Web 服务器构造详细设计说明书

1 引言

1.1 编写目的

本说明书确定简单的 Web 服务器系统的详细功能模块，为下阶段的代码编写、项目开发工作提供主要依据。

1.2 背景

- **软件系统名**：简单的访问文件 Web 服务器
- **项目提出者**：《计算机网络第 2 章.pptx》
- **项目开发者**：Lolipop（2018091202000）

1.3 参考资料

《计算机网络——自顶向下方法（第七版）》

2 任务描述

2.1 目标

使用 Python 语言构造一个基于套接字的简单 Web 服务器，安装服务器后可以在其他主机上用浏览器（如 IE）向服务器端请求一个文件。

2.2 详细设计方法和工具

（1）方法

- 程序流程图。

（2）工具

- **JetBrains PyCharm**：编写 Python 程序的 IDE;
- **Microsoft Visio**：软件工程建模软件。

2.3 运行环境

（1）硬件环境

- **windows 服务器**：CPU1 核 2G/操作系统 windows 10 1903/SSD 固态硬盘 10G
- **Linux 服务器**：CPU1 核 2G/操作系统 Ubuntu Server 18.04.4 LTS/SSD 固态硬盘 10G

（2）软件环境

开发系统：windows 10 1903

网络协议：TCP/IP

浏览器：Chrome 80.0.3987.149 (64 位)

3 系统设计

3.1 设计原则

- 实现的服务器端应具有一定的可靠性和自纠错性；
- 可以在不同的浏览器访问服务器文件资源。

3.2 程序文件

序号	文件名称	说明
1	TCPServer.py	实现服务器端功能的核心文件，客户端可以通过浏览器访问服务器上的文件
2	神秘嘉宾.txt	中文测试文件
3	hello/world.txt	英文测试文件

4 模块设计

4.1 概述

本 Web 服务器系统仅包括获取服务器文件模块，针对该模块特点，下面将构造程序流程图展示模块实现过程。

4.2 获取服务器文件模块

（1）概述

服务器创建套接字获取 Http 请求，并解析 Http 请求报文。在服务器端显示请求报文的各字段名和值，并进行注释说明。根据请求报文获取对象文件路径名并打开文件，封装文件内容到 Http 响应报文。最后使用套接字发送响应报文到客户端，将文件内容显示在 Web 页面上。

