实验二 套接字编程

一、实验目的

通过本实验，熟练掌握Socket编程。

二、实验要求

根据教材第2章中的习题28，29或套接字编程作业完成相关作业并撰写实验报告。

三、实验内容

Web Server Lab

实验任务

在本次实验中，我们将学习 Python 中 TCP 连接的套接字编程基础：如何创建套接字，将其绑定到特定的地址和端口，并发送和接收 HTTP 数据包。我们还将学习一些 HTTP 头格式的基础知识。

我们将开发一个一次只处理一个 HTTP 请求的 Web 服务器。我们的 Web 服务器应：

1. 当一个客户（浏览器）联系时创建一个连接套接字
2. 从这个连接接受并解析 HTTP 请求，
3. 从服务器的文件系统中获取请求的文件
4. 创建包含请求文件和头部信息的 HTTP 响应消息
5. 经TCP连接将响应直接发送给客户端（浏览器）。如果服务器上没有请求的文件，服务器应向客户端发送一个 HTTP “404 Not Found” 错误消息。

实验步骤

完善配套网站中提供的 Web 服务器的框架代码，完成这部分代码。需要填写代码的位置用 **# Fill in start** 和 **# Fill in end** 标记。每个标记处可能需要填写一行或多行代码。

Skeleton Python Code for the Web Server

```
#import socket module
from socket import *
import sys # In order to terminate the program

serverSocket = socket(AF_INET, SOCK_STREAM)
#Prepare a sever socket
#Fill in start

#Fill in end
while True:
 #Establish the connection
```

```
 print('Ready to serve...')
 connectionSocket, addr = #Fill in start  #Fill in end
 try:
 message = #Fill in start   #Fill in end
 filename = message.split()[1]
 f = open(filename[1:])
 outputdata = #Fill in start  #Fill in end
 #Send one HTTP header line into socket
 #Fill in start

 #Fill in end
 #Send the content of the requested file to the client
 for i in range(0, len(outputdata)):
 connectionSocket.send(outputdata[i].encode())
 connectionSocket.send("\r\n".encode())

 connectionSocket.close()
 except IOError:
 #Send response message for file not found
 #Fill in start

 #Fill in end
 #Close client socket
 #Fill in start

 #Fill in end
serverSocket.close()
sys.exit()  #Terminate the program after sending the corresponding data
```

Running the Server

将一个 HTML 文件（例如，HelloWorld.html）放置在与服务器程序相同的目录中。运行服务器程序后，确定运行服务器的主机的 IP 地址（例如10.72.77.228:6789）。在另一台主机上打开浏览器并输入对应的 URL。例如：

http://10.68.248.155:3456/HelloWorld.html

`HelloWorld.html` 是我们放置在服务器目录中的文件的名称。请注意，在 URL 中冒号后的端口号需要替换为我们在服务器代码中使用的端口号。例如，在上面的例子中使用了端口号 6789。浏览器访问 URL 后，应显示 HelloWorld.html 的内容。如果省略了 :6789，浏览器会默认使用端口 80，只有当我们的服务器监听端口为 80 时，才能成功获取网页内容。接下来，尝试请求服务器中不存在的文件。此时，您应该收到一条 “404 Not Found” 错误消息。

