实验报告

课程名称	：	**计算机网络原理**
实验题目	：	**WEB服务器编程实验**
学号	：	**21281280**
姓名	：	**柯劲帆**
班级	：	**物联网2101班**
指导老师	：	**常晓琳**
报告日期	：	**2024年3月28日**

目录

1. 实验目的

本实验旨在通过运用各种语言实现WEB服务器软件，达到以下几点要求：

1. 处理一个HTTP请求。
2. 接收并解析HTTP请求。
3. 从服务器文件系统中获得被请求的文件。
4. 创建一个包括被请求的文件的HTTP响应信息。
5. 直接发送该信息到客户端。

2. 实验环境

- **Server OS**：WSL2 Ubuntu-22.04 (Kernel: 5.15.146.1-microsoft-standard-WSL2)
- **Client Browser**：Windows Google Chrome
- **Python**：version 3.11.5

3. 实验原理

3.1. 连接原理

HTTP连接的建立是通过申请套接字(Socket)实现的。客户打开一个套接字并把它约束在一个端口上，如果成功，就相当于建立了一个虚拟文件。以后就可以在该虚拟文件上写数据并通过网络向外传送。

3.2. 解析请求

Server收到的请求格式举例如下：

```
GET /index.html HTTP/1.1
Host: 172.28.40.76
Connection: keep-alive
...
```

其中第1行中`/index.html`是请求的文件路径，第2行中`172.28.40.76`是请求源的IP地址。简单来说只需要将请求的文件通过socket发送给请求的IP地址即可。

另外，其他部分信息包括：

- GET /index.html HTTP/1.1
  - `GET`：HTTP请求方法，用于请求访问指定资源。GET方法表示请求获取指定的资源。
  - `HTTP/1.1`：HTTP协议版本。说明此次请求使用的是HTTP 1.1版本。
- Connection: keep-alive
  - 告诉服务器客户端希望保持连接开放，以便发送或接收后续的请求，而不是在完成这一请求后立即关闭连接。

3.3. 发送响应

Server返回的响应格式举例如下：

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 333

<!DOCTYPE html>
...
```

其中包括信息：

- HTTP/1.1 200 OK
  - `HTTP/1.1`：响应使用的HTTP版本，这里是HTTP 1.1。
  - `200 OK`：HTTP状态码及其描述，200代表请求成功，资源被正常的GET或POST等方法取得，"OK"是对应的状态描述。
    - 状态码表示类型：`1xx`保留、`2xx`表示请求成功地接收、`3xx`为完成请求客户需进一步细化请求、`4xx`客户错误、`5xx`服务器错误。
    - 常用状态码：`2xx`OK、`403`Forbidden、`404`Not Found、`502`Bad Gateway。

- Content-Type: text/html
  - 指定返回内容的MIME类型，这里是`text/html`，表示响应体中的内容是HTML格式的文本。
- Content-Length: 333
  - 表示响应体的长度或大小为333字节。客户端可以使用这个信息来读取响应体的正确字节数。
- <!DOCTYPE html> ...
  - 响应体部分，是一个HTML格式的网页内容。

4. 实验过程

4.1. 编写代码

首先编写`start_server`函数处理服务器收发文件的基本功能：

1. 创建一个socket对象，设置允许地址重用。将socket对象绑定到指定的主机地址（0.0.0.0）和端口（80），并开始监听连接请求。
2. 无限循环等待客户端连接：
    1. 接收客户端连接请求。
    2. 读取并解析HTTP请求。
    3. 提取请求行，解析出方法、路径和HTTP版本。
    4. 调用`serve_file`函数处理请求，并向客户端发送响应。
    5. 处理完请求后，关闭客户端socket连接。

接着编写`serve_file`函数处理文件请求:

1. 解析HTTP请求的URL路径，处理URL编码，并设置默认页面为`/index.html`。
2. 构建请求文件的绝对路径，并确保该路径不会跳出根目录以保证安全。否则返回403 Forbidden。
3. 判断文件是否存在：
    - 如果存在，读取文件内容，并根据文件类型设置`Content-Type`（HTML或其他），然后创建一个HTTP 200响应返回文件内容。
    - 如果请求路径不合法（例如，试图访问根目录之外的文件），则返回403 Forbidden响应。
    - 如果文件不存在，返回404 Not Found响应。

同时还设置了一些异常处理，捕获并处理可能发生的异常，例如连接重置错误或其他意外错误。

另外，编写两个html文件，并在内容中互相引用。

以上代码见附录。

4.2. 运行实验

将代码文件放在合适的位置，文件夹构建如下：

