

Project 1 mini web server report

Shuhao Sun UID: 304653090

Shuangyu Li UID: 805035359

Introduction

In this project , a mini web server was developed in C/C++ . The purpose of this project is to help us know the basics of socket programming and HTTP protocols. It could help us understand more about how a browser and a web server work through HTTP messages.

Server design

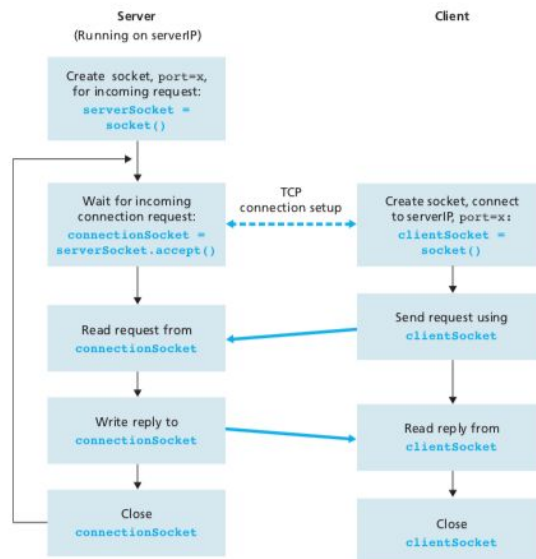


Fig: socket programming

We use an object-oriented design with C++ and C libraries to build the server. In part A, we created a least-working server. The core part is to use `create_socket()` method

and set up a socket at server side, which can receive message from clients. After that we used `server_listen()` method which in a while loop accepts the TCP requests and dumps message to console. In part B, we change the core loop in `server_listen()` method to be a complete web server which can send responses back. Within the while loop of `server_listen()` method, the HTTP request messages from socket are parsed by `parser()` method to get filename from the request message. The typical HTTP request we got is shown below. Then, we used `get_contentType()` method to get the corresponding file type for HTTP response header later. The last part is to build HTTP response with the requested file and send it back the content. If the file requested is not found, `send_404()` method will be called to send proper HTTP response.

```
GET /a.html HTTP/1.1
Host: localhost:5555
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Upgrade-Insecure-Requests: 1
Cookie: _xsrf=2140a92dbf1c7f95cb912e2df5168e6b3e96831ceac11524447322; username-localhost-8889="211:010:1524331807|23:username-localhost-8889|44:MMU0NWU4M2I1MGIxNDQyZTk4MzliYzgzYzhjYjgwYjk=|eb64408ea37a3e6038116c4cacc1d3c7d845b49a65eb34950e250b61dfbc039a"; username-localhost-8888="211:010:1524256657|23:username-localhost-8888|44:NTY1MzU5NjFiNDg4NDEzOWE2NWY1YjNkMWFmZjY1YzY=|5b5c729fca65c2983da258afff3a558cd82f6cfa82b245be3f2974dbb1d9b47b"
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_4) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/11.1 Safari/605.1.15
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Difficulties

1. Case sensitivities

The filename we got should be case insensitive, which is a difficulty we face. We can't simply open the files according to the file names. We implemented a new way which

instead search the whole working directory and compare file names with the file name we want one by one.

2. String comparison

We made naive mistake when try to compare file extension and desired file type. Because we just printed out the file extension to make sure it follows our thought but ignore the empty space followed by the string variable. Therefore, we spent some time to print all the variable types and size to make sure to find this bug.

3. Read file to socket

We choose to use C++ io library to the binary io. The difficulty is that socket interface uses the simple file descriptor structure from C but the C++ uses different file objects. To solve this problem we used a buffer as intermediate to read binary file in C++'s method and then write to socket in C's method.

User Manual

All the C/C++ code are included in the *server.cpp*. The makefile is written for this project. Please run the code below to test

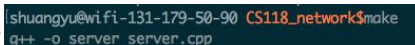
```
>> make
```

```
>> ./server <port_number>
```

Then input ***http://localhost:<port_number>/<file_name>*** to get the desire file/image/page.

Sample Output







1. Compile and run normally
2. HTTP request



```
shuangyu@wifi-131-179-50-90 CS118_network$ make  
g++ -o server server.cpp
```

localhost:10000/a.html

Index of /courses/cs118/project1-exampl

	Name	Last modified	Size	Description
	Parent Directory	-	-	-
	Makefile	2018-04-09 13:55	359	
	README	2018-04-09 13:55	78	
	client.c	2018-04-09 13:55	2.1K	
	server.c	2018-04-09 13:55	2.0K	

Apache/2.4.7 (Ubuntu) Server at metro.cs.ucla.edu Port 80

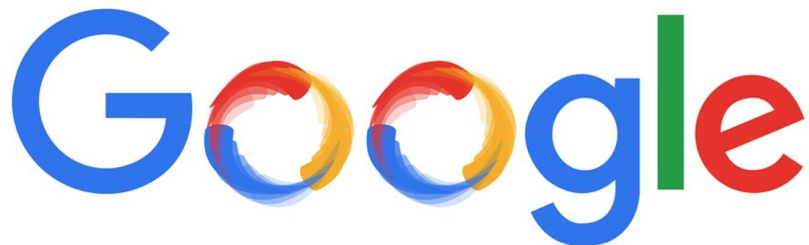
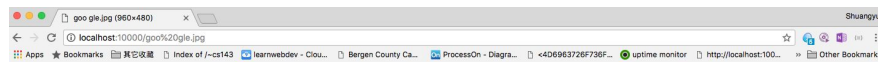
```
GET /a.html HTTP/1.1
Host: localhost:10000
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Upgrade-Insecure-Requests: 1
Cookie: __srf=2140a92dbf1c7f95cb912e2df5168e6b3e96831ceac1524447322; username=localhost-8889="211:0110:1524331807|23:use
rname=localhost-8889|44:MMU0NNU4M2I1MGIxNDQyZTk4MzliYzgzYzhjYjgwYjk=|eb64408ea37a3e6038116c4cacc1d3c7d845b49a65eb34950e25
0b61dfbc039a"; username=localhost-8888="211:0110:1524256657|23:username=localhost-8888|44:NTY1MzUSNjFiNDg4NDZzOWE2NWY1YjN
kMWFmZjY1YzY=|5b5c729fca65c2983da258afff3a558cd82f6cfa82b245be3f2974dbb1d9b47b"
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_4) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/11.1 Safari/
605.1.15
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

3. Image

```
GET /google.jpg HTTP/1.1
Host: localhost:10000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.139 S
afari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7,zh-TW;q=0.6
```



4. name with space



```
GET /goo%20gle.jpg HTTP/1.1
Host: localhost:10000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.139 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7,zh-TW;q=0.6
```