山东大学网络空间安全学院

Python高级编程 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202100450069 | 姓名：孟庆丹 | | 班级：网安21.2 |
| 实验题目： | | | |
| 实验学时：2 | | 实验日期：2022.11.01 | |
| 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：  处理器 AMD Ryzen 5 5600H with Radeon Graphics 3.30 GHz  机带 RAM 16.0 GB (13.9 GB 可用) | | | |
| 软件环境：  Windows 11 家庭中文版 21H2  Python 3.10.7 | | | |
| 实验步骤与内容：   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。要求支持多线程。   要实现TCP服务端的多线程响应，可以利用socketserver.ThreadingTCPServer，对于每个新的请求都会启动一个单独的线程来处理这个请求。  要实现智能聊天机器人，需要对客户端发送的数据进行处理，并与预定义的问题进行比对，猜测提出的问题并进行回答，可以利用os.path.commomprefix()方法检测客户端发送问题与预定义问题的相似程度，在不存在高相似度的问题时回答最高重合度的问题的答案。  代码实现：  #服务端  import socketserver  from os.path import commonprefix  words = {'how are you?': 'Fine,thank you.',           'how old are you?': '10',           'what is your name?': 'xiaoluo',           'where do you work?': 'Lavatory',           'whats your favourite?': 'majsoul',           'areYouManAWoman?': 'man',           'whatDoYouLikeToEat?': 'cake',           'bye': 'Bye'}  class MyTCPHandler(socketserver.BaseRequestHandler):      def handle(self):          print("get connection from : ", self.client\_address)          while True:              # 接收客户端发来的数据              self.data = self.request.recv(1024).decode()              if not self.data:                  break              if self.data == 'exit':                  break              m = 0              key = ''              for k in words.keys():                  self.data = ' '.join(self.data.split())                  if len(commonprefix([k, self.data])) > len(k)\*0.7:                      key = k                      break  # 跳出for,表示找到对应的问题                  # 选择一个重合度较高的“键”                  length = len(set(self.data.split()) & set(k.split()))                  if length > m:                      m = length                      key = k              self.request.sendall(words.get(key, 'Sorry.').encode())  # 回答          self.request.close()  server = socketserver.ThreadingTCPServer(('', 5007), MyTCPHandler)  server.serve\_forever()  # 客户端  import socket  import sys  # 服务端主机IP地址和端口号  HOST = '60.205.212.22'  # 运行服务端的服务器IP  PORT = 5007  s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  try:      # 连接服务器      s.connect((HOST, PORT))  except Exception as e:      print('Server not found or not open')      sys.exit()  while True:      c = input('Input the content you want to send:')      # 发送数据      s.sendall(c.encode())      # 从服务端接收数据      data = s.recv(1024).decode()      print(data)  多线程运行结果：     1. 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。   代码实现：  # 服务端  import socket  import time  s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  s.bind(('', 5008))  # 绑定端口和端口号，空字符串表示本机任何可用IP地址  print("UDP开始")  while True:      data, addr = s.recvfrom(1024)      data = data.decode()      print('received message:{0} from PORT {1[1]} on {1[0]}'.format(data, addr))      Time = str(time.asctime(time.localtime(time.time())))      s.sendto(Time.encode(), addr)      if data.lower() == 'bye':          break  s.close()  # 客户端  import socket  import time  while True:      s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)      s.sendto('ask for time'.encode(), ("127.0.0.1", 5008))      data, addr = s.recvfrom(1024)      print(data.decode())      s.close()      time.sleep(1)  运行结果：  、   1. 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。   代码实现：  #服务端  import socketserver  class MyTCPHandler(socketserver.BaseRequestHandler):      def handle(self):          print("get connection from : ", self.client\_address)          while True:              # 接收客户端发来的数据              self.data = self.request.recv(1024).decode()              if self.data == 'exit':                  break              self.request.sendall(self.data.upper().encode())  # 回答          self.request.close()  server = socketserver.ThreadingTCPServer(('', 5009), MyTCPHandler)  server.serve\_forever()  运行结果：     1. 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。   利用socket模块对IP的TCP端口尝试进行连接，成功则为端口开放。因为判断端口开放需要较长时间，我们利用threading模块进行多线程扫描。  代码实现：  import socket  import threading  def portScanner(host, port):      socket.setdefaulttimeout(0.5)      try:          s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)          s.connect((host, port))          print("port", port, " open/tcp")          s.close()      except:          return  for p in range(0, 65536):      my\_thread = threading.Thread(          target=portScanner, args=("60.205.212.22", p))      my\_thread.start()  扫描结果：    利用nmap进行端口开放扫描： | | | |
| 结论分析与体会：  通过本节课的学习，我了解了TCP协议原理和标准库socket 的用法，在实验的过程中熟悉了Socket 编程，为之后的学习打下了良好的基础 | | | |