中南大学

《分布式系统与云计算》

课程实验报告



学生姓名 ： 项岩

学 号 ： 8208180127

实验名称 ： 数据包 socket 应用

指导教师 ： 余腊生

学 院 ： 计算机学院

专业班级 ： 大数据1801

# 数据包 socket 应用

## 一、实验目的

1.理解数据包 s o c k e t 的 应 用

2.实现数据包 s o c k e t 通信

3.了解python并 行 编 程 的 基 本 方 法

## 二、实验环境

1.独立计算机或计算机网络；

2.Windows 操作系统。

3.python工具包

## 三、实验原理

分布式计算的核心是进程通信。操作系统、网卡驱动程序等应用从不同抽象 层面提供了对进程通信的支持。

Socket是一种作为 IPC 提供对系统低层抽象的机制。尽管应用人员很少需要在该层编写代码，但理解 socket非常重要，因为：1，高层设施是构建于 socket之上的，即他们是利用 socket提供的操作来实现；2，对于以响应时间要求较高或运行于有限资源平台上的应用来说，socket可能是最适合的。

在 Internet 网络协议体系结构中，传输层上有 UDP 和 TCP 两种主要协议，UDP 允许在 传送层使用无连接通信传送，被传输报文称为数据包。

一个线程是比进程更小的执行粒度。 Java 虚拟机允许应用程序有多个执行线程同时运 行。有两种方法来创建一个新线程的执行。一个是声明一个类是一个线程的子类。

## 四、实验代码

### 服务器端

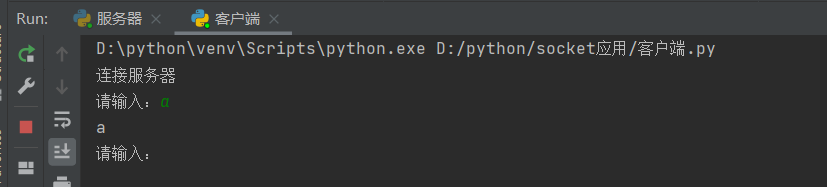
import socket  
from threading import Thread  
import socketserver  
  
g\_socket = None  
g\_conn\_pool = []  
  
def handle\_client():  
 while True:  
 client, addr = g\_socket.accept()  
 print(addr)  
 g\_conn\_pool.append(client)  
 t = Thread(target=message\_handle, args=(client,))  
 t.setDaemon(True)  
 t.start()  
  
def message\_handle(client):  
 while True:  
 data = client.recv(1024).decode('utf-8')  
 print(data)  
 if data == 'exit':  
 g\_conn\_pool.remove(client)  
 client.send(data.encode('utf-8'))  
 print('下线了')  
 break  
 client.send(data.encode('utf-8'))  
  
def main():  
 global g\_socket, g\_conn\_pool  
 g\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 g\_socket.bind(('127.0.0.1', 12306))  
 g\_socket.listen(5)  
 print("服务端已启动，等待客户端连接...")  
  
 t = Thread(target=handle\_client)  
 t.setDaemon(True)  
 t.start()  
 while True:  
 cmd = input("请输入操作：")  
 if cmd == '':  
 continue  
 if int(cmd) == 1:  
 print("--------------------------")  
 print("当前在线人数：", len(g\_conn\_pool))  
 if cmd == 'exit':  
 exit()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

### 客户端

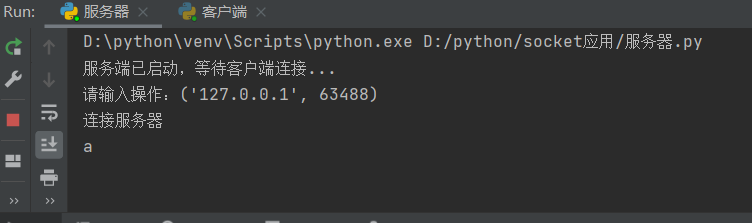
import socket  
import time  
import queue  
from threading import Thread  
  
q = queue.Queue()  
  
def recv\_message(s):  
 while True:  
 data = s.recv(1024).decode('utf-8').strip()  
 print(data)  
 if data == 'exit':  
 q.put(1)  
 break  
 send\_message(s)  
 s.close()  
  
def send\_message(s):  
 data = input('请输入：')  
 s.send(data.encode('utf-8'))  
  
def main():  
 s = socket.create\_connection(('127.0.0.1', 12306))  
 s.send('连接服务器'.encode('utf-8'))  
  
 Thread(target=recv\_message, args=(s,)).start()  
 time.sleep(1)  
 while q.empty():  
 pass  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

## 五、实验结果

### 服务端



### 客户端



## 六、思考题

1.如何避免数据包丢失而造成的无限等待问题?  
答:我认为可在发包时设定一个定时器，若发出去的包在一定时间内没有收到答应，则再发一次。为了避免接受者接到重复的包，可以给数据包加个序号，接受者收包时查看序即可。

2.如何实现全双工的数据包通信?  
答:利用端口套接字之间的通信功能。