云南大学数学与统计学院  
《计算机网络实验》上机实践报告

|  |  |  |
| --- | --- | --- |
| 课程名称：计算机网络实验 | 年级：2015级 | 上机实践成绩： |
| 指导教师：陆正福 | 姓名：刘鹏 | 专业：信息与计算科学 |
| 上机实践名称：基于TCP协议与Socket接口的网络通信编程实验 | 学号：20151910042 | 上机实践日期：2018-10-25 |
| 上机实践编号：No.03 | 组号： |  |

# 实验目的

1. 熟悉基于TCP协议与Socket接口的网络通信编程实验
2. 熟悉教材第二章的基本概念

# 实验内容

1. 掌握基于TCP协议与Socket接口的网络通信编程的流程；
2. 使用Java实现基于TCP协议与Socket接口的PC网络通信编程；
3. 使用Java和Android实现基于TCP协议与Socket接口的移动网络通信编程（**选做**）；
4. 使用Python实现基于TCP协议与Socket接口的网络通信编程（**选做**）；

**说明**：3和4至少选做1项；调试所用实例可以源自书本和网络，但是应有属于自己的修改，不可以照搬照抄。

# 实验平台

Windows 10 Pro 1803；

MacOS Mojave 10.14；

Cygwin GCC编译器；

Android Studio 3.2.1 for MacOS；

Android Studio 3.2.1 for Windows；

# 程序代码

## 基于TCP协议与Socket接口的网络编程流程（叙述内容）

1. **配置服务器** 网络编程中比较经典的场景是Client-Server模式，P2P模式可以视为Client-Server模式的一种拓展。把服务器的IP确定下来，保持平稳运行、网络畅通、无故障。
2. **配置服务器端代码** 服务端的负责通讯的代码比较简单。TCP通信时，需要创建一个Socket，然后时刻监听。具体步骤如下：



1. **编写客户端程序** 客户端的程序需要精心编写，而且要在获知服务器地址（或者有解析的域名）的情况下进行。由于客户端需要设计好用的UI，所以在Java中牵扯到了创建进程对象，并调用其start方法。具体实现细节可以参看具体的代码。

这里的客户端使用Android模拟了一个手机App，仅仅是echo一个字符串。除此之外没有运算。



1. 测试与客户端的通信是否正常。

## 使用Java实现基于TCP协议与Socket接口的网络编程（给出实例）

主要把Java给出的Socket用好就可以了。详见1.1。

## 使用Java实现基于TCP协议与Socket接口的网络编程

本选做题选择Java，因为Java比较全面，各种库比较完善，而且在应用中，Java也是核心语言，相比Python就比较弱势。本例主要参考https://www.jianshu.com/p/fb4dfab4eec1的相关指导与说明。应用层的编程，不需要考虑如何建立路由表、防止比特错误以及纠错，只需要考虑建立连接以及通过连接收发消息。这类似于通过邮件系统收发邮件，需要做的仅仅是知道对方的邮箱地址、邮编。

本次使用的Socket是由TCP协议规定的、由Java语言实现的一种面向应用层的接口。

### 应用设计

服务器开启一个端口号为2222的TCP进程，一直监听来自外部的访问，然后经过简单的计算给出回答。而这里假设的外部客户端是一台手机上的App。本例没有采取任何复杂的计算，只是单纯地回复一个字符串。

### 实验过程分析

由于实验中需要设计一个App，所以比较理想的状态是采取开源的Android平台，用Java语言进行开发。这个App十分简易，但是需要连接网络，所以在App的设计中，需要赋予其连接网络的权限。Android应用需要用触摸交互式操作，所以设计对UI对象的操作，即把对基本UI组件的操作映射到相应的代码块上。

