//需求：使用UDP进行消息循环发送与接收

发送端：

package xhu.edu.udp;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

/\*问题与总结：

\* ---为了防止广播信息的骚扰，一般只能在同网段进行广播，不同网段的广播信息无法收到，会被过滤掉，

\* 系统一般会采用反向路径过滤，所以即是我们这里是程序测试，向不同网段发送了信息，但是由于我们的测试程序

\* 位于同一台计算机上，那么消息会兜兜转转回到此处，但是会被误认为是不同网段的地址，所以会被过滤掉，但是

\* 这里还有一个可能就是，我们的客户端的IP并没有真正的绑定对到相应的网段。

\*

\* ---InetAddress.getName(参数)，该参数既可以是IP，也能是域名、还能是主机名

\*

\* ---connect方法的意思是将socket与指定地址绑定，绑定成功后只能往该地址发送数据。

\* 并不是建立连接，也就是TCP是在网络中建立一条稳定的线路，然后通信，

\* 而UDP的连接只是指定一个地址，并不确定线路，数据报文在网络中临时选择相应的路由进行发送

\*

\* ---UDP广播传输，发送方使用使用广播地址，并指定端口号，而接收方则到指定端口接收，想想频道接收电台，

\* 所以广播地址的范围确定很重要，而UDP传输一般用于公共服务，或是将数据报进行加密处理

\*

\* ---DatagramSocket 套接字，不指定绑定的IP，默认为本机 IP，那么如果你的IP是动态的，它也是动态。

\*

\* ---UDP也能通过connect（）进行与指定的IP和端口好进行连接建立。

\* ---对象名.this

\* ---getLocalAddress是本地的地址，联网的话是一个分配的地址，不联网的话是本地localhost

\*

\* 在DatagramSocket中有个getInnetAddress()和getPort()方法，

\* 调用的时候发现是null和-1，问题也在这里。如果你调用了connet()方法，

\* 那么这两个方法的返回值就是connet中的地址和端口。你以前从来没有调用过connet()方法，

\* 所以才返回null和-1。

\*

\* ---native:本地，有点c的系统调用韵味，在Linux中自己写一个系统调用然后通过编译进内核，再写一个对应的函数，

\* 进行映射配置，就能被外部进行使用。这里就是将JNI（Java Native interface）生成ddl（动态连接库），而在类中

\* 使用Native只不过进行声明，将其封装入类中，方便类的使用

\*/

public class UdpServer {

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

// 发送方：创建一个数据报套接字，并绑定到本主机的1000端口号上，默认启动了广播（这里相当于是用应用层，等下要去调用packet）

DatagramSocket datagramSocket = new DatagramSocket();

// 创建数据报：并指定其要传送的到的主机以及端口号（这里相当于传输层，将要发送的数据以及接收方信息进行封装，当这个信息进行发送时，应用层的信息发送方信息也会被封装到头部信息）

byte[] data = new byte[1024];

DatagramPacket datagramPacket = new DatagramPacket(data, data.length, InetAddress.getByName("127.168.0.106"),

2000);

// 使用缓冲流从控制台输入数据

BufferedReader bReader = new BufferedReader(new InputStreamReader(System.in));

// 循环输入数据并发送

while (true) {

String string = bReader.readLine();

byte[] buff = new byte[1024];

buff = string.getBytes();

datagramPacket.setData(buff);

datagramSocket.send(datagramPacket);

if ("--bye--".equals(string)) {

break;

}

}

// 关闭资源

bReader.close();

datagramSocket.close();

}

}

接收端：

**package** xhu.edu.udp;

**import** java.io.BufferedWriter;

**import** java.io.IOException;

**import** java.io.OutputStreamWriter;

**import** java.net.DatagramPacket;

**import** java.net.DatagramSocket;

**import** java.net.InetAddress;

**public** **class** UdpClient {

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

// 接收方：建立数据报套接字，监听2000端口，等待消息

DatagramSocket datagramSocket = **new** DatagramSocket(2000, InetAddress.*getByName*("127.168.0.106"));

// 创建数据报接收数据

**byte**[] data = **new** **byte**[1024];

DatagramPacket datagramPacket = **new** DatagramPacket(data, data.length);

// 使用缓冲流从控制台输出数据

BufferedWriter bWriter = **new** BufferedWriter(**new** OutputStreamWriter(System.***out***, "Unicode"));

// 循环接收数据并输出

**while** (**true**) {

datagramSocket.receive(datagramPacket);

// 获取新发送的准确的长度的字符串，因为getData中的数据并不会自动刷新。getLength()能获取最近一次接收到的实际接收到的字节长度。

// 这也证明了发送端的数据报是会刷新的

String string = **new** String(datagramPacket.getData()).substring(0, datagramPacket.getLength());

bWriter.write(string);

bWriter.newLine();

bWriter.flush();

**if** ("--bye--".equals(string)) {

**break**;

}

}

// 关闭资源

bWriter.close();

datagramSocket.close();

}

}

运行截图：



