CSE 4344 Lab # 1 – Web Server Programming Spring 2025

Objectives:

1. Understanding Socket programming
2. Understanding Web and HTTP protocol
3. Using packet sniffing tool to analyze network traffic

Due: 11:59pm, 02/28, 2025 Difficulty: Easy

Project Specification:

This is an individual project.

For this project, Python/Java is the required language.

You can learn from available codes online but you cannot copy directly the code and submit. You are also required to document the code and provide clear comments.

In this project, we will develop a multi-thread web server, capable of serving multiple requests in parallel. You can refer to the textbook for this assignment as well.

Basic Requirements:

The basic functional requirements of the web server are as below:

1) A webpage is stored on your server (assuming it is localhost). We assume it is index.html but the name is your choice. This index.html shall contain at least one image.
2) A process will be running on the server and listening to the specified port. We assume it is 8080 but the port number is your choice.
3) In your web browser, if you type in http://localhost:8080/index.html, your web server process shall fetch the index.html (or requested objects) from the file system, compose the http response, and send it back to the browser.
4) If you web browser can display the page correctly, the web server process is functioning as required for a Status Code 200 response.
5) You are required to implement 301, 404 Status Codes as well. ***You will need to study (BY YOURSELF) meanings of these codes and how to implement the proper additional header fields***, if any.

More Explanation and Requirements:

First, if you do not know how to compose an html page, you can take a look at https://www.w3schools.com/html/html_basic.asp. You can copy and use any html page from the web for this project.

Recall that HTTP 1.0 creates a separate TCP connection for each request/response pair. **A separate thread handles each of these connections.** There will also be a main thread, in which the server listens for clients that want to establish connections. To simplify the programming task, you can develop the code in two stages. In the first stage, you can write a multi-thread server that simply displays the contents of the HTTP request message that it receives. After this program is running properly, you will add the code required to generate an appropriate response.

As you develop the code, you can test your server with a web browser. Remember that you are not serving through the standard port 80. You will need to specify the port number within the URL given to the web browser. For example, if your host name is host.someschool.edu, your server is listening on port 6789, and you want to retrieve the file index.html, then you would specify the URL in the browser as

http://host.somechool.edu:6789/index.html.

You can test the web server on your own machine using:

http://localhost:6789/index.html or http://127.0.0.1:6789/index.html.

As we have discussed in class, localhost or 127.0.0.1 is pointing to your local machine.

Notice that while there are multiple types of methods in the HTTP protocol, you are required mainly to implement the response for the GET method. Also, while there may be multiple status codes possible in the response as in the HTTP protocol, you are required only to implement the 200, 301, and 404 codes, where the browser should be able to display properly the responses.

For 301 code, assume that you have a page named *page1.html, which has been permanently moved to page2.html. In you browser, when visiting* http://localhost:6789/page1.html, the browser shall be redirected to http://localhost:6789/page2.html. **This shall be hard coded in your program.**

For 404 code, you are quired to use a customized 404 page. In other words, the browser shall display **a 404 page that you have designed for that purpose**. An example of a customized 404 page can be found here: https://www.uta.edu/test404for4344.html

When testing your server code, **at least use one image object in your html page**. Please note the image object should be stored locally at your server, not referring to an URL to other servers. Notice that the images will be fetched by the browser after the html base page has been successfully received by the browser.

Debug:

In additional to any conventional debugging tools, you should use the developer tools of the browser to observe the client.

You shall also use wireshark to monitor http messages.

References:

There are various resources available regarding socket programming on the Web. You are encouraged to browse and learn from them.

Writeup:

Your write-up should include instructions on how to compile and run your program. Ideally it should be complete enough that the TA can test your program without your being there. Your writeup should include any known bugs and limitations in your programs. If you made any assumptions such as limits on the size of a word or phrase you should document what you decided and why. This writeup should be in PDF format and should be submitted along with your code.

For this project, you should also include screenshots of the outgoing http requests and incoming http responses (for codes 200, 301, and 404). This can be done using WireShark to capture the corresponding http messages and take screen shots.

Submission Guidelines:

All submissions are through Canvas. You should zip your source files and other necessary items like project definitions, classes, special controls, data files, DLLs, etc. and your writeup into a single file. The name of this file should be your 1000ID.zip, e.g. 1000123456.zip. Be sure that you include everything necessary to unzip this file on another machine and compile and run it. This might include forms, modules, classes, config. files, etc. DO NOT INCLUDE ANY RUNNABLE EXECUTABLE (binary) program. We will recompile your programs anyway.

Make sure your name and your student ID are listed in your writeup, and in comments in your source code. You may resubmit the project at any time but total number of submissions is limited to 3. Late submissions will be accepted at a penalty as announced in the class website. This penalty will apply regardless of whether you have other excuses. In other words, it may pay you to submit this project early. If the TA can not run your program based on the information in your writeup then he will email you to schedule a demo. The TA may optionally decide to require all students to demonstrate their labs. In that case we will announce it to the class.

If your program is not working by the deadline, submit it anyway and review it with the TA for partial credit. Do not take a zero or excessive late penalties just because it isn't working yet. We will make an effort to grade you on the work you have done.

Grading: Total 100 Points

Some point elements are listed below

25 -- Correctly parse a HTTP request
10 -- Correctly generate the status code
25 -- Correctly compose the HTTP response message (200,301,404)
10 -- Correctly handle multiple requests simultaneously
15 -- Correctly capture http requests and responses using wireshark
05 -- Proper comments in code
10 - TA discretion for things the instructor overlooked

To receive full credit for comments in the code you should have **brief** headers at the start of every module/ subroutine/ function explaining the inputs, outputs and function of the module. You should have a comment on every data item explaining what it is about. (Almost) every line of code should have a comment explaining what is going on. A

comment such as /* Add 1 to counter */ will not be sufficient. The comment should explain what is being counted.

Deductions for failing to follow directions:

-5 Including absolute/ binary/ executable module in submission
-5 Submitting writeup in other than PDF format
-5 Submitted file has a name other than student's campus Login_Id.zip
-5 Not submitting a make file or the equivalent for the IDE, if applicable.

Important Note:

You may discuss the problem definition and tools with other students. You may discuss the lab requirements. You may discuss or share project designs. All coding work must be your own. You may use any book, WWW reference or other people's programs (but not those of other students in the class) as a reference as long as you cite that reference in the comments. If you use parts of other programs or code from web sites or books YOU MUST CITE THOSE REFERENCES. If we detect that portions of your program match portions of any other student's program it will be presumed that you have collaborated unless you both cite some other source for the code. You must not violate UTA, state of Texas or US laws or professional ethics. Any violations, however small, will not be tolerated. Students who do not submit anything get a grade of 0. Therefore students who break the rules may receive a negative grade – most likely a -50 on this lab assignment.