**《计算机网络》课程设计**

**周进度报告**

**第二周：实现 HEAD、GET 和 POST 方法**

**学 号 3020244294**

**姓 名 石子跃**

**学 院 智能与计算**

**专 业 网络安全**

**年 级 2020**

**任课教师 仇超**

**2022年 3月 23日**

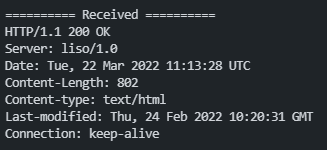
1. 设计HEAD、GET 和 POST 方法
2. Get方法：根据http请求行里的url在服务器中去搜索文件，更改响应报文中的响应状态行Content-Length，Content-Type，最后将文件流send到客户端即可。
3. Head方法：与上述Get方法不同，只需根据url中的文件地址去搜索文件，此方法被用来获取请求实体的元信息而不需要传输实体主体（entity-body）。此方法经常被用来测试超文本链接的有效性，可访问性，和最近的改变。更改报文中的响应状态行，Content-Length，Content-Type，Date，Last-modified等
4. Post方法：被用于请求源服务器接受请求中的实体作为请求资源的一个新的从属物。本次实验只需要echo回去post请求本体就可，但是我实现了echo回需要post的数据。
5. 支持 5 种 HTTP 1.1 出错代码：400，404，408，501，505。准确判别客户端消息，并发回响应消息。400 = "HTTP/1.1 400 Bad request " 404 = "HTTP/1.1 404 Not Found " 501 = "HTTP/1.1 501 Not Implemented " 505 = "HTTP/1.1 505 HTTP Version not supported”，查看httprfc手册。
6. 持久连接（persistent connection）：长连接的退出动作由客户端发出，也就是利用connection头域可以产生终止连接的信号，在客户端发出带有close的一条指令后，服务器也会返回响应状态connection：close，在完成响应报文send后，客户端便不可再想此连接提出任何新请求。
7. 创建简化的日志记录模块，记录格式化日志。
   1. 以下是apache的access\_log日志格式样例：
      1. 127.0.0.1 - - [21/Mar/2022:20:55:14 -0700] "GET / HTTP/1.1" 304 -
      2. 127.0.0.1 - - [21/Mar/2022:20:55:48 -0700] "GET / HTTP/1.1" 200 45
      3. 127.0.0.1 - - [21/Mar/2022:20:55:48 -0700] "GET /favicon.ico HTTP/1.1" 404 196
      4. 127.0.0.1 - - [23/Mar/2022:06:21:15 -0700] "GET / HTTP/1.1" 200 45
      5. 127.0.0.1 - - [23/Mar/2022:06:21:15 -0700] "GET /favicon.ico HTTP/1.1" 404 196

于是自己定义新的格式如下：

（时间，host，method，url，响应代码）

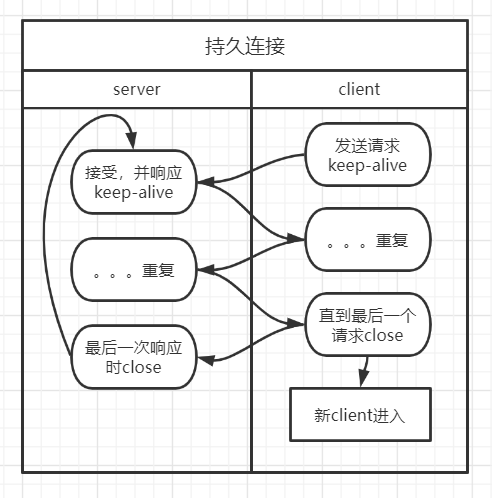
# 具体实现实现

1. GET方法，HEAD方法，POST方法集成在response.c中，将server需要使用的命令在response.h中。相比于我最初把这三个方法分别编写.c.h文件，只用一个.c文件更好，避免了文件之间函数的重复引用，更简单明了。
2. GET：客户端会传递url来访为相应位置的文件，于是编写以下函数：
   1. get\_file\_size(char\* file\_name) 获取响应文件大小
   2. get\_file\_type(char\* filename) 获取文件类型
3. HEAD：

这是一个head请求的响应格式，观察得到Content-type，Content-length在get中有所实现，但Date，Last-modified首部行还未实现，于是编写函数如下：

