



Simple HTTP server-client

Name: Eman Rafik Abd el-Kader

ID: 11

Overview

The developed application is mainly composed of two main parts: the multi-threaded web server, that is designed to serve http requests from multiple clients, and the http web client, that is designed to send http requests and receive responses.

Design

Sever

The server is a multi-threaded program, where it creates a socket with the specified port then it accepts connections from multiple clients and assigns a running thread to serve an individual client. Each thread listens on the socket to receive requests from the specified client and it delegates another thread to handle these requests and send responses back to the client to achieve a better throughput. The server implements HTTP 1.1, so it supports persistent connection, where it keeps every connection open then it timeouts after ten seconds if no requests were received.

Client

The client program firstly tries to connect to the server socket and once connected it has to running threads: one for sending requests and one for processing the received responses. The requests are read from the input file then for each request the http message is composed and sent through the socket. Upon receiving a response, it is displayed on the console and the file is saved in the client folder. Since the server runs on http 1.1, the client sends all requests without waiting for each response, and once received all responses it closes the connection. The client timeout and close the connection if it didn't receive responses for too long.


Implementation

Sever

The server program has two classes: main and helper.

Main class:

The main class initializes the server socket and accepts connections from clients. It has a pool of threads (clientsThreadsPool) to prevent memory consumption. The threads'



function serves the clients in the waiting queue once being idle. The delegated thread listens to the socket infinitely, unless timed out, and delegates another thread (process_message_thread) to process these requests.

Helper class:

Responsible for parsing requests and handle both get and post requests then send responses to the client. Each client thread has its own helper.

Data Structures:

clientsThreadPool: An array of threads to serve clients.

clients: queue of waiting clients to be served.

clientArgs: a struct to keep pointers on a client's arguments to be passed to the process_message_thread.

buffer: character array for each client to receive requests on.

sendBuffer: a character array in the helper class to send responses through.

Client

The client program has two main classes: the main class and the helper class.

Main class:

The main class firstly establishes a connection with the server then it calls the helper to send requests, listens for responses, and creates a thread to process them.

Helper class:

A class with a group of functions to parse the input file, construct the http messages and send requests to the server

Data Structures:

receiveBuffer: character array for each client to receive responses on.

sendBuffer: a character array in the helper class to send requests through.

paths: a vector of strings to keep the paths in requests to save files after response.

Assumptions

1. The server program has only one parameter, the port number.
2. The client program has three parameters: the folder name, the server and the port.
3. Each client must have a folder with a file named "input.txt" containing the requests.
4. Files retrieved from the server are directly saved to the client folder path.