### Server端Java代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60 | **import** java**.**io**.**BufferedReader**;**  **import** java**.**io**.**IOException**;**  **import** java**.**io**.**InputStream**;**  **import** java**.**io**.**InputStreamReader**;**  **import** java**.**io**.**OutputStream**;**  **import** java**.**net**.**ServerSocket**;**  **import** java**.**net**.**Socket**;**  public class Server **{**  public static void main**(**String**[]** args**)** **{**  **try** **{**  // 为了看流程，我就把所有的代码都放在main函数里了,  // 也没有捕捉异常，直接抛出去了。实际开发中不可取。  // 1.新建ServerSocket对象，创建指定端口的连接  ServerSocket serverSocket **=** **new** ServerSocket**(**2222**);**  System**.**out**.**println**(**"服务端监听开始了~~~~"**);**  // 2.进行监听  Socket socket **=** serverSocket**.**accept**();**  // 3.拿到输入流（客户端发送的信息就在这里）  InputStream is **=** socket**.**getInputStream**();**  // 4.解析数据  InputStreamReader reader **=** **new** InputStreamReader**(**is**);**  BufferedReader bufReader **=** **new** BufferedReader**(**reader**);**  String s **=** **null;**  StringBuffer sb **=** **new** StringBuffer**();**  **while** **((**s **=** bufReader**.**readLine**())** **!=** **null)** **{**  sb**.**append**(**s**);**  **}**  System**.**out**.**println**(**"服务器：" **+** sb**.**toString**());**  // 关闭输入流  socket**.**shutdownInput**();**  OutputStream os **=** socket**.**getOutputStream**();**  os**.**write**((**"我是服务端,客户端发给我的数据就是：" **+** sb**.**toString**()).**getBytes**());**  os**.**flush**();**  // 关闭输出流  socket**.**shutdownOutput**();**  os**.**close**();**  // 关闭IO资源  bufReader**.**close**();**  reader**.**close**();**  is**.**close**();**  socket**.**close**();**// 关闭socket  serverSocket**.**close**();**// 关闭ServerSocket  **}** **catch** **(**IOException e**)** **{**  e**.**printStackTrace**();**  **}** **catch** **(**Exception e**)** **{**  e**.**printStackTrace**();**  **}**  **}**  **}** |

在后来的Windows平台的调试过程中，我发现由于JDF for Windows是只识别GBK编码，直接从Mac OS拷贝过来的Java源代码是UTF-8编码，是无法编译成字节码的。然而，用Visual Studio Code将字符集更改为GB2312之后，虽然可以编译成功，但是客户端返回的中文字符又变成了乱码。可见，服务端的输出在Windows平台下是GBK输出。

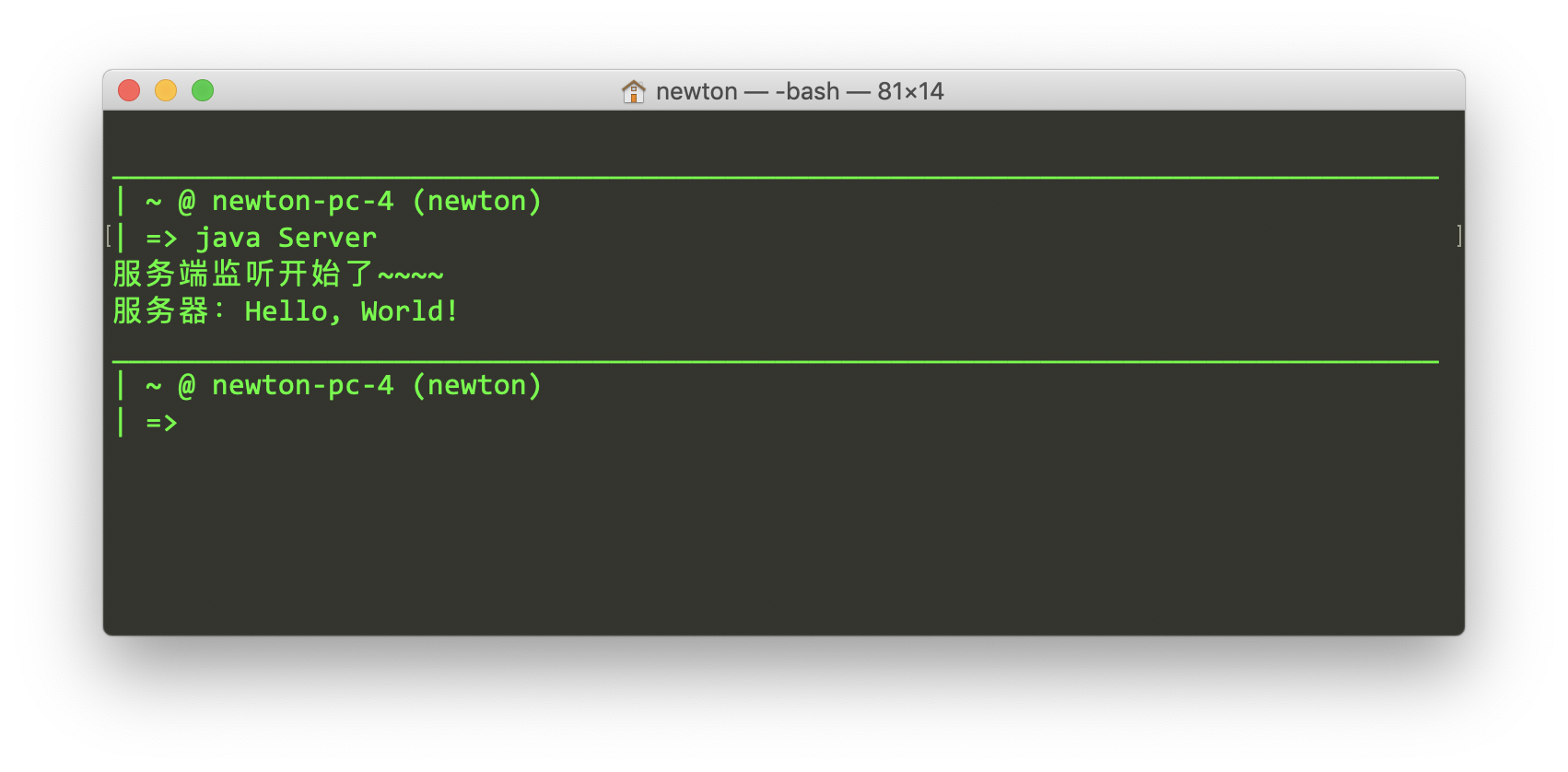
解决这个问题的方法有很多，下面试列举两个。（1）所有的代码、注释都用英文，这样可以绝对保证不会出问题；（2）统一平台的编码，更改Android Studio的字符集比较困难，最好是令JDK的字符集更改为默认UTF-8，这可以通过添加环境变量：变量名JAVA\_TOOL\_OPTIONS，变量值-Dfile.encoding=UTF-8，改完之后就不存在这种问题了。

