Aidan Smith

CSE 130

Design Document for Assignment 1

1.0 Introduction

This document describes all data, functional and behavioral requirements for httpserver.

1.1 Goals and objectives

This program is designed to create and executable named httpserver that acts as a simple server with GET, PUT, and HEAD commands.

1.2 Statement of scope

Httpserver will take in different commands and will act as if it were a server.

2.0 Design

A description of the design of httpserver.

2.1 server.c

server.c is made up of three main functions and an int main function. These are expanded on in 3.0 Functions.

2.2 Makefile

Makefile is used to run httpserver cleanly, its usage is described in the makefile itself.

3.0 Function Parameters

A description of the parameters used for all three functions.

3.1 ssize_t client_sockd

This parameter is the file descriptor of the client, so that information can be read from and written to them.

3.2 struct httpObject* message

This parameter is the struct used to contain all relevant information in regard to the client and their request. This is updated throughout the functions so that the correct information is relayed between functions and ultimately sent to the client.

4.0 Functions

A description of functions used in server.c.

4.1 void read_http_request(ssize_t client_sockd, struct httpObject* message)

This is the first function, the only one to be called within int main(). This function handles reading the header from the client and the point where the content is reached. The content that is reached is saved to the struct message, previously described in 3.2.

4.2 void process_request(ssize_t client_sockd, struct httpObject* message)

This is the second function, nested inside of the previous. This function is only called if no errors are reached beforehand. This function determines what request was made (get, put, or head) and runs the necessary commands to complete the request.

4.3 void construct_http_response(ssize_t client_sockd, struct httpObject* message)

This is the last function, purely used to send a properly formatted response. This function can be called within either of the previously mentioned functions. If it is called in the first function, it will be due to an error. If it is called from within the second function, it could be due to success or error. This function reads the status_code from the struct message, which was set before this function was called from within a previous function. This code tells the function what to print, error or success, and it also looks to the struct message for content length. The content length is only nonzero if a get is called, and it is only nonzero if it runs properly up until the response is called.

4.4 int main(int argc, char** argv)

This is the main function that starts the server. Within here is where the server is created, and then used to allow clients to connect. Inside of an infinite while loop a client will connect. Once accepted, the server will call the first function, read_http_response(), to begin the process of reading and completing the client's request. Once the function calls are complete, the client will be closed, successful or not. The struct message will be cleared afterwards, to ensure no information between clients causes disruptions in data.

5.0 User interface

This is described in README.md.

6.0 Restrictions and differences

httpserver is supposed perform on a very similar level to a real server with curl calls. The program is a simple version of an http server with only GET, PUT, and HEAD commands. There is a limited amount of error testing in comparison to a full written http server, thus most errors will be bunched up under 500 (internal error) and not specified further.