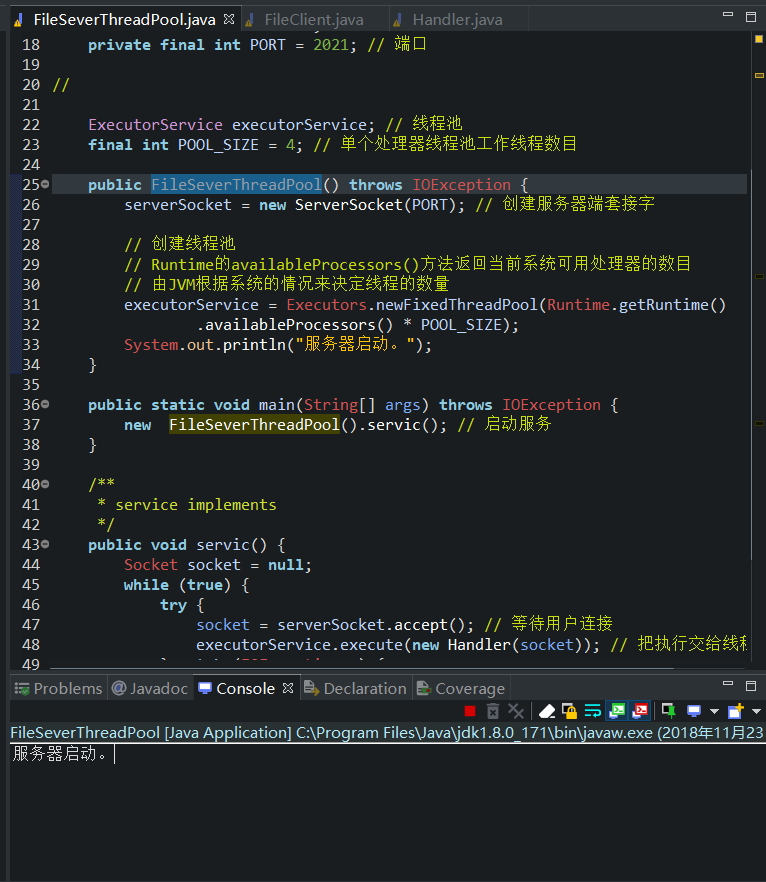
**Exercise 1 Network File Service**

**Based on TCP&amp;UDP**

Startup program

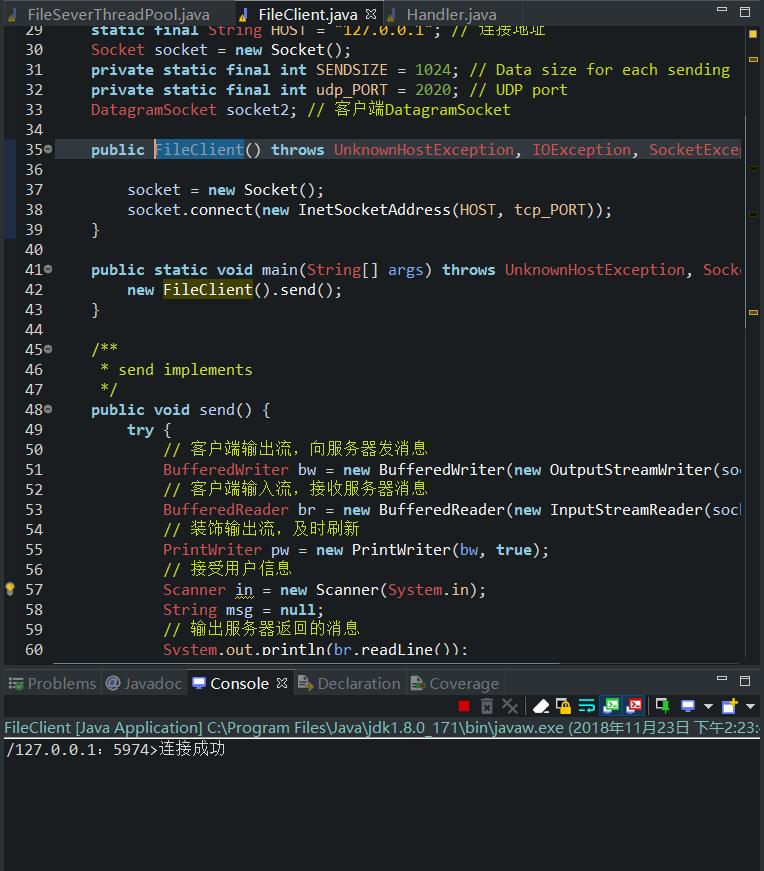