```
.
├── src
|   └── main.py
└── www
    ├── dir
    |   └── hello.html
    └── index.html
```

使用`ip a`或`ifconfig`命令获取ip地址：

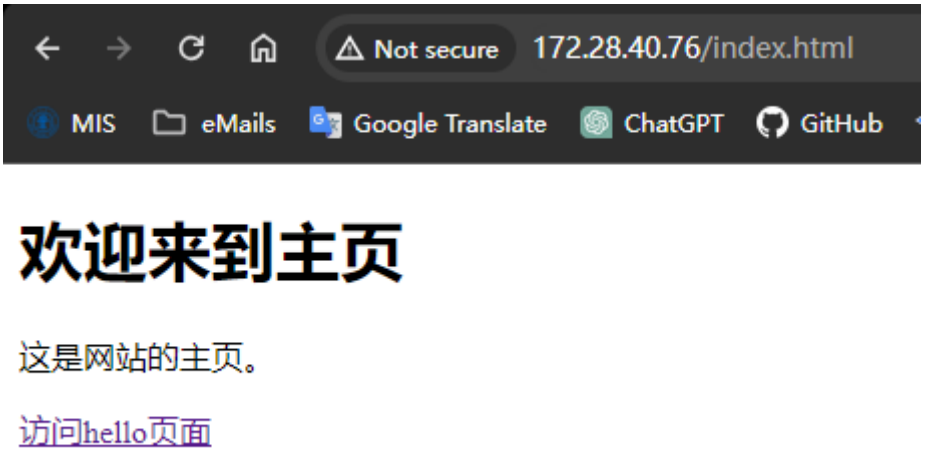
```
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1480 qdisc mq state UP group default qlen 1000
    link/ether 00:15:5d:67:27:0d brd ff:ff:ff:ff:ff:ff
    inet 172.28.40.76/20 brd 172.28.47.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe67:270d/64 scope link
       valid_lft forever preferred_lft forever
```

在`src`目录运行`main.py`代码。代码中设置服务器端口为`80`，这个端口需要管理员权限。执行：

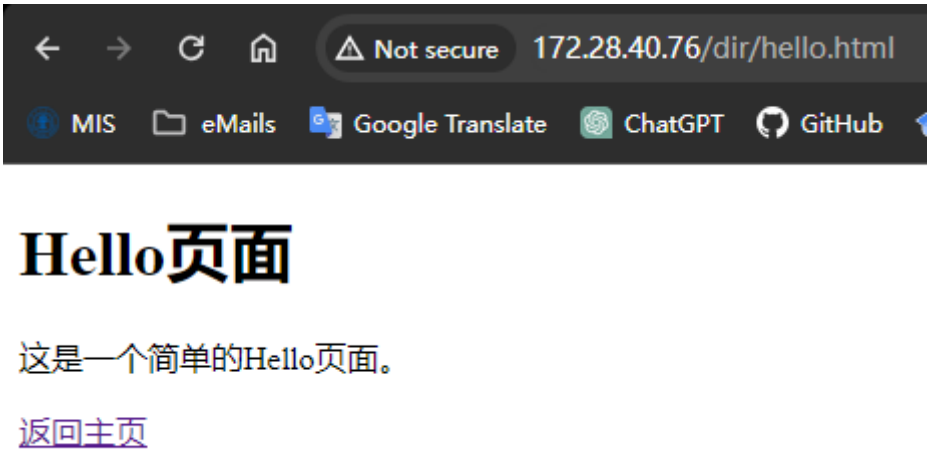
```
$ sudo python3 main.py
Server listening on 0.0.0.0:80
Connection from ('172.28.32.1', 56763)
Connection from ('172.28.32.1', 56764)
...
```

代码中设置了打印请求源的IP和端口，可以看到Windows主机浏览器使用的是`56763`等大于`1024`的端口发起请求。

在Windows主机端的浏览器访问ip`172.28.40.76`，即可向服务器发送请求，收到服务器的响应后，内容会被浏览器自动解析，截图如下：



点击该超链接，跳到`hello.html`页面，截图如下：



5. 实验问题及解决方案

- **在超链接中，引用的文件`dir/hello.html`会不会被误解析为客户端本地的文件而不是服务器上的文件?**

  经过查阅资料，这个文件路径会自动接上IP，对服务器发出请求，而不会访问本地的文件。当点击这个链接时，浏览器会解释这个路径为相对于当前页面的URL的路径。服务器运行在`http://172.28.40.76:80/`，点击链接后浏览器会请求`http://172.28.40.76:80/dir/hello.html`，这是一个网络请求，指向的是服务器上的资源，而不是客户端机器上的本地文件。

  相反，浏览器不允许网页通过链接直接访问或打开客户机上的文件，因为这样做可能会泄露用户数据或被用于恶意目的。

6. 总结和感想

通过本次实验，我深刻理解了WEB服务器的工作原理以及如何使用Python进行简单WEB服务器的编程。实验不仅加深了我对计算机网络原理的理解，还提高了我的编程能力，尤其是在网络编程方面。以下是我的一些具体感想：

1. **深入理解HTTP协议：** 实验过程中，我通过编写代码来处理HTTP请求和响应，这让我更加熟悉了HTTP协议的各种细节，如请求方法、状态码、请求头和响应头等。通过逐行分析和构造HTTP消息，我能更好地理解这些组件在实际通信中的作用。
2. **掌握基本的网络编程技巧：** 在实验中，我使用Python的socket库来实现服务器和客户端之间的通信。这个过程中，我学习到了如何创建socket，如何监听端口，以及如何接收和发送数据。这些技能对我今后进行更复杂的网络编程或者研究网络安全等方面有很大帮助。
3. **认识到安全性的重要性：** 在实现WEB服务器时，我意识到了保证服务器安全性的重要性。例如，需要确保请求的文件路径不会跳出服务器的根目录，防止访问服务器上的敏感文件。这让我意识到，编程时不仅要关注功能的实现，还要充分考虑到安全因素。
4. **体验到解决问题的成就感：** 实验过程中遇到了一些问题，如处理文件路径的安全性问题、响应静态文件请求等，通过查阅资料和不断尝试，我最终解决了这些问题。这个过程虽然有些困难，但解决问题后的成就感让我对编程有了更深的喜爱。

5. **感受到实践的重要性：** 通过这次实践活动，我深刻体会到了“实践出真知”的道理。理论学习固然重要，但将所学知识应用到实际中，解决实际问题，才能更深刻地理解和掌握这些知识。

这次实验不仅让我学到了许多关于网络编程的知识和技能，还锻炼了我的问题解决能力，提高了我的自学能力。我将继续保持学习的热情，不断提升自己的专业技能和综合素质。

7. 附录

`src/main.py`:

