UNIVERSITY OF CALIFORNIA, LOS ANGELES

CS 118

Project 1 Report

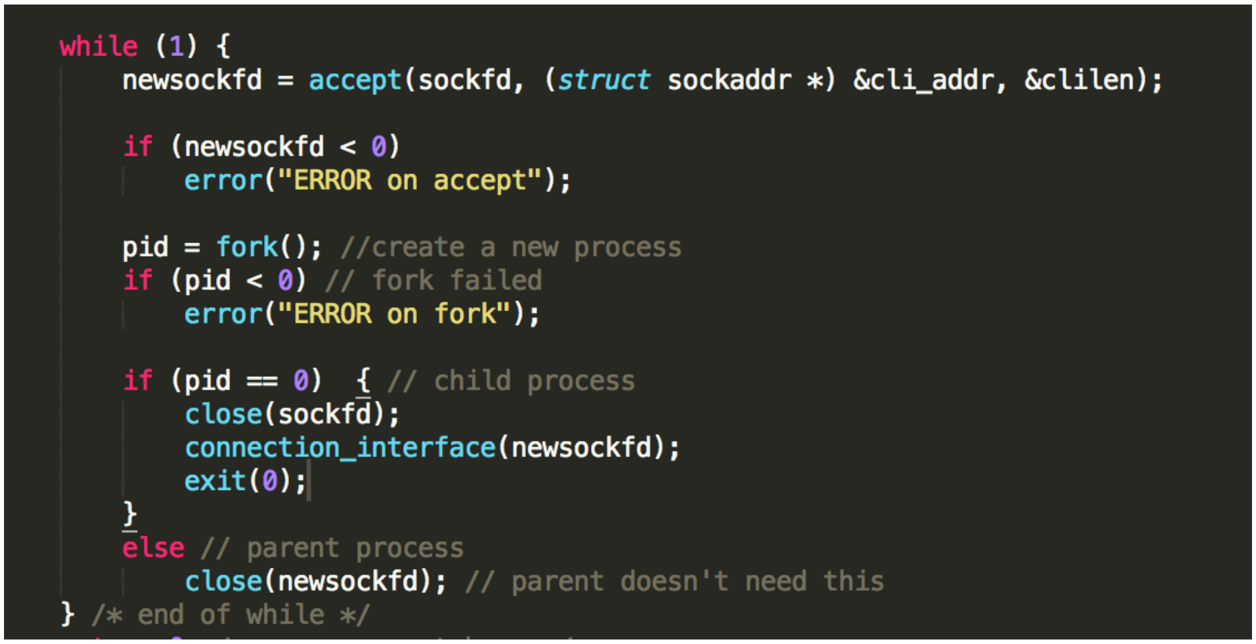
Daxuan Shu
UID: 2**-***-**1

Spring 2018

April 27, 2018

High-Level Description

I used the provided server demo as the skeleton code. When testing the webserver, I used the browser Google Chrome on my laptop. I connected to the server through the port number that I had defined in the console(terminal) when running the executable file. After running the server with a port number, the server will hold and listen up to five requests to build a TCP connection by using listen() function and set the parameter to 5. The client then use the port number provided and the browser to connect to the web server. Each time a client has connected to the server, there will be a fork() function to create a new process for the client. Since the fork() returns the child pid to the parent process and returns 0 to the child process on success. fork() returns -1 to parent process and no child process is created. I can determine which process is running the code by checking the pid value and do the corresponding functions.