最后，由于这个工程的系统代码太多，有很多XML文件，所以这里仅仅贴出核心代码，具体工程文件参见GitHub仓库。

### Android App端Java代码

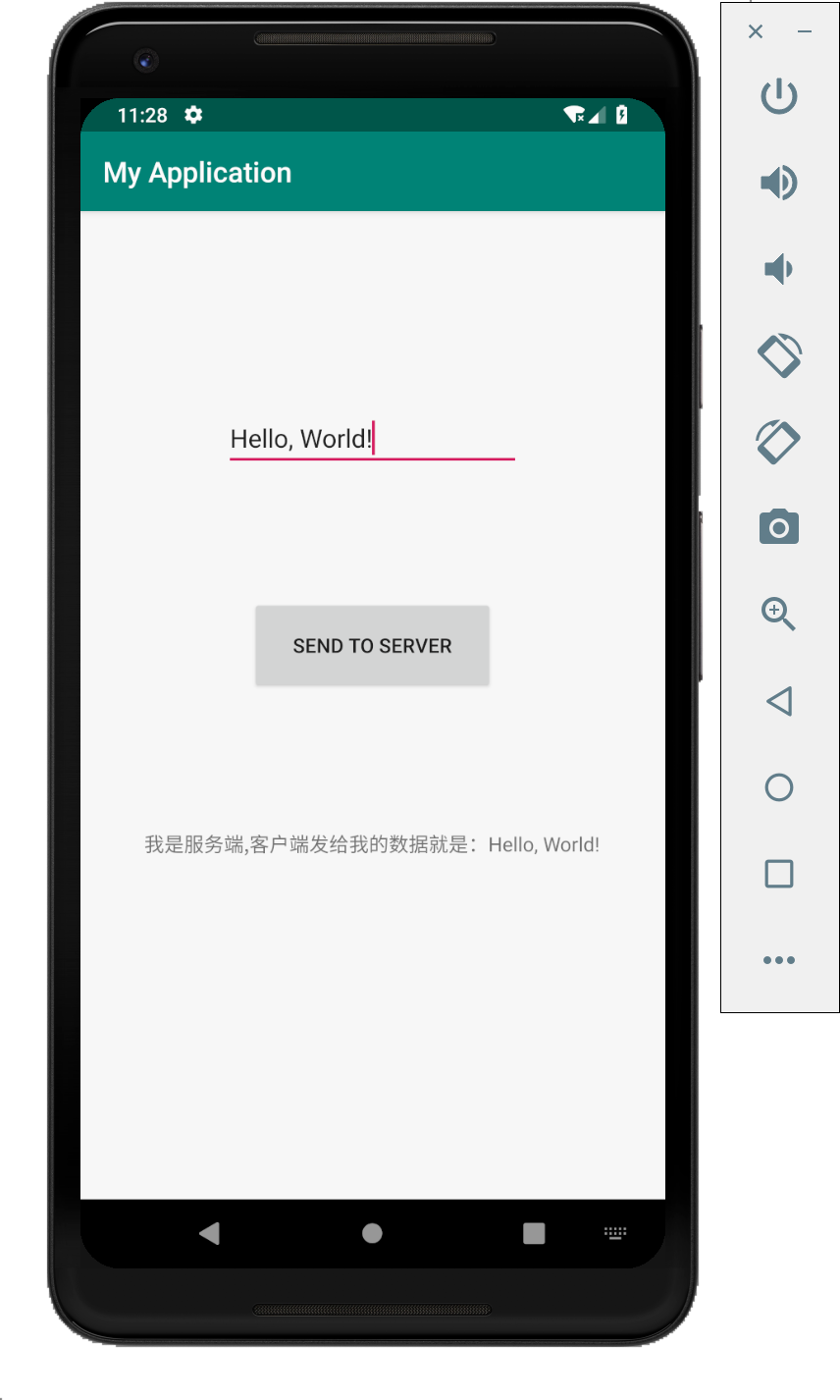
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90 | package com**.**example**.**newton**.**myapplication**;**  **import** android**.**support**.**v7**.**app**.**AppCompatActivity**;**  **import** android**.**os**.**Bundle**;**  **import** java**.**io**.**BufferedReader**;**  **import** java**.**io**.**IOException**;**  **import** java**.**io**.**InputStream**;**  **import** java**.**io**.**InputStreamReader**;**  **import** java**.**io**.**OutputStream**;**  **import** java**.**net**.**Socket**;**  **import** android**.**view**.**View**;**  **import** android**.**widget**.**Button**;**  **import** android**.**widget**.**EditText**;**  **import** android**.**widget**.**TextView**;**  **import** java**.**net**.**UnknownHostException**;**  public class MainActivity **extends** AppCompatActivity **{**  EditText et**;**  TextView tv**;**  Button button**;**  @Override  protected void onCreate**(**Bundle savedInstanceState**)** **{**  **super.**onCreate**(**savedInstanceState**);**  setContentView**(**R**.**layout**.**activity\_main**);**  et **=** findViewById**(**R**.**id**.**et**);**  tv **=** findViewById**(**R**.**id**.**tv**);**  button **=** findViewById**(**R**.**id**.**button**);**  **}**  public void onClick**(**View view**){**  **new** Thread**(){**  @Override  public void run**()** **{**  **super.**run**();**  **try** **{**  // 1.创建监听指定服务器地址以及指定服务器监听的端口号  Socket socket **=** **new** Socket**(**"192.168.1.79"**,** 2222**);**  // 2.拿到客户端的socket对象的输出流发送给服务器数据  OutputStream os **=** socket**.