

1 环境

1.1 开发环境

本次实验在 Linux 下进行，发行版 Manjaro 21.1.6，CPU i5-1135G7，内存 16GB。

考虑到本次实验主题是套接字编程，我选择了 Java 语言进行开发，其 socket api 更加清晰通用，易于开发。IDE 是 IDEA，JDK 版本 13。

1.2 运行环境

因为使用的 Java 语言，所以编译生成的.class 文件在安装了 JRE 的机器上都可以运行。

2 系统功能需求

总的来说，本次实验需要开发一个 Web 服务器，类似 Nginx、Apache Web。结合现有的 Web 服务器的功能，将本次实验的功能需求细化如下：

2.1 基本需求

1. 将 Web 服务器监听的 IP 地址、端口，Web 服务器的基路径都写到一个配置文件中，修改配置的时候不用重新编译程序；
2. 能够监听给定的地址，当浏览器（或者其他客户端）向给定地址发起的请求时，能够处理请求、根据请求定位文件、构建响应报文、回报文给请求方；
3. 能够识别请求文件的 MIME 类型，使浏览器能够正确显示请求结果；
4. 具备日志功能，能够打印每个请求的来源 IP、端口号、HTTP 命令行等信息和请求文件的结果到控制台

2.2 进阶需求

1. 抵御路径遍历攻击；
2. 提供良好、完整的异常处理机制。

3 系统设计

应该设计一个 `HttpServer` 类作为主类，做一些初始化、善后的工作，并在给定的地址上监听，每收到一个请求，就交由 `Receiver` 处理。

应该设计一个 `ServerUtils` 类来读取配置文件、提供配置信息。

应该设计一个 `Receiver` 类处理收到的请求，包括将请求的读写流抽出来，分别交给 `Request` 类和 `Response` 类。

应该设计一个 `Request` 类根据请求的 `InputStream` 来分析请求体，并保存下来分析结果。

应该设计一个 `Response` 类根据 `Request` 分析的结果定位文件、构造回复、返回回复。

各模块之类的关系如图 1 所示：

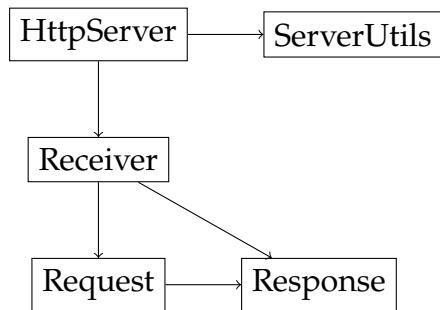


图 1: 模块关系图

4 系统实现

4.1 HttpServer

4.1.1 成员变量

有成员变量 `serverSocket`，为监听套接字。

4.1.2 main() 方法

`HttpServer.main()` 方法首先尝试调用 `ServerUtils.load()`，初始化配置。这些配置通过 `ServerUtils` 相应 Getter 访问。

因为本程序没有 GUI，所以通过 `ctrl+c` 来结束程序，因此需要捕获结束信号。`HttpServer.main()` 接下来通过 `Runtime.getRuntime().addShutdownHook()` 来截获 `ctrl+c` 信号，当用户试图 `ctrl+c` 时，先打印出关闭服务器提示，接下来检查 `serverSocket` 的关闭状态，如果没有关闭就关闭它。注册完控制程序以后，打印关闭服务器的提示信息。

注册完关闭程序以后创建监听在 `ServerUtils` 中给出的地址的套接字，赋给 `serverSocket`。然后在死循环里试图进行 `serverSocket.accept()`，若无请求到达自然在阻塞状态，若有请求

就根据请求的 socket 创建一个 Receiver，新开一个线程执行 Receiver.run()。

这些方法中语句可能出现的异常都用 try-catch 语句捕捉并打印提示信息。

4.2 ServerUtils

4.2.1 成员变量

有 localIp、port、basePath 三个静态变量，分别代表本 Web 服务器监听的 IP、端口号、基路径。不支持热更换配置。

4.2.2 load() 方法

Web 服务器的配置都存放在 config.properties 中。properties 是 Java 常用的配置文件，一行一个 key=value。

ServerUtils.load() 方法首先读取 config.properties 到 localIp、port、basePath 三个变量中。读取的过程中对 basePath 的合法性进行校验，如果其不存在或者是不是目录，校验都不通过。