1. open FileSeverThreadPool

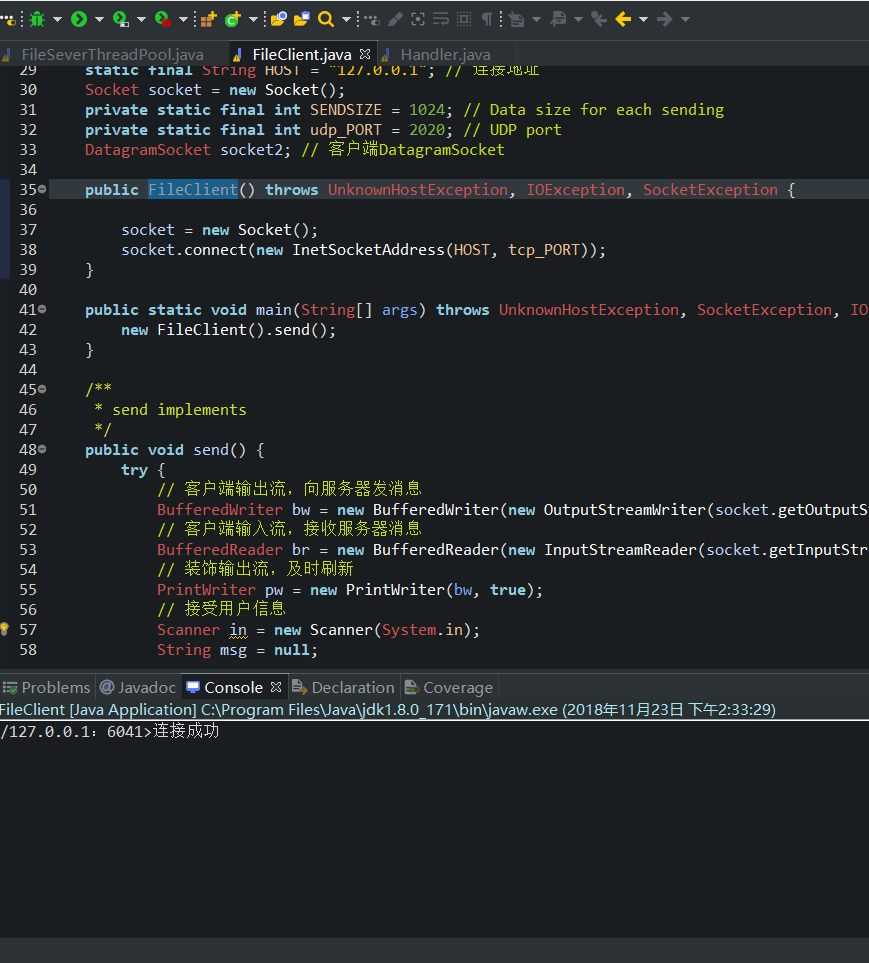


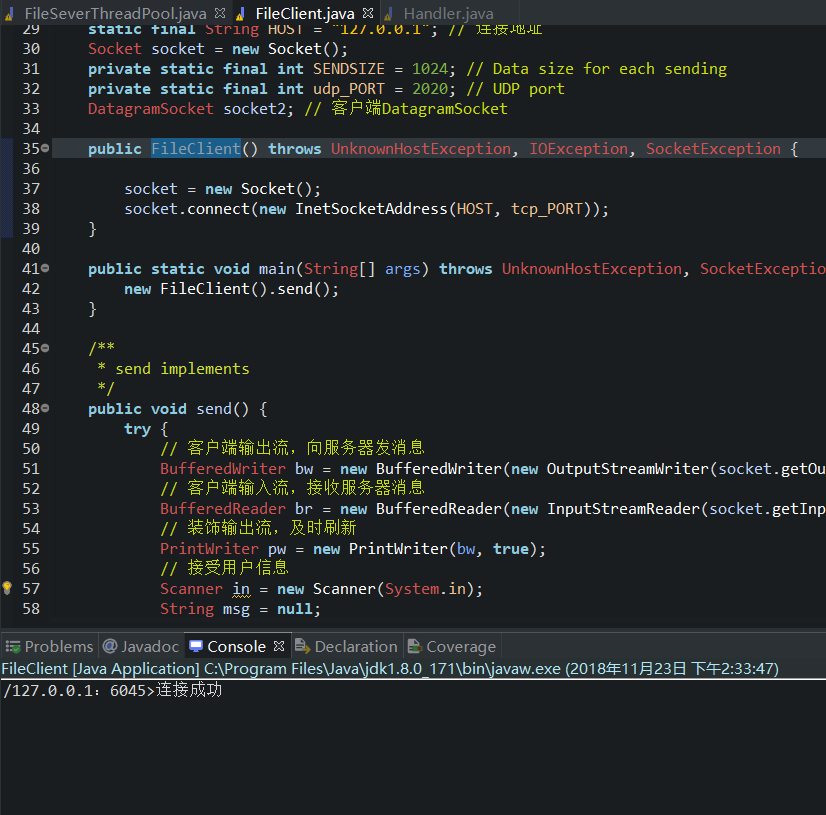
1. open FileClient

Multiple clients can be opened at the same time.

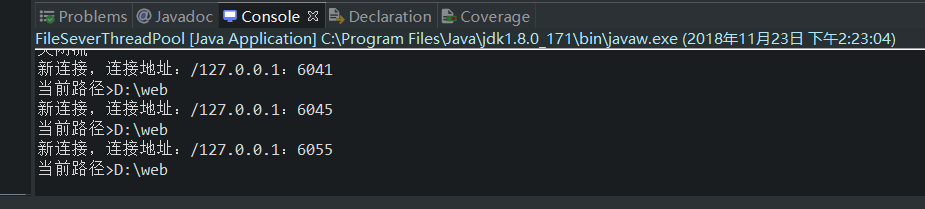
The screenshot is as follows:







Client screenshots are as follows:



