****

**课程实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| **课程编号** |  | **课程名称** | **计算机网络** |
| **实验学分** |  | **任课教师** | **叶松涛** |
| **学生学号** | **190320020** | **行政班级** | **信科1901** |
| **学生姓名** | **田章** | **联系方式** | **17680314099** |

**20 22 -2022学年度2学期**

**教务处制**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **实验** | | | | |
| **实验编号** | **实验1** | **实验名称** | **Web服务器** | |
| **指导教师** | **叶松涛** | **实验日期** | **2022.5.18** | |
| **实验同组人** | **未分组** |  |  | |
| 1. **实验目的和要求**   目的：1.学习Python中TCP连接的套接字编程的基础知识：如何创建套接字，将其绑定到特定的地址和端口，以及发送和接收HTTP数据包。2.学习一些HTTP首部格式的基础知识。  要求：开发一个处理一个HTTP请求的Web服务器，该服务器接受并解析HTTP请求，然后从服务器的文件系统获取所请求的文件，创建一个由响应文件组成的HTTP响应消息，前面是首部行，然后将响应直接发送给客户端。如果请求的文件不存在于服务器中，则服务器应该向客户端发送“404 Not Found”差错报文。 | | | | |
| 1. **实验设备、软件**   笔记本上使用VScode  Python环境：3.10.1  客户端：谷歌浏览器+火狐浏览器（练习2在cmd中运行客户端程序client.py） | | | | |
| 1. **实验内容（步骤、计划、方案）**   创建一个html静态资源的展示界面，同时将服务器端程序代码与该界面放在同一个目录下，由于文本编辑器可能会出现一些莫名的路径错误，这里我们都放在实验1文件夹下，然后用编辑器VScode打开实验1即可。  具体步骤如下：   * 创建一个HelloWorld.html文件作为响应给客户端的资源文件，这里我们使用VScode自带的emmet插件来生成html文件基本格式(快捷键：!+tab键)，然后在body标签中写一个一级标题Hello World! * 创建一个server.py文件，这是我们手写的一个Web服务器程序。创建服务器端的套接字ServerSocket，并绑定端口号6789，同时设置最大连接数为1，调用accept()表示接收来自客户端的socket。在正常运行（try语句块）中设置响应头为HTTP/1.1 200 OK，这个响应头规定了HTTP请求的协议版本号为1.1，状态码200表示正常访问，即访问成功；在出现错误（比如资源路径错误）的except语句块中将请求头中的状态码设置为404。 * 打开谷歌浏览器，在访问地址栏路径中输入主机名+端口号和请求资源路径，即<http://localhost:6789/HelloWorld.html>，其中127.0.0.1表示本机,这里可以将127.0.0.1写成localhost是因为在"C:\Windows\System32\drivers\etc\hosts”文件中设置了127.0.0.1 localhost,NDS域名解析器会自动将域名localhost解析成127.0.0.1。 * 至此，我们在浏览器客户端中可以看到大大的Hello World!，如果将请求路径中的资源路径/HelloWorld.html写成了其他，将会抛出一个404错误，并且由于最大连接数为1，我们将浏览器关闭后服务器将自动停止运行，再次访问资源路径也将访问失败。 | | | | |
| 1. **实验原始记录、结果分析（可以附页）**   **HelloWorld.html代码如下：**  <!DOCTYPE *html*>  <html *lang*="zh-CN">  <head>      <meta *charset*="UTF-8">      <meta *http-equiv*="X-UA-Compatible" *content*="IE=edge">      <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">      <title>首页</title>  </head>  <body>      <h1>Hello World!</h1>  </body>  </html>  **Server.py代码如下：**  #import socket module  from socket import \*  serverSocket = socket(AF\_INET, SOCK\_STREAM)  #Prepare a sever socket  serverPort = 6789  serverSocket.bind(("", serverPort))  serverSocket.listen(1)  while True:      #Establish the connection      print('Ready to serve...')      connectionSocket, addr = serverSocket.accept()      try:          message = connectionSocket.recv(1024)          filename = message.split()[1]          f = open(filename[1:])          outputdata = f.read()          f.close()          #Send one HTTP header line into socket          outputdata = 'HTTP/1.1 200 OK\r\n\r\n' + outputdata          #Send the content of the requested file to the client          for i in range(0, len(outputdata)):              connectionSocket.send(outputdata[i].encode())          connectionSocket.close()          print("OK!")      except *IOError*:          #Send response message for file not found          outputdata = 'HTTP/1.1 404 Not Found\r\n\r\n'          #Close client socket          for i in range(0, len(outputdata)):              connectionSocket.send(outputdata[i].encode())          connectionSocket.close()  serverSocket.close()  **成功访问截图：**    **访问资源错误截图**    **关闭浏览器后后，服务器自动停止运行** | | | | |
