CSC358 Assignment 1

Due: February 11th, 2022. 11:59pm

This assignment is designed to give you a chance to practice with socket programming and threads. The core functionality of the assignment is fairly simple, but there are a wide variety of cases that will need your attention to handle. You may work in groups of 1-3 for this assignment. Be sure to declare your groups on MarkUs far in advance of the deadline. Members of the group must all be able to attend the same tutorial for the code presentation, but it is ok if you are not all registered for that tutorial. Please notify your TA(s) at least one week in advance to let them know which tutorial you will attend to present your work.

HTTP Server

Your first task will be to implement a server for the HTTP 1.0 protocol that is capable of handling GET method requests for files with the following extensions: .html .css .js .txt .jpg.

Your server must also support conditional GET requests (such as an If-Modified-Since request header). Your server must also be written to handle, at a minimum, requests for MIME text/html, text/plain, text/css, text/javascript, and image/jpeg files. You may choose to handle additional file types.

Adding Persistence

Your second task will be to implement a server for the HTTP 1.0 + 1.1 protocols capable of handling all the same method requests as your previous server, but that also allows persistent connections to be maintained. In addition to making adjustments to your HTTP headers from part 1, you will also need to efficiently manage memory. Your solution must not keep too many unnecessary connections open, nor should it close connections that are still likely to be used.

Adding Pipelining

Your third task will be to add pipelining, now that you have a version of your server with persistent connections. This should allow users to send pipelined requests that will be responded to in a pipelined fashion.

Command-line Execution

You will write 3 separate programs to implement the various versions of your server:

- SimpleServer.c
- PersistentServer.c
- PipelinedServer.c

Your servers should take 2 command-line arguments:

- port #
- http root path

port # is a TCP port number, http root path is a string representing a relative or absolute file system path.

The marker will start your server with their own port # and http root path, so make sure you test your server with these parameters supplied on the command line at startup. If your submission has its port # and http root path hard-coded, it will receive a mark of 0.

Performance Testing

You should now have 3 different versions of your server, we want to see how much of a difference all those extra features made. Your task is to compare:

- Your simple http server
- Your persistent http server
- Your pipelined http server
- The Apache server

and write a report on what you found. To help you with your testing, you should make use of httperf. httperf sends a configurable-volume of HTTP requests to a server, and will record data about the responses received from that server. Your report should be written as a formal technical report, detailing the differences in your servers, and providing a clear explanation of how the differences affected their performance. It should be written in a way that a person who has experience with networking, but not necessarily the details of this assignment (or of httperf) could follow your reasoning.

Submission

By the deadline, you must submit the following to MarkUs:

- A1.tar.gz - A compressed file containing the three servers and any additional files necessary to run them
- A1.pdf - Your performance testing report file

In your tutorial following the submission deadline, you will present your report and your code to one of the TAs. You will have no more than 5 minutes to formally present your code and your findings, followed by a brief period where your TA may ask you questions for clarification. If you fail to show up for your scheduled tutorial you will receive a zero for your assignment grade.

Evaluation

Your assignment grade will be calculated as follows:

- 25% Working file submission
- 25% Code quality and documentation
- 25% Performance Testing Report
- 25% Presentation