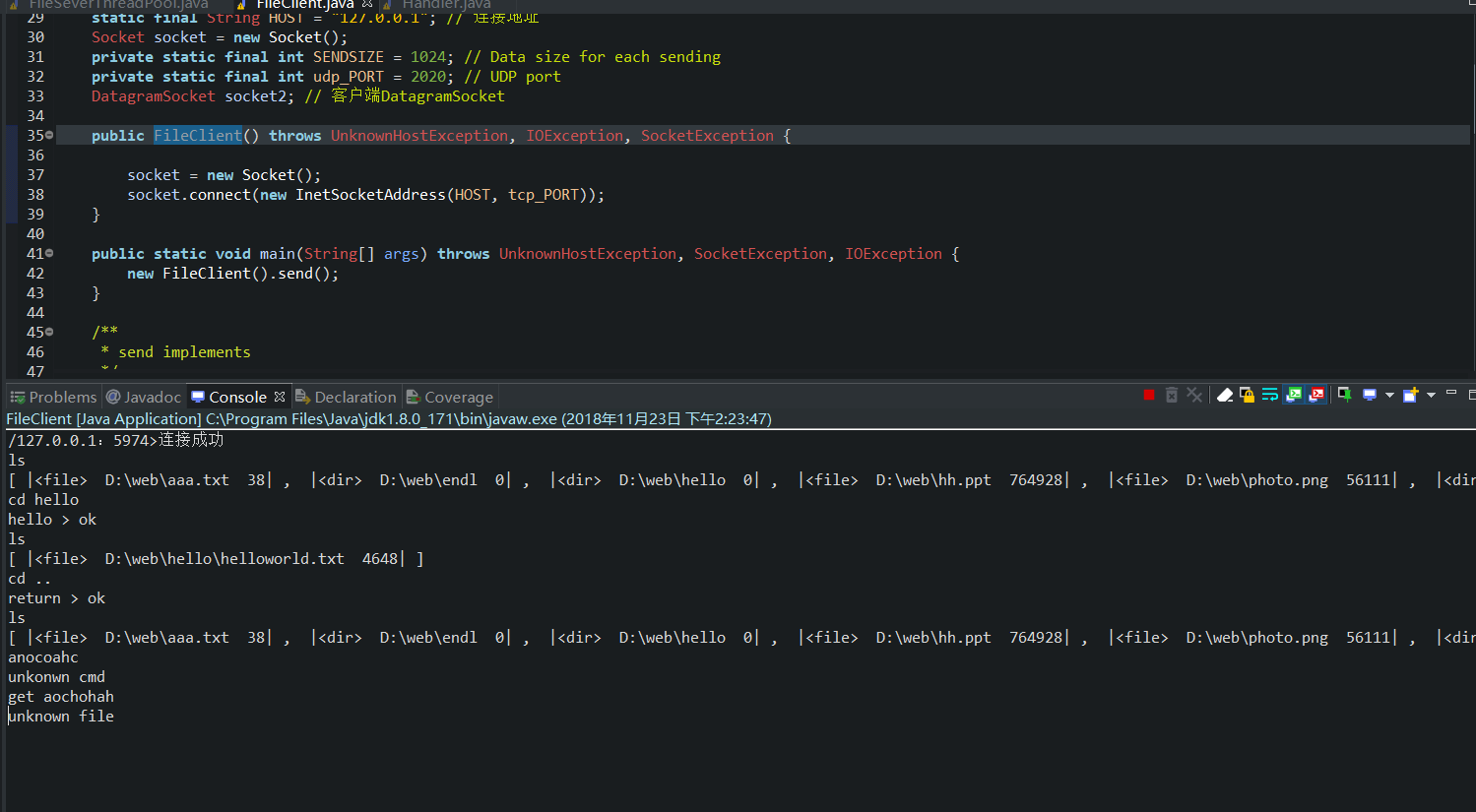
File directory access

1. Enter “ls” to get all files and folders in the default file path

2. Enter “cd Hello” into the Hello folder, and then enter “ls” to get the contents of the current folder.

3. Enter “cd..” Return to the previous directory

4. Enter the wrong file name and return "unknown file"

5. Enter the error command and return "unknown cmd"

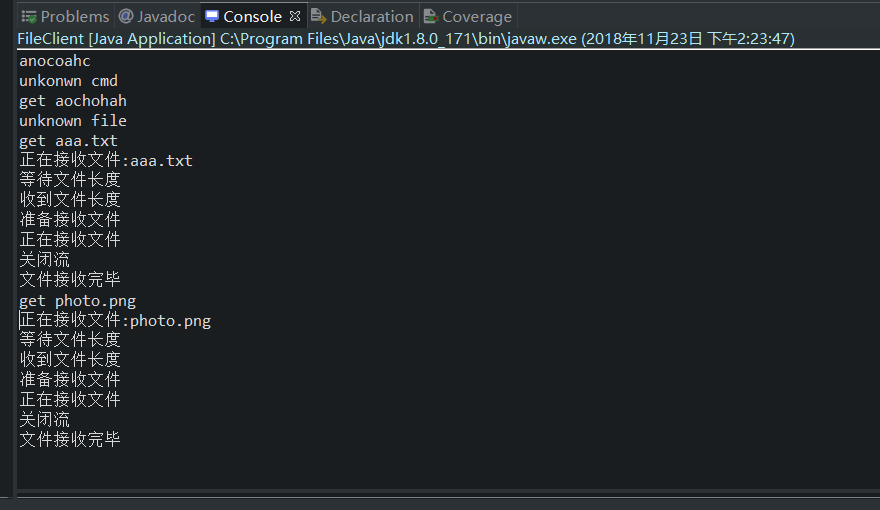
File transfer

1. Enter “get + filename. Suffix”

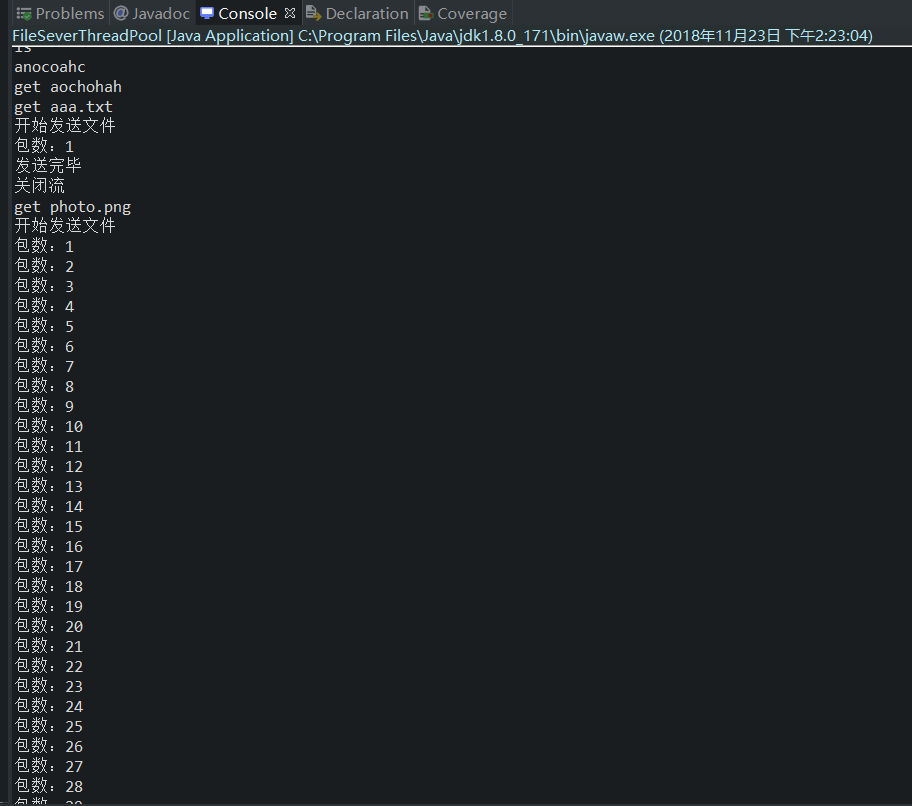
2. Waiting for file transfer

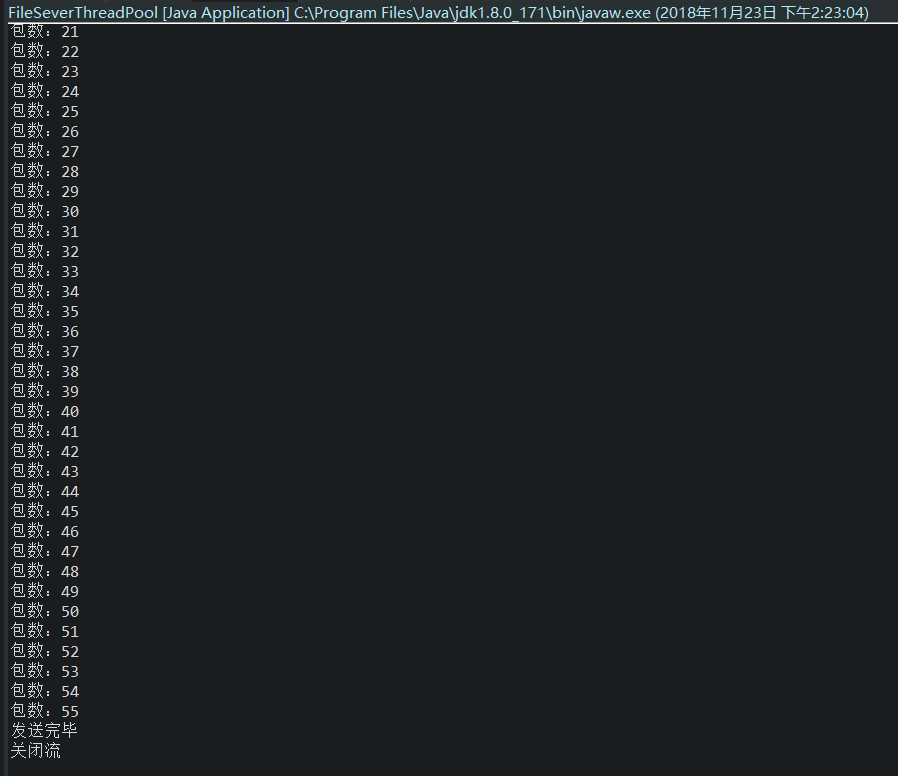
3. Successful file reception

Client screenshots are as follows:



Server-side screenshots are as follows:





Source code:

Class FileClient

package ex01;

import java.io.BufferedOutputStream;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.io.PrintWriter;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetSocketAddress;

import java.net.Socket;

import java.net.SocketAddress;

import java.net.SocketException;

import java.net.UnknownHostException;

import java.util.Scanner;

/\*\*

\* tcp , client side

\*

\* @author 史嘉辉

\*

\*/

public class FileClient {

