CSE3027: Computer Networks
Project 1:
Concurrent Web Server using BSD Sockets
Fall 2017
Instructor: Prof. Suk-Bok Lee

Due: 11:59 PM, Friday, October 13, 2017

1 Goal

In this project, we are going to develop a Web server in C/C++. Also, we will take the chance to learn a little bit more about how Web browser and server work behind the scene.

2 Getting Started

Reading Chapter 2 of the textbook will help you to understand how HTTP works. However, there are some differences:

- You must program in C/C++ (not Java) . Examples of socket programming Client/Server code in C are available on the CourseWeb..
- Instead of using port 80 or 8080 or 6789 for the listening socket, you can pick your own port to avoid conflicts. You need to make this server port number as input in your web server execute command. It is suggested NOT to use port numbers 0-1024.
- Your program should run on a linux machine. Your submitted code will be tested and evaluated on linux machines.

3 Instructions

1. The project consists of Part A and Part B, which are specified as following:
   - Part A: Implement a "Web Server" that dumps request messages to the console. This is a good chance to observe how HTTP works. So start a browser, e.g. Internet Explorer, Firefox, Chrome, etc (browser is the client side), connect to your server, record the request message, and find out what the fields in the message mean by looking up in the textbook or *RFC 1945*.
   - Part B: Based on the code you have done in Part A, add one more function to the "Web server", that is, the "Web server" parses the HTTP request from the browser, creates an HTTP response message consisting of the requested file preceded by header lines, then sends the response directly to the client.
2. Pay attention to the following issues when you are implementing the project:
   - Your web server has only one command line:
     %myserver <port number>
     <port number> is a port number of your web server for the listening socket. This way you can input your server port number when you start the web server.

- If you are running the browser and server on the same machine, you may use localhost or 127.0.0.1 as the name of the machine.
- Make sure your *"Content-Type"* function recognizes at least html files. We will worry about other types of files later.

3. After you're done with every part (Part A and Part B), you need to test your server. You first put a html file in the directory of your server, or more exactly, where you run your server. Connect to your server from a browser with the URL of http://<machine name>:<port number>/<html file name> and see if it works. For your server in Part A, you should be able to see the HTTP request format in the console of your server machine. For your server in Part B, your browser should be able to show the content of the requested html file.

4. Back to the *"Content-Type"* function. Besides HTML files, add in the support for GIF, JPEG, MP3, and PDF files. Your browser should be able to correctly show/support all these type files.

4 Project Submission

Note that the programming language that you use should be C/C++ and your program should run on a linux machine.

1. Put all your files into a directory, named "project1_UID_YourName" where UID is your student ID and YourName is your name (e.g., Hong Kildong → project1_xxx_Hong_Kildong).

2. In the directory that contains "project1_UID_YourName", type the following command in UNIX shell

   tar cvf project1_UID_YourName.tar project1_UID_YourName

   For example, if you have a "cse3027" directory in your home directory, and "project1_UID_YourName" directory within "cse3027". Once you login from ssh, you need to

   cd cse3027
   tar cvf project1_UID_YourName.tar project1_UID_YourName

3. Submit the file "project1_UID_YourName.tar" via online submission in Hanyang Portal courseweb.

4. The directory "project1_UID_YourName" should include the following files:

   - All **commented** source files.
   - A Makefile.
     - Graders will only type "make" to compile your code; make sure your Makefile works!
   - Short report (report.doc, report.pdf). 1-3 pages.
     - You need to include the following items in your report:
       * Give a high-level description of your server's design
       * What difficulties did you face and how did you solve them?
       * Include and briefly explain some sample outputs of your client-server (e.g. in Part A you should be able to see an HTTP request)

The project due date is October 13, 2017.