private MyTextField userName;

private MyTextField userPassword;

private MyButton login;

private MyButton cancel;

private JLabel tip;

private JLabel loginLogo;

private JLabel signUp;

static ImageIcon deleteIcon;

static ImageIcon nameIcon;

static ImageIcon passwordIcon;

static ImageIcon loginLogoIcon;

{

if(deleteIcon==null){

deleteIcon=new ImageIcon(this.getClass().getResource("/resourse/删除.png"));

nameIcon=new ImageIcon(this.getClass().getResource("/resourse/用户.png"));

passwordIcon=new ImageIcon(this.getClass().getResource("/resourse/密码.png"));

loginLogoIcon=new ImageIcon(this.getClass().getResource("/resourse/loginlogo.png"));

}

//deleteIcon=new ImageIcon(LoginFrame.class.getClassLoader().getResource("resourse/删除.png"));

//nameIcon=new ImageIcon(LoginFrame.class.getClassLoader().getResource("resourse/用户.png"));

//passwordIcon=new ImageIcon(LoginFrame.class.getClassLoader().getResource("resourse/密码.png"));

//loginLogoIcon=new ImageIcon(LoginFrame.class.getClassLoader().getResource("resourse/loginlogo.png"));

}

static final int SIGN\_IN\_TYPE=0;

static final int SIGNED\_UP\_TYPE=1;

private int type = SIGN\_IN\_TYPE;

public LoginFrame(){

super();

//setLayout(null);

setSize(500, 400);

setLocationRelativeTo(null);

try {

connection=new MyConnection(MyConnection.SIGN\_IN);

} catch (NumberFormatException | IOException e ) {

e.printStackTrace();

new SetHostFrame();

dispose();

return;

} catch(MyTimeOutException e){

e.printStackTrace();

} catch(Exception e){

e.printStackTrace();

}

if(UIManager.getLookAndFeel().isSupportedLookAndFeel()){

final String platform = UIManager.getSystemLookAndFeelClassName();

if (!UIManager.getLookAndFeel().getName().equals(platform)) {

try {

UIManager.setLookAndFeel(platform);

} catch (Exception exception) {

exception.printStackTrace();

}

}

}

onPanel=new JPanel(){

/\*\*

\* 20160405

\*/

private static final long serialVersionUID = 1L;

protected void paintComponent(Graphics g){

super.paintComponent(g);

Paint arcRectPaint =new GradientPaint(0, 0, new Color(19,146,255),

0, 400, new Color(188,240,255));

Graphics2D g2=(Graphics2D)g.create();

g2.setPaint(arcRectPaint);

g2.fill(new RoundRectangle2D.Double(0,0,getWidth(),getHeight(),30,30));

}

};

onPanel.setLayout(null);

add(onPanel);

loginLogo=new JLabel();

loginLogo.setIcon(loginLogoIcon);

loginLogo.setLocation(0, 20);

loginLogo.setSize(500,120);

onPanel.add(loginLogo);

Font font=new Font("微软雅黑",Font.PLAIN,20);

userName=new MyTextField("用户名", 260, 50, 25, deleteIcon);

userName.setLocation(140, 160);

userName.setSize(260,50);

userName.setFont(font);

userName.enableInputMethods(true);

//这是为了实现文本框的那张删除图片的点击反应

userName.addMouseListener(new MouseAdapter() {

public void mouseClicked (MouseEvent e){

if(userName.clickTheImage(e.getPoint())){

userName.setText("");

}

}

});

onPanel.add(userName);

signUp=new JLabel("注册");

signUp.setLocation(400, 160);

signUp.setSize(50,50);

signUp.setFont(new Font("微软雅黑",Font.PLAIN,14));

signUp.setForeground(Color.DARK\_GRAY);

signUp.setVerticalAlignment(JLabel.CENTER);

signUp.setHorizontalAlignment(JLabel.CENTER);

signUp.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked (MouseEvent e){

new UserItem("用户须知",LoginFrame.this);

}

@Override

public void mouseEntered(MouseEvent e) {

signUp.setFont(new Font("微软雅黑",Font.PLAIN,17));

signUp.setForeground(Color.red);

}

@Override

public void mouseExited(MouseEvent e) {

signUp.setFont(new Font("微软雅黑",Font.PLAIN,14));

signUp.setForeground(Color.DARK\_GRAY);

}

});

onPanel.add(signUp);

userPassword=new MyTextField("密码", 260, 50, 25, deleteIcon);

userPassword.setLocation(140, 230);

userPassword.setSize(260,50);

userPassword.setFont(font);

userPassword.enablePassword();

userPassword.addMouseListener(new MouseAdapter() {

public void mouseClicked (MouseEvent e){

if(userPassword.clickTheImage(e.getPoint())){

userPassword.setText("");

}

}

});

//用于输完密码后直接回车就登录

userPassword.addKeyListener(new KeyAdapter() {

public void keyTyped (KeyEvent e){

if(e.getKeyChar()==KeyEvent.VK\_ENTER){

if(type==SIGN\_IN\_TYPE){

if(LoginFrame.this.loginIn()){

new MainFrame(connection);

LoginFrame.this.dispose();

}else{

tip.setForeground(Color.red);

tip.setText("用户名或密码错误");

}

}else{

if(LoginFrame.this.loginUp()){

new MainFrame(connection);

LoginFrame.this.dispose();

}else{

tip.setForeground(Color.red);

tip.setText("用户名已存在");

}

}

}

}

});

onPanel.add(userPassword);

tip=new JLabel();

tip.setFont(new Font("微软雅黑",Font.PLAIN,15));

tip.setSize(500, 20);

tip.setForeground(Color.red);

tip.setLocation(0, 290);

tip.setHorizontalAlignment(JLabel.CENTER);

onPanel.add(tip);

font=new Font("微软雅黑",Font.PLAIN,15);

cancel=new MyButton("取消");

cancel.setFont(font);

cancel.setButtonColor(new Color(200,226,251),new Color(128,208,254),new Color(241,248,252),new Color(161,211,253));

cancel.setSize(80, 30, 20);

cancel.setLocation(150,325);

cancel.addMouseListener(new MouseAdapter() {

public void mouseClicked (MouseEvent e){

LoginFrame.this.dispose();

System.exit(0);

}

});

onPanel.add(cancel);

login=new MyButton("登录");

login.setFont(font);

login.setButtonColor(new Color(200,226,251),new Color(128,208,254),new Color(241,248,252),new Color(161,211,253));

login.setSize(80,30,20);

login.setLocation(270,325);

login.addMouseListener(new MouseAdapter() {

public void mouseClicked (MouseEvent e){

if(type==SIGN\_IN\_TYPE){

if(LoginFrame.this.loginIn()){

new MainFrame(connection);

LoginFrame.this.dispose();

}else{

tip.setForeground(Color.red);

tip.setText("用户名或密码错误");

}

}else{

if(LoginFrame.this.loginUp()){

new MainFrame(connection);

LoginFrame.this.dispose();