    static final int tcp\_PORT = 2021; // 连接端口

    static final String HOST = "127.0.0.1"; // 连接地址

    Socket socket = new Socket();

    private static final int SENDSIZE = 1024; // Data size for each sending

    private static final int udp\_PORT = 2020; // UDP port

    DatagramSocket socket2; // 客户端DatagramSocket

    public FileClient() throws UnknownHostException, IOException, SocketException {

        socket = new Socket();

        socket.connect(new InetSocketAddress(HOST, tcp\_PORT));

    }

    public static void main(String[] args) throws UnknownHostException, SocketException, IOException {

        new FileClient().send();

    }

    /\*\*

     \* send implements

     \*/

    public void send() {

        try {

            // 客户端输出流，向服务器发消息

            BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

            // 客户端输入流，接收服务器消息

            BufferedReader br = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            // 装饰输出流，及时刷新

            PrintWriter pw = new PrintWriter(bw, true);

            // 接受用户信息

            Scanner in = new Scanner(System.in);

            String msg = null;

            // 输出服务器返回的消息

            System.out.println(br.readLine());

            Cd cmd = new Cd();

            while ((msg = in.nextLine()) != null) {

                pw.println(msg); // 发送给服务器端

                String ls = br.readLine();

                System.out.println(ls);// 输出接收到的信息

                if (ls.startsWith("正在接收文件:")) {

                    DatagramPacket dp = new DatagramPacket(new byte[SENDSIZE], SENDSIZE);

                    socket2 = new DatagramSocket(udp\_PORT);// Create UDP socket

                    byte[] info = new byte[SENDSIZE];

                    String filename = cmd.cdcmd(msg);

                    FileOutputStream fos = new FileOutputStream(new File(("D:\\temp\\") + filename));// 文件对象

                    System.out.println("等待文件长度");

                    long fileLength = Long.parseLong(br.readLine());// 接收文件的长度

                    System.out.println("收到文件长度");

                    int count = (int) (fileLength / SENDSIZE) + ((fileLength % SENDSIZE) == 0 ? 0 : 1);// 判断需要接收多少个包

                    System.out.println("准备接收文件");

                    while ((count--) > 0) {

                        socket2.receive(dp);

                        info = dp.getData();

