Programming Assignment 1 for CS560

Due: 2/28/2018

This programming assignment does not specify any particular programming language you choose. You may choose any mainstream programming language such as Java, Python, or C to implement the requirements. You do, however, are required to provide detailed testing results in the form of screenshots on how to run your code, what results you obtain, and how you tested your results to know that they are correct.

Overview

The goal is to implement a basic HTTP server that supports directory listing, static files, and CGI scripts. The server needs to run on a Linux server environment, such as those supported in our lab machines. Additionally, you will also have to write a sample CGI shell script that can be executed by your web server.

Specifically, your web server needs to support the following features:

1. Single connection mode (serial) and multiple connection mode (parallel with multiple threads)
2. HTTP GET requests with query and header parsing
3. Automatic directory listing
4. Static file transport
5. Basic CGI support by running a sample CGI script on the server side

Design

Generally, the implementation of a HTTP server is straightforward. First, the code needs to allocate a server socket, bind it to a port, and then listen for incoming connections. Next, the server accepts an incoming client connection and parse the input data stream into a HTTP request. Based on the request's parameters, it then forms a response and sends it back to the client. In a loop, the server continues to perform steps 2 and 3 for as long as the server is running. If we are in multi-thread mode, then we simply fork a thread after we accept a connection and let the child thread handle parsing and responding to the request. In practice, you may choose to use either a process (as in Linux processes) or a thread to handle incoming calls.

To observe what a web browser such as Firefox or Chrome would send to a HTTP server, you can use netcat to simulate a server. For instance, run the fake server with nc -l -p 10010 and then use your web browser to go to http://hostname:10010. You should then see all HTTP request and headers the web browser sends to the HTTP server. For example, you may see something like following (using Chrome as the browser):

Connection: keep-alive

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari

/537.36

Upgrade-Insecure-Requests: 1

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.9

In general, a valid HTTP response looks like this:

HTTP/1.0 200 OK

Content-Type: text/html

<html>

...

</html>

The first line contains the HTTP status code for the request. The second line instructs the client (i.e. web browser) what type of data to expect (i.e. mime-type). Each of these lines should be ended with \r\n. Additionally, it is important that after the Content-Type line you include a blank line consisting of only \r\n. Most web clients will expect this blank line before parsing for the actual content.

As noted above, your implementation should support both single connection mode (the default) and threading mode. In single mode, one connection is handled at a time and thus there is no concurrency. In threading mode, a child process/thread is forked after accepting a request and used to handle the request, thus allowing for multiple requests to be processed at the same time.

Once you have implemented a webserver and verified it is working by accessing web pages using a web browser such as Firefox or Chrome, and by using command line tools such as curl or wget, then you are to write a CGI shell script. What this script does is up to you, but it must involve some form of processing and dynamic content generation. When the CGI script is embedded into a webpage as response, it will be executed by the server side and the results embedded into the response automatically.

Extra credit (15pts)

The extra credit part requires you to implement basic user interaction on submitting a form. Specifically, you should be able to support users to submit a basic HTTP form using the web server, and write the results into a local file. The details on the implementation will depend on your design choices.

Deliverables

You need to provide a design document on your decision choices. You will also need to provide screenshots on how you use various tools such as browser to test your code to ensure its correctness. The code needs to be properly documented. Each group should submit original code, meaning that the code should be not obtained from anywhere else. Violations of this policy will be considered as plagiarism and will taken very seriously.