| 1. **实验思考题及其解答**   **练习1：目前，这个Web服务器一次只处理一个HT1TP请求。请实现一个能够同时处理多个请求的多线程服务器。使用线程，首先创建一个主线程，在固定端口监听客户端请求。当从客户端收到TCP连接请求时，它将通过另一个端口建立TCP连接，并在另外的单独线程中为客户端请求提供服务。这样在每个请求/响应对的独立线程中将有一个独立的TCP连接。**  思考：将实验1的服务器端程序中的最大连接数设置为10，为了能够同时接收到多个客户端的访问，应该开启多线程，需要导入threading模块，在实验1客户端的基础上，我们将请求封装成一个方法。  **server\_thread.py代码如下：**  #import socket module  from socket import \*  import threading  def webProcess(*connectionSocket*):      try:          message = connectionSocket.recv(1024)          filename = message.split()[1]          f = open(filename[1:], "rb")          outputdata = f.read()          outputdata = outputdata.decode()          f.close()          #Send one HTTP header line into socket          outputdata = 'HTTP/1.1 200 OK\r\n\r\n' + outputdata          #Send the content of the requested file to the client          for i in range(0, len(outputdata)):              connectionSocket.send(outputdata[i].encode())          connectionSocket.close()          print("OK!")      except *IOError*:          #Send response message for file not found          outputdata = 'HTTP/1.1 404 Not Found\r\n\r\n'          #Close client socket          for i in range(0, len(outputdata)):              connectionSocket.send(outputdata[i].encode())          connectionSocket.close()  serverSocket = socket(AF\_INET, SOCK\_STREAM)  #Prepare a sever socket  serverPort = 6789  serverSocket.bind(("", serverPort))  serverSocket.listen(10)  while True:      #Establish the connection      print('Ready to serve...')      connectionSocket, addr = serverSocket.accept()      thread = threading.Thread(*target* = webProcess, *args* = (connectionSocket, ))      thread.start()  serverSocket.close()  **运行server\_thread.py文件，此时我们发现谷歌浏览器访问之后关闭浏览器，服务器并没有终止，并且可以谷歌浏览器与火狐浏览器同时进行访问**      **练习2不使用浏览器，编写自己的HTTP客户端来测试你的服务器。您的客户端将使用一个TCP连接用于连接到服务器，向服务器发送HTTP请求，并将服务器响应显示出来。您可以假定发送的HTTP请求将使用GET方法。**  **思考：客户端程序需要设置主机名，由于我们实在本机上运行的服务器，因此主机名设置为127.0.0.1或者localhost都可，在请求过程中，请求路径包含协议、主机名、端口号、请求资源路径以及请求方式，因此我们还需要绑定端口号，这里设置为6789,请求资源为客户端所在目录下的HelloWorld.html，请求方式为GET请求。**  **Client.py代码如下：**  from socket import \*  serverName = '127.0.0.1'  serverPort = 6789  clientSocket = socket(AF\_INET, SOCK\_STREAM)  clientSocket.connect((serverName, serverPort))  outputdata = 'GET /HelloWorld.html HTTP/1.1\r\nHost: 127.0.0.1\r\n\r\n'  clientSocket.send(outputdata.encode())  data = 1  while data:      data = clientSocket.recv(1024)      print(data.decode(), *end* = '')  clientSocket.close()  **全部代码的目录结构如下：**    **开启服务器后，在cmd中运行客户端程序，由于我们的主机名与端口号与服务器是一致的，并且访问的资源也是存在，因此访问是成功的**    **如果我们将资源路径改为一个不存在的资源** | | | |
| 1. **实验评分**   **教师签名：**    **年 月 日** | | | |