**getOutputStream**();** // 客户机的output  // 写入要发送给服务器的数据  os**.**write**(**et**.**getText**().**toString**().**getBytes**());**  // System.out.println(os.toString());  os**.**flush**();**  socket**.**shutdownOutput**();** // OK, 不再需要写字了，准备发送吧！  // 拿到socket的输入流，这里存储的是服务器返回的数据  InputStream is **=** socket**.**getInputStream**();**  // 解析服务器返回的数据  InputStreamReader reader **=** **new** InputStreamReader**(**is**);**  BufferedReader bufReader **=** **new** BufferedReader**(**reader**);**  String s **=** **null;**  final StringBuffer sb **=** **new** StringBuffer**();**  **while((**s **=** bufReader**.**readLine**())** **!=** **null){**  sb**.**append**(**s**);**  **}**  runOnUiThread**(new** Runnable**()** **{**  @Override  public void run**()** **{**  tv**.**setText**(**sb**.**toString**());**  **}**  **});**  // 3、关闭IO资源  bufReader**.**close**();**  reader**.**close**();**  is**.**close**();**  os**.**close**();**  socket**.**close**();**  **}** **catch** **(**UnknownHostException e**)** **{**  e**.**printStackTrace**();**  **}** **catch** **(**IOException e**)** **{**  e**.**printStackTrace**();**  **}**  **}**  **}.**start**();**  **}**  **}** |

### Server端运行结果



本次实验没有采用多机环境，一是考虑到多机环境配置网络可能比较繁琐，再者是单机的不同进程之间也可以通过Socket进行网络通信，所以在一台机器上进行了实验。

### Android客户端运行结果



# 实验体会

Android编程比较面向应用，对于实现某个特定的功能考虑较多，最初对于安卓的开发环境非常不熟悉，后来逐步完成代码与UI设计，中间遇到了一些困难，但是还可以通过考虑系统的可能设计加上搜索由此而得到的问题逐一解决。

# 参考文献

[1] https://www.jianshu.com/p/fb4dfab4eec1

[2] java NIO: https://www.jianshu.com/p/093b7c408dba

[3] java NIO: https://docs.oracle.com/javase/7/docs/api/java/nio/package-summary.html

[4] java NET: https://docs.oracle.com/javase/7/docs/api/java/net/package-summary.html