Asgn 1 Design Document

Samuel Shin
Cruzid: sayshin

CSE 130, Fall 2019

1 Goal

The goal of this program is to create a server that can take simple GET and PUT requests and process them and execute. The program is able to take in ip addresses and hostnames and port numbers and connect to them. In a GET request, the server gets a local file and outputs its content to the client. In a PUT request, the client is putting one of its local files onto the server's directory. The server is supposed to respond with appropriate http protocols to let the client know what went right and wrong.

2 Assumptions

I'm assuming that the testing will be done with "curl http://localhost:8080 --request-target filename_27_character" for GET requests and "curl -T localfile http://localhost:8080 --request-target filename_27_character" for PUT requests. I'm assuming that my GET works fine because on Piazza it says that it's okay if we're getting this warning "Warning: Binary output can mess up your terminal. Use "--output -" to tell Warning: curl to output it to your terminal anyway, or consider "--output Warning: " to save to a file." I'm assuming that the user will input something like 127.0.0.1 for the address name when running ./httpserver [address name] [optional port num] because my program can't handle something like http://localhost. I'm assuming that if using the default port number 80, use sudo ./httpserver [address name] to start the server.

3 Design

The way I am approaching this problem is first being able to establish basic connection between server and client and communicate basic messages. Then I will parse the GET/PUT headers and check for legality of the filename given. Next, I will read and write whatever data I need depending on if it's a GET or PUT request. And then I'm going to handle edge cases, response codes, and code cleaning.

4 Pseudocode

Procedure httpserver

If num of args < 2 || > 3
    Error out
Call socket(), setsockopt(), bind(), listen(), accept() in this order to establish connection

Somewhere in above steps, set port number if port number was given, otherwise default is 80
recv() request(s) from client
Parse http header

If header has GET
    Parse and check if filename is legal
    Basically do the dog algorithm we did, but instead of writing to stdout
    Write to the client by doing a send(sockfd, buffer, bytes, 0)
Else If header has PUT
    Parse and check if filename is legal
    Get the content length to see how many bytes to read
    Call recv again to get the data to put into the file
    Then place file contents into given filename
Else
    Send a error 500 code