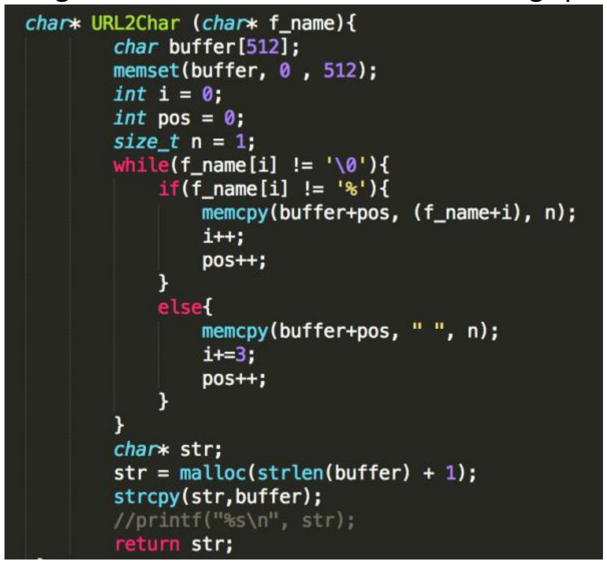


```
while (1) {
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);

    if (newsockfd < 0)
        error("ERROR on accept");

    pid = fork(); //create a new process
    if (pid < 0) // fork failed
        error("ERROR on fork");

    if (pid == 0)  { // child process
        close(sockfd);
        connection_interface(newsockfd);
        exit(0);
    }
    else // parent process
        close(newsockfd); // parent doesn't need this
} /* end of while */
```

The connection_interface() function reads the request message from the socket first and then prints out the request message to the console. Then, it parses the filename and calls the URL2Char() function to deal with the file name including space cases. After that, based on the file name, it checks the server and gives feedback. The make_header()function is used inside the connection_interface() function to help to print out correct header in the reponse messages.

Difficulties:

The most challenging part in this project is to handle request-files whose name including space. I first tried to parse the file_name using the strtok()funtions to get the second tokens in the request message. However, since I used “ ”as the delimiter, I could not deal with the case with space inside the file names. Then, I tried to find the index of the positions that are located at before and end of the file name. Then retrieve the char(name might contain space) out of the buffer. Then call the URL2Char function to decode URL ‘%20’ to “ ”. In this case, my server can recognize the files whose names including spaces.



```
char* URL2Char (char* f_name){
        char buffer[512];
        memset(buffer, 0 , 512);
        int i = 0;
        int pos = 0;
        size_t n = 1;
        while(f_name[i] != '\0'){
            if(f_name[i] != '%'){
                memcpy(buffer+pos, (f_name+i), n);
                i++;
                pos++;
            }
            else{
                memcpy(buffer+pos, " ", n);
                i+=3;
                pos++;
            }
        }
        char* str;
        str = malloc(strlen(buffer) + 1);
        strcpy(str,buffer);
        //printf("%s\n", str);
        return str;
```

Brief manual to compile and run the my source code:

The first step is to generate the executable file by compile the webserver.c source code using terminal: (Suppose in directory containing all source files)

```
gcc webserver.c -o webserver
```

After generating the executable file, running the server with specific port number like:

```
./webserver 1025
```

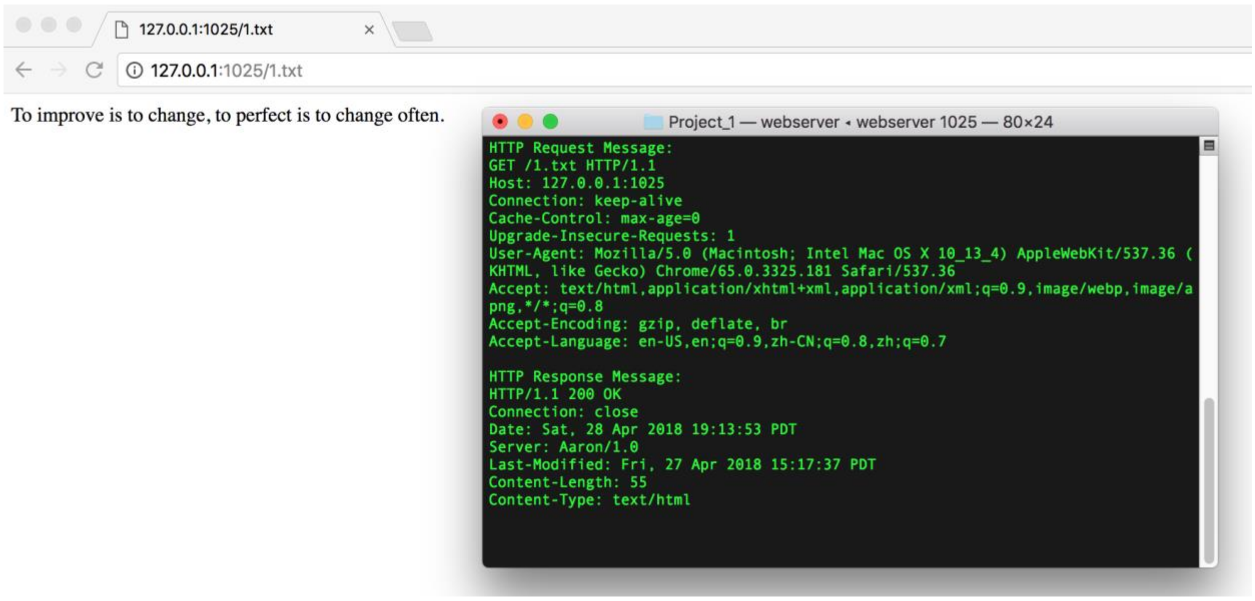
Then the server is running and listening connections.
Then, you should connect to the server from a browser using the URL http://<machinename>:<port>/<filename> and see if it works.
If testing on your own machine, the URL should like:
127.0.0.1:1025/teemo.gif
To request the gif file called teemo.gif