```
import os
import socket
import urllib.parse

# 设置服务器的根目录
WEB_ROOT = os.path.abspath("../www")

def serve_file(client_socket:socket.socket, path):
    # 解析请求的URL路径，移除开头的/，并处理URL编码
    requested_path = urllib.parse.unquote(path)
    if requested_path == "/":
        requested_path = "/index.html"  # 默认页面

    # 构建文件的绝对路径
    file_path = os.path.join(WEB_ROOT, requested_path.lstrip('/'))

    # 获取安全的绝对路径，以确保它不会跳出根目录
    safe_path = os.path.abspath(file_path)
    if not safe_path.startswith(WEB_ROOT):
        # 如果请求的路径不是根目录的子路径，则返回403 Forbidden
        response = b"HTTP/1.1 403 Forbidden\r\n"
        response += b"Content-Type: text/html\r\n"
        response += b"\r\n"
        response += b"<html><body><h1>403 Forbidden</h1></body></html>"
    else:
        try:
            # 尝试打开并读取文件
            with open(safe_path, "rb") as file:
                content = file.read()
            # 根据文件类型设置Content-Type（这里简化处理，仅对HTML进行了处理）
            content_type = "text/html" if safe_path.endswith(".html") else
"application/octet-stream"
            response = b"HTTP/1.1 200 OK\r\n"
            response += f"Content-Type: {content_type}\r\n".encode()
            response += b"Content-Length: " + str(len(content)).encode() +
b"\r\n"
            response += b"\r\n"
            response += content
        except FileNotFoundError:
            # 文件未找到，返回404响应
            response = b"HTTP/1.1 404 Not Found\r\n"
            response += b"Content-Type: text/html\r\n"
            response += b"\r\n"
```

```
            response += b"<html><body><h1>404 Not Found</h1></body></html>"

    client_socket.send(response)

def start_server(host="0.0.0.0", port=80):
    socket_server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    socket_server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1) # 允许地址重用
    socket_server.bind((host, port))
    socket_server.listen(5)
    print(f"Server listening on {host}:{port}")

    while True:
        client_socket = None
        try:
            client_socket, address = socket_server.accept()
            print(f"Connection from {address}")

            # 接收客户端请求
            request = client_socket.recv(1024).decode("utf-8")
            if not request:
                print("Empty request received.")
                continue

            # 解析HTTP请求的第一行获取请求路径
            request_lines = request.split("\r\n")
            if request_lines and len(request_lines[0].split(" ")) >= 3:
                method, path, _ = request_lines[0].split(" ", 2)
                # 根据路径返回相应的文件
                serve_file(client_socket, path)
            else:
                print("Invalid request line:", request_lines[0])
            # 关闭客户端连接

        except ConnectionResetError:
            print("Connection reset by peer.")
        except Exception as e:
            print(f"Unexpected error: {e}")
        finally:
            if client_socket:
                client_socket.close()

if __name__ == "__main__":
    start_server()
```

`www/index.html`:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>主页</title>
</head>
<body>
    <h1>欢迎来到主页</h1>
    <p>这是网站的主页。</p>
    <a href="dir/hello.html">访问hello页面</a>
</body>
</html>
```

`www/dir/hello.html`:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hello页面</title>
</head>
<body>
    <h1>Hello页面</h1>
    <p>这是一个简单的Hello页面。</p>
    <a href="../index.html">返回主页</a>
</body>
</html>
```