COP4635 Sys & Net II **Project 1: HTTP Client Server** **Spring 2018**

HTTP Server and Client

OVERVIEW

Program an HTTP server in C. The server will listen on a valid port. Make sure you use a port number from the valid port number range (60001 – 60099). When client makes a request for an HTML file using a valid HTTP/1.1 method like 'GET', the HTTP server will check if the file is a valid HTML file with the server. If the file is a valid HTML file, the server should send it to the client with proper http header. If the file requested by the client is not a valid file with the server, or if the client request for the file is not recognized, the server should respond with a small, but descriptive error message that should be received by the client and displayed. Your HTTP server should be verbose, and display all activity going on at the server.

To test the file access through the server further, make sure you incorporate two hyperlinks in the requested html file. Make the first link point to an image object that gets displayed on the client browser on clicking the link. Make the other link on the html page point to a non-existent html page.

Program an HTTP client in C, which will request the user to enter the name of the http server followed by the name of a file on that server. The client program should send a valid HTTP/1.1 request to the server for the file. The client program should display all response received from the server. Make sure that your client program guides the user about the right usage (way to request for the file) before prompting the user and waiting on a client request. Also make sure you let the user request for more files in the same way through the client, and not exit immediately. If the server is still active and listening, it is supposed to respond accordingly. If the server has stopped working, this should also be indicated on the client. Make sure that you have provisions for handling both - normal and abnormal terminations of the client program, displaying appropriate codes and descriptive messages.

Your server should also respond to any standard browser program that is chosen to run as the client. If a user makes a request for the HTML page files while the server is running, the server should send the file to the browser if the request is valid. In case of the valid request/s the bowser would receive the HTML file and display it on the screen.

THE PROGRAM

The bulletin board system consists of two programs: a httpClient and a httpServer. The httpServer program serves HTML files to the httpClient. Both programs use TCP for communication.

IMPLEMENTATION SUGGESTIONS

I suggest reading lecture notes and RFC for details on HTTP protocol and TCP networks discussed in class. You may also examine the reading material and the provided sample program on TCP programming. Read these pages thoroughly before starting this project. For your implementation you need to use the following system calls (not an exhaustive list) to create a socket, bind it, send and receive messages and perform file service.

socket()	connect()	gethostname()	bind()
read()	write()	close()	listen()
fflush()	sprintf()	getaddrinfo()	freeaddrinfo()
fscanf()	sendto()	recvfrom()	sigaction()

DELIVERABLES & EVALUATION

Your project submission should follow the instructions below. Any submissions that do not follow the stated requirements will not be graded.

1. Follow the submission requirements of the instructor as published on eLearning under the Content area.

2. You should submit all the following files for this assignment:

   a. source code files for httpClient and httpServer, and supporting code files if any
   b. a single Makefile to compile the client and server programs,
   c. Relevant HTML and image files
   d. a protocol document
   e. a README file containing names of all project members, directions on compiling and running your files indicating usage and examples, and any other relevant details like an analysis if project wasn't completed
   f. Screen captures of your running program showing all windows involved, in two or three stages of communication.

The protocol document must describe the protocols used by the system to make it work. At the minimum the document must describe the message exchange between httpClient and httpServer. The README file should only be included if you submit a partial solution. In that case, the README file must describe the work you were able to complete.

Your program will be evaluated according to the steps shown below. Notice that the instructor/GA will not fix your syntax errors. However, they will make grading quick!

1. Program compilation with Makefile. The options –g and –Wall must be enabled in the Makefile. See the sample Makefile that I uploaded in eLearning.
   - If errors occur during compilation, the instructor/GA will not fix your code to get it to compile. The project will be given zero points.
   - If warnings occur during compilation, there will be a deduction. The instructor/GA will test your code, though.
2. Program documentation and code structure.
   - The source code must be properly documented and the code must be structured to enhance readability of the code.
   - Each source code file must include a header that describes the purpose of the source code file, the name of the programmer(s), the date when the code was written, and the course for which the code was developed.
3. Perform several test-runs with input of the grader's own choosing. At a minimum, the test runs address the following questions.
   - Do the httpServer and httpClient program compile properly and run, and is the communication between the two processes evident?
   - Does the httpServer work equally well with a standard browser program?

- Can valid and invalid file links be tested with the browser as a client for your httpServer?
- Is there ample and evident feedback on what is going on in the httpClient and httpServer programs while they are running, and when they exit?