Sample Outputs

I prepared 4 files: 1.txt , Meme.jpeg, Teemo.gif , ab c.txt
Notice that there is a space in the last file “ab c.txt” to test the space file name cases:

Below are the results:

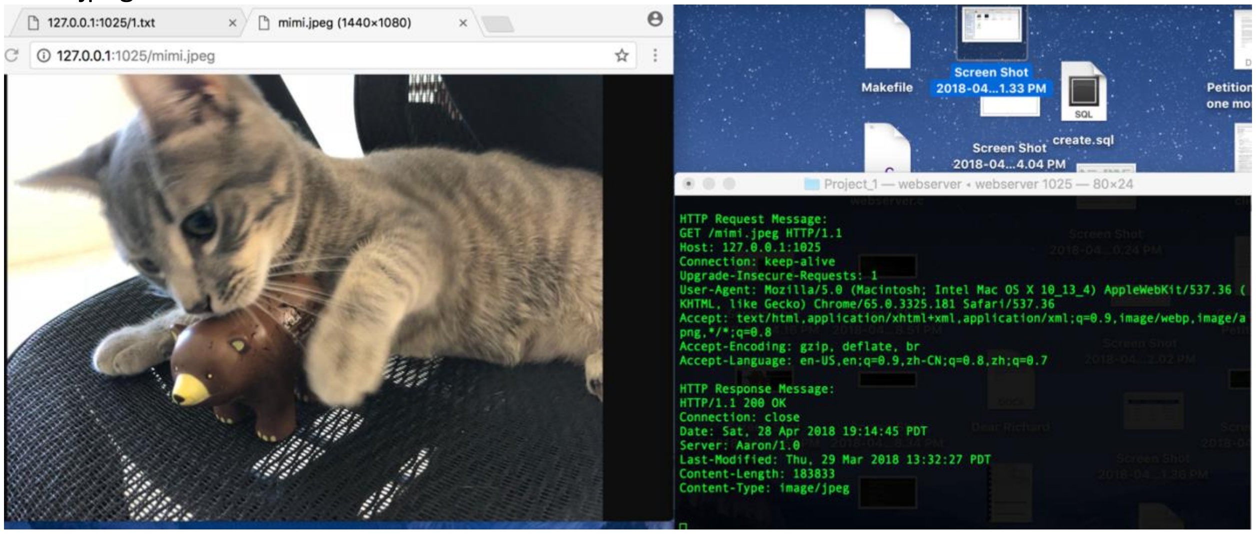
1. 1.txt



To improve is to change, to perfect is to change often.