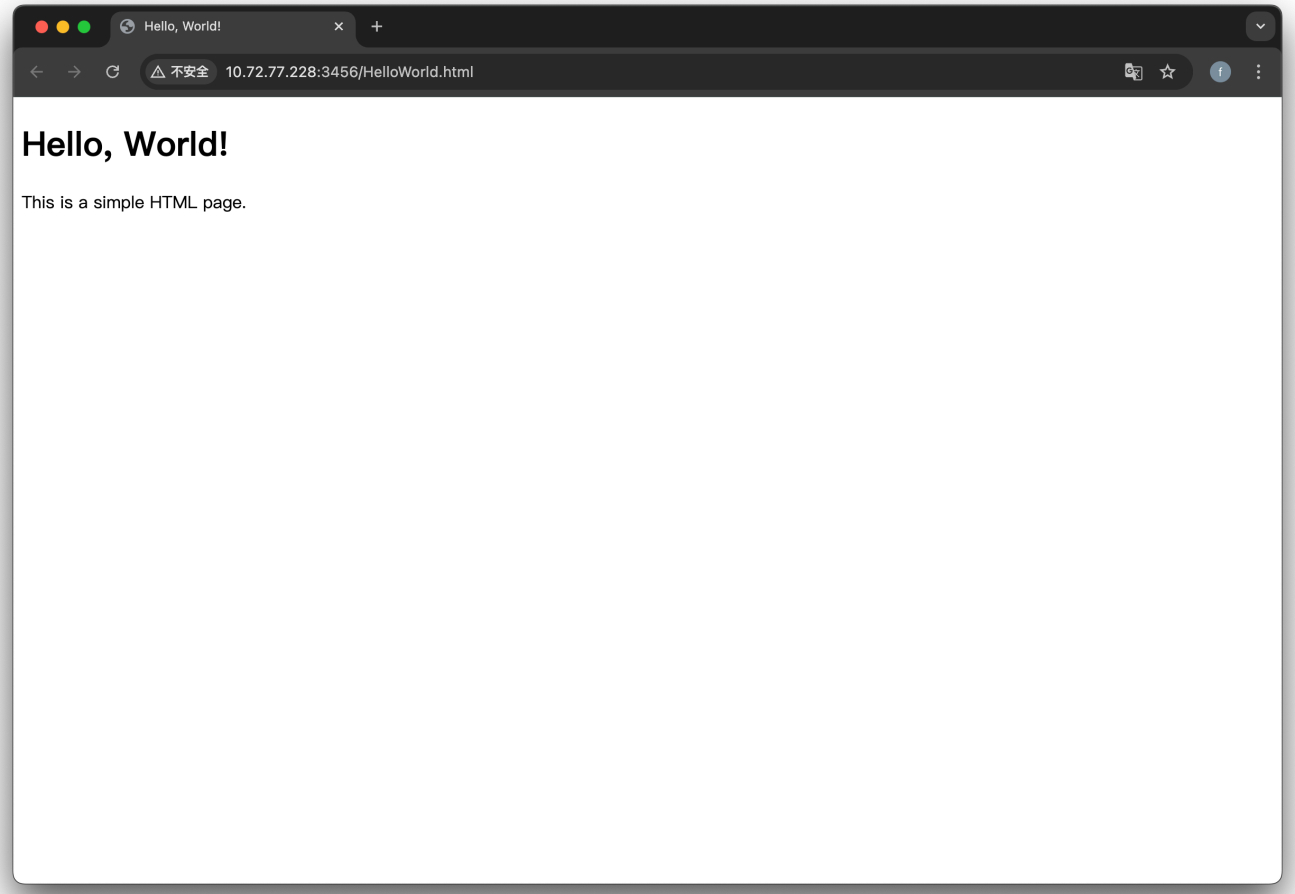
What to Hand in

1. 完整的服务器代码。

```
# import socket module
from socket import *
import sys  # In order to terminate the program

serverSocket = socket(AF_INET, SOCK_STREAM)
# Prepare a server socket
```

```python
# Fill in start
serverPort = 6789  # 可以选择端口号
serverSocket.bind(('', serverPort)) #serverSocket.bind((host, port)): host-空字符串表示绑定到所有可用的网络接口
serverSocket.listen(1)
# Fill in end

while True:
    # Establish the connection
    print('Ready to serve...')
    # Fill in start
    connectionSocket, addr = serverSocket.accept() #accpt会返回连接创建的套接字对象,客户端的地址信息
    # Fill in end

    try:
        # Fill in start
        message = connectionSocket.recv(1024).decode()
        # Fill in end
        filename = message.split()[1]
        f = open(filename[1:])  # 去掉文件路径前的 "/"
        # Fill in start
        outputdata = f.read()
        # Fill in end

        # Send one HTTP header line into socket
        # Fill in start
        connectionSocket.send("HTTP/1.1 200 OK\r\n\r\n".encode())
        # Fill in end

        # Send the content of the requested file to the client
        for i in range(0, len(outputdata)):
            connectionSocket.send(outputdata[i].encode())
        connectionSocket.send("\r\n".encode())

        connectionSocket.close()
    except IOError:
        # Send response message for file not found
        # Fill in start
        connectionSocket.send("HTTP/1.1 404 Not Found\r\n\r\n".encode())
        connectionSocket.send("<html><head></head><body><h1>404 Not Found</h1></body></html>\r\n".encode())
        # Fill in end

        # Close client socket
        # Fill in start
        connectionSocket.close()
        # Fill in end

serverSocket.close()
sys.exit()  # Terminate the program after sending the corresponding data
```