}else{

tip.setForeground(Color.red);

tip.setText("用户名已存在");

}

}

}

});

onPanel.add(login);

string sql = "insert allfiles values('" + string(temp) + "',\'\','" + to\_string(filelen) + "')";

MYSQL\_RES \*res = runSQL(sql.c\_str());

free(temp);

mysql\_free\_result(res);

return true;

}

//文件是否是用户的,返回文件的大小

int MySQLManager::fileIsPerson(char \*filename, char \*name)

{

char \*temp1 = charto16(name, 33);

char \*temp2 = charto16(filename, 33);

string sql = "select filesize from file" + string(temp1+4)+ string(" where fileid='") + string(temp2) + "'";

MYSQL\_RES \*res = runSQL(sql.c\_str());

MYSQL\_ROW row;

free(temp1);

free(temp2);

if (res == NULL)

{

return -1;

}

if (row=mysql\_fetch\_row(res))

{

int size;

sscanf\_s(row[0], "%d", &size);

mysql\_free\_result(res);

return size;

}

else

{

mysql\_free\_result(res);

return -1;

}

}

//某人是否是某人的好友

bool MySQLManager::personIsFriend(char \*user, char \*friendname)

{

char \*temp1 = charto16(user, 33);

char \*temp2 = charto16(friendname, 33);

string sql = "select friendid from friend" + string(temp1+6) + " where friendid ='" + string(temp2) + "';";

MYSQL\_RES \*res = runSQL(sql.c\_str());

free(temp1);

free(temp2);

if (res == NULL)

{

return false;

}

if (mysql\_fetch\_row(res) != NULL)

{

mysql\_free\_result(res);

return true;

}

else

{

mysql\_free\_result(res);

return false;

}

}

bool MySQLManager::personLogin(char \*name, char \*password)

{

char \*temp1 = charto16(name, 33);

char \*temp2 = charto16(password, 33);

string sql = "select status from users where usernamehash ='" + string(temp1) + "'and userpassword ='" + string(temp2) + "'and status=0;";

MYSQL\_RES \*res = runSQL(sql.c\_str());

if (res == NULL)

{

return false;

}

if (mysql\_fetch\_row(res)!=NULL)

{

//要及时修改状态

mysql\_free\_result(res);

sql = "update users set status=1 where usernamehash ='" + string(temp1) + "';";

res = runSQL(sql.c\_str());

mysql\_free\_result(res);

return true;

}

else

{

mysql\_free\_result(res);

return false;

}

}

//用户登出

bool MySQLManager::personLogout(char \*name)

{

char \*temp = charto16(name, 33);

string sql = "update users set status=0 where usernamehash ='" + string(temp) + "';";

MYSQL\_RES \*res = runSQL(sql.c\_str());

free(temp);

if (res == NULL)

{

return false;

}

if (mysql\_fetch\_row(res) == NULL)

{

mysql\_free\_result(res);

return false;

}

else

{

mysql\_free\_result(res);

return true;

}

}

bool MySQLManager::personInsert(char \*userid, char \*password, char \*pkey, char \*name, int namelen)

{

char \*table = charto16(userid, 33);

string sql = "create table if not exists file" + string(table+4) +

"(id int,fileid char(65),loc int,father int,filename text,filesize int,filekey text(260),type int)";

mysql\_free\_result(runSQL(sql.c\_str()));

sql = "create table if not exists friend" + string(table+6) +

"(friendid char(65) unique,friendname text,friendkey char(132))";

mysql\_free\_result(runSQL(sql.c\_str()));

char \*temp1 = charto16(password, 33);

char \*temp2 = charto16(pkey, 66);

//参数化插入防名字部分注入

person\_bind[0].buffer = table;

person\_bind[1].buffer = temp1;

person\_bind[2].buffer = name;

person\_bind[2].buffer\_length = namelen + 1;

person\_bind[3].buffer = temp2;

mysql\_stmt\_bind\_param(person\_stmt, person\_bind); //要绑定输入缓冲区

WaitForSingleObject(lock, INFINITE);

if (mysql\_stmt\_execute(person\_stmt) != 0)

{

ReleaseMutex(lock);

free(table);

free(temp1);

free(temp2);

return true;

}

else

{

ReleaseMutex(lock);

free(table);

free(temp1);

free(temp2);

return false;

}

}

#include"user.h"

#include"MySQLManager.h"

MySQLManager \*Mysql;

int ramp\_k;

int ramp\_m;

int ramp\_w;

int blocksize;

int packetsize;

int \*matrix;

int \*bitmatrix;

int \*\*schedule;

map<string, User \*> all\_users;

User::User(char \*name,char \*password,SSL \* clntSock)

{

userid = new char[33];

for (int i = 0; i < 32; i++)

{

userid[i] = name[i];

}

userid[32]=0;

this->clntSock = clntSock;

userpassword = new char[33];

for (int i = 0; i < 32; i++)

{

userpassword[i] = password[i];

}

userpassword[32] = 0;

}

User::~User()

{

delete[] userid;

delete[] userpassword;

}

void User::handleClnt()

{

char buffer[1];

//收到的字节数

int readnum = 0;

int flag = 1;

while (flag)

{

//printf("子线程%d\n", GetCurrentThreadId());

//printf("等待");

readnum = SSL\_read(clntSock, buffer, 1);

if (readnum <= 0)

{

printf("用户已断开连接!\n");

sign\_out();

flag = 0;

return;

}

else

{

switch (buffer[0]){

case SIGN\_OUT:

//这个判断的目的主要是判断数据库有没有出错

if (sign\_out())

{

printf("一用户已退出...\n");

//一旦退出就返回

return;

}

break;

case REQUIRE\_FILES:

//除非数据库出现错误，否则请求文件列表一定成功

require\_files();

printf("一用户已取得文件列表！\n");

break;

case REQUIRE\_FRIENDS:

require\_friends();

printf("一用户取得好友列表成功！\n");

break;

case ADD\_FRIEND:

if(add\_friend())

printf("一用户已添加一好友！\n");

else

printf("一用户添加好友失败！\n");

break;

case DELETE\_FILE:

if (deleteFile())

printf("一用户删除文件成功！\n");

else

printf("一用户删除文件失败！\n");

break;

case CREATE\_DIR:

if (createDir())

printf("一用户已创建目录\n");

else

printf("一用户创建目录失败\n");

break;

case SHARE\_FILE:

if (share\_file())

{

printf("一用户分享文件成功！\n");

}

else

{

printf("一用户分享文件失败！\n");

}

break;

default:

flag = 0;

break;

}

}

}

printf("用户不合法的行为导致连接被主动关闭！\n");

}

int User::sign\_out()

{

return Mysql->personLogout(userid);

}

bool User::require\_files()

{

char \*buffer=NULL;

int readnum = 0;

unsigned char \*sint = new unsigned char[4];

readnum = SSL\_read(clntSock, (char \*)sint, 4);

while (readnum != 4)

{

if (readnum <= 0)

{

delete[] sint;

return false;

