套接字编程作业1：Web服务器

信息安全专业智息技术班　杨乙　21307130076

完整的服务器代码：

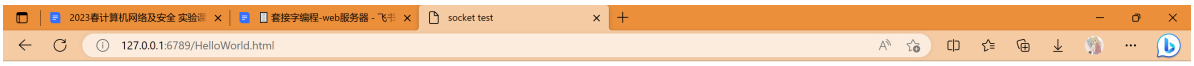
```python
# import socket module
from socket import *
serverSocket = socket(AF_INET, SOCK_STREAM)
# Prepare a sever socket
# Fill in start
serverPort = 6789
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
# Fill in end
while True:
    # Establish the connection
    print('Ready to serve...')
    connectionSocket, addr = serverSocket.accept() #Fill in start # end
    try:
        message = connectionSocket.recv(1024) # Fill in start # Fill in end
        filename = message.split()[1] # filename = 'helloworld.html'
        f = open(filename[1:])
        outputdata = f.read() # Fill in start # Fill in end
        # Send one HTTP header line into socket
        # Fill in start
        import time
        t = time.strftime('%a, %d %b %Y %H:%M:%S', time.gmtime())
        header = 'HTTP/1.1 200 OK\r\n' + \
                 'Date: ' + t + ' GMT' + '\r\n' + \
                 'Connection: keep-alive\r\n' + \
                 'Content-Length: %d\r\n' % (len(outputdata)) + \
                 'Content-Type: text/html; charset=utf-8\r\n\r\n'
        connectionSocket.send(header.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.close()
    except IOError:
        # Send response message for file not found
        # Fill in start
        header = 'HTTP/1.1 404 NOT FOUND\r\n\r\n'
        connectionSocket.send(header.encode())
        # Fill in end
        # Close client socket
        # Fill in start
        connectionSocket.close()
        # Fill in end
serverSocket.close()
```

对代码的解释：

- 3 - 8 行：建立 TCP 欢迎套接字，将欢迎套接字绑定到指定端口 6789，将服务器最大连接客户端数目设置为 1
- 12 - 13 行：建立 TCP 连接套接字，打印准备消息
- 15 - 18 行：字符串 message 当中存储了请求报文的内容。通过字符串的分割和截取得到 html 文件名，从文件中读取消息主体内容，存储在 `outputdata` 字符串中
- 21 - 24 行：准备并发送响应报文。按照作业要求，报文具体内容如下：
  - 状态行：请求成功，为 `HTTP/1.1 200 OK`
  - Date：用 `time` 模块中的 `gmtime()` 获取 GMT 时间，用 `strftime()` 进行格式化，拼接到响应报文字符串中
  - Connection：查看请求头的 Connection 字段为 `keep-alive`；在本例中，由于没有消息主体没有图片等信息，客户端仅进行一次请求，因此为了减少资源占用，将响应头的 Connection 字段设置为 `close`，在请求完成后立刻关闭连接
  - Content-Length：消息主体内容的长度，即字符串 `outputdata` 的长度
  - Content-Type：内容类型，定义了网络文件的类型和网页的编码

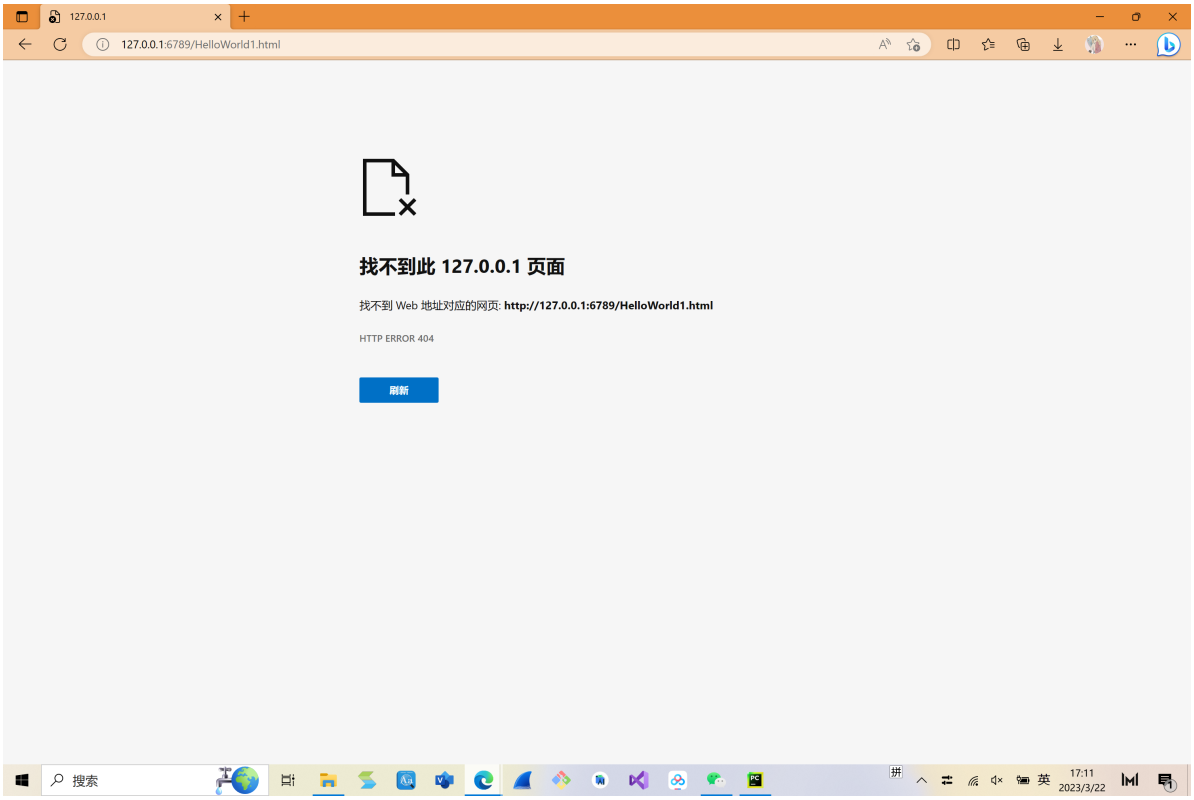
  最后需要将响应头通过 `send()` 函数发送出去，这里需要使用 `encode()` 函数将字符串进行 UTF-8 编码
- 27 - 29 行：发送消息主体内容
- 30 - 40 行：当要访问不存在的网页（即打开不存在的文件）时，捕获到 IOError 异常，发送状态码为 404 的响应报文，最后关闭 TCP 连接套接字和 TCP 欢迎套接字