2. 客户端浏览器的截图，用于验证你确实从服务器接收到 HTML 文件的内容。

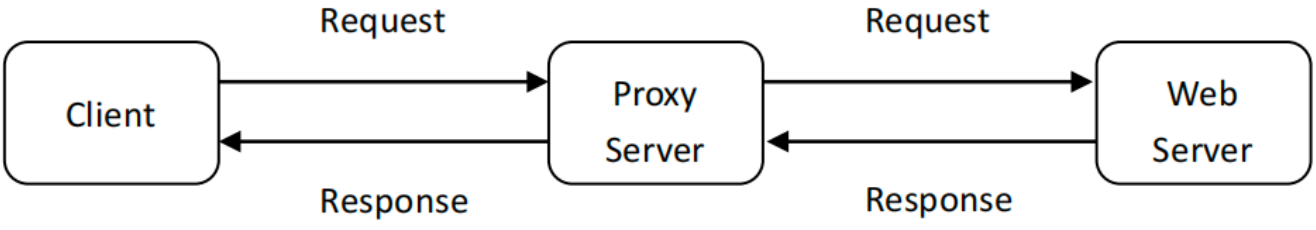


HTTP Web Proxy Server

在本次实验中将学习 Web 代理服务器的工作原理及其基本功能之一——缓存。

实验任务

开发一个小型 Web 代理服务器，该服务器能够缓存网页。这个代理服务器非常简单，只能处理简单的 GET 请求，但可以处理所有类型的对象，不仅包括 HTML 页面，还包括图片等。通常情况下，当客户端发出请求时，该请求会被发送到 Web 服务器。Web 服务器处理请求后，将响应消息发送回请求的客户端。为了提高性能，我们在客户端和 Web 服务器之间创建一个代理服务器。客户端发送的请求消息和 Web 服务器返回的响应消息都会经过代理服务器。换句话说，客户端通过代理服务器请求资源。当该代理服务器从一个浏览器接收到对某个对象的HTTP请求时，代理服务器会生成相同对象的HTTP请求转发给 Web 服务器，然后 Web 服务器生成包括该对象的HTTP响应并发送给代理服务器，代理服务器再将响应发送给客户端。



实验步骤

完成以下客户端框架代码。需要填写代码的部分已用 **#Fill in start** 和 **#Fill in end** 标注。每个标注部分可能需要一行或多行代码。

Skeleton Python Code for the Proxy Server

```
from socket import *
import sys

if len(sys.argv) <= 1:
  print('Usage : "python ProxyServer.py server_ip"\n[server_ip : It is the IP Address Of Proxy Server')
  sys.exit(2)
# Create a server socket, bind it to a port and start listening
tcpSerSock = socket(AF_INET, SOCK_STREAM)
# Fill in start.
# Fill in end.
while 1:
  # Strat receiving data from the client
  print('Ready to serve...')
  tcpCliSock, addr = tcpSerSock.accept()
  print('Received a connection from:', addr)
  message = # Fill in start. # Fill in end.
  print(message)
  # Extract the filename from the given message
  print(message.split()[1])
  filename = message.split()[1].partition("/")[2]
  print(filename)
  fileExist = "false"
  filetouse = "/" + filename
  print(filetouse)
  try:
    # Check wether the file exist in the cache
    f = open(filetouse[1:], "r")
    outputdata = f.readlines()
    fileExist = "true"
    # ProxyServer finds a cache hit and generates a response message
    tcpCliSock.send("HTTP/1.0 200 OK\r\n")
    tcpCliSock.send("Content-Type:text/html\r\n")
    # Fill in start.
    # Fill in end.
      print('Read from cache')
  # Error handling for file not found in cache
  except IOError:
    if fileExist == "false":
      # Create a socket on the proxyserver
      c = # Fill in start. # Fill in end.
      hostn = filename.replace("www.","",1)
      print(hostn)
      try:
        # Connect to the socket to port 80
```

```
        # Fill in start.

        # Fill in end.
        # Create a temporary file on this socket and ask port 80 for the file
requested by the client
        fileobj = c.makefile('r', 0)
        fileobj.write("GET "+"http://" + filename + " HTTP/1.0\n\n")
        # Read the response into buffer
        # Fill in start.

        # Fill in end.
        # Create a new file in the cache for the requested file.
        # Also send the response in the buffer to client socket and the corresponding
file in the cache
        tmpFile = open("./" + filename,"wb")
        # Fill in start.
        # Fill in end.
      except:
        print("Illegal request")
    else:
      # HTTP response message for file not found
      # Fill in start.

      # Fill in end.
  # Close the client and the server sockets
  tcpCliSock.close()
# Fill in start.

# Fill in end.
```

