**《计算机网络》课程设计**



**周进度报告**

**第一周 实现简单的Echo Web Server**

**学 号 3020244294**

**姓 名 石子跃**

**学 院 智能与计算**

**专 业 网络安全**

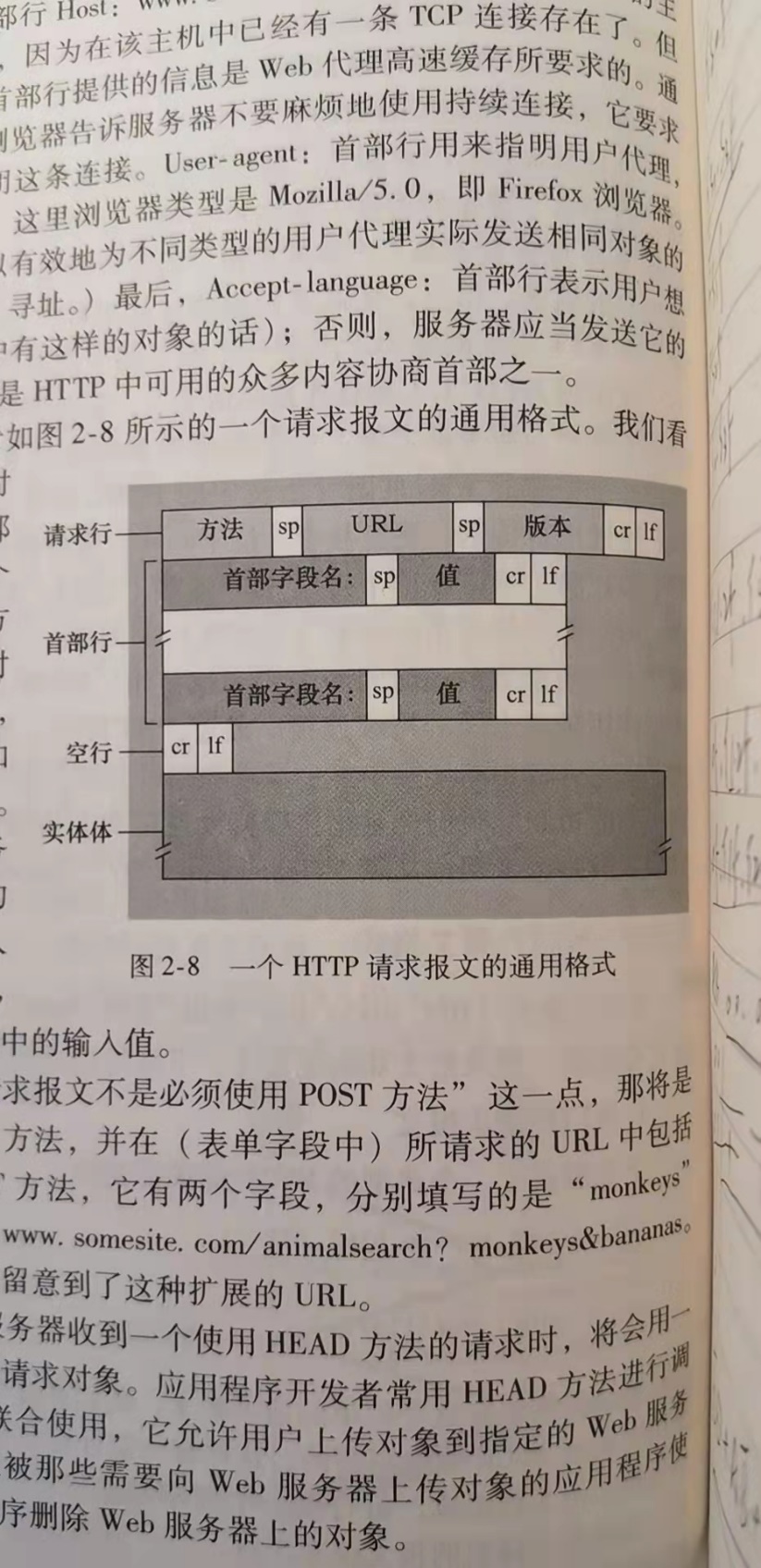
**年 级 2020**

**任课教师 仇超**

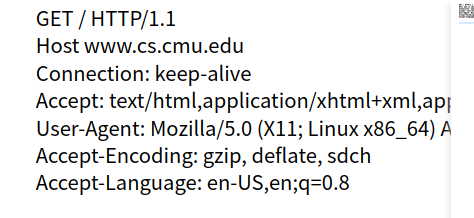
**2022年 3月 19日**

# 一、协议设计

1.协议头部、协议规则：



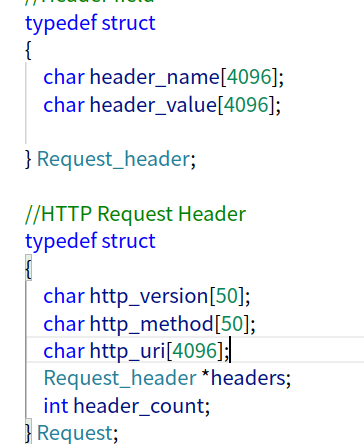
样例：



上图GET / HTTP/1.1 分别对用方法，URL ，版本

2.数据结构

其中请求行与其他请求头部不一样要单独拿出方便解析。在parse.h中定义http请求的数据结构



其中http\_version,http\_method,http\_url分别对应分别对应协议规则中的版本，方法，URL。这是http请求行的结构

3.消息解析方法

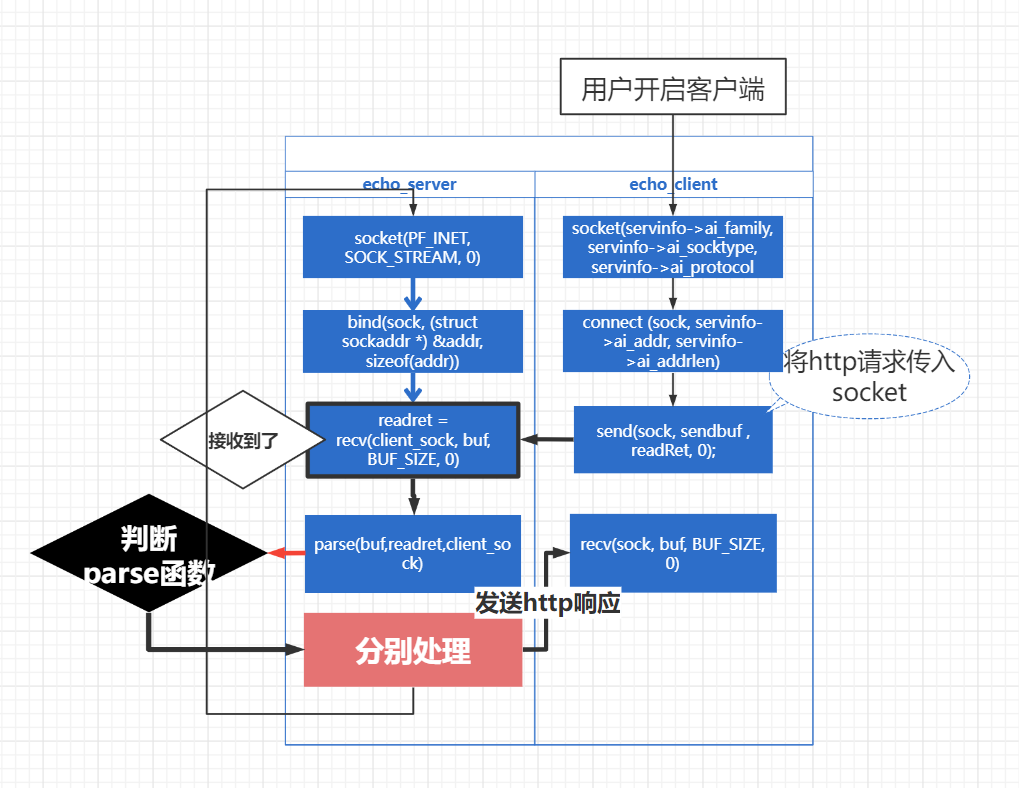
使用正则表达式匹配。

将请求行单独匹配后，匹配请求头部，后面的结构相似。在parser.y中进行匹配，当识别一个request\_header后，再去找下一个。若最终不符合http请求规范则直接返回NULL。运行yacc处理语法文件，生成一个解析器  
说明语法：  
-编写.y语法文件  
-编写一个词法分析器来处理输入并将标记传递给处解析器。这里可以使用Lex完成。  
-编写一个函数，使用yyparse()开始解析  
-编写错误处理例程 如yaerror()  
编译yacc生成的代码以及其他相关源文件  
