山东大学网络空间安全学院

Python高级程序设计课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202100460065 | 姓名：李昕 | | 班级：21级密码2班 |
| 实验题目：实验六 socket编程1 | | | |
| 实验学时：2 | | 实验日期：2022/11/1 | |
| 实验目的：了解TCP协议原理、标准库socket 的用法、熟悉Socket 编程。 | | | |
| 硬件环境：  AMD Ryzen 7 5800H with Radeon Graphics     3.20 GHz  机带 RAM 16.0 GB (13.9 GB 可用) | | | |
| 软件环境：  系统：Windows11  编译器：IDLE | | | |
| 实验步骤与内容：  本次实验包括四个部分：  1、使用TCP协议实现智能聊天机器人。编写聊天程序的服务端代码和客户端代码。完成后，先启动服务端代码，然后启动客户端程序用输入问题，服务端可以返回相应的答案。要求服务端代码具有一定的智能，能够根据不完整的问题识别客户端真正要问的问题。（问题和答案是预先定义好的）要求支持多线程。  2、使用UDP协议打造在线时间服务器。服务端监听特定的端口，如果收到客户端发来的请求就把服务器上的当前时间发给客户端，而客户端收到时间之后立刻打印输出。  3、使用socketserver模块建立基于tcp协议通信的服务，收到客户端发来的英文字符串之后，将其变为大写发回客户端。  4、编写代码对网络上的ip地址进行端口扫描，收集“ip+开放端口”信息。进一步的，尝试了解和使用流行的网络扫描工具进行扫描，如zmap，nmap，并写出使用过程和扫描结果。  【智能聊天机器人】  首先，使用 socket() 函数来创建套接字socket，其具体语法格式为：socket.socket([family[, type[, proto]]])  其中的family: 套接字家族可以是 AF\_UNIX 或者 AF\_INET  type: 套接字类型可以根据是面向连接的还是非连接分为SOCK\_STREAM或SOCK\_DGRAM，由于本题是TCP协议，故选择SOCK\_STREAM。  protocol: 一般不填默认为0。  **代码实现：**  服务端：   1. import socket 2. from os.path import commonprefix 3. words={"what's your name?":'lixin', 4. 'what time is it?':'It is 8:00 am', 5. 'how old are you?':'20 years old hahaha', 6. 'where do you work?':'SDU!!', 7. 'how are you?':'Fine,thank you!', 8. 'bye':'Bye!'}*#创建智能识别的问题库* 9. HOST=""*#服务端主机IP地址和端口号，空字符串表示本机任何可用IP地址* 10. PORT=1234 11. s = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)***#创建socket对象*** 12. s.bind((HOST,PORT))*#绑定端口号* 13. s.listen(1)*#设置最大连接数，超过后排队* 14. print('Listening on port:',PORT) 15. conn,addr=s.accept() 16. print('Connected by:',addr) 17. while True: 18. data=conn.recv(1024).decode()*#解码* 19. if not data:break 20. print('Received message:',data) 21. *#尽量猜测对方要表达的意思* 22. m=0 23. key='' 24. for k in words.keys(): 25. data=' '.join(data.split()) *#删除多余的空白字符* 26. *#与某个键非常接近，就直接返回* 27. if len(commonprefix([k,data]))>len(k)\*0.7: 28. key=k;break 29. length=len(set(data.split())&set(k.split())) 30. if length>m: 31. m=length;key=k 32. conn.sendall(words.get(key,'Sorry.').encode()) 33. conn.close() 34. s.close()   但是显然该代码不能处理多线程问题，所以下面给出使用socketserver模块实现的多线程服务器，即在上述代码中进行模块化修改。  多线程服务端：   1. import socketserver 2. from os.path import commonprefix 3. from os.path import commonprefix 4. words={"what's your name?":'lixin', 5. 'what time is it?':'It is 8:00 am', 6. 'how old are you?':'20 years old hahaha', 7. 'where do you work?':'SDU!!', 8. 'how are you?':'Fine,thank you!', 9. 'bye':'Bye!'}*#创建智能识别的问题库* 10. class MyTCPhanler(socketserver.BaseRequestHandler): 11. def handle(self): 12. while True: 13. try: 14. data = self.request.recv(1024).decode() 15. print('Received message:',data) 16. *#尽量猜测对方要表达的意思* 17. m=0 18. key='' 19. for k in words.keys(): 20. data=' '.join(data.split()) *#删除多余的空白字符* 21. *#与某个键非常接近，就直接返回* 22. if len(commonprefix([k,data]))>len(k)\*0.7: 23. key=k;break 24. length=len(set(data.split())&set(k.split())) 25. if length>m: 26. m=length;key=k 27. self.request.send(words.get(key,'Sorry.').encode()) 28. except ConnectionResetError: 29. break 30. self.request.close() 31. if \_\_name\_\_ == '\_\_main\_\_': 32. server=socketserver.ThreadingTCPServer(('127.0.0.1',1234),MyTCPhanler) 33. server.serve\_forever()***# 链接循***   客户端（服务端是否多线程与客户端无关，故公用一套客户端）：   1. import socket 2. import sys 3. HOST='127.0.0.1' *#服务器IP地址和端口号* 4. PORT=1234 5. s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM) 6. try: 7. s.connect((HOST,PORT)) *#连接服务器* 8. except Exception as e: *#捕捉异常* 9. print('not found or not open, please try again!') 