}

readnum += SSL\_read(clntSock, (char \*)sint + readnum, 4 - readnum);

}

int id = bytes\_to\_int(sint);

readnum = SSL\_read(clntSock, (char \*)sint, 4);

while (readnum != 4)

{

if (readnum <= 0)

{

delete[] sint;

return false;

}

readnum += SSL\_read(clntSock, (char \*)sint + readnum, 4 - readnum);

}

int deep = bytes\_to\_int(sint);

int sendnum = Mysql->getFiles(userid, id, deep, &buffer);

SSL\_write(clntSock, buffer, sendnum);

delete[] buffer;

delete[] sint;

return true;

}

int User::require\_friends()

{

char \*buffer=NULL;

int len = Mysql->getFriends(userid, &buffer);

SSL\_write(clntSock, buffer, len);

delete[] buffer;

return 0;

}

int User::add\_friend()

{

char \*buffer=NULL;

char \*namehash = new char[33];

memset(namehash, 0, 33\*sizeof(char));

int readnum = SSL\_read(clntSock, namehash, 32);

while (readnum != 32)

{

if (readnum <= 0)

{

return 0;

}

readnum += SSL\_read(clntSock, namehash + readnum, 32 - readnum);

}

bool b = Mysql->getPerson(namehash, &buffer);

if (!b)

{

SSL\_write(clntSock, buffer, 1);

delete[] namehash;

delete[] buffer;

return 0;

}

SSL\_write(clntSock, buffer, 66);

Mysql->friendInsert(userid, namehash);

Mysql->friendInsert(namehash, userid);

delete[] namehash;

delete[] buffer;

return 1;

}

int User::createDir()

{

int readnum;

unsigned char sint[4];

readnum = SSL\_read(clntSock, (char \*)sint, 1);

int namelen = sint[0];

char \*name = new char[namelen+1];

readnum = SSL\_read(clntSock, name, namelen);

while (readnum != namelen)

{

if (readnum <= 0)

{

delete[] name;

return false;

}

readnum += SSL\_read(clntSock, name + readnum, namelen - readnum);

}

name[namelen] = 0;

readnum = SSL\_read(clntSock, (char \*)sint, 4);

while (readnum != 4)

{

if (readnum <= 0)

{

delete[] name;

return false;

}

readnum += SSL\_read(clntSock, (char \*)sint + readnum, 4 - readnum);

}

int id = bytes\_to\_int(sint);

readnum = SSL\_read(clntSock, (char \*)sint, 4);

while (readnum != 4)

{

if (readnum <= 0)

{

delete[] name;

return false;

}

readnum += SSL\_read(clntSock, (char \*)sint + readnum, 4 - readnum);

}

int deep = bytes\_to\_int(sint);

char temp1[33];

memset(temp1, 0xff, 32);

temp1[32] = 0;

char temp2[130];

memset(temp2, 0xff, 129);

temp2[129] = 0;

int tid = Mysql->writeFile(userid, name, namelen, 0, temp1, temp2, id, deep, 0);

if (tid!=0)

{

delete[] name;

char s[5];

s[0] = SUCCESS\_CREATE;

int\_to\_bytes(tid, (unsigned char \*)s + 1);

SSL\_write(clntSock, s, 5);

return true;

}

return false;

}

int User::deleteFile()

{

int readnum;

unsigned char sint[4];

readnum = SSL\_read(clntSock, (char \*)sint, 4);

while (readnum != 4)

{

if (readnum <= 0)

{

return false;

}

readnum += SSL\_read(clntSock, (char \*)sint + readnum, 4 - readnum);

}

int id = bytes\_to\_int(sint);

readnum = SSL\_read(clntSock, (char \*)sint, 4);

while (readnum != 4)

{

if (readnum <= 0)

{

return false;

}

readnum += SSL\_read(clntSock, (char \*)sint + readnum, 4 - readnum);

}

int deep = bytes\_to\_int(sint);

if (Mysql->deleteFile(userid, id, deep))

{

char c[1];

c[0] = SUCCESS\_DELETE;

SSL\_write(clntSock, c, 1);

return true;

}

else

{

char c[1];

c[0] = FAILED\_ACTION;

SSL\_write(clntSock, c, 1);

return false;

}

}

int User::upload\_files(SSL \* upSock)

{

int readnum = 0;

char c[4];

//获取文件名大小

SSL\_read(upSock, c, 1);

int namelen = (unsigned char)c[0];

char \*filename = new char[namelen+1];

//获取文件名

readnum = SSL\_read(upSock, filename, namelen);

while (readnum != namelen)

{

if (readnum <= 0)

{

delete[] filename;

return -1;

}

readnum += SSL\_read(upSock, filename + readnum, namelen - readnum);

}

filename[namelen] = 0;

//获取文件大小

readnum = SSL\_read(upSock, c, 4);

while (readnum != 4)

{

if (readnum <= 0)

{

delete[] filename;

return -1;

}

readnum += SSL\_read(upSock, c + readnum, 4 - readnum);

}

int filesize = bytes\_to\_int((unsigned char \*)c);

char fileID[33];

//获取文件的哈希的哈希

readnum = SSL\_read(upSock, fileID, 32);

while (readnum != 32)

{

if (readnum <= 0)

{

delete[] filename;

return -1;

}

readnum += SSL\_read(upSock, fileID + readnum, 32 - readnum);

}

fileID[32] = 0;

//获取经过加密的文件密钥

char fileKey[130];

readnum = SSL\_read(upSock, fileKey, 129);

while (readnum != 129)

{

if (readnum <= 0)

{

delete[] filename;

return -1;

}

readnum += SSL\_read(upSock, fileKey + readnum, 129 - readnum);

}

unsigned char \*locLen = new unsigned char[4];

readnum = SSL\_read(upSock, (char \*)locLen, 4);

while (readnum != 4)

{

if (readnum <= 0)

{

delete[] filename;

delete[] locLen;

return -1;

}

readnum += SSL\_read(upSock, (char \*)locLen + readnum, 4 - readnum);

}

//获取文件要放的位置

int deep = bytes\_to\_int(locLen);

int loc;

readnum = SSL\_read(upSock, (char \*)locLen, 4);

while (readnum != 4)

{

if (readnum <= 0)

{

delete[] filename;

return -1;

}

readnum += SSL\_read(upSock, (char \*)locLen + readnum, 4 - readnum);

}

loc = bytes\_to\_int(locLen);

if (Mysql->fileExist(fileID))

{

srand((unsigned int)time(0));

int left = ((((int)rand()) << 8) + rand()) % filesize;

int temp = rand() % 32;

int right = (left + temp) >= filesize ? filesize - 1 : left + temp;

if (upload\_partfile(filesize, fileID,upSock,left,right) && Mysql->writeFile(userid,filename, namelen, filesize, fileID, fileKey, loc, deep,1))

{

printf("用户校验文件成功！\n");

delete[] filename;

delete[] locLen;

return 1;

}

else

{

printf("用户上传文件失败！\n");

delete[] filename;

delete[] locLen;

return 0;