Keep in mind that documentation of source code is an essential part of computer programming. The better your code is documented, the better it can be maintained and reused. If you do not include comments in your source code, points will be deducted. You should refactor your code to make it more manageable and to avoid memory leaks. Points will be deducted if you don't refactor your code or if we encounter memory leaks in your program during testing.

DUE DATE

The project is due as indicated by the Dropbox for Project 1 in eLearning. Upload your complete solution to the dropbox and the shared drive. I will not accept submissions emailed to me or the GA. Upload ahead of time, as last-minute uploads may fail.

TESTING

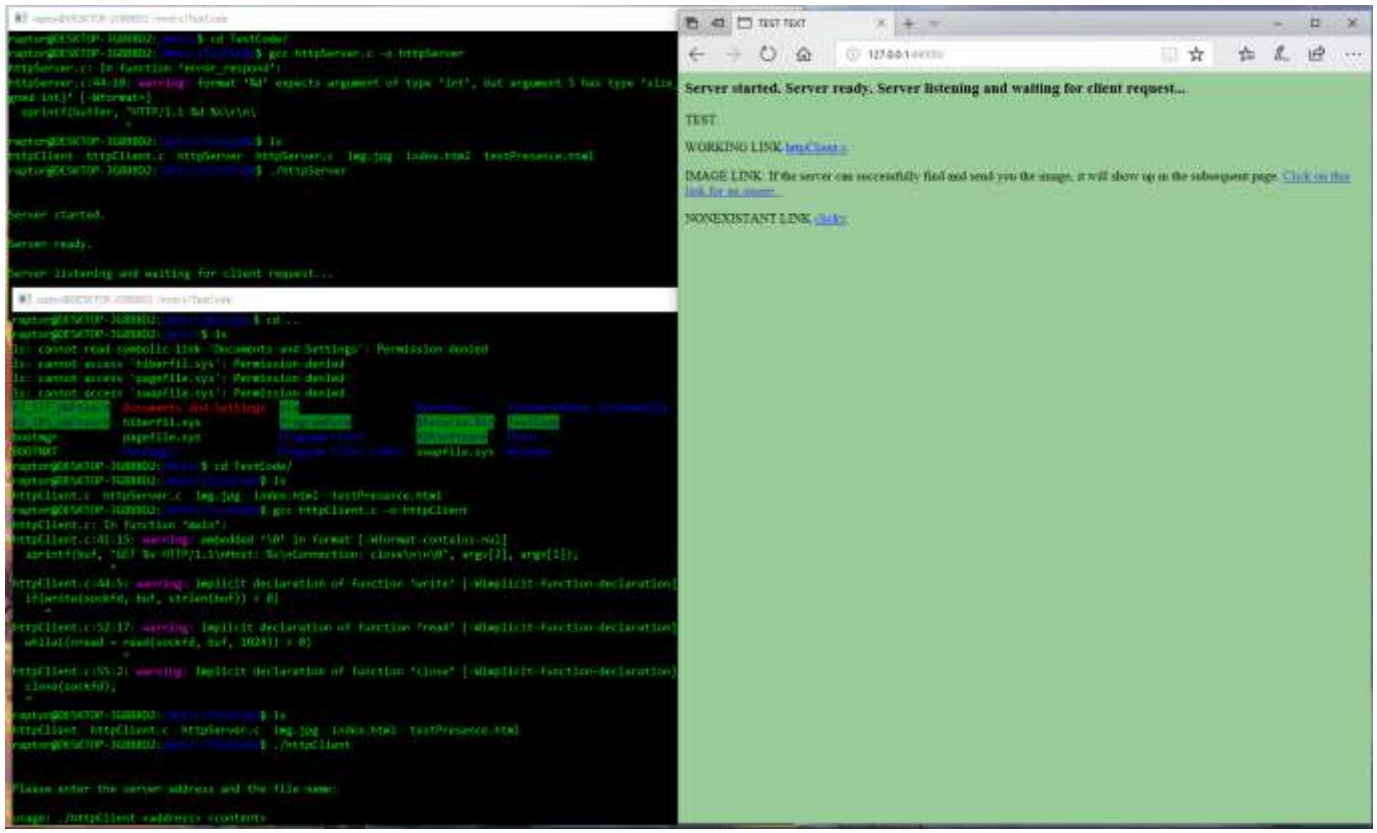
Your solution needs to compile and run on the CS department's SSH server. I/the GA will compile and test your programs on the SSH server and a Linux VM. Therefore, to receive full credit for your work it is highly recommended that you test & evaluate your solution to make sure that we will be able to run your programs and test them successfully. For security reasons, most of the ports on the servers have been blocked from communication. Ports that are open for communication between the public servers' range in values from 60,000 to 60,099.