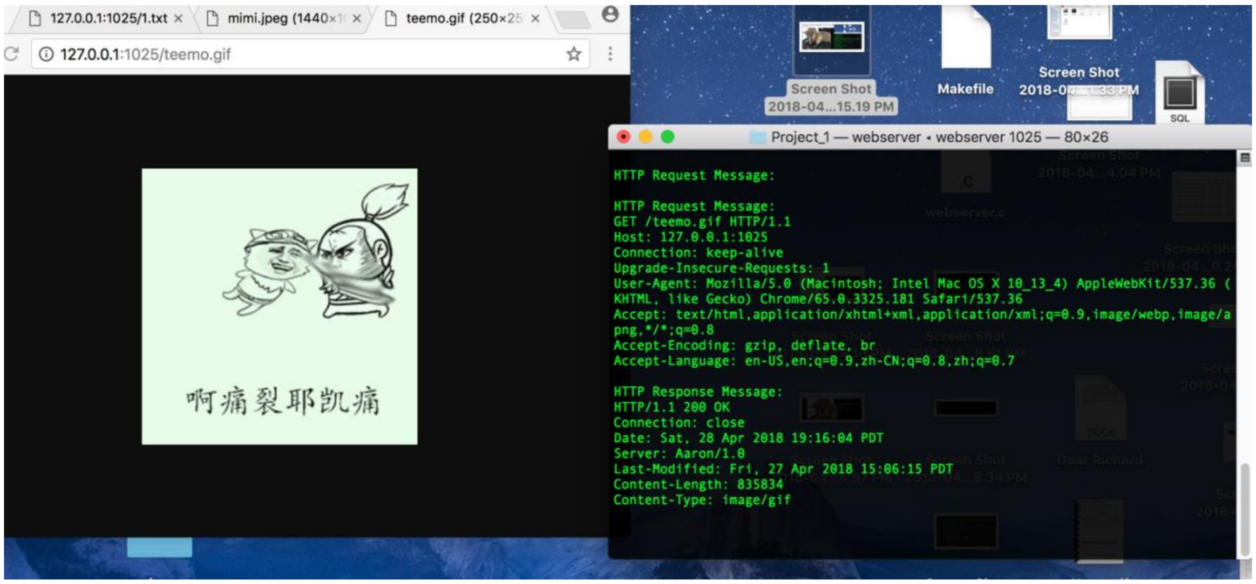
```
HTTP Request Message:
GET /1.txt HTTP/1.1
Host: 127.0.0.1:1025
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_4) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/a
png,*/*;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

HTTP Response Message:
HTTP/1.1 200 OK
Connection: close
Date: Sat, 28 Apr 2018 19:13:53 PDT
Server: Aaron/1.0
Last-Modified: Fri, 27 Apr 2018 15:17:37 PDT
Content-Length: 55
Content-Type: text/html
```

2. Mimi.jpeg


127.0.0.1:1025/1.txt
mimi.jpeg (1440×1080)
127.0.0.1:1025/mimi.jpeg
Makefile
Screen Shot 2018-04...1.33 PM
Petition one mo
Screen Shot 2018-04...4.04 PM
create.sql
Project_1 — webserver ◂ webserver 1025 — 80×24

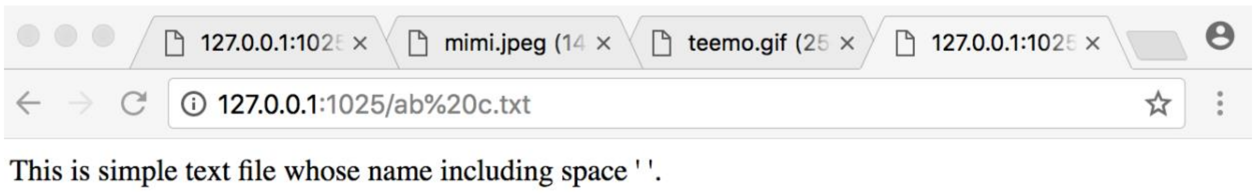
```
HTTP Request Message:
GET /mimi.jpeg HTTP/1.1
Host: 127.0.0.1:1025
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_4) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/a
png,*/*;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

HTTP Response Message:
HTTP/1.1 200 OK
Connection: close
Date: Sat, 28 Apr 2018 19:14:45 PDT
Server: Aaron/1.0
Last-Modified: Thu, 29 Mar 2018 13:32:27 PDT
Content-Length: 183833
Content-Type: image/jpeg
```