}

}

else

{

char buffer[25];

buffer[0] = SEND\_FILE;

randomiv(buffer + 1);

int\_to\_bytes(1, (unsigned char \*)(buffer + 17));

int\_to\_bytes(filesize, (unsigned char \*)(buffer + 21));

SSL\_write(upSock, buffer, 25);

SSL\_read(upSock, c, 1);

if (c[0] == UPLOAD\_FILE && recvfile(filesize, fileID, upSock) && Mysql->writeFile(userid, filename, namelen, filesize, fileID, fileKey, loc, deep, 1))

{

Mysql->writeFile(filesize, fileID);

printf("用户上传文件成功！\n");

delete[] filename;

delete[] locLen;

return 1;

}

else

{

printf("用户上传文件失败！\n");

delete[] filename;

delete[] locLen;

return 0;

}

}

}

bool User::upload\_partfile(int filesize, char \*fileID, SSL \* upSock, int left,int right)

{

//k个分开后的文件的指针数组

FILE \*\*fd;

//m个冗余块的文件的指针数组

FILE \*\*fs;

//分为k块的文件的数据数组的指针

char \*\*data;

//分为m块的冗余码的数组的指针

char \*\*coding;

//缓冲区的指针，存放k块blocksize大小的文件数据内容，也就是data数据

char \*buffer;

char \*checkdata;

//文件划分的块的总数

int blocknum;

//缺失的块的编号数组，最后一位为-1

int erasures[100];

//标示缺失的块的数组，某个数为1说明其下标对应的块缺失

int erase[100];

int e = 0;

//临时文件名

char fname[1000];

int size = filesize \* 16;

//printf("验证开始：%ld\n", clock());

memset(erase, 0, sizeof(char)\* 100);

memset(erasures, 0, sizeof(char)\* 100);

data = (char \*\*)malloc(sizeof(char\*)\*ramp\_k);

coding = (char \*\*)malloc(sizeof(char\*)\*ramp\_m);

fd = (FILE \*\*)malloc(sizeof(FILE\*)\*ramp\_k);

fs = (FILE \*\*)malloc(sizeof(FILE\*)\*ramp\_m);

for (int i = 0; i < ramp\_m; i++)

{

coding[i] = (char \*)malloc(sizeof(char)\*blocksize);

memset(coding[i], 0, sizeof(char)\*ramp\_w\*packetsize);

}

//保证文件夹的存在

char mname[256];

for (int i = 0; i < ramp\_k; i++)

{

sprintf\_s(mname, 100, "file\_k\_%d", i);

if (\_access(mname, 0) == -1)

{

\_mkdir(mname);

}

}

for (int i = 0; i < ramp\_m; i++)

{

sprintf\_s(mname, 100, "file\_m\_%d", i);

if (\_access(mname, 0) == -1)

{

\_mkdir(mname);

}

}

char \*temp;

//打开各分块和冗余码所在文件

for (int i = 0; i < ramp\_k; i++)

{

temp = charto16(fileID, 33);

sprintf\_s(fname, 100, "file\_k\_%d/%s", i, charto16(fileID, 33));

free(temp);

fopen\_s(fd + i, fname, "rb");

//获得blocknum

if (\*(fd + i) != NULL)

{

fseek(\*(fd + i), 0, SEEK\_END);

blocknum = ftell(\*(fd + i)) / blocksize;

fseek(\*(fd + i), 0, SEEK\_SET);

}

else{

erasures[e++] = i;

erase[i] = 1;

temp = charto16(fileID, 33);

sprintf\_s(fname, 100, "file\_k\_%d/%s", i, charto16(fileID, 33));

free(temp);

fopen\_s(fd + i, fname, "wb");

}

}

for (int i = 0; i < ramp\_m; i++)

{

temp = charto16(fileID, 33);

sprintf\_s(fname, 100, "file\_m\_%d/%s", i, charto16(fileID, 33));

free(temp);

fopen\_s(fs + i, fname, "rb");

if (\*(fs + i) != NULL)

{

fseek(\*(fs + i), 0, SEEK\_END);

blocknum = ftell(\*(fs + i)) / blocksize;

fseek(\*(fs + i), 0, SEEK\_SET);

}

else{

erasures[e++] = i + ramp\_k;

erase[i + ramp\_k] = 1;

temp = charto16(fileID, 33);

sprintf\_s(fname, 100, "file\_m\_%d/%s", i, charto16(fileID, 33));

free(temp);

fopen\_s(fs + i, fname, "wb");

}

}

//获得解码后的文件的大小

size = blocknum\*blocksize\*ramp\_k;

erasures[e] = -1;

int xl = left \* 16 / (blocksize\*ramp\_k);

int xr;

if (left \* 16 % (blocksize\*ramp\_k) == 0)

xl--;

if (xl < 0)

xl++;

xr = xl + 1;

if (e == ramp\_m + ramp\_k)

{

for (int i = 0; i < ramp\_k; i++)

{

fflush(fd[i]);

fclose(fd[i]);

}

free(fd);

for (int i = 0; i < ramp\_m; i++)

{

fflush(fs[i]);

fclose(fs[i]);

}

free(fs);

free(data);

for (int i = 0; i < ramp\_m; i++)

free(coding[i]);

free(coding);

buffer = (char \*)malloc(50 \* sizeof(char));

char \*c = (char \*)malloc(sizeof(char)\* 4);

memset(buffer, 0, sizeof(char)\* 50);

buffer[0] = SEND\_FILE;

randomiv(buffer + 1);

int\_to\_bytes(1, (unsigned char \*)(buffer + 17));

int\_to\_bytes(filesize, (unsigned char \*)(buffer + 21));

SSL\_write(upSock, buffer, 25);

SSL\_read(upSock, c, 1);

if ((unsigned char)c[0] == UPLOAD\_FILE&&recvfile(filesize, fileID, upSock) == 1)

{

free(c);

free(buffer);

return true;

}

else

{

free(c);

free(buffer);

return false;

}

}

//假如存在块缺失的话，要先把缺失的块补齐

if (erasures[0] != -1)

{

recover\_file(fd, fs, erase, erasures, blocknum);

}

checkdata = (char \*)malloc(sizeof(char)\* 2 \* blocksize\*ramp\_k);

buffer = (char \*)malloc(sizeof(char)\* blocksize\*ramp\_k);

int ct = xr - xl;

for (int i = 0; i < ramp\_k; i++)

{

//fseek移动不过去，所以采用关闭文件再打开的方式

//fseek(fd[i], xl\*blocksize, SEEK\_SET);

/\*printf("%d\n", fseek(fd[i], xl\*blocksize, SEEK\_SET));\*/

fclose(fd[i]);

temp = charto16(fileID, 33);

sprintf\_s(fname, 100, "file\_k\_%d/%s", i, temp);

fopen\_s(fd + i, fname, "rb");

free(temp);

fseek(fd[i], xl\*blocksize, SEEK\_SET);

}

memset(buffer, 0xff, sizeof(char)\*blocksize\*ramp\_k);

while (ct--)