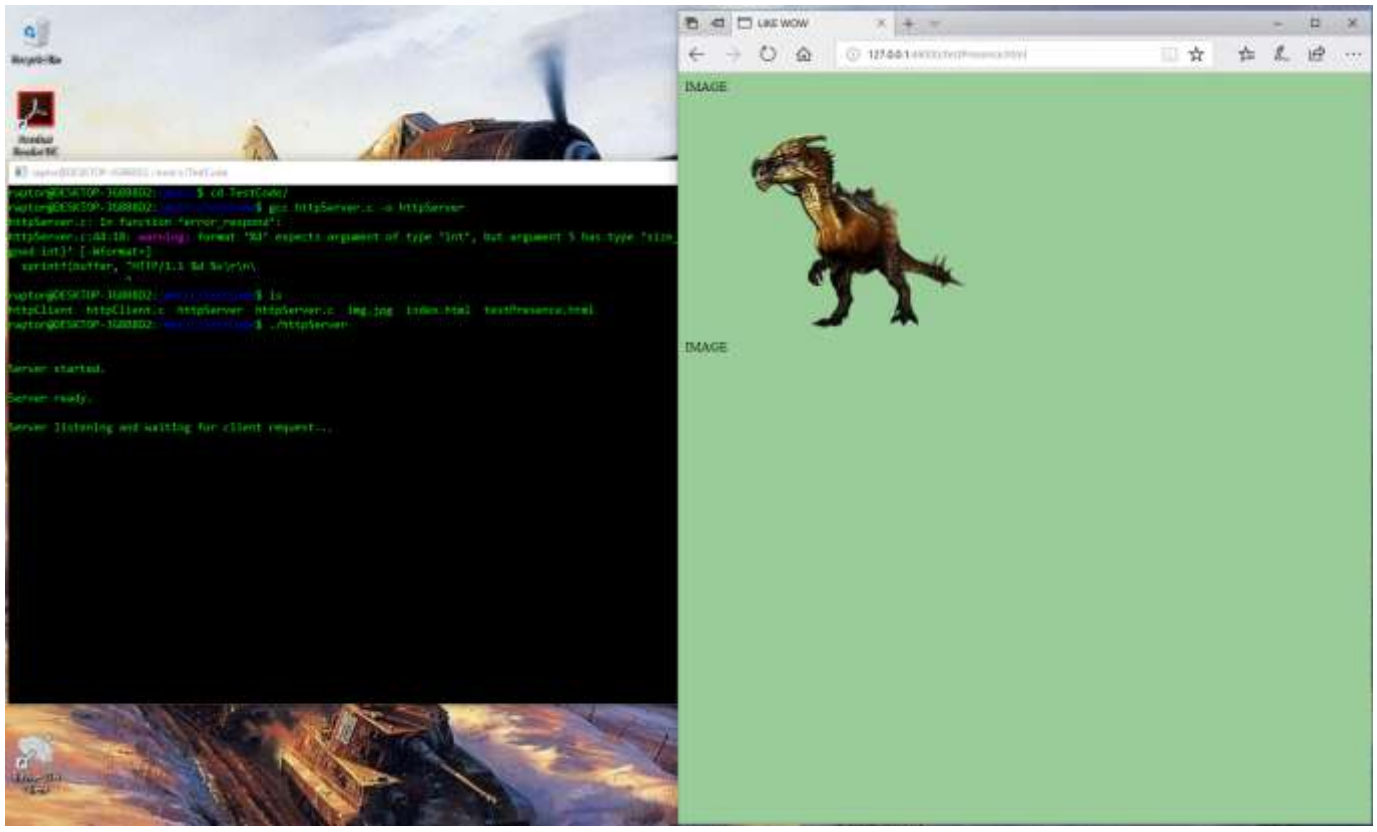
GRADING

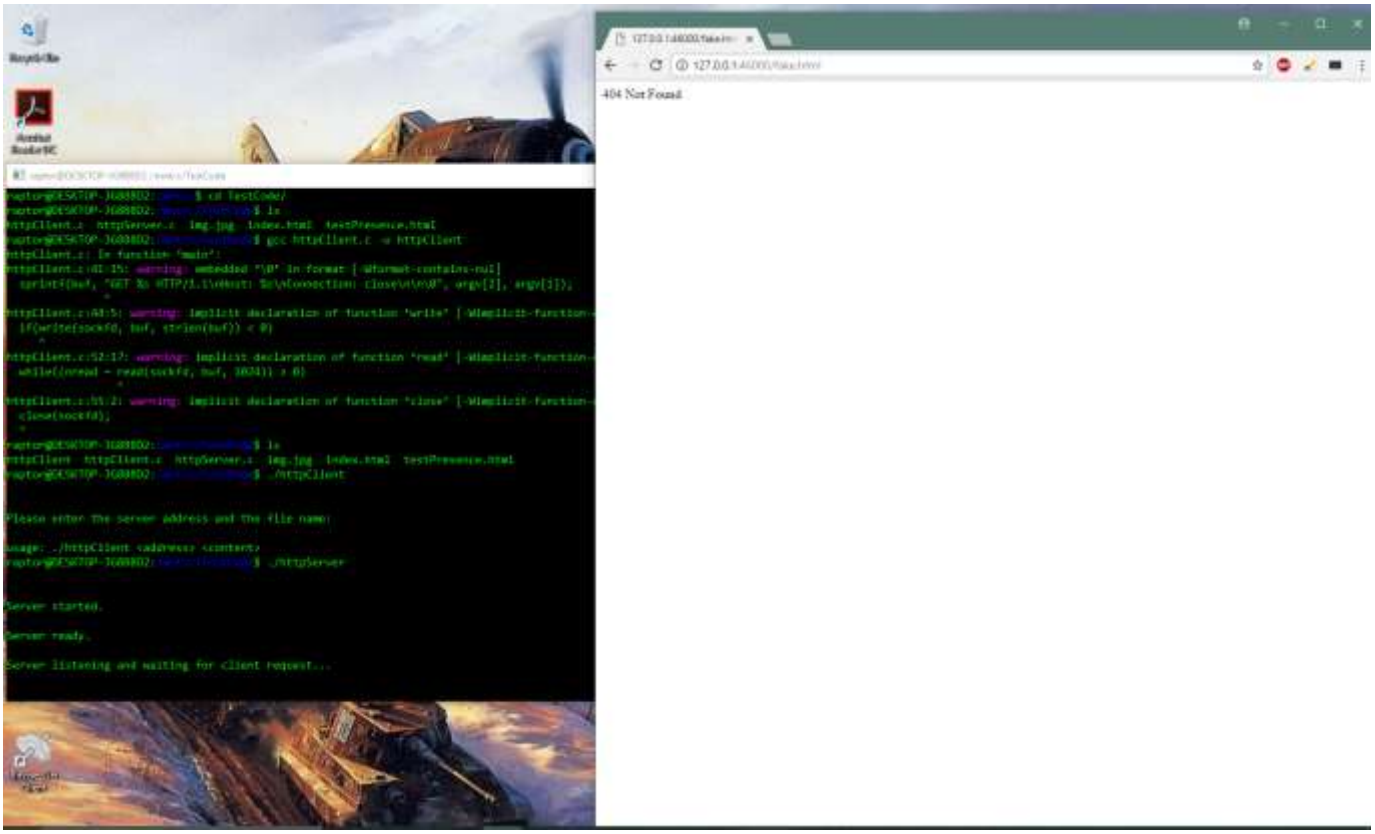
This project is worth 100 points total. I will be updating this file with the rubric and will email you of this update. Please remember that there will be deductions if your code has memory leaks, crashes, has endless loops or is otherwise, poorly documented or organized and there will be zero points for code that does not compile.