* 1. getime() 获取当前时间
  2. getlast(char \*path) 获取文件最后修改时间

1. Post：起初实现的可以根据Content-Length，返回已post的函数，考虑到首部行中传输的格式为ASCII，于是编写了一个int char2int(const char\* str)函数，来读出文本长度，最后，echo回客户端
2. 持久连接：大致实现如下：



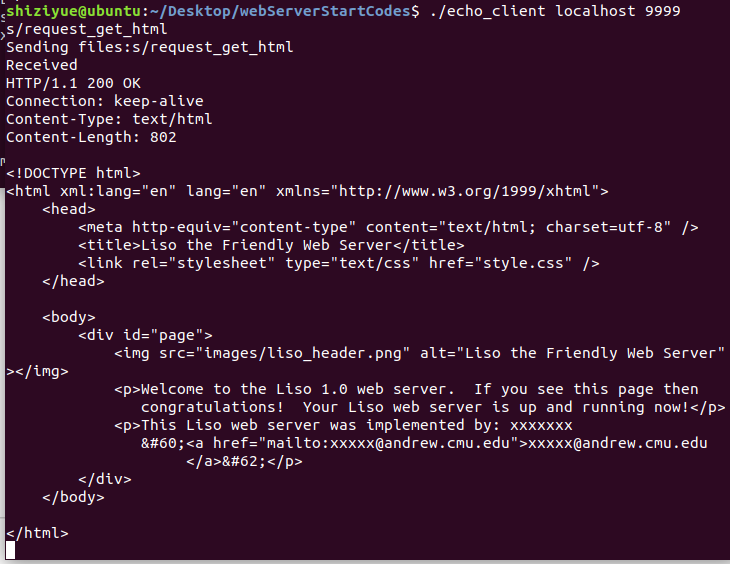
需要编写函数：

addalive(Request \*request,char\* buff) 用于当服务器接收到带有connection：keep-alive的请求头部时进行正确的操作