{

memset(buffer, 0, sizeof(char)\*blocksize\*ramp\_k);

for (int i = 0; i < ramp\_k; i++)

{

fread(buffer + i\*blocksize, sizeof(char), blocksize, fd[i]);

//printf("%s\n", charto16(buffer + i\*blocksize));

}

int n = blocksize\*ramp\_k;

memcpy\_s(checkdata + ct\*blocksize\*ramp\_k, blocksize\*ramp\_k, buffer, blocksize\*ramp\_k);

}

xl = ((left - 1) \* 16 + blocksize\*ramp\_k) % (blocksize\*ramp\_k);

char \*sbuffer;

//发送请求文件某部分的文件帧

sbuffer = (char \*)malloc(50 \* sizeof(char));

sbuffer[0] = SEND\_FILE;

memcpy\_s(sbuffer + 1, 16, checkdata + xl, 16);

int\_to\_bytes(left, (unsigned char \*)(sbuffer + 17));

int\_to\_bytes(right, (unsigned char \*)(sbuffer + 21));

//printf("%s\n", charto16(checkdata + xl));

SSL\_write(upSock, sbuffer, 25);

char c[1];

SSL\_read(upSock, c, 1);

bool ret;

//帧格式正确并且验证文件正确，就写入数据

if ((unsigned char)c[0] == UPLOAD\_FILE&&check\_partfile(upSock, checkdata + xl, right - left + 1))

{

ret = true;

//printf("验证结束：%ld\n", clock());

}

else

{

ret = false;

}

free(buffer);

free(sbuffer);

free(checkdata);

for (int i = 0; i < ramp\_k; i++)

{

fflush(fd[i]);

fclose(fd[i]);

}

free(fd);

for (int i = 0; i < ramp\_m; i++)

{

fflush(fs[i]);

fclose(fs[i]);

}

free(fs);

free(data);

for (int i = 0; i < ramp\_m; i++)

free(coding[i]);

free(coding);

return ret;

}

//恢复文件

void User::recover\_file(FILE \*\*fd, FILE \*\*fs, int \*erase, int \*erasures, int blocknum)

{

//分为k块的文件的数据数组的指针

char \*\*data;

//分为m块的冗余码的数组的指针

char \*\*coding;

//缓冲区的指针，存放k块blocksize大小的文件数据内容，也就是data数据

char \*buffer;

//printf("恢复开始：%ld\n", clock());

data = (char \*\*)malloc(sizeof(char\*)\*ramp\_k);

coding = (char \*\*)malloc(sizeof(char\*)\*ramp\_m);

buffer = (char \*)malloc(ramp\_k\*blocksize);

for (int i = 0; i < ramp\_m; i++)

{

coding[i] = (char \*)malloc(sizeof(char)\*blocksize);

memset(coding[i], 0, sizeof(char)\*ramp\_w\*packetsize);

}

while (blocknum--)

{

memset(buffer, 0, sizeof(char)\*blocksize\*ramp\_k);

//读取数据到data和coding

for (int i = 0; i < ramp\_k; i++)

{

if (erase[i] == 0)

fread(buffer + i\*blocksize, sizeof(char), blocksize, fd[i]);

data[i] = buffer + i\*blocksize;

}

for (int i = 0; i < ramp\_m; i++)

{

if (erase[i + ramp\_k] == 0)

fread(coding[i], sizeof(char), blocksize, fs[i]);

}

jerasure\_schedule\_decode\_lazy(ramp\_k, ramp\_m, ramp\_w, bitmatrix, erasures, data, coding, packetsize\*ramp\_w, packetsize, 1);

//假如有数据块或编码块丢失的话，恢复这些块

for (int i = 0; i < ramp\_k; i++)

{

if (erase[i] != 0)

fwrite(data[i], sizeof(char), blocksize, fd[i]);

}

for (int i = 0; i < ramp\_m; i++)

{

if (erase[i + ramp\_k] != 0)

fwrite(coding[i], sizeof(char), blocksize, fs[i]);

}

}

//printf("恢复结束：%ld\n", clock());

for (int i = 0; i < ramp\_k; i++)

{

fflush(fd[i]);

}

for (int i = 0; i < ramp\_m; i++)

{

fflush(fs[i]);

}

free(data);

for (int i = 0; i < ramp\_m; i++)

free(coding[i]);

free(coding);

free(buffer);

}

//检查一部分的文件是否相同

bool User::check\_partfile(SSL \* upSock, char \*checkdata, int ct)

{

int readnum = 0;

char \*SSL\_readdata;

SSL\_readdata = (char \*)malloc(sizeof(char)\*ct \* 16);

//接收传来的部分文件

readnum = SSL\_read(upSock, SSL\_readdata, ct \* 16);

while (readnum != ct \* 16)

{

if (readnum <= 0)

{

free(SSL\_readdata);

return false;

}

readnum += SSL\_read(upSock, SSL\_readdata + readnum, ct \* 16 - readnum);

}

/\*printf("%s\n", charto16(checkdata));

printf("-%s\n", charto16(SSL\_readdata));\*/

//检查文件是否相同

for (int i = 0; i < ct \* 16; i++)

{

if (checkdata[i] != SSL\_readdata[i])

{

free(SSL\_readdata);

return false;

}

}

free(SSL\_readdata);

return true;

}

int User::download\_file(SSL \* doSock)

{

char fileID[33];

memset(fileID, 0, 33 \* sizeof(char));

int readnum = 0;

int sendnum = 0;

readnum = SSL\_read(doSock, fileID, 32);

//读取文件的哈希的哈希

while (readnum != 32)

{

if (readnum <= 0)

{

return -1;

}

readnum += SSL\_read(doSock, fileID + readnum, 32 - readnum);

}

fileID[32] = 0;

//如果用户拥有该文件，就给该用户传该文件

int size = Mysql->fileIsPerson(fileID, userid);

if (size>=0)

{

char \*firstchar = (char \*)malloc(sizeof(char));

firstchar[0] = SEND\_FILE;

SSL\_write(doSock, firstchar, 1);

send\_file(doSock, fileID, size);

//free(filename);

free(firstchar);

return 1;

}

return 0;

}

bool User::share\_file()

{

int readnum = 0;

char myfriend[33];

char fileID[33];

char fileKey[130];

char c[4];

bool ret = true;

//获取好友名的哈希

readnum = SSL\_read(clntSock, myfriend, 32);

while (readnum != 32)

{

if (readnum <= 0)

{

return false;

}

readnum += SSL\_read(clntSock, myfriend + readnum, 32 - readnum);

}

myfriend[32] = 0;

if (!Mysql->personIsFriend(userid, myfriend))

{

return false;

}

//获取根节点数目

readnum = SSL\_read(clntSock, c, 4);

while (readnum != 4)

{

if (readnum <= 0)

{

return false;

}

readnum += SSL\_read(clntSock, c + readnum, 4 - readnum);

}

int rootnum = bytes\_to\_int((unsigned char \*)c);

char \*filename=NULL;

int filesize;

int namelen;

for (int i = 0; i < rootnum; i++)