这些方法中语句可能出现的异常都用 try-catch 语句捕捉并打印提示信息。

4.3 Receiver

4.3.1 成员变量

有成员变量 socket，为待处理的请求套接字。

4.3.2 构造函数

构造函数接收一个 socket，保存到自己的成员变量中。

4.3.3 run() 方法

考虑到 Receiver 实现了 Runnable 接口，故应 override Receiver.run() 方法。它首先取得 socket 的 inputStream 和 outputStream，根据 inputStream 构造 Request 对象，传入请求的 IP 地址和端口号调用 Request.parse() 方法进行分析；然后根据 outputStream 构造 Response 对象，传入分析完毕的 Request 对象调用 Response.constructResponse() 以构造返回，最后调用 Response.fillResponse() 写入响应。

Receiver.run() 方法最后调用 Receiver.socketClose() 来关闭请求套接字。

4.3.4 socketClose() 方法

Receiver.socketClose() 方法检查请求套接字，如果其存在且未被关闭就关闭它。

这些方法中语句可能出现的异常都用 try-catch 语句捕捉并打印提示信息。

4.4 Request

4.4.1 成员变量

inputStream 代表请求套接字的读入流,parseSuccess 标记着识别请求是否成功,method、uri、httpVersion 和 HTTP 请求对应，分别为请求的方法、URI、协议。

4.4.2 构造函数

把传入的读取流保存到成员变量。

4.4.3 parse() 方法

Request.parse() 首先读取读入流的第一行，按空格 split 成三部分，分别作为 method、uri、httpVersion。如果 split 失败则报告识别错误并返回。特别注意 uri 要按照 UTF-8 解码，否则定位本地中文文件名文件会出问题。

按照一定的格式打印出日志头，随即忠实地打印出请求的其他部分（包括请求头）作为日志。

只支持 HTTP 请求。

这些方法中语句可能出现的异常都用 try-catch 语句捕捉并打印提示信息。

4.5 Response

4.5.1 成员变量

outputStream 代表请求套接字的写入流。statusCode 是将返回的状态码，会根据这个构造返回。willFile 是期待的返回文件。

4.5.2 构造函数

把传入的写入流保存到成员变量。

4.5.3 `constructResponse()` 方法

`Response.constructResponse()` 方法首先判断请求分析是否成功，不成功或者请求方法不是 GET 都设置状态码为 400，意为 bad request。将请求 URI 正规化以后和基路径拼接再正规化就能够避免路径遍历攻击，从而得到了 willFile。

如果 willFile 是文件夹，根据 Web 服务器的常见实现，应该尝试返回文件夹下的 `index.html` 文件。如果 willFile 不存在，则设置状态码为 404，意为 not found。

其他情况状态码为 200，即 OK。

4.5.4 `fillResponse()`

`Response.fillResponse()` 方法根据状态码来构造不同的返回。如果不是 200，就直接返回设计好的报文。如果是 200，就先写入一些报文头，计算文件的类型、长度并写入报文，然后将文件写入报文中。

文件类型的计算直接使用 `Files.probeContentType()` 方法。

这些方法中语句可能出现的异常都用 try-catch 语句捕捉并打印提示信息。

5 系统测试与结果说明

配置 config.properties 和启动以后的结果如图 2 所示。

```
LocalIp=10.12.51.195
Port=9213
BasePath=/home/poorpool/Documents/HUST/计算机网络/实验/lab1/testfiles/
HttpServer x
/usr/lib/jvm/java-13-jdk/bin/java -javaagent:/home/poorpool/.local/
Start HttpServer, located on 10.12.51.195:9213, base path is /home/p
Press ctrl+c to shutdown
Listening on 10.12.51.195:9213
```