Running the Proxy Server

运行代理服务器程序后，可以使用命令提示符启动程序，然后通过浏览器请求网页。将请求指向代理服务器的IP地址和端口号。

例如，可以在浏览器地址栏输入以下内容：

http://10.63.43.209:8888/www.baidu.com

http://127.0.0.1:1234/www.baidu.com

http://localhost:8888/www.baidu.com

如果需要在不同的计算机上使用代理服务器（即浏览器和代理位于不同的计算机上），则需要使用运行代理服务器的计算机的IP地址。在这种情况下，运行代理时需要将“localhost”替换为运行代理服务器计算机的IP地址。此外，还需要注意端口号。在服务器代码中使用的端口号，例如这里的“8888”，也需要替换为您的服务器程序中设置的监听端口号。

Configuring your Browser

直接在网页浏览器中配置代理。具体操作取决于使用的浏览器：

- **在 Internet Explorer 中**，可以通过以下步骤设置代理：

- 转到 **工具 > Internet 选项 > 连接选项卡 > 局域网设置**（LAN Settings）。

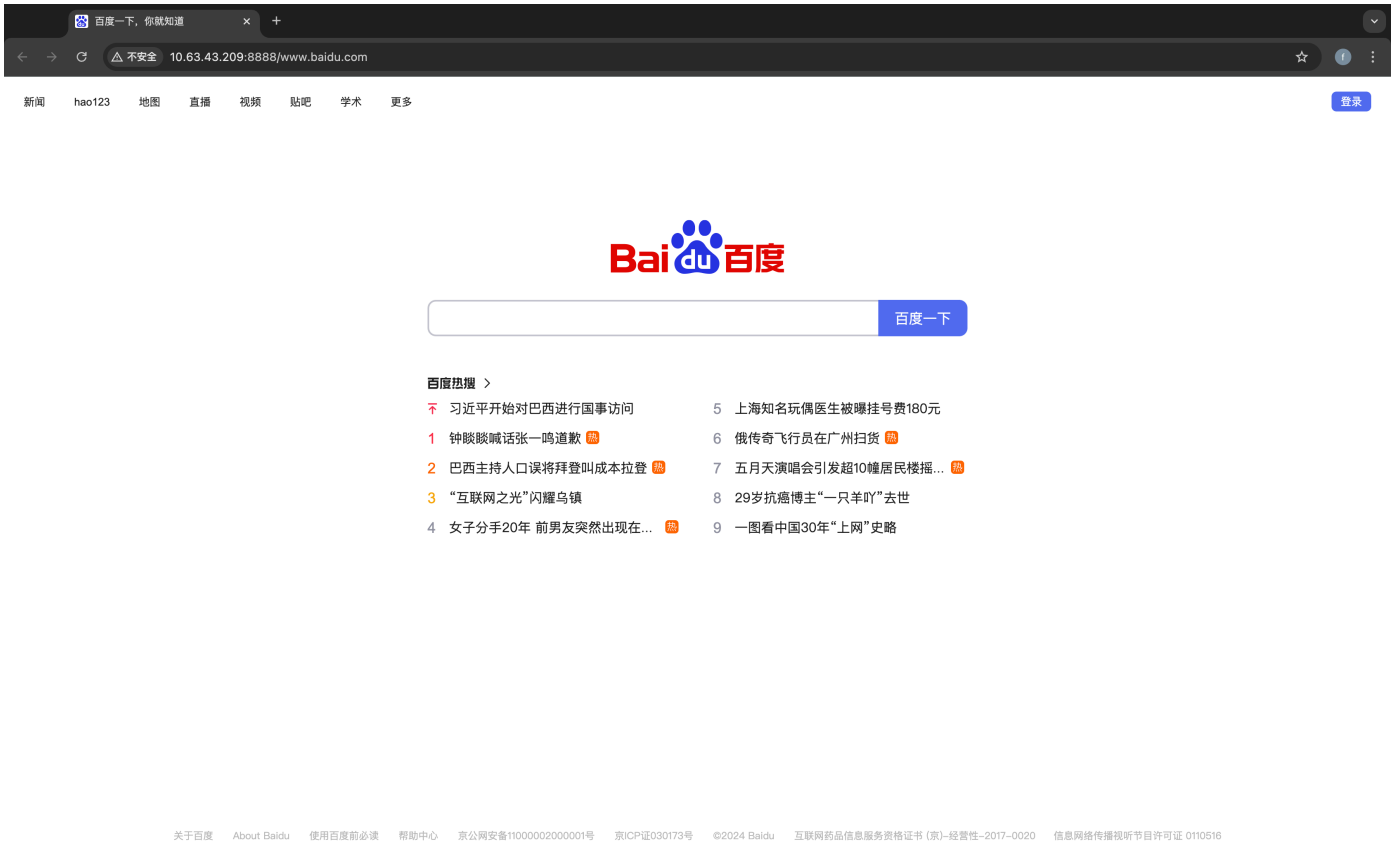
- **在 Netscape（以及 Mozilla 等衍生浏览器中）**，可以通过以下步骤设置代理：
  - 转到 **工具 > 选项 > 高级选项卡 > 网络选项卡 > 连接设置**（Connection Settings）。