{

//获取文件名大小

SSL\_read(clntSock, c, 1);

filename = (char \*)malloc((unsigned char)c[0] \* sizeof(char)+1);

namelen = (unsigned char)c[0];

//获取文件名

readnum = SSL\_read(clntSock, filename, namelen);

while (readnum != readnum)

{

if (readnum <= 0)

{

free(filename);

return false;

}

readnum += SSL\_read(clntSock, filename + readnum, namelen - readnum);

}

filename[namelen] = 0;

//获取文件大小

readnum = SSL\_read(clntSock, c, 4);

while (readnum != 4)

{

if (readnum <= 0)

{

free(filename);

return false;

}

readnum += SSL\_read(clntSock, c + readnum, 4 - readnum);

}

filesize = bytes\_to\_int((unsigned char \*)c);

//获取文件类型

SSL\_read(clntSock, c, 1);

//文件夹的情况

if (c[0] == 0){

//获取子节点数目

int childnum;

readnum = SSL\_read(clntSock, c, 4);

while (readnum != 4)

{

if (readnum <= 0)

{

free(filename);

return false;

}

readnum += SSL\_read(clntSock, c + readnum, 4 - readnum);

}

childnum = bytes\_to\_int((unsigned char \*)c);

char temp1[33];

memset(temp1, 0xff, 32);

temp1[32] = 0;

char temp2[130];

memset(temp2, 0xff, 129);

temp2[129] = 0;

int loc;

if (Mysql->fileIsPerson(fileID, userid))

{

loc= Mysql->writeFile(myfriend, filename, namelen, 0, temp1, temp2, 0, 1, 0);

ret &= writeTreeFiles(myfriend, loc, 2, childnum);

}

else

{

ret &= false;

}

}

else

{

//文件的情况

//获取文件的哈希的哈希

readnum = SSL\_read(clntSock, fileID, 32);

while (readnum != 32)

{

if (readnum <= 0)

{

free(filename);

return false;

}

readnum += SSL\_read(clntSock, fileID + readnum, 32 - readnum);

}

fileID[32] = 0;

//获取经过加密的文件密钥

readnum = SSL\_read(clntSock, fileKey, 129);

while (readnum != 129)

{

if (readnum <= 0)

{

free(filename);

return false;

}

readnum += SSL\_read(clntSock, fileKey + readnum, 129 - readnum);

}

fileKey[129] = 0;

//要是用户可以分享的话，要保证文件该用户有，且分享的是该用户的好友

if (Mysql->fileIsPerson(fileID, userid))

{

Mysql->writeFile(myfriend, filename, namelen, filesize, fileID, fileKey, 0, 1, 1);

}

else

{

ret &= false;

}

}

}

if (ret)

{

c[0] = SUCCESS\_SHARE;

SSL\_write(clntSock, c, 1);

free(filename);

return true;

}

else

{

c[0] = FAILED\_ACTION;

SSL\_write(clntSock, c, 1);

free(filename);

return false;

}

}

//向好友写入树状的文件信息

bool User::writeTreeFiles(char \*friendID,int loc, int deep, int num)

{

char c[4];

char \*filename=NULL;

int namelen;

int readnum;

int filesize;

bool ret = true;

char fileID[33];

char fileKey[130];

for (int i = 0; i < num; i++)

{

//获取文件名大小

SSL\_read(clntSock, c, 1);

filename = (char \*)malloc((unsigned char)c[0] \* sizeof(char)+1);

namelen = (unsigned char)c[0];

//获取文件名

readnum = SSL\_read(clntSock, filename, namelen);

while (readnum != readnum)

{

if (readnum <= 0)

{

free(filename);

return false;

}

readnum += SSL\_read(clntSock, filename + readnum, namelen - readnum);

}

filename[namelen] = 0;

//获取文件大小

readnum = SSL\_read(clntSock, c, 4);

while (readnum != 4)

{

if (readnum <= 0)

{

free(filename);

return false;

}

readnum += SSL\_read(clntSock, c + readnum, 4 - readnum);

}

filesize = bytes\_to\_int((unsigned char \*)c);

//获取文件类型

SSL\_read(clntSock, c, 1);

//文件夹的情况

if (c[0] == 0){

//获取子节点数目

int childnum;

readnum = SSL\_read(clntSock, c, 4);

while (readnum != 4)

{

if (readnum <= 0)

{

free(filename);

return false;

}

readnum += SSL\_read(clntSock, c + readnum, 4 - readnum);

}

childnum = bytes\_to\_int((unsigned char \*)c);

char temp1[33];

memset(temp1, 0xff, 32);

temp1[32] = 0;

char temp2[130];

memset(temp2, 0xff, 129);

temp2[129] = 0;

int newloc = 0;

if (Mysql->fileIsPerson(fileID, userid))

{

int newloc = Mysql->writeFile(friendID, filename, namelen, 0, temp1, temp2, loc, deep, 0);

ret &= writeTreeFiles(friendID,newloc, deep + 1, childnum);

}

else

{

ret &= false;

}

}

else

{

readnum = SSL\_read(clntSock, fileID, 32);

while (readnum != 32)

{

if (readnum <= 0)

{

free(filename);

return false;

}

readnum += SSL\_read(clntSock, fileID + readnum, 32 - readnum);

}

fileID[32] = 0;

//获取经过加密的文件密钥

readnum = SSL\_read(clntSock, fileKey, 129);

while (readnum != 129)

{

if (readnum <= 0)

{

free(filename);

return false;

}

readnum += SSL\_read(clntSock, fileKey + readnum, 129 - readnum);

}

fileKey[129] = 0;

if (Mysql->fileIsPerson(fileID, userid))

{

Mysql->writeFile(friendID, filename, namelen, filesize, fileID, fileKey, loc, deep, 1);

}

else

{

ret &= false;

}

}

}

if (ret)

{

free(filename);

return true;

}

else

{

free(filename);

return false;

}

}

void User::send\_file(SSL \* clntSock, char \*filename, int size)

{

//k个分开后的文件的指针数组

FILE \*\*fd;

//m个冗余块的文件的指针数组

FILE \*\*fs;

//分为k块的文件的数据数组的指针

char \*\*data;

//分为m块的冗余码的数组的指针

char \*\*coding;

//缓冲区的指针，存放k块blocksize大小的文件数据内容，也就是data数据

char \*buffer;

//文件划分的块的总数

int blocknum;

//缺失的块的编号数组，最后一位为-1

int erasures[100];

//标示缺失的块的数组，某个数为1说明其下标对应的块缺失

int erase[100];

int e = 0;

//临时文件名

char fname[1000];

size = size \* 16;

memset(erase, 0, sizeof(erase));

data = (char \*\*)malloc(sizeof(char\*)\*ramp\_k);

coding = (char \*\*)malloc(sizeof(char\*)\*ramp\_m);

buffer = (char \*)malloc(ramp\_k\*blocksize);

fd = (FILE \*\*)malloc(sizeof(FILE\*)\*ramp\_k);

fs = (FILE \*\*)malloc(sizeof(FILE\*)\*ramp\_m);

for (int i = 0; i < ramp\_m; i++)

