COM SCI 118 Computer Network Fundamentals

Spring 2009

Project 1

Title : Concurrent Web Server in C/C++
Due time. 11:59 PM May 12 th, 2009

I. 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.

II. Lab Work Environment

You need a plain-text editor to create the source code. Notepad in Windows, vi/pico/emacs in Unix will do the job. Since C is a cross-platform language, you should be able to compile and run your code on any machine with C compiler and BSD socket library installed.
Students should use Solaris workstations in the SEASnet to finish the project, because

1. C/C++ compiler (gcc) and BSD socket library on those workstations have been verified.
2. Your submitted code will be tested and evaluated on them.

AIX workstations and NT workstations are **NOT** recommended. They have their own idiosyncracies which may be too much for this simple project. Also, TAs will not be able to help you on any of these platforms.
For those of you who do not have a SEASnet account:

1. If you are engineering major, you can get your own account from the SEASnet office (2684 BH, x66864).
2. If you are from outside the engineering school, you can still get a class account, only that this account expires when quarter ends. Get an application from the SEASnet office, complete it, and TAs will be happy to sign it for you.

For more information about the SEASnet, go to their homepage at *http://www.seas.ucla.edu/seasnet/*

III. Instructions

1. Read Chapter 2 of the textbook carefully. The textbook does help you to understand how HTTP works. However, there are some differences:

   - You must program in C/C++ (not Java). A set of socket programming template code in C has been available since the 2nd week of this quarter.
   - Instead of using port 80 or 8080 or 6789 for the listening socket, you should pick your own to avoid conflicts. Keep away from port number 0-1024.

2. 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., Microsoft Internet Explore, 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.

3. Pay attention to the following issues when you're implementing the project:

    - 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 contentType function recognizes at least html files. We will worry about other types of files later.

3. After you're done with either Part A or 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 start 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 contentType function. Add in the support for GIF and JPEG images. Besides that, find out another 5 different types of MIME files and add them to your code. Your browser should be able to correctly show all these MIME files.

IV. Materials to turn in

1. Your source code files(e.g. webserver.c)
2. A report of your project (pdf or word format) should not exceed 8 pages (except for source codes).
3. The cover page of the report should include the name of the course and project, your partner's name, student id, and SEASnet login name.
4. Include a brief manual (must be less than 100 words) in the report to explain how to compile and run your source code. If TAs can't compile and run your source code by reading your manual, the project is considered to be not finished and the grade will be 0.
5. Include your source code in the report, too.
6. Include the result of step 4 and your explanation of the request message fields in the report.
7. Method of submission is described in Section VI.

V. How to submit

1. Put all your files into a directory, say, "project1" (I will assume "project1" is used in following).
2. In the directory that contains "project1", type the following command in SEAS UNIX shell

    tar cvf project1.tar project1

    For example, if you have a "cs118" directory in your home directory, and "project1" directory within "cs118". Once you login from ssh, you need to

    cd cs118
    tar cvf project1.tar project1

3. Submit the file "project1.tar" via SEAS online submission in course webpage.
4. In "project1" direcotry, it should contain at least 3 files.
    - your web server source code (can be multiple files)
    - a readme.html file
    - a Makefile
5. The readme file (readme.html) should contain at least the following
    - Student names and Student IDs at the very beginning.
    - You source code
6. The TAs will only type "make" to compile your code, make sure your Makefile works under SEAS.
7. Each group just needs to submit one set of files.