将目标文件链接到适当的可执行解析器库

# 二、协议实现

Socket是应用层与TCP/IP协议族通信的中间软件抽象层，它是一组接口。在设计模式中，Socket其实就是一个门面模式，它把复杂的TCP/IP协议族隐藏在Socket接口后面，对用户来说，一组简单的接口就是全部，让Socket去组织数据，以符合指定的协议。

实现原理如下



服务器端先初始化Socket，然后与端口绑定(bind)，对端口进行监听(listen)，调用accept阻塞，等待客户端连接。在这时如果有个客户端初始化一个Socket，然后连接服务器(connect)，如果连接成功，这时客户端与服务器端的连接就建立了。客户端发送数据请求，服务器端接收请求并处理请求，然后把回应数据发送给客户端，客户端读取数据，最后关闭连接，一次交互结束。

这次的代码使用本地ip localhost 127.0.0.1 端口号9999，如果服务器bind失败会提示socket bind failed，如果客户端connect失败会提示connect，在客户端收到一次来自服务器的响应后，echo\_server就会关闭socket，而server还将继续监听这个端口。

# 三．实验过程

1.更改example，parse等文件对http请求案例进行正确解析

  首先观察http请求协议的标准格式，学习并根据parser.y，yacc的匹配规则和语法，修改代码使得parse(buf,readRet,fd\_in)函数可以正确解析buf中的字符流。

example.c调用open(),read()函数进行对文件的读，将其字符流保存在buf中。

parse()首先会判断传入的数据中是不是有两个CRLF，这是判断http请求报文最基础的。然后会进入yyparse()，这就是根据parser.y语法文件，所执行的语法解析器。

