Design Document

Rishab Jain
CruzID: rjain9

CSE:130, Fall 2019

1 Goal

The goal of this program is to create a server that can handle PUT and GET commands until the server is closed by the user.

2 Assumptions

For the assignment I am assuming that the filename that is passed in the header from the client doesn't contain the '/' character. Also assuming that the response header are sent to the client and that no messages are printed to the server. Also if a file exists but it isn't a correct 27 character ascii name that the program will respond with a forbidden error. Assume for PUT case where there is no content length that the client must close the connection and the server will purposely keep reading from client until the connection is closed.

3 Design

The general design of my program is to set up the initial client server socket connection. Then I created an endless while loop that would read the header from the client and store that into a vector. Then I would go through the vector and determine if it was a GET or PUT command. Depending on which one it was I would call a different function to deal with the functionality. In the GET and PUT functions is would read the contents from the client and store them in a file in the server directory if it was a PUT command, or it would read the contents from a file and send that data to the client if it was a GET command.

4 Pseudocode

SetPort()

- For loop for all argv[] arguments
    - Set portNumber variable if argv[2] exists

Parser(buffer,vector)

- strtok() on header
- While loop while strtok() element is not NULL

    - Push back each element into vector

asciCheck(string)

- If length of string is not 27 return false
- For loop for each character in the string
    - Check if letter,number, “-”, or “_” and if it isn’t return false

getFuctionality(vector, socket number)

- Open file with name taken from vector
- If file throws error then
    - Print 404 not found header
- Else if aschiCheck() returns false then
    - Print forbidden error
- Else if fstat() return error then
    - Print 400 bad request
- Else
    - Print ok header message
    - While read from file is > 0
        - Send buffer to client
- Close the file

putFunctionality(vector, socket number)

- Go through vector and look for content length
    - If found set boolean to true and grab store content length number in variable
- Open file with create flags filename from vector
- If file throws an error then
    - Send 500 internal error
- Else if asciCheck returns false then
    - Send forbidden header
- If size of content length is 0 then
    - Send created message and send file that's empty
- If content length is false then

        - Send created header
        - While recv > 0
            - Write to file
- Else
    - Send created header
    - While recv > 0
        - Write to file
        - Keep incrementing characters read
        - Break when content length is the same as total bytes read
- Close file

main()

- Socket
- Setsockopt
- Set socket to either ip or host name
- bind()
- listen()
- While(1)
    - accept()
    - read(buffer)
    - parser(buffer)
    - For vector elements
        - Comapare first element to either GET or PUT
        - Call approriate function
        - If neither get or put
            - Send 400 bad request
        - Closer socket
- Closer server

-