Sockets Programming Project: A Web Server

Objectives

- Learn about connection- and connectionless-oriented communication.
- Utilize the client-server model in applications.

Specification

You will develop and implement a Web server that is capable of processing multiple simultaneous service requests in parallel. You should be able to demonstrate that your Web server is capable of delivering your home page to a Web browser.

We are going to implement version 1.0 of HTTP, as defined in RFC 1945, where separate HTTP requests are sent for each component of the Web page. The server will be able to handle multiple simultaneous service requests in parallel. In the main program/process, the server listens to a fixed port. When it receives a TCP connection request, it sets up a TCP connection through another port and services the request in a separate process.

When the server encounters an error, it sends a response message with the appropriate HTML source so that the error information is displayed in the browser window.

The first item available in the client's input stream will be the HTTP request line. After obtaining the request line of the message header, we obtain the header lines. Since we don't know ahead of time how many header lines the client will send, we must get these lines within a looping operation. The header lines end with a blank line (CRLF).

The server must analyze the request and send an appropriate response. We are going to ignore the information in the header lines, and use only the file name contained in the request line. In fact, we are going to assume that the request line always specifies the GET method, and ignore the fact that the client may be sending some other type of request, such as HEAD or POST. Because the browser precedes the filename with a slash, we prefix a dot so that the resulting pathname starts within the current directory. Now that we have the file name, we can open the file as the first step in sending it to the client. If the file does not exist, we should return an error message to the user's browser.

There are three parts to the response message: the status line, the response headers, and the entity body. The status line and response headers are terminated by the character sequence CRLF. We are going to respond with a status line, and a single response header. In the case of a request for a nonexistent file, we return *404 Not Found* in the status line of the response message, and include an error message in the form of an HTML document in the entity body.

When the file exists, we need to determine the file's MIME type and send the appropriate MIME-type specifier. We make this determination by examining the file extension. If a file ends with .htm or .html, it is a text web page and if it ends with .jpg, it is a jpg image. These are the only two MIME types that your server must be able to handle. If the file extension is unknown, we return the type application/octet-stream.

Now we can send the status line and our single header line to the browser by writing into the socket's output stream. Now that the status line and header line with delimiting CRLF have been placed into the output stream on their way to the browser, it is time to do the same with the entity body. If the requested file exists, we send the file. If the requested file does not exist, we send the HTML-encoded error message that we have prepared.

After sending the entity body, the work in this process has finished, so we close the streams and socket before terminating.

Usage Clause

The usage clause for your program will be:

webserver <port_number>

Notes

- The server should include several interrupt handlers. If the server receives a SIGINT (ctrl-c) or SIGHUP, it should ignore it. If the server receives a SIGTERM or SIGQUIT, it should ***gracefully*** terminate.
- As you are developing the code, you can test your server from a Web browser. But remember that you are not serving through the standard port 80, so you need to specify the port number within the URL that you give to your browser. For example, if your machine's name is host.kutztown.edu, your server is listening to port 6789, and you want to retrieve the file index.html, then you would specify the following URL within the browser:

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

  If you omit ":6789", the browser will assume port 80 which most likely will not have a server listening on it.

Grading

- You will receive a 0 if there are any compilation errors.
- Your program should not core dump, hang or end prematurely.
- The following items will be considered when grading this assignment
    - Style and format
    - Documentation
    - Error checking
    - Implements specification as described

Turn in

- Source code
- Makefile
- Readme file that must include at least the following
    - Description of program (what it does)
    - How to compile
    - How to run
    - Design overview
    - How you may have handled any ambiguities in the specification
    - Any known bugs or problems