10. sys.exit() 11. while True: 12. c=input('Please input the content you want to send:') 13. s.sendall(c.encode())*#发送数据同时使用UTF-8编码* 14. data=s.recv(1024)*#接收数据，接收小于 1024 字节的数据* 15. data=data.decode() 16. print('Received：',data) 17. if c.lower()=='bye': 18. break 19. s.close()*#关闭连接*   运行结果：  客户端：    服务端：    【在线时间服务器】  1.由于 UDP并不会持续建立连接，因此适用于对效率要求相对较高而对准确性要求相对较低的场合，例如视频在线点播、网络语音通话等等。  2、UDP实现即服务端监听特定的端口，如果收到客户端发来的请求就把服务器上的当前时间发给客户端，而客户端收到时间之后立刻打印输出。  服务端：   1. import socket 2. from datetime import datetime 3. s=socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM) 4. HOST="" 5. PORT=50007 6. s.bind((HOST,PORT)) 7. while True: 8. data,addr=s.recvfrom(1024) 9. print("received message:{0} from {1}".format(data.decode(),addr[0])) 10. now =str(datetime.now())[:19] 11. s.sendto(now.encode(),addr) 12. s.close()   用户端：   1. import socket 2. import sys 3. import time 5. HOST="127.0.0.1" 6. PORT=50007 7. while True: 8. s = socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM)*#SOCK\_DGRAM表示UDP协议* 9. s.sendto('ask for time'.encode(),(HOST,PORT)) 10. data,addr=s.recvfrom(1024)*#接收数据，接收小于 1024 字节的数据* 11. print(data.decode()) 12. s.close() 13. time.sleep(1)     执行结果：    【字符串转大写】  要实现一项服务，必须派生一个request handler class请求处理类，并重写父类的handle()方法。handle方法就是用来专门是处理请求的。该模块是通过服务类和请求处理类组合来处理请求的。  代码实现：  服务端：   1. import socketserver 2. class MyTCPhanler(socketserver.BaseRequestHandler): 3. def handle(self): 4. while True: 5. try: 6. data = self.request.recv(1024) 7. if len(data) == 0: 8. break 9. print('-->收到客户端的消息: ', data) 10. self.request.send(data.upper()) 11. except ConnectionResetError: 12. break 13. self.request.close() 14. if \_\_name\_\_ == '\_\_main\_\_': 15. server=socketserver.ThreadingTCPServer(('127.0.0.1',8081),MyTCPhanler) 16. server.serve\_forever() *# 链接循环*   用户端：   1. from socket import \* 2. client = socket(AF\_INET, SOCK\_STREAM) 3. client.connect(('127.0.0.1', 8081)) 4. *# 通信循环* 5. while True: 6. sdata = input("please enter the data:") 7. if sdata == 'q': 8. break 9. client.send(sdata.encode('utf-8'))*#发送数据同时使用UTF-8编码* 10. data=client.recv(1024)*#接收数据，接收小于 1024 字节的数据* 11. print(data) 12. client.close()*#关闭连接*   执行结果：  [收集“ip+开放端口”信息]  题目要求编写代码对网络上的ip地址进行端口扫描，收集“ip+开放端口”信息。进一步尝试了解和使用流行的网络扫描工具进行扫描，如zmap，nmap，并写出使用过程和扫描结果。  我首先编写了扫描ip地址端口的程序，并尝试使用其扫描开放端口，调用socket模块，使用connect\_ex()方法，如果连接成功则返回0，使用print输出信息已开放。  但经过运行发现扫描速度较慢，开放端口较少，故经过网络检索了解并安装了扫描软件Nmap，并尝试使用默认的TCP扫描命令“namp ip地址”对指定ip地址101.76.250.1进行扫描，部分结果如下：    接着通过该结果缩小了python代码中的扫描范围，成功扫描得到部分结果。  代码实现：   1. import socket 2. import re 3. def scan(ip): 4. begin,end = (134,140) *#输入范围范围值* 5. for i in range(int(begin),int(end)+1): 6. s = socket.socket() 7. conn = s.connect\_ex((ip,i)) *#该方法如果链接成功会返回0* 8. if conn ==0: 9. print("主机:",ip,"端口:",i,"开放！！！！") 10. else: 11. print("主机:",ip,"端口:",i,"未开放") 12. s.close() 13. ip = input('请输入要扫描的ip：') 14. ip1= re.compile('((2(5[0-5]|[0-4]\d))|[0-1]?\d{1,2})(\.((2(5[0-5]|[0-4]\d))|[0-1]?\d{1,2})){3}') 15. if(ip1.match(ip)): *#判断格式是否正确* 16. scan(ip) 17. else: 18. print("格式错误！")   部分运行结果：    经对比，运行结果与网络流行扫描软件Nmap相同，代码结果正确。 | | | |
| 结论分析与体会：  通过本次练习，我熟悉了解了Python的socket网络编程，明白了如何让利用socketserver实现TCP和UDP协议的多线程，同时，我了解了如何进行端口扫描，下载安装并学习了网络扫描工具Nmap。 | | | |