                        fos.write(info, 0, dp.getLength());

                        // fos.flush();

                    }

                    System.out.println("正在接收文件");

                    socket2.close();

                    System.out.println("关闭流");

                    fos.close();

                    System.out.println("文件接收完毕");

                }

                if (msg.equals("bye")) {

                    break; // 退出

                }

            }

        } catch (IOException e) {

            e.printStackTrace();

        } finally {

            if (null != socket) {

                try {

                    socket.close(); // 断开连接

                } catch (IOException e) {

                    e.printStackTrace();

                }

            }

        }

    }

}

Class FileSeverThreadPool

package ex01;

import java.io.IOException;

import java.net.DatagramSocket;

import java.net.InetSocketAddress;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

public class FileSeverThreadPool {

    ServerSocket serverSocket;

    private final int PORT = 2021; // 端口

    ExecutorService executorService; // 线程池

    final int POOL\_SIZE = 4; // 单个处理器线程池工作线程数目

    public FileSeverThreadPool() throws IOException {

        serverSocket = new ServerSocket(PORT); // 创建服务器端套接字

        // 创建线程池

        // Runtime的availableProcessors()方法返回当前系统可用处理器的数目

        // 由JVM根据系统的情况来决定线程的数量

        executorService = Executors.newFixedThreadPool(Runtime.getRuntime().availableProcessors() \* POOL\_SIZE);

        System.out.println("服务器启动。");

    }

    public static void main(String[] args) throws IOException {

        new FileSeverThreadPool().servic(); // 启动服务

    }

    /\*\*

     \* service implements

     \*/

    public void servic() {

        Socket socket = null;

        while (true) {

            try {

                socket = serverSocket.accept(); // 等待用户连接

                executorService.execute(new Handler(socket)); // 把执行交给线程池来维护

            } catch (IOException e) {

                e.printStackTrace();

            }

        }

    }

}

Handler

package ex01;

import java.io.BufferedInputStream;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.io.PrintWriter;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.Inet4Address;

import java.net.InetAddress;

import java.net.InetSocketAddress;

import java.net.Socket;

import java.net.SocketAddress;

import java.net.SocketException;

import java.util.ArrayList;

import java.util.concurrent.TimeUnit;

public class Handler implements Runnable {

    private Socket socket;

    BufferedReader br;

    BufferedWriter bw;

    PrintWriter pw;

    private static final String HOST = "127.0.0.1"; // IP address

    final int UDP\_PORT = 2020; // UDP port

    private static final int SENDSIZE = 1024; // The number of bytes for each transport

    DatagramSocket socket2; // 客户端DatagramSocket

    SocketAddress socketAddres;

    public Handler(Socket socket) {

        this.socket = socket;

    }

    // 初始化输入输出流对象方法

    public void initStream() throws IOException {

        br = new BufferedReader(new InputStreamReader(socket.getInputStream()));

        bw = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

        pw = new PrintWriter(bw, true);

    }

    String paths = null;// 实时文件夹路径

    String path = "D:\\web";// 默认路径

    // 处理输入的命令

    Cd cmd = new Cd();

    // 读取文件夹

    public ArrayList<String> readfiles(String path) {

        ArrayList<String> files = new ArrayList<String>();

        File file = new File(path);

        File[] tempList = file.listFiles();

        if (tempList != null) {

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

                if (tempList[i].isDirectory()) {

                    // System.out.println("<dir> " + tempList[i] + " " + tempList[i].length());

                    files.add(("|<dir> " + tempList[i] + " " + tempList[i].length() + "|"));

                }

                if (tempList[i].isFile()) {

                    // System.out.println("<file> " + tempList[i] + " " + tempList[i].length());

                    files.add(("|<file> " + tempList[i] + " " + tempList[i].length() + "|"));

                }

            }

            return files;

        } else {

            return files;

        }

    }

    public void run() { // 执行的内容

        try {

            System.out.println("新连接，连接地址：" + socket.getInetAddress() + "：" + socket.getPort()); // 客户端信息

            initStream(); // 初始化输入输出流对象

            String info = null;