（2）程序流程图

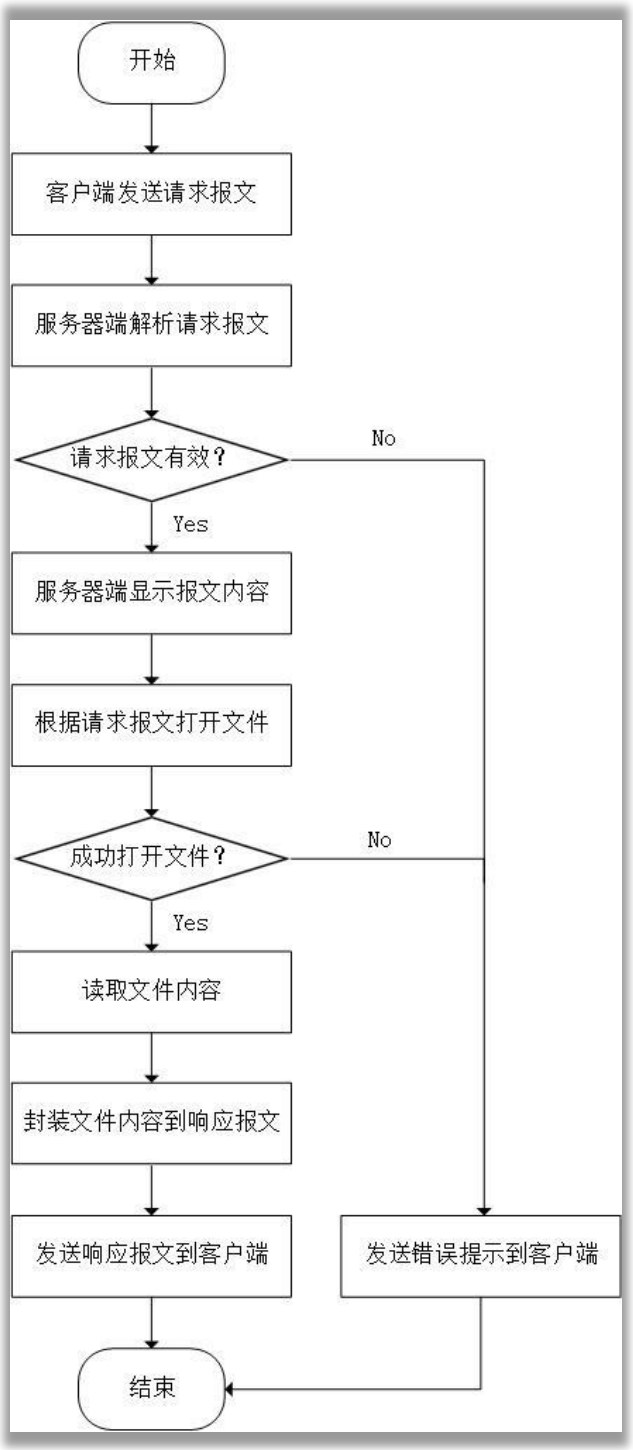

图 1 程序流程图

5 附录

Python 代码 - TCPServer.py

```
# test on Chrome version. 80.0.3987.149
# TCPServer.py
from socket import *
import os

root = os.getcwd()  # 获取当前路径
serverPort = 8888
bufSize = 1024

serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))      # 监听本地 8888 端口
serverSocket.listen(5)
print('The server is on serving at port: ' + str(serverPort))
while True:
    connectionSocket, address = serverSocket.accept()    # 握手
    request = connectionSocket.recv(bufSize).decode()    # 获取请求
    count = 0
    # method 行
    methodLine = request.split('\n')[0]
    srcPath = methodLine.split()[1]      # 获取文件路径
    methodNote = '\n\t 其中' + methodLine.split()[0] + '表示读取请求\t' + srcPath + '表示
URL 路径\t' +\
        methodLine.split()[2] + '表示 HTTP 协议版本\n'
    # Host 行
    hostLine = request.split('\n')[1]
    hostNote = '\n\t 表示请求域名\n'
    # Connection 行
    connectionLine = request.split('\n')[2]
    connectionNote = '\n\t 表示请求完成后服务器的操作\n'
    # Cache-Control 行
    cacheControlLine = request.split('\n')[3]
    cacheControlNote = '\n\t 表示缓存存储的相关内容\n'
    # DNT 行
    DNTLine = request.split('\n')[4]
    DNTNote = '\n\t 表示是否禁止追踪\n'
    # Upgrade-Insecure-Requests 行
    upgradeInsecureRequestsLine = request.split('\n')[5]
    upgradeInsecureRequestsNote = '\n\t 表示是否可以处理 https 协议\n'
    # User-Agent 行
    userAgentLine = request.split('\n')[6]
```

```
        userAgentNote = '\n\t 表示浏览器自身身份\n'
        # Sec-Fetch-Dest  行
        secFetchDestLine = request.split('\n')[7]
        secFetchDestNote = '\n\t 说明所获取的数据将如何使用\n'
        # Accept  行
        acceptLine = request.split('\n')[8]
        acceptNote = '\n\t 表示客户端希望接受的数据类型\n'
        # Sec-Fetch-Site  行
        secFetchSiteLine = request.split('\n')[9]
        secFetchSiteNote = '\n\t 表示请求发起者的原点和该资源的原点之间的关系\n'
        # Sec-Fetch-Mode  行
        secFetchModeLine = request.split('\n')[10]
        secFetchModeNote = '\n\t 表示该请求的模式\n'
        # Sec-Fetch-User  行
        secFetchUserLine = request.split('\n')[11]
        secFetchUserNote = '\n\t 表示导航请求是否是由用户激活触发\n'
        # Accept-Encoding  行
        acceptEncodingLine = request.split('\n')[12]
        acceptEncodingNote = '\n\t 表示客户端能够理解的内容编码方式\n'
        # Accept-Language  行
        acceptLanguageLine = request.split('\n')[13]
        acceptLanguageNote = '\n\t 表示客户端声明可以理解的自然语言\n'
        # request  添加注释以后
        requestAfter = methodLine + methodNote + hostLine + hostNote + connectionLine +
connectionNote + cacheControlLine +\
            cacheControlNote + DNTLine + DNTNote + upgradeInsecureRequestsLine +
upgradeInsecureRequestsNote +\
            userAgentLine + userAgentNote + secFetchDestLine + secFetchDestNote +
acceptLine + acceptNote +\
            secFetchSiteLine + secFetchSiteNote + secFetchModeLine + secFetchModeNote +
secFetchUserLine +\
            secFetchUserNote + acceptEncodingLine + acceptEncodingNote +
acceptLanguageLine + acceptLanguageNote
        #  处理请求
        print('request info:\n' + requestAfter)
        try:
            file = open(root + srcPath, mode='r', encoding='utf-8')
            context = file.read()
            header = 'HTTP/1.1 200 OK\r\n\r\n'
            file.close()
        except FileNotFoundError:     #  文件不存在时
            header = 'HTTP/1.1 404 NOT FOUND\r\n\r\n'
            context = 'No such file. Read failed.'
        except PermissionError:         #  无文件访问权限时
```

```
            header = 'HTTP/1.1 404 NOT FOUND\r\n\r\n'
            context = 'You can\'t access this file. Read failed.'
        except OSError:         # 访问错误
            header = 'HTTP/1.1 404 NOT FOUND\r\n\r\n'
            context = 'Wrong request. Read failed.'
        connectionSocket.sendall(bytes(header.encode('gbk')))
        connectionSocket.sendall(bytes(context.encode('gbk')))    # 返回处理结果
        connectionSocket.close()        # 结束连接
```

测试文件 1 - 神秘嘉宾.txt

```
下一秒，谁是神秘嘉宾
小心翼翼，揭开了面具
掌声鼓励

……

你按了我的门铃
我终于，从呵欠中苏醒
紧张兮兮，对你说一句
“欢迎光临”
```

测试文件 2 - world.txt

```
Hi, here is the common txt file:
    Hello, world!

Good luck!
```