isalive(char\* buff) 用于客户端正确识别服务器是否支持继续持久连接

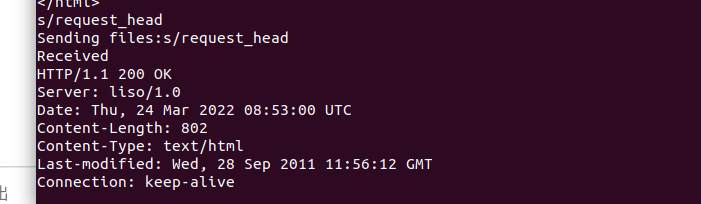
**三、实验结果及分析**

1. Get



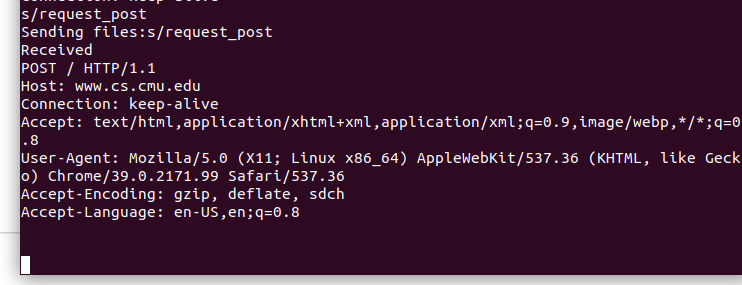
结果和上文描述一致，同时客户端可继续向server发送请求。也实现了持久连接。

1. Head：

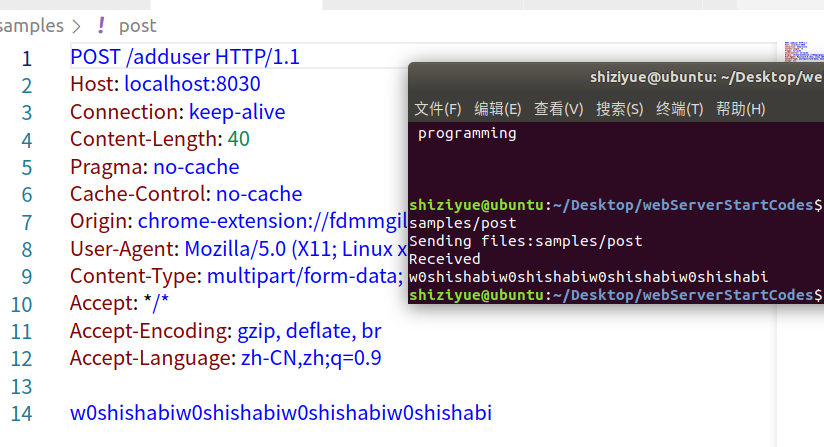


返回和上文一中叙述一致。

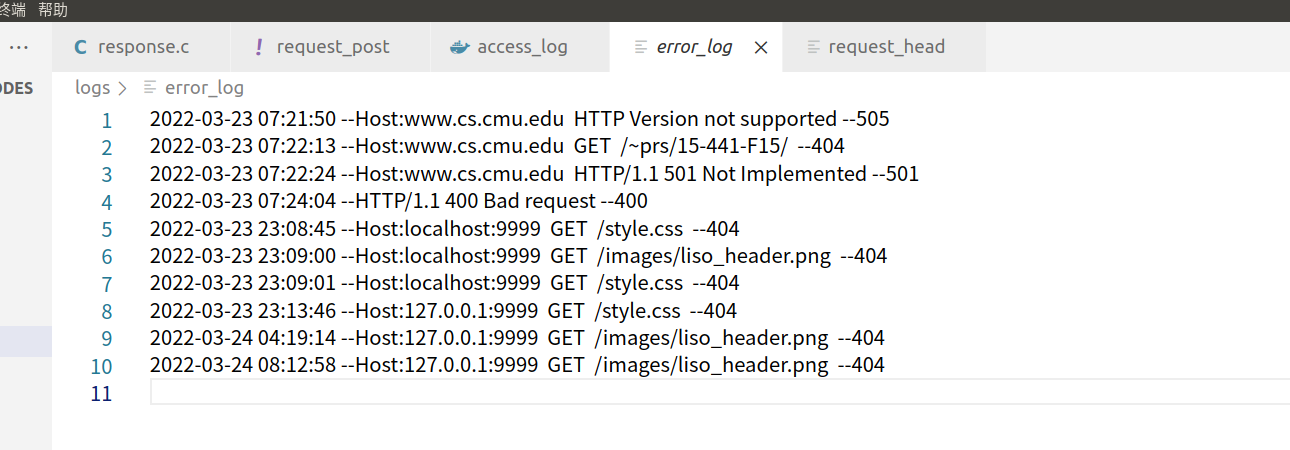
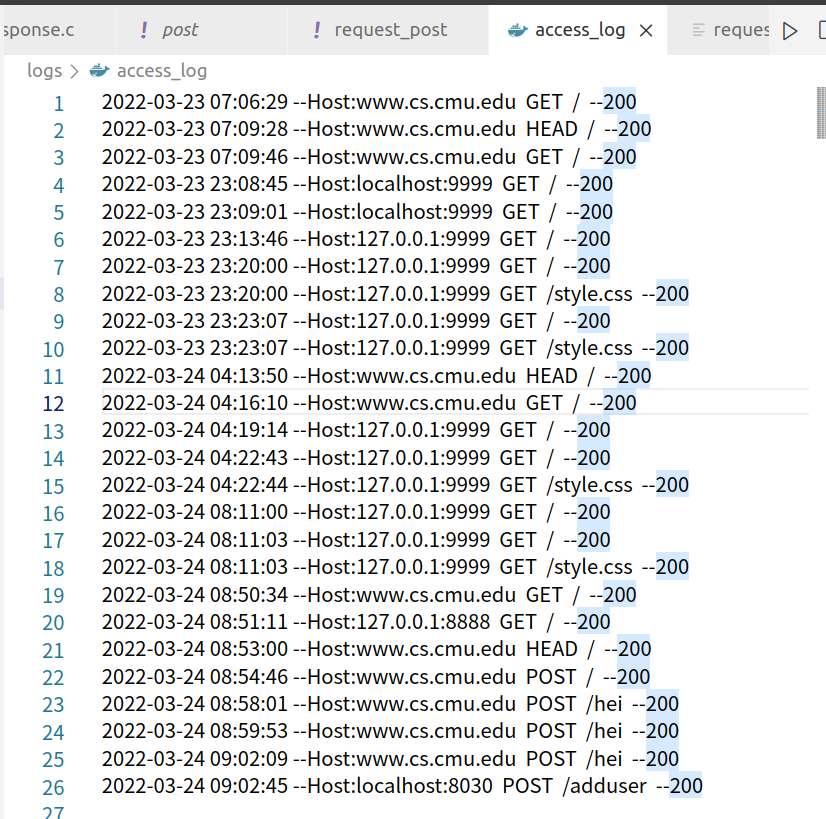
1. Post：