3. Teemo.gif

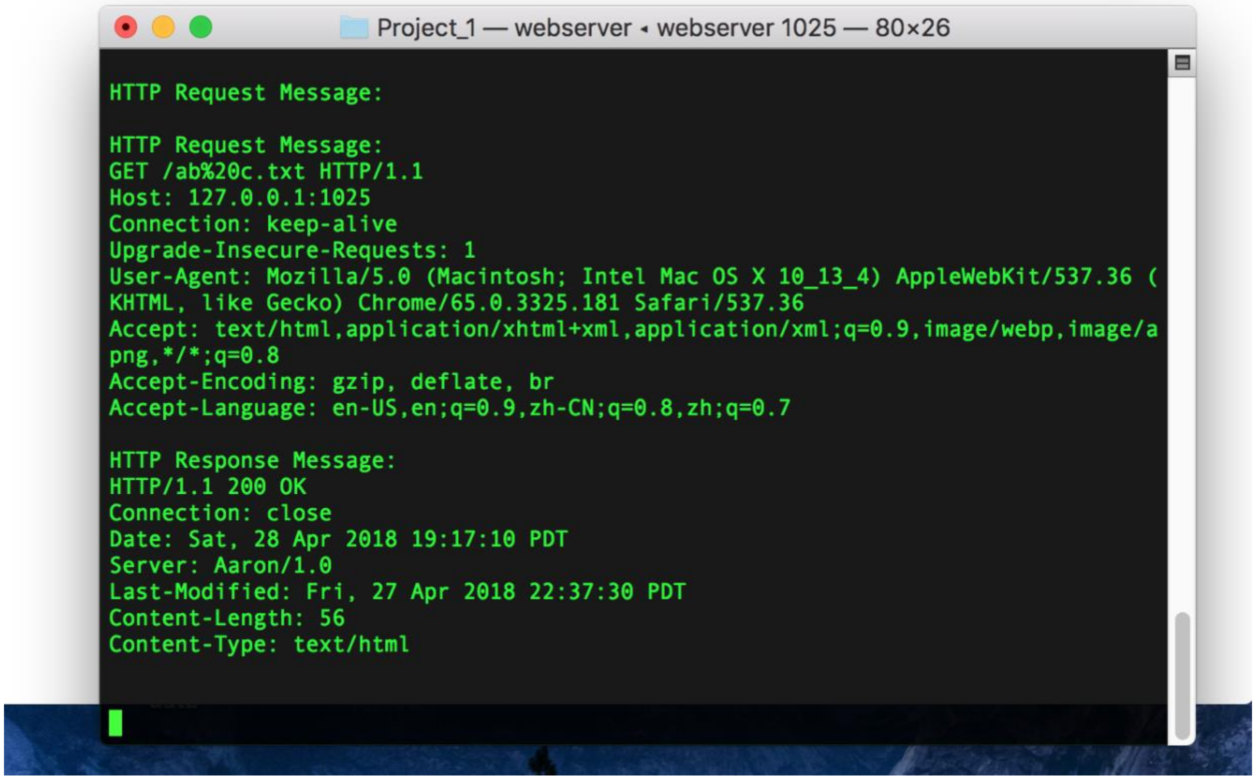

127.0.0.1:1025/1.txt
mimi.jpeg (1440×1
teemo.gif (250×25
127.0.0.1:1025/teemo.gif
啊痛裂耶凯痛
Screen Shot 2018-04...15.19 PM
Makefile
Screen Shot 2018-04...1.33 PM
SQL
Project_1 — webserver ◂ webserver 1025 — 80×26

```
HTTP Request Message:

HTTP Request Message:
GET /teemo.gif HTTP/1.1
Host: 127.0.0.1:1025
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/65.0.3325.181 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/a
png,*/*;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

HTTP Response Message:
HTTP/1.1 200 OK
Connection: close
Date: Sat, 28 Apr 2018 19:16:04 PDT
Server: Aaron/1.0
Last-Modified: Fri, 27 Apr 2018 15:06:15 PDT
Content-Length: 835834
Content-Type: image/gif
```

4. ab c.txt



This is simple text file whose name including space ' '.



```
HTTP Request Message:

HTTP Request Message:
GET /ab%20c.txt HTTP/1.1
Host: 127.0.0.1:1025
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/65.0.3325.181 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/a
png,*/*;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

HTTP Response Message:
HTTP/1.1 200 OK
Connection: close
Date: Sat, 28 Apr 2018 19:17:10 PDT
Server: Aaron/1.0
Last-Modified: Fri, 27 Apr 2018 22:37:30 PDT
Content-Length: 56
Content-Type: text/html
```