图 2: 配置与启动

服务器将展示图 3 中的内容，这是我的静态个人博客，包括多种类型的文件。

打开提供的 URL 进入主页面如图 4 所示。

这个过程中，请求和部分响应报文如图 5 所示。

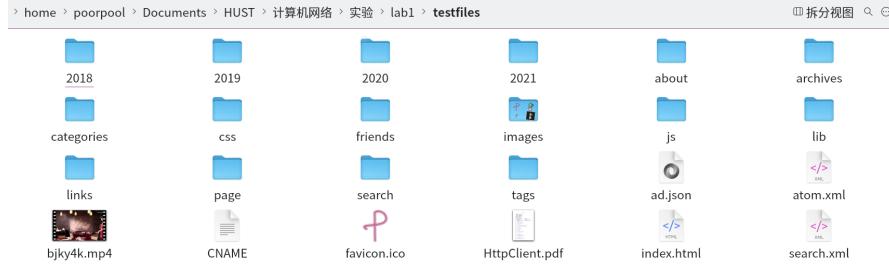


图 3: 服务器的基路径内容



文章

标签

Android C++ Java Linux Spring Web go lisp shell tastyPOST vue 乱搞 前端 动态规划——决策单调性 动态规划——区间dp 动态规划——斜率优化 动态规划——普通dp
动态规划——树形dp 动态规划——状压dp 图论——LCA 图论——二分图 图论——割点与桥 图论——基环树 图论——拓扑排序 图论——最短次短K短路 图论——树论 图论——点分治
图论——网络流 多线程 多进程 字符串——AC自动机 字符串——KMP 字符串——后缀数组 字符串——编码 底层 开发笔记 思维 垃圾 数学——FFT, NTT, 多项式 数学——博弈论 数学——反演和筛法 数学——微积分 数学——数论 数学——概率与期望 数学——生成函数 数学——线性基 数学——计算几何 数据库 数据结构——LCT 数据结构——RMQ 数据结构——ST表、倍增、RMQ 数据结构——主席树 数据结构——哈希表 数据结构——线段树 数据结构——虚树 埃波那契堆 杂记 无形dp 模拟退火 游记 笔记 算法竞赛 编译原理 网络 联创 计算机网络 贪心 搞表 错 非传统题 面向对象

最新的文章

2021-08-20	上了github actions
2021-03-12	牛客和leetcode 及类似OJ的部分题目刷题记录
2021-03-01	华科Java笔记1
2021-02-12	CS144 Lab记录
2021-02-12	面试突击人
2021-02-10	模拟退火
2021-01-27	《编译原理》第二版（龙书）学习笔记
2020-12-20	败亦可喜
2020-12-06	HUST ACM 2020 新生赛补题

图 4: 服务器主页面

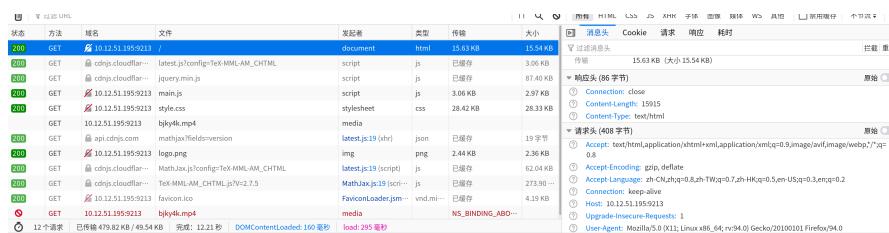


图 5: 请求展示与响应报文

在服务器侧，可以看到图 6 所示的日志。

```
[Tue Nov 16 21:05:19 CST 2021]
Received request from 10.12.51.195:33122 for /js/main.js {
    GET /js/main.js HTTP/1.1
    Host: 10.12.51.195:9213
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:94.0) Gecko/20100101 Firefox/94.0
    Accept: */*
    Accept-Encoding: gzip, deflate
    Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
    Referer: http://10.12.51.195:9213/
}
Want /home/poorpool/Documents/HUST/计算机网络/实验/lab1/testfiles/js/main.js
Get file ok, type application/javascript
socket closed
```