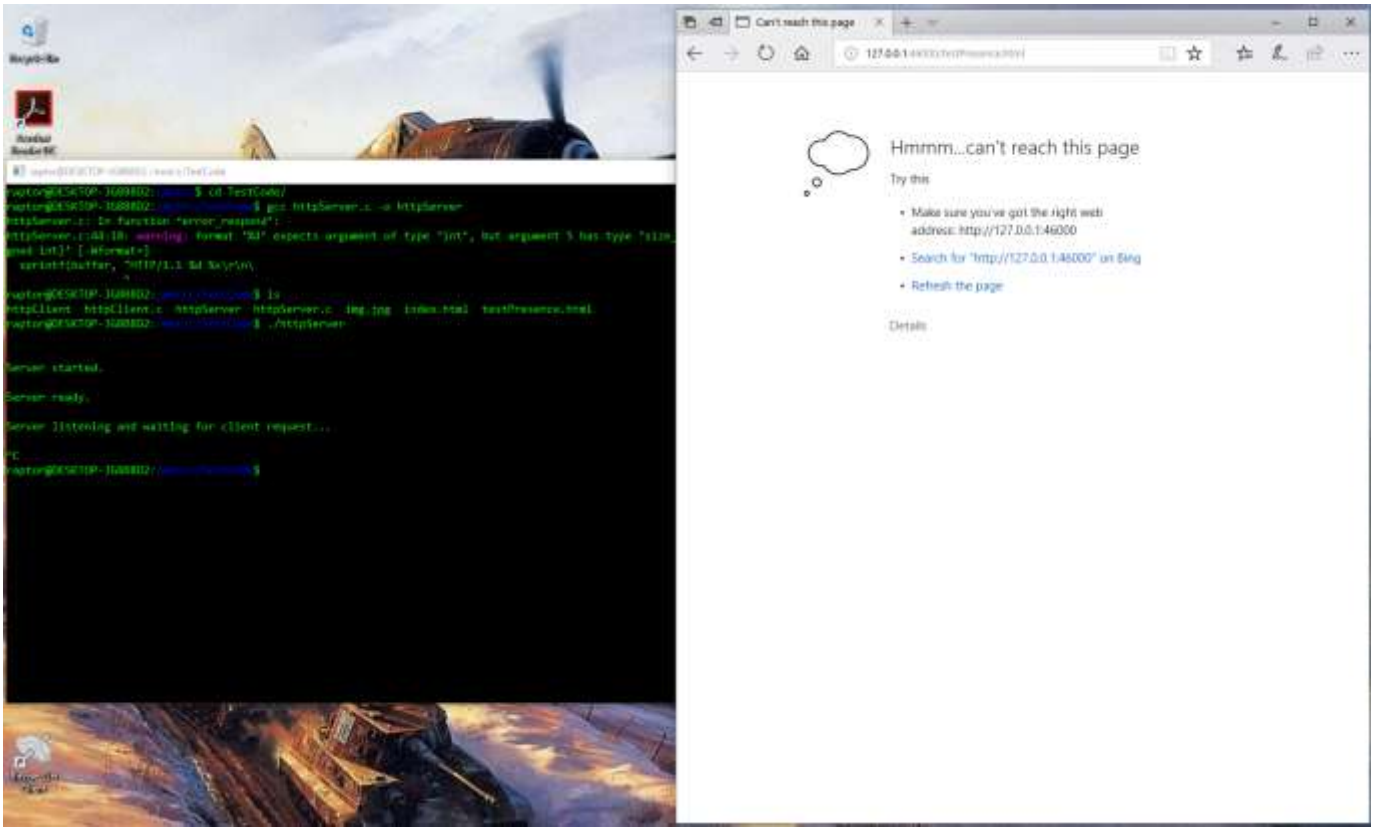
SAMPLE OUTPUTS

Following are some sample outputs from barebones HTTP server and client code in C. This code isn't as elaborate, efficient or informative as your project code should be. I used port address 46000 locally, for the test. The three screenshots show the server and client working together, then the client is terminated and a standard browser is tested for requesting and displaying the valid and existent pages, followed by testing a non-existent file called fake.html, and finally testing the browser client with the server shut down.


TEST TEXT
Server started. Server ready. Server listening and waiting for client request...
TEST
WORKING LINK test/Client.c
IMAGE LINK If the server can successfully find and send you the image, it will show up in the subsequent page. Click on this link for an answer
NONEXISTANT LINK clicky
$ cd TestCode/
$ gcc httpServer.c -o httpServer
$ ./httpServer
Server started.
Server ready.
Server listening and waiting for client request...
$ gcc httpClient.c -o httpClient
$ ./httpClient
Please enter the server address and the file name:
usage: ./httpClient <address> <content>


LIKE WOW
IMAGE
IMAGE
$ cd TestCode/
$ gcc httpServer.c -o httpServer
$ ./httpServer
Server started.
Server ready.
Server listening and waiting for client request...


404 Not Found


Hmmm...can't reach this page
Try this
Make sure you've got the right web address: http://127.0.0.1:46000
Search for "http://127.0.0.1:46000" on Bing
Refresh the page
Details