   在request\_header这条规则下，添加它的有递归式使得可以一直连续的识别header。(提示：每个header长得都一样）

同时还要注意对request-headers的动态分配内存以防请求头过多导致溢出。

2.完善好parse函数后将其移植到echo\_server中，使服务器可正确解析出来自客户端的请求

   客户端和服务端全部使用buf来缓存数据，当服务器接收到信息后进行解析，再通过send函数返还echo给client，就可开始第三步。echo\_client如何读取文件可以直接从example.c找到。其中由于server调用了parse函数，还需要在makfile中进行相应修改。在进行编译时，将与解析有关的.o文件和server进行连接，生成可执行文件。

3.服务器对不同请求做出不同响应

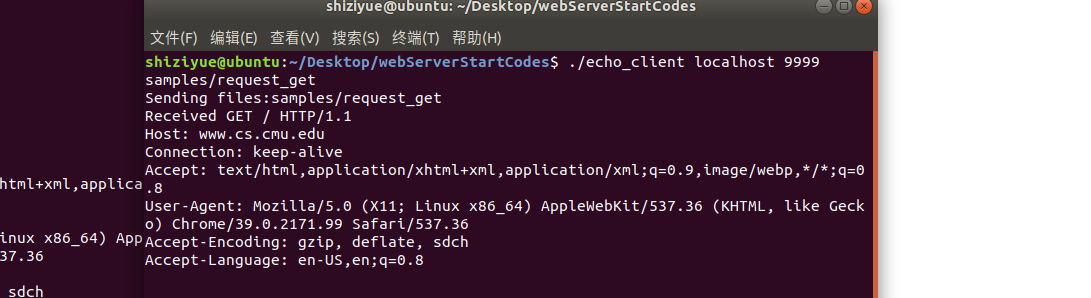
   阅读http手册，和RFC手册，了解不同状态码对应的含义，在parse中对畸形的请求进行筛选，在echo\_server中对为未部署和已部署的request\_method进行不同的响应，比如501，400等。我目前的做法是畸形的请求交给parse.c处理，未部署的请求放在eecho\_server中处理。

# 四、实验结果及分析

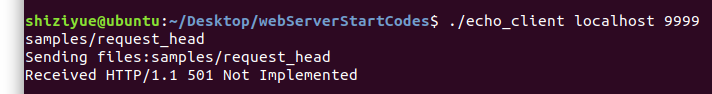
测试所实现任务的功能和性能，并对结果进行分析。需要针对考察点逐一展开。

实验结果：

1. 如果收到客户端发来的是 GET, HEAD 和 POST 方法，则 echo 回去，即重新封装（encapsulation）消息并返回给客户端。

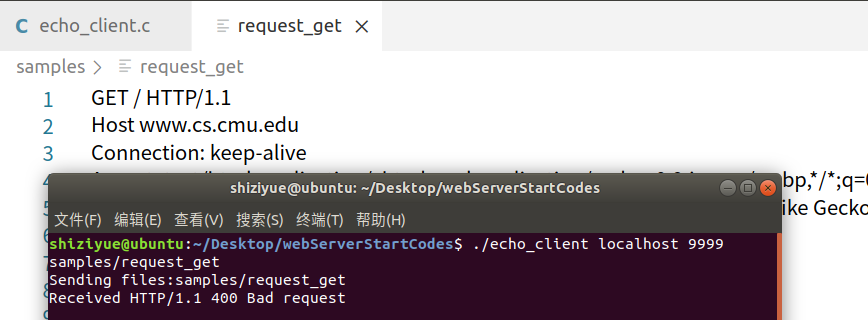


1. 如果收到客户端发来的是除 GET, HEAD 和 POST 以外的其它方法，服务 器 并 没 有 实 现 ， 则 需 要 返 回 响 应 消 息 “ HTTP/1.1 501 Not mplemented\r\n\r\n”。



3） 如果收到的客户端消息的格式错误，应能够识别出来，并返回错误代码

为400的HTTP 响应消息“HTTP/1.1 400 Bad request\r\n\r\n”



但是也遇到一个问题，在服务器第一次返回400之后，在输入一个正确的http请求，parse还会在上次错误的地方继续，但是观察buf已经更新，很神奇。

# 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、阅读HTTP/1.1的标准文档RFC2616 | √ |  |  |
| 2、搭建编程环境 | √ |  |  |
| **3、**熟悉Socket编程方法； | √ |  |  |
| 4、掌握lex和yacc正确解析消息（message）的方法 | √ |  |  |
| 5.1实现简单的echo web server。Echo GET, HEAD, POST | √ |  |  |
| 5.2 响应没有实现的方法 | √ |  |  |
| 5.3 响应错误的方法 | √ |  |  |
| 6、功能测试 | √ |  |  |

**上周任务改进表**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 上周任务 | 改进内容 | 备注 |
| 1 |  |  |  |
| 2 |  |  |  |
|  |  |  |  |
|  |  |  |  |