            System.out.println("当前路径>" + path);

            ArrayList<String> ok = readfiles(path);

            pw.println(socket.getInetAddress() + "：" + socket.getPort() + ">连接成功");

            int i = 1;

            paths = path;

            // 服务器端口

            int remotePort = socket.getPort();

            // gg

            while ((info = br.readLine()) != null) {

                System.out.println(info);

                ArrayList<String> ok1 = readfiles(paths);

                // 目录列表

                if (info.equals("ls")) {

                    pw.println(ok1);

                }

                // 输入cd命令

                else if (info.startsWith("cd")) {

                    // 获得命令中的目录

                    String cm = cmd.cdcmd(info);

                    if (cm.equals("null")) {

                        pw.println("unkonw cmd");

                    }

                    // 返回上一级

                    if (cm.equals("..")) {

                        // 进入前一个目录

                        ArrayList<String> ok3 = readfiles(path);

                        if (ok3.toString() == "[]") {

                            pw.println("unkonwn dir");

                        } else {

                            paths = path;

                            pw.println("return > ok");

                        }

                    }

                    // 进入指定文件夹

                    else {

                        paths = path + "\\" + cm;

                        ArrayList<String> ok2 = readfiles(paths);

                        if (ok2.toString() == "[]") {

                            pw.println("no file");

                        } else {

                            pw.println(cm + " > ok");

                        }

                    }

                }

                // 输入get命令

                else if (info.startsWith("get")) {

                    String cm2;

                    cm2 = cmd.cdcmd(info);

                    if (cm2.equals("null"))

                        pw.println("unkonwn cmd");

                    else {

                        String p = paths + "\\" + cm2;// 需要传输的文件lu'jing

                        File file = new File(p);// 创建文件对象

                        if (file.exists()) {

                            pw.println("正在接收文件:" + cm2);

                            // 发送文件

                            sendFile(file);

                        } else {

                            pw.println("unknown file");

                        }

                    }

                }

                // 退出

                else if (info.equals("bye")) {

                    break;

                }

                // 输入错误命令

                else {

                    pw.println("unkonwn cmd");

                }

            }

        } catch (IOException e) {

            e.printStackTrace();

        } catch (InterruptedException e) {

            // TODO Auto-generated catch block

            e.printStackTrace();

        } finally {

            if (null != socket) {

                try {

                    socket.close();

                } catch (IOException e) {

                    e.printStackTrace();

                }

            }

        }

    }

    /\*

     \* 发送文件 sendFile（）

     \*/

    private void sendFile(File file) throws SocketException, IOException, InterruptedException {

        socket2 = new DatagramSocket();

        socketAddres = new InetSocketAddress(HOST, UDP\_PORT);

        DatagramPacket dp;

        byte[] sendInfo = new byte[SENDSIZE];

        dp = new DatagramPacket(sendInfo, sendInfo.length, socketAddres);// 新建DatagramPacket对象

        BufferedInputStream bfdIS = new BufferedInputStream(new FileInputStream(file));// 文件流

        pw.println(file.length());// 发送数据长度

        byte[] sendDataByte = new byte[1024];// 创建数据包，存放拆分的文件

        int read = 0;// 判断数据包是否发送完毕

        System.out.println("开始发送文件");

        int i = 1;

        while (true) {

            if (bfdIS != null) {

                read = bfdIS.read(sendDataByte);

            }

            if (read == -1) {

                break;

            }

            dp = new DatagramPacket(sendDataByte, sendDataByte.length,

                    (Inet4Address) Inet4Address.getByName("localhost"), UDP\_PORT);// 创建传输流

            socket2.send(dp);// 发送

            TimeUnit.MICROSECONDS.sleep(1000);// 限制传输速度

            System.out.println("包数：" + i);

            i++;

        }

        System.out.println("发送完毕");

        socket2.close();

        bfdIS.close();

        System.out.println("关闭流");

    }

}