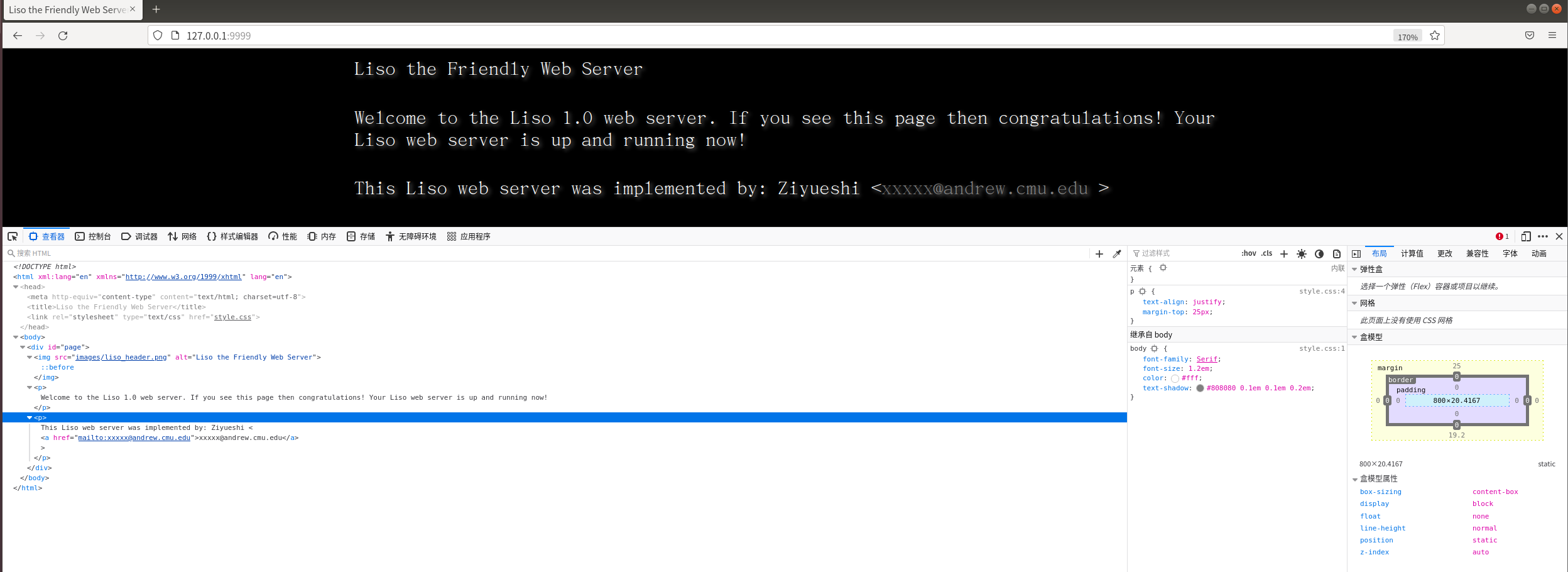
相比于本次实验要求的我还新增了一个功能，即可将post的数据返回



1. Log日志



1. 最后浏览器运行测试



此项目前存在以下问题：

* 1. 不知道浏览器何时会发送close的指令
  2. 不清楚png图片的具体传输方法，虽然看出可以传，但是好像颜色不一样

# 四、进度总结

**本周任务完成表**。

在“完成”“没完成”列对应打“√”

|  |  |  |  |
| --- | --- | --- | --- |
| 本周任务  要求 | 完成 | 没完成 | 备注 |
| 1、GET, HEAD, POST | √ |  |  |
| 2、支持 5 种 HTTP 1.1 出错代码 | √ |  |  |
| 3、创建简化的日志记录模块，记录格式化日志 | √ |  |  |
| 4、功能测试 | √ |  |  |

**上周任务改进表**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 上周任务 | 改进内容 | 备注 |
| 1 |  |  |  |