随堂测试1 （编程题2）；

在程序中直接将读入的网络数据流转换为等长字节数组存在安全问题，因为在网络获取数据不流畅，数据流的传送会断断续续，不能保证一次能读取全部数据。特别是在读取大容量网络数据时问题更严重。假设有以下场景，应用程序servelet当前收到http（协议）请求流，请设计程序怎么保证当前程序接收到所有请求流数据？请写出主要代码和思路。

已知http请求流的获得指令：

ServletInputStream inStream = request.getInputStream(); //取HTTP请求流

int size = request.getContentLength(); //取HTTP请求流长度

参考思路：

在读取数据时检测实际读到的长度，如果没有读完已知长度的数据就应该再次读取，以此循环检测，直到实际读取的长度累加与已知的长度相等。

ServletInputStream inStream = request.getInputStream(); //取HTTP请求流

int size = request.getContentLength(); //取HTTP请求流长度

byte[] buffer = new byte[size]; //用于缓存每次读取的数据

byte[] in\_b = new byte[size]; //用于存放结果的数组

int count = 0;

int rbyte = 0;

while (count < size) { //循环读取

　rbyte = inStream.read(buffer); //每次实际读取长度存于rbyte中

　for(int i=0;i;)

　　in\_b[count + i] = buffer[i];

}

count += rbyte;

}

随堂测试2 （编程题2）

客户端编程：创建一个简单的HTTP客户端程序：EasyHTTPClient 类，它访问www.szu.edu.cn/xxgk/xxjj.htm, 把得到的HTTP响应结果保存到本地文件系统的一个文件中。

参考代码：

import java.net.\*;

import java.io.\*;

public class EasyHTTPClient{

String host="www.szu.edu.cn/xxgk/xxjj.htm";

int port = 80;

Socket socket;

public void createSocket() throws Exception {

socket = new Socket(host, 80);

}

public void communicate() throws Exception {

StringBuffer sb = new StringBuffer("GET " + "/index.jsp" + " HTTP/1.1\r\n");

sb.append("Host: "+host+"\r\n");

sb.append("Accept: \*/\*\r\n");

sb.append("Accept-Language: zh-cn\r\n");

sb.append("Accept-Encoding: gzip, deflate\r\n");

sb.append("User-Agent: MyHTTPClient\r\n");

sb.append("Connection: Keep-Alive\r\n\r\n");

// 发出HTTP请求

OutputStream socketOut = socket.getOutputStream();

socketOut.write(sb.toString().getBytes());

socketOut.flush();

// 接收响应结果

InputStream socketIn = socket.getInputStream();

FileOutputStream fileOut = new FileOutputStream("response.data");

byte[] buff = new byte[1024];

int len = -1;

while ((len = socketIn.read(buff)) != -1) {

fileOut.write(buff, 0, len);

}

fileOut.close();

socket.close();

System.out.println("响应数据已经保存到response.data文件中");

}

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

EasyHTTPClient client = new EasyHTTPClient();

client.createSocket();

client.communicate();

}

}

UDP编程练习：

编写一个PictureServer服务器程序和PictureClient客户程序。PictureServer利用DatagramSocket发送客户端的一个图片文件。PictureClient利用DatagramSocket接收图片文件。（提示：服务器端发送结束后，会发送一个文本消息end,以通知客户端文件发送完毕）

参考代码：

**PictureClient:**

import java.net.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import javax.swing.\*;

public class

extends JFrame implements Runnable, ActionListener {

public static final int PORT=8899;

JButton b=new JButton("显示图片");

ImagePanel imagePanel;

public ImageClient() {

super("图片展示");

b.addActionListener(this);

Container container=getContentPane();

container.add(b,BorderLayout.NORTH);

imagePanel=new ImagePanel();

container.add(imagePanel,BorderLayout.CENTER);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(600,600);

setVisible(true);

}

public void actionPerformed(ActionEvent event){

byte b[]="please send picture".trim().getBytes();

try{

InetAddress address=InetAddress.getByName("localhost");

DatagramPacket data=new DatagramPacket(b,b.length,address,ImageServer.PORT);

DatagramSocket mailSend=new DatagramSocket();

mailSend.send(data);

System.out.println("客户端请求获取图片");

Thread thread=new Thread(this);

thread.start();

} catch(Exception e){

e.printStackTrace();

}

}

public void run() {

DatagramPacket pack=null;

DatagramSocket clientSocket=null;

byte b[]=new byte[8192];

ByteArrayOutputStream out=new ByteArrayOutputStream();

try {

pack=new DatagramPacket(b,b.length);

clientSocket=new DatagramSocket(PORT);

}catch(Exception e){

e.printStackTrace();

}

try {

while(true){

clientSocket.receive(pack);

String message=new String(pack.getData(),0,pack.getLength());

if(message.startsWith("end")) {

System.out.println("图片数据接收完毕");

break;

}

out.write(pack.getData(),0,pack.getLength());

System.out.println("正在接收图片数据");

}

byte imagebyte[]=out.toByteArray();

out.close();

Toolkit tool=getToolkit();

Image image=tool.createImage(imagebyte);

imagePanel.setImage(image);

imagePanel.repaint();

validate();

}catch(IOException e){

e.printStackTrace();

}finally{

try{ clientSocket.close(); }catch(Exception e){}

}

}

public static void main(String args[]) {

new ImageClient();

}

}

**pictureServer:**

import java.net.\*;

import java.io.\*;

public class PictureServer {

public static final int PORT=7000;

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

DatagramPacket pack=null;

DatagramSocket mainSocket=null;

byte b[]=new byte[8192];

InetAddress address=null;

pack=new DatagramPacket(b,b.length);

mainSocket=new DatagramSocket(PORT);

System.out.println("服务器启动");

while(true){

try {

mainSocket.receive(pack);

address=pack.getAddress();

System.out.println("收到来自客户端的请求:"+address);

} catch(IOException e){

System.out.println(e.getMessage());

}

if(address!=null) {

new ImageSender(address).start();

} else {

continue;

}

}

}

}

class ImageSender extends Thread{

InetAddress address;

DataOutputStream out=null;

DataInputStream in=null;

String s=null;

ImageSender(InetAddress address) {

this.address=address;

}

public void run() {

FileInputStream in;

byte b[]=new byte[8192];

DatagramSocket imageSocket=null;

try {

imageSocket=new DatagramSocket();

in=new FileInputStream("javathinker.png");

int n=-1;

while((n=in.read(b))!=-1) {

DatagramPacket data=new DatagramPacket(b,n,address,ImageClient.PORT);

imageSocket.send(data);

System.out.println("正在发送图片数据");

}

in.close();

byte end[]="end".getBytes();

DatagramPacket data=new DatagramPacket(end,end.length,address,ImageClient.PORT);

imageSocket.send(data);

System.out.println("图片数据发送完毕");

} catch(Exception e){

e.printStackTrace();

}finally{

try{ imageSocket.close();}catch(Exception e){}

}

}

}