图 6: 打印请求

如图 7 所示，可以展示图片。

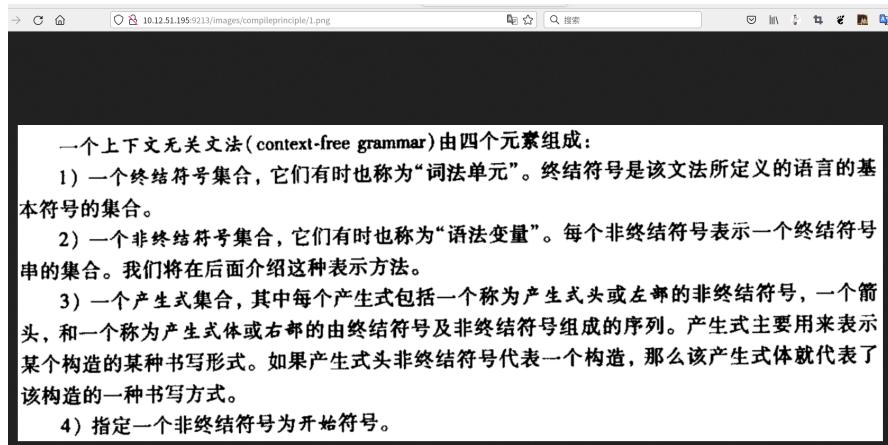


图 7: 图片显示

如 8 所示，也支持视频的传输。

试图访问不存在的页面时，会展示 404，如 9 所示。

如果配置得当，那么在其他的设备上也可以访问网站，如图 10 所示。

6 改进方向

线程池 每收到一个请求，就开一个线程去处理，显然是一种野蛮、粗糙的做法。在请求量大时，合理的做法是开一个线程池，限制可以执行的线程数，同时不至于频繁创建关闭线程。

硬编码 在代码里硬编码返回报文的一部分也是不好的做法。可以考虑将返回报文的模板保存到特殊的文件夹，根据模板拼接报文。

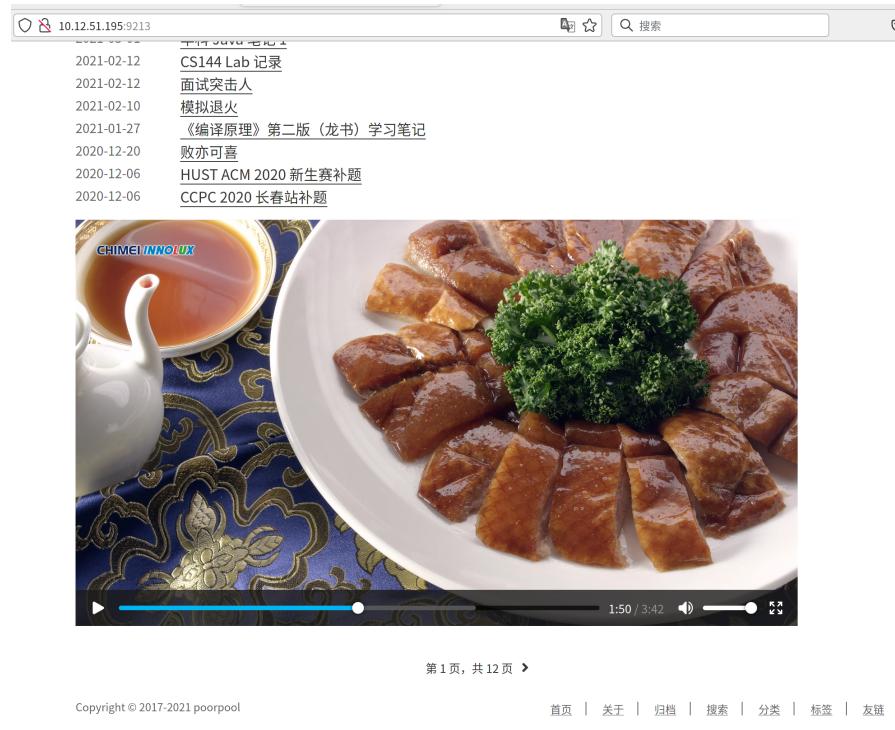


图 8: 播放视频

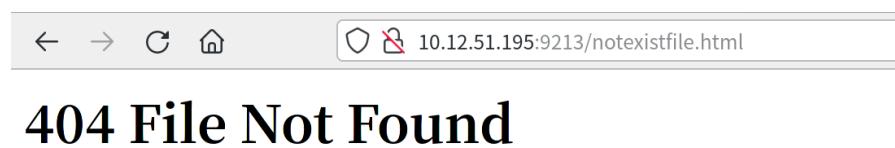


图 9: 访问不存在页面



图 10: 其他设备访问

视频分片 视频虽然能播放，但是不能随意拖动，只能看已缓冲的部分。将来可以加入对视频分片的支持。

多线程日志混乱 直接打印到控制台会引发多线程日志混乱打印的问题，应该合理使用日志库。

7 总结

本次实验完成了一个 Web 服务器，总体完成度不错，已经具备良好的可用性。但在大规模请求、多线程部分还有待改善。从这次实验中，我强化了 TCP、HTTP 的概念，进一步了解了 socket 编程的步骤，感受到了计算机网络的魅力。