{

coding[i] = (char \*)malloc(sizeof(char)\*blocksize);

memset(coding[i], 0, sizeof(char)\*ramp\_w\*packetsize);

}

//保证文件夹的存在

char mname[256];

if (\_access("user\_file", 0) == -1)

{

\_mkdir("user\_file");

}

for (int i = 0; i < ramp\_k; i++)

{

sprintf\_s(mname, 100, "file\_k\_%d", i);

if (\_access(mname, 0) == -1)

{

\_mkdir(mname);

}

}

for (int i = 0; i < ramp\_m; i++)

{

sprintf\_s(mname, 100, "file\_m\_%d", i);

if (\_access(mname, 0) == -1)

{

\_mkdir(mname);

}

}

char \*temp;

//打开各分块和冗余码所在文件

for (int i = 0; i < ramp\_k; i++)

{

temp = charto16(filename, 33);

sprintf\_s(fname, 100, "file\_k\_%d/%s", i, temp);

fopen\_s(fd + i, fname, "rb");

free(temp);

//获得blocknum

if (\*(fd + i) != NULL)

{

fseek(\*(fd + i), 0, SEEK\_END);

blocknum = ftell(\*(fd + i)) / blocksize;

fseek(\*(fd + i), 0, SEEK\_SET);

}

else{

erasures[e++] = i;

erase[i] = 1;

temp = charto16(filename, 33);

sprintf\_s(fname, 100, "file\_k\_%d/%s", i, charto16(filename, 33));

fopen\_s(fd + i, fname, "wb");

free(temp);

}

}

for (int i = 0; i < ramp\_m; i++)

{

temp = charto16(filename, 33);

sprintf\_s(fname, 100, "file\_m\_%d/%s", i, charto16(filename, 33));

free(temp);

fopen\_s(fs + i, fname, "rb");

if (\*(fs + i) != NULL)

{

fseek(\*(fs + i), 0, SEEK\_END);

blocknum = ftell(\*(fs + i)) / blocksize;

fseek(\*(fs + i), 0, SEEK\_SET);

}

else{

erasures[e++] = i + ramp\_k;

erase[i + ramp\_k] = 1;

temp = charto16(filename, 33);

sprintf\_s(fname, 100, "file\_m\_%d/%s", i, charto16(filename, 33));

fopen\_s(fs + i, fname, "wb");

free(temp);

}

}

//获得解码后的文件的大小

size = blocknum\*blocksize\*ramp\_k;

erasures[e] = -1;

//printf("下载开始：%ld\n", clock());

while (blocknum--)

{

memset(buffer, 0, sizeof(char)\*blocksize\*ramp\_k);

//读取数据到data和coding

for (int i = 0; i < ramp\_k; i++)

{

if (erase[i] == 0)

fread(buffer + i\*blocksize, sizeof(char), blocksize, fd[i]);

data[i] = buffer + i\*blocksize;

}

for (int i = 0; i < ramp\_m; i++)

{

if (erase[i + ramp\_k] == 0)

fread(coding[i], sizeof(char), blocksize, fs[i]);

}

if (erasures[0] != -1)

jerasure\_schedule\_decode\_lazy(ramp\_k, ramp\_m, ramp\_w, bitmatrix, erasures, data, coding, packetsize\*ramp\_w, packetsize, 1);

//一次向文件中写入的字节数

int n = blocksize\*ramp\_k;

//除去填充

if (blocknum == 0)

{

while (buffer[--n] == 0);

}

SSL\_write(clntSock, buffer, n);

//假如有数据块或编码块丢失的话，仍要恢复这些块

for (int i = 0; i < ramp\_k; i++)

{

if (erase[i] != 0)

fwrite(data[i], sizeof(char), blocksize, fd[i]);

}

for (int i = 0; i < ramp\_m; i++)

{

if (erase[i + ramp\_k] != 0)

fwrite(coding[i], sizeof(char), blocksize, fs[i]);

}

}

//释放所有申请的空间

for (int i = 0; i < ramp\_k; i++)

{

fflush(fd[i]);

fclose(fd[i]);

}

free(fd);

for (int i = 0; i < ramp\_m; i++)

{

fflush(fs[i]);

fclose(fs[i]);

}

free(fs);

free(data);

for (int i = 0; i < ramp\_m; i++)

free(coding[i]);

free(coding);

free(buffer);

return;

}

bool User::recvfile(int size,char \*filename,SSL \* upSock)

{

FILE \*\*fd;

FILE \*\*fs;

char \*\*data;

char \*\*coding;

char \*buffer;

int blocknum;

char fname[1000];

size = size \* 16;

data = (char \*\*)malloc(sizeof(char\*)\*ramp\_k);

coding = (char \*\*)malloc(sizeof(char\*)\*ramp\_m);

buffer = (char \*)malloc(ramp\_k\*blocksize);

memset(buffer, 0, ramp\_k\*blocksize\*sizeof(char));

fd = (FILE \*\*)malloc(sizeof(FILE\*)\*ramp\_k);

fs = (FILE \*\*)malloc(sizeof(FILE\*)\*ramp\_m);

for (int i = 0; i < ramp\_m; i++)

{

coding[i] = (char \*)malloc(sizeof(char)\*blocksize);

memset(coding[i], 0, sizeof(char)\*ramp\_w\*packetsize);

}

char mname[256];

for (int i = 0; i < ramp\_k; i++)

{

sprintf\_s(mname, 100, "file\_k\_%d", i);

if (\_access(mname, 0) == -1)

{

\_mkdir(mname);

}

}

for (int i = 0; i < ramp\_m; i++)

{

sprintf\_s(mname, 100, "file\_m\_%d", i);

if (\_access(mname, 0) == -1)

{

\_mkdir(mname);

}

}

char \*temp;

for (int i = 0; i < ramp\_k; i++)

{

temp = charto16(filename, 33);

sprintf\_s(fname, 100, "file\_k\_%d/%s", i, temp);

fopen\_s(fd + i, fname, "wb");

free(temp);

}

for (int i = 0; i < ramp\_m; i++)

{

temp = charto16(filename, 33);

sprintf\_s(fname, 100, "file\_m\_%d/%s", i, charto16(filename,33));

fopen\_s(fs + i, fname, "wb");

free(temp);

}

if (fd[0] == NULL)

{

free(fd);

free(fs);

free(data);

for (int i = 0; i < ramp\_m; i++)

free(coding[i]);

free(coding);

free(buffer);

return false;

}

blocknum = size / (blocksize\*ramp\_k) + 1;

while (blocknum--)

{

int haveread = 0;

if (blocknum != 0)

{

haveread = SSL\_read(upSock, buffer, blocksize\*ramp\_k);

while (haveread != blocksize\*ramp\_k)

{

if (haveread <= 0)

{

for (int i = 0; i < ramp\_k; i++)

