计算机网络编程
实验报告

班级：07111707

组长：1120171189 崔程远

成员：1120172149 吴沁璇

1120172153 张澈

1120172163 王晓媛

1120172736 张鉴昊

1120172765 曾煜瑾

1120173326 曾紫飞

北京理工大学

计算机学院

2020 年 5 月

第六章 实验 2　TCP 协议服务器和客户

1. 实验目的

学习 TCP 通信协议，掌握 TCP socket 通信方法。

2. 实验内容

客户发送命令行文本给服务器，服务器转换大写后返回给客户并显示。

配置文件关键要点：

无，对方的 IP 地址、端口以及发送串以命令行参数的形式提供程序运行

3. 实验原理

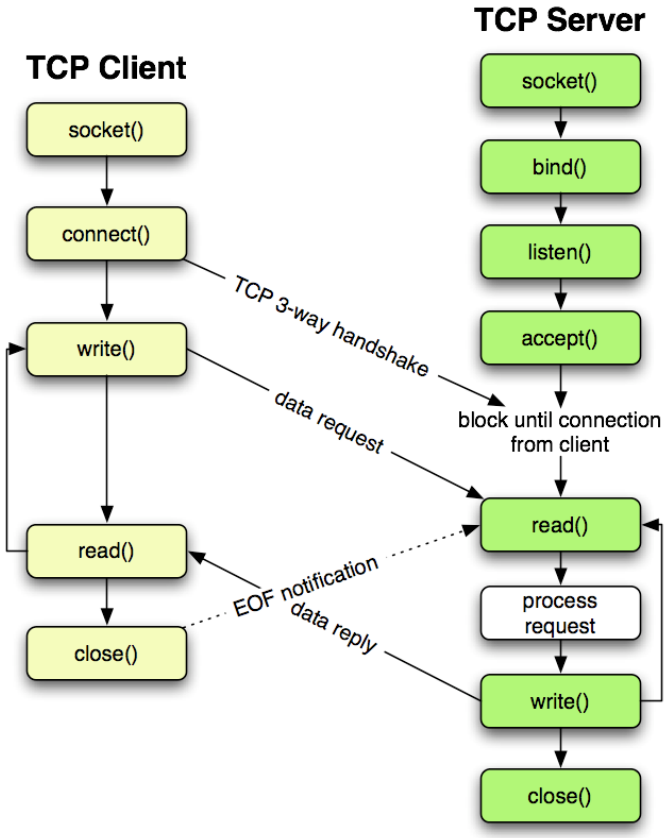
socket 起源于 Unix，而 Unix/Linux 基本哲学之一就是“一切皆文件”，都可以用“打开 open –> 读写 write/read –> 关闭 close”模式来操作。我的理解就是 Socket 就是该模式的一个实现，socket 即是一种特殊的文件，一些 socket 函数就是对其进行的操作（读/写 IO、打开、关闭）。

TCP 编程的服务器端一般步骤是：

1. 创建一个 socket，用函数 socket()；
2. 绑定 IP 地址、端口等信息到 socket 上，用函数 bind()；
3. 开启监听，用函数 listen()；
4. 接收客户端上来的连接，用函数 accept()；
5. 收发数据，用函数 send()和 recv()，或者 read()和 write()；
6. 关闭网络连接；
7. 关闭监听；

TCP 编程的客户端一般步骤是：

1. 创建一个 socket，用函数 socket()；
2. 绑定 IP 地址、端口等信息到 socket 上，用函数 bind();* 可选
3. 设置要连接的对方的 IP 地址和端口等属性；
4. 连接服务器，用函数 connect()；
5. 收发数据，用函数 send()和 recv()，或者 read()和 write()；
6. 关闭网络连接；



图源网络

4. 实验环境

语言	集成开发环境	编译器
C++	Visual Studio 2017	gcc version 4.8.1
Java	Eclipse 2019	java version "1.8.0_65"
Python	Pycharm 2017	Python 3.7.0

5. 实验步骤

各个语言的实现都分为服务端和客户端。

• C 语言版本

服务端首先判断接收参数数量是否符合要求，之后创建 Socket 并绑定到本地端口；通过 listen()阻塞等待连接；通过 accept()阻塞等待；通过 recv()接收信息；处理后通过 send()发送；关闭 Socket。

客户端首先判断接收参数数量是否符合要求，之后创建 Socket；通过 connect()连接服务端；send()发送字符串;recv()接收处理后的字符串；关闭 Socket。

• Java 语言版本

服务端首先判断接收参数数量是否符合要求，之后创建 ServerSocket 对象完成绑定；通过 ServerSocket 对象的 accept()方法创建 Socket 对象；通过 Socket 对象获得输入输出流 InputStream OutputStream 对象；通过输入输出流对象获得待处理信息并将处理后信息发送；关闭输入输出流和 Socket。

客户端首先判断接收参数数量是否符合要求，之后创建 Socket 对象；通过 Socket 对象获得输入输出流 InputStream OutputStream 对象；通过输入输出流

对象发送待处理信息并获得处理后信息；关闭输入输出流和 Socket。

• Python 语言版本

服务端首先判断接收参数数量是否符合要求，之后创建 Socket 并绑定到本地端口；通过 listen()阻塞等待连接；通过 accept()阻塞等待；通过 recv()接收信息；处理后通过 send()发送；关闭 Socket。

客户端首先判断接收参数数量是否符合要求，之后创建 Socket；通过 connect()连接服务端；send()发送字符串;recv()接收处理后的字符串；关闭 Socket。

6. 运行结果

Send 函数中将 01100000000000111110000000000110 作为待发送的数据信息。

• C

服务端：输入开放的端口号

```
[chez@chez-laptop C]$ ./TCPserver 10010
Waiting for message
Message recived
Message:aslfjasf
Sending message
Message sent
```

客户端：依次输入目标 IP、目标端口、待处理字符串

```
[chez@chez-laptop C]$ ./TCPclient 127.0.0.1 10010 aslfjasf
Connection ready
Sending Message
Message sent
Waiting for Message
Message recived
toUpper:ASLFJASF
```

• Java

服务端：输入开放的端口号

```
[chez@chez-laptop Java]$ java TCPserver 10010
Waiting for connection
Message received
Message:asdfpwejdv
Sending message
Message sent
```

客户端：依次输入目标 IP、目标端口、待处理字符串

```
[chez@chez-laptop Java]$ java TCPclient 127.0.0.1 10010 asdfpwejdv
Sending message
Message sent
Waiting for message
Message received
toUpper:ASDFPWEJDV
```

• Python

服务端：输入开放的端口号

```
[chez@chez-laptop Python]$ python TCPserver.py 10010
Waiting for Connection
Waiting for Message
Message recived
Message:aljx,cnzwie
Sending message
Message sent
```

客户端：依次输入目标 IP、目标端口、待处理字符串

```
[chez@chez-laptop Python]$ python TCPclient.py 127.0.0.1 10010 aljx,cnzwie
Connecting...
Sending message
Message sent
Waiting for Message
Message recived
toUpper:ALJX,CNZWIE
```

7. 实验总结

实验内容相对简单，通过上一个实验后基本熟悉了 Socket 编程方法， TCP Socket 编程和 UDP 相比有所不同，但了解了各个语言实现 TCP 通信的基本函数和类后还是能很快完成功能。由于各种语言对 UDP socket 的实现机制不同，不同系统对 socket 的支持也不同（POSIX 标准和 Winsocket），因此需要查阅相关资料。

总的来说，这个实验使我加深了对 TCP 通信过程的理解，提高了编程能力。Socket 极大地方便了网络编程。