以下是客户端浏览器的屏幕截图：

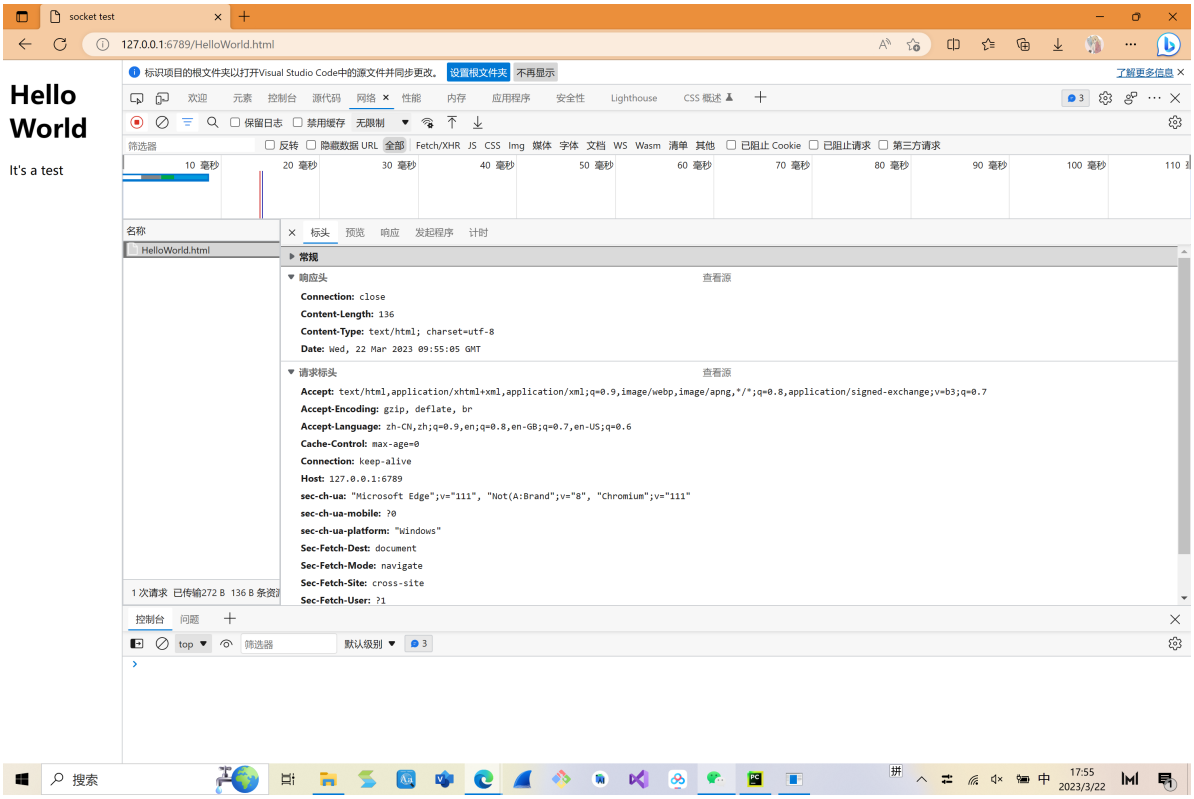


Hello World

It's a test



以下是响应报文和请求报文：



可选练习

更改后的服务器代码：

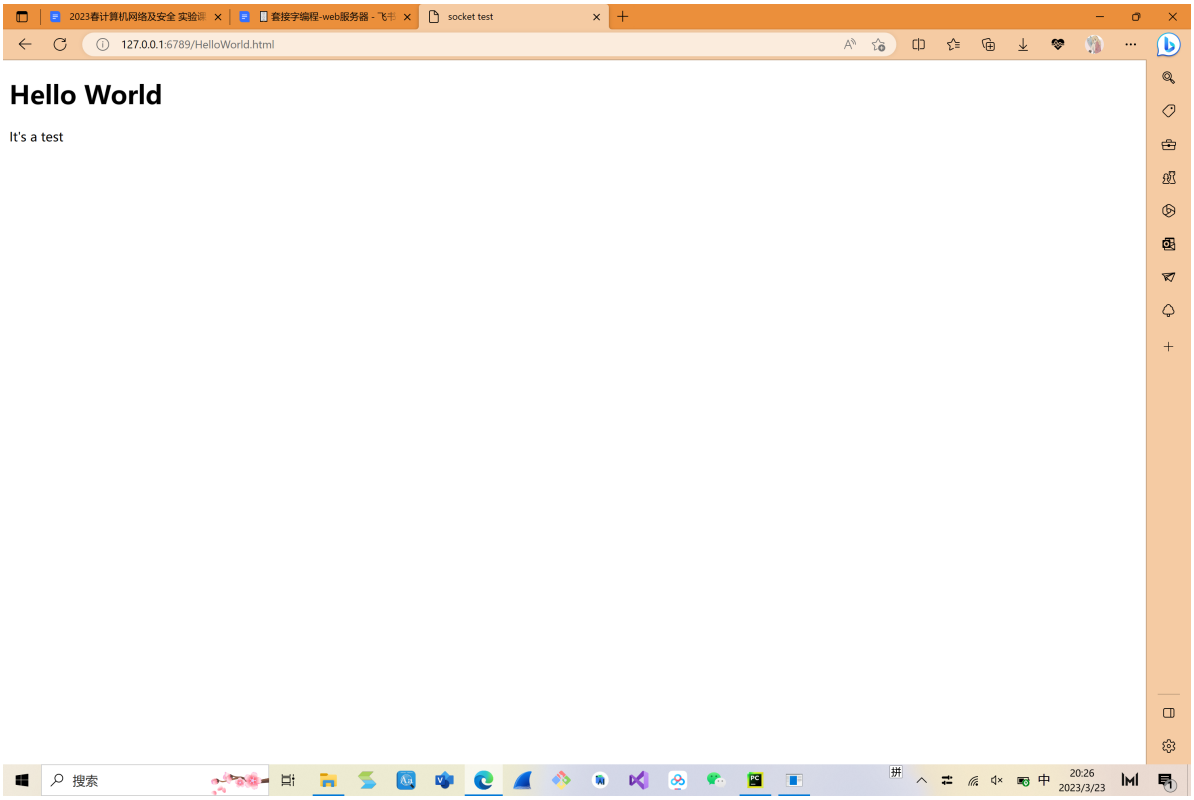
```
import threading
import time
from socket import *


def new_task(connectionSocket):
    try:
        message = connectionSocket.recv(1024)
        filename = message.split()[1]
        f = open(filename[1:])
        outputdata = f.read()
        t = time.strftime('%a, %d %b %Y %H:%M:%S', time.gmtime())
        header = 'HTTP/1.1 200 OK\r\n' + \
                 'Date: ' + t + ' GMT' + '\r\n' + \
                 'Connection: keep-alive\r\n' + \
                 'Content-Length: %d\r\n' % (len(outputdata)) + \
                 'Content-Type: text/html; charset=utf-8\r\n\r\n'
        outputdata = header + outputdata
        for i in range(0, len(outputdata)):
            connectionSocket.send(outputdata[i].encode())
        connectionSocket.close()
    except IOError:
        header = 'HTTP/1.1 404 NOT FOUND\r\n\r\n'
        connectionSocket.send(header.encode())
        connectionSocket.close()


serverPort = 6789
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen()

while True:
    try:
        print('Ready to serve...')
        connectionSocket, addr = serverSocket.accept()
        new_thread = threading.Thread(target=new_task,
                                      kwargs={"connectionSocket":
connectionSocket})
        new_thread.start()
    except KeyboardInterrupt:
        serverSocket.close()
```

以下是客户端浏览器的屏幕截图：



对代码的解释：

和之前的服务器代码原理基本相同，不同之处在于收到 TCP 连接请求时在新的线程中为客户端请求提供服务，使得每个请求/响应的独立线程中有一个独立的 TCP 连接。

因此，每次调用 `accept()` 函数建立 TCP 连接套接字之后，需要建立一个新的线程，在新的线程中为客户端请求提供服务

P.S.

作业中说 “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。”

但是我觉得 ”通过另一个端口建立 TCP 连接“ 可能是无法实现的，因为 `accept()` 函数是在原来的端口，也就是监听端口上进行连接，似乎也无法再在代码里指定一个新的端口。