{

fflush(fd[i]);

fclose(fd[i]);

}

free(fd);

for (int i = 0; i < ramp\_m; i++)

{

fflush(fs[i]);

fclose(fs[i]);

}

free(fs);

free(data);

for (int i = 0; i < ramp\_m; i++)

free(coding[i]);

free(coding);

free(buffer);

return false;

}

haveread += SSL\_read(upSock, buffer + haveread, blocksize\*ramp\_k - haveread);

}

}

else

{

haveread = SSL\_read(upSock, buffer, size % (blocksize\*ramp\_k));

while (haveread != size % (blocksize\*ramp\_k))

{

if (haveread <= 0)

{

for (int i = 0; i < ramp\_k; i++)

{

fflush(fd[i]);

fclose(fd[i]);

}

free(fd);

for (int i = 0; i < ramp\_m; i++)

{

fflush(fs[i]);

fclose(fs[i]);

}

free(fs);

free(data);

for (int i = 0; i < ramp\_m; i++)

free(coding[i]);

free(coding);

free(buffer);

return false;

}

haveread += SSL\_read(upSock, buffer + haveread, size % (blocksize\*ramp\_k) - haveread);

}

buffer[size % (blocksize\*ramp\_k)] = 1;

}

for (int i = 0; i < ramp\_k; i++)

{

data[i] = buffer + i\*blocksize;

}

jerasure\_schedule\_encode(ramp\_k, ramp\_m, ramp\_w, schedule, data, coding, blocksize, packetsize);

for (int i = 0; i < ramp\_k; i++)

{

fwrite(data[i], sizeof(char), blocksize, fd[i]);

}

for (int i = 0; i < ramp\_m; i++)

{

fwrite(coding[i], sizeof(char), blocksize, fs[i]);

}

memset(buffer, 0, sizeof(char)\*blocksize\*ramp\_k);

}

for (int i = 0; i < ramp\_k; i++)

{

fflush(fd[i]);

fclose(fd[i]);

}

free(fd);

for (int i = 0; i < ramp\_m; i++)

{

fflush(fs[i]);

fclose(fs[i]);

}

free(fs);

free(data);

for (int i = 0; i < ramp\_m; i++)

free(coding[i]);

free(coding);

free(buffer);

return true;

}

User \*sign\_in(SSL \* clntSock)

{

char userid[33];

char userpassword[33];

char cookies[33];

int readnum = 0;

readnum = SSL\_read(clntSock, userid, 32);

while (readnum != 32)

{

if (readnum <= 0)

return NULL;

readnum += SSL\_read(clntSock, userid + readnum, 32 - readnum);

}

userid[32] = 0;

readnum = SSL\_read(clntSock, userpassword, 32);

while (readnum != 32)

{

if (readnum <= 0)

return NULL;

readnum += SSL\_read(clntSock, userpassword + readnum, 32 - readnum);

}

userpassword[32] = 0;

if (Mysql->personLogin(userid,userpassword))

{

char f[1];

f[0] = SUCCESS\_LOGN;

randomiv(cookies);

randomiv(cookies + 16);

SSL\_write(clntSock, f, 1);

SSL\_write(clntSock, cookies, 32);

char \*hexcookies=charto16(cookies,33);

hexcookies[65] = 0;

User \*user = new User(userid, userpassword, clntSock);

all\_users[hexcookies] = user;

free(hexcookies);

return user;

}

else

{

char f[1];

f[0] = FAILED\_ACTION;

SSL\_write(clntSock, f, 1);

return NULL;

}

}

User \*sign\_up(SSL \* clntSock)

{

char userid[33];

char userpassword[33];

char cookies[33];

int readnum = 0;

readnum = SSL\_read(clntSock, userid, 32);

while (readnum != 32)

{

if (readnum <= 0)

return NULL;

readnum += SSL\_read(clntSock, userid + readnum, 32 - readnum);

}

userid[32] = 0;

readnum = SSL\_read(clntSock, userpassword, 32);

while (readnum != 32)

{

if (readnum <= 0)

return NULL;

readnum += SSL\_read(clntSock, userpassword + readnum, 32 - readnum);

}

userpassword[32] = 0;

char c[1];

SSL\_read(clntSock,c, 1);

int namelen = (unsigned char)c[0];

char \*name = (char \*)malloc((namelen + 1)\*sizeof(char));

readnum = SSL\_read(clntSock, name,namelen);

while (readnum != namelen)

{

if (readnum <= 0)

{

return NULL;

}

readnum += SSL\_read(clntSock, name + readnum, namelen - readnum);

}

name[namelen] = 0;

char userPkey[66];

readnum = SSL\_read(clntSock, userPkey, 65);

while (readnum != 65)

{

if (readnum <= 0)

{

return NULL;

}

readnum += SSL\_read(clntSock, userPkey + readnum, 65 - readnum);

}

userPkey[65] = 0;

if (!Mysql->personExist(userid))

{

Mysql->personInsert(userid, userpassword, userPkey, name, namelen);

char f[1];

f[0] = SUCCESS\_LOGN;

randomiv(cookies);

randomiv(cookies + 16);

SSL\_write(clntSock, f, 1);

SSL\_write(clntSock, cookies, 32);

char \*hexcookies = charto16(cookies, 33);

hexcookies[65] = 0;

User \*user = new User(userid, userpassword, clntSock);

all\_users[hexcookies] = user;

free(hexcookies);

free(name);

return user;

}

else

{

char f[1];

f[0] = FAILED\_ACTION;

SSL\_write(clntSock, f, 1);

free(name);

return NULL;

}

}

int bytes\_to\_int(unsigned char \*b)

{

int i = 0;

for (int j = 3; j >= 0; j--)

{

i = i << 8;

i += b[j];

}

return i;

}

void int\_to\_bytes(int i, unsigned char \*b)

{

memset(b, 0, sizeof(unsigned char)\* 4);

for (int j = 0; j < 4; j++)

{

b[j] = i % 256;

i = i / 256;

}

return;

}

void randomiv(char \*iv)

{

srand((unsigned int)time(0));

for (int i = 0; i < 16; i++)

iv[i] = (char)(rand() % 256);

return;

}

char \*charto16(char \*s,int size)

{

char \*c = (char \*)malloc(size\*2 \* sizeof(char));

memset(c, 0, size \* 2 \* sizeof(char));

for (int i = 0; i < size-1; i++)

sprintf\_s(c + 2 \* i, size \* 2 - 2 \* i, "%02x ", (unsigned char)s[i]);

return c;

}

void hextochar(char \*src, char \*des, int size)

{

memset(des, 0, size);

for (int i = 0; i < size; i++)

{

if (src[2\*i] <= '9'&&src[2\*i] >= '0')

des[i] += src[2\*i] - '0';

else

des[i] += src[2\*i] - 'a' + 10;

des[i] \*= 16;

if (src[2\*i+1] <= '9'&&src[2\*i+1] >= '0')

des[i] += src[2\*i+1] - '0';

else

des[i] += src[2\*i+1] - 'a' + 10;

}

}