在这两种情况下，需要输入代理服务器的地址和端口号（即运行代理服务器时使用的地址和端口号）。通过这种方式，可以在同一台计算机上运行代理服务器和浏览器而不会出现问题。在这种配置下，要通过代理服务器获取网页，只需在浏览器中输入要访问网页的 URL 即可。

For e.g. http://www.google.com

What to Hand in

提交完整的代理服务器代码，以及客户端的截图，以验证您确实通过代理服务器获取了网页。



运行结果：

```
[redacted] WebServerLab % python3 ProxyServer.py 10.63.43.209
Ready to serve...
Received a connection from: ('10.63.43.209', 61372)
GET /www.baidu.com HTTP/1.1
Host: 10.63.43.209:8888
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: ABTEST=4|1732101534|v17


baidu.com.html
www.baidu.com
```

```python
from socket import *
```

```python
import sys

if len(sys.argv) <= 1:
    print('Usage : "python ProxyServer.py server_ip"\n[server_ip : It is the IP Address Of Proxy Server')
    sys.exit(2)
# Create a server socket, bind it to a port and start listening
tcpSerSock = socket(AF_INET, SOCK_STREAM)
# Fill in start.
proxy_port = 8888
tcpSerSock.bind(('', proxy_port))
tcpSerSock.listen()

# Fill in end.
while 1:
    # Start receiving data from the client
    print('Ready to serve...')
    tcpCliSock, addr = tcpSerSock.accept()
    print('Received a connection from:', addr)
    # Fill in start.
    message = tcpCliSock.recv(4096).decode()
    # Fill in end.
    print(message)
    # Extract the filename from the given message
    url = message.split()[1]
    # 如果路径是以 / 开头的，尝试从路径中提取主机
    if url.startswith("/"):
        url = url[1:]  # 移除开头的斜杠
        host = url.split("/")[0]  # 提取主机名
        path = "/" + "/".join(url.split("/")[1:])  # 提取路径
    else:
        # 处理完整的 URL
        if "://" in url:  # 去掉协议部分
            protocol, url = url.split("://")
        host = url.split("/")[0]  # 主机名
        path = "/" + "/".join(url.split("/")[1:])  # 路径
    fileExist = "false"
    filename = path.split("/")[-1] or f"{host.replace("www.", "", 1)}.html"  # filename = "" (空字符串)
    print(filename)
    try:
        # Check whether the file exist in the cache
        f = open(filename, "rb")
        outputdata = f.readlines()
        fileExist = "true"
        # ProxyServer finds a cache hit and generates a response message
        tcpCliSock.send(b"HTTP/1.1 200 OK\r\n")
        tcpCliSock.send(b"Content-Type:text/html\r\n")
        # Fill in start.
        # Fill in end.
        print('Read from cache')
        # Error handling for file not found in cache
```

```
    except IOError:
        if fileExist == "false":
            # Create a socket on the proxyserver
            # Fill in start.
            c = socket(AF_INET, SOCK_STREAM)
            # Fill in end.
            hostn = host
            print(hostn)
            try:
                # Connect to the socket to port 80
                # Fill in start.
                c.connect((hostn, 80))
                # Fill in end.
                # Create a temporary file on this socket and ask port 80 for the file requested by the client
                target_request = f"GET {path} HTTP/1.1\r\nHost: {hostn}\r\n\r\n"
                c.send(target_request.encode())
                # Read the response into buffer
                # Fill in start.
                write = ""
                while True:
                    response = c.recv(4096)
                    write += response.decode()
                    if len(response) > 0:
                        tcpCliSock.send(response)
                    else:
                        break
                # Fill in end.
                # Create a new file in the cache for the requested file.
                # Also send the response in the buffer to client socket and the corresponding file in the cache
                tmpFile = open('./' + filename, "wb")
                # Fill in start.
                tmpFile.write(write.encode())  # 将响应内容写入缓存文件
                tcpCliSock.send(b"HTTP/1.1 200 OK\r\n")
                tcpCliSock.send(b"Content-Type:text/html\r\n")
                # Fill in end.
            except:
                print("Illegal request")
        else:
            # HTTP response message for file not found
            # Fill in start.
            response_line = "HTTP/1.1 404 Not Found\r\n"
            content_type = "Content-Type: text/html\r\n"
            blank_line = "\r\n"
            error_message = "<html><body><h1>404 Not Found</h1></body></html>"

            tcpCliSock.send(response_line.encode())
            tcpCliSock.send(content_type.encode())
            tcpCliSock.send(blank_line.encode())
            tcpCliSock.send(error_message.encode())
            # Fill in end.
```

```
    # Close the client and the server sockets
    tcpCliSock.close()
    # Fill in start.
    tcpSerSock.close()
    # Fil in end.
```

套接字作业28、29

P28

P28. 在一台主机上安装编译 TCPClient 和 UDPClient Python 程序，在另一台主机上安装编译 TCPServer 和 UDPServer 程序。

a. 假设你在运行 TCPServer 之前运行 TCPClient，将发生什么现象？为什么？

b. 假设你在运行 UDPServer 之前运行 UDPClient，将发生什么现象？为什么？

c. 如果你对客户端和服务器端使用了不同的端口，将发生什么现象？

a.如果在TCPServer之前运行TCPClient，TCPClient 会尝试与 TCPServer 建立连接，但由于 TCPServer 未启动，连接将失败。

因为TCP 是面向连接的协议，客户端和服务器需要建立可靠的连接（三次握手）才能通信。在 TCPClient 运行时，它会尝试连接指定的服务器地址和端口。如果服务器（TCPServer）未运行，目标地址的端口未处于监听状态，操作系统会返回错误。因此，客户端无法建立连接。

b.如果在UDPServer之前运行UDPClient,UDPClient应正常工作，因为UDP 是无连接协议，客户端无需与服务器建立连接即可发送数据。如果服务器未启动，客户端发送的数据包无法被接收或处理，但协议本身不会检查目标是否可达，因此客户端不会报错。数据包通常会在网络设备或目标主机的网络堆栈中被丢弃。

c.

TCP

1. 现象：
    - 客户端会尝试连接到指定的服务器端口，但如果服务器监听的端口不同，连接会失败。
2. 原因：
    - TCP 是基于端口的协议，客户端需要准确地连接到服务器监听的端口。如果端口不匹配，客户端的连接请求会被目标主机拒绝，操作系统会返回错误。

UDP

1. 现象：
    - 客户端可以发送数据包，但服务器不会接收到这些数据包。
2. 原因：
    - UDP 数据包是通过目标 IP 地址和端口号进行路由的，如果客户端发送到的端口和服务器监听的端口不同，数据包不会被服务器接收，可能被丢弃。

p29

P29. 假定在 UDPClient. py 中在创建套接字后增加了下面一行：

```
clientSocket.bind(('', 5432))
```

有必要修改 UDPServer. py 吗？UDPClient 和 UDPServer 中的套接字端口号是多少？在变化之前它们是多少？

不需要修改 UDPServer.py,UDPClient和UDPServer中的套接字端口号为5432，变换前UDPClient在创建端口时未指定端口号,是系统分配的随机端口号。查看UDPServer，可知UDPServer通过从客户端接收的数据报确定客户端地址信息和端口号，所以服务器可以接收到来自任何源端口的数据包（包括现在固定的 5432）。