Design Document

Ricky Zhao
CruzID: rjzhao
CSE 130, Fall 2019

1 Goal

The goal of this program is to create a simple single-thread HTTP server. The commands we will be implementing is PUT and GET, which allows for client to read and write data from the server.

2 Assumptions

I am assuming that a GET request will request for the content of a file and print it out to the content of the file to standard output.

I am also assuming that the server will only handle one request per connection since there it is difficult to test multiple request using curl.

3 Design

The first step is to create a socket for the server. I will be using the function socket(), setsockop(), bind(), listen(), and accept() to create a socket for the server.

When binding the server, I will check if the arguments of the main function is greater than 1. If it is, I will check if there are two or three arguments. If there are two arguments, I will take the second argument converting it into an address and set the socket address to the second argument. If there are three arguments, I will set the second argument as the socket address and the third argument as the port number of the socket. If there is more than 3, I will throw an error of invalid number of arguments.

When the connection is accepted, we will have the program receive the HTTP header from the client server by using recv(). We will then parse the header to get the operation the client is trying to perform and the name of the file that the client specified.

Once we have parsed the header, we will check if the filename is 27 characters long and contains only the allowed characters. If the file name is not 27 characters long or contains special characters that are not allowed, we will send a 400 Bad Request code to the client. If the filename is formatted correctly, we will continue to perform the function the client requested.

If the request is a GET, we will check if the file the client is trying to get exist within the server’s directory or have permission to read it. If the file does not exist, we will send a 404

File not found code to the client. If we don’t have permission to read the file, we will send the client a 403 Forbidden code to the client. If the file is in the Server’s directory, we will open the file and attempt to open and read the file. Then we will try to send the content to the client using send (). Once we successfully sent the file to the client, we will send a 200 OK code to the message to the client.

If the request is a PUT, we will check if the file already exist in our server’s directory. If the file does not exist, we will create a new file with the filename we just parsed. If the file already exists, we will check if we have permission to read/write to the file. If we do not have permission, we will send the client a 403 Forbidden code. We will also continue to parse the HTTP Header to get the content length. We will then receive the data from the client and write it to the file specified by the client. If we have created a new file, we will send the client a 201 Created code. If the file already exists, we will send a 200 OK code instead.

If the client requested any functions other than GET or PUT, we will send the client a 500 Internal Server Error.

4 Pseudocode

Httpserver

```
Parse_header function ( header,index)
  String word
  While header[i] is not a space
     Word += header[i]
   Return word
Valid_filename(filename)
   For i=0 to filename.length()
      If filename[i] is not a-z,A-Z,”-“ or “_”
         Return false
   Return true


Create socket
  socket()
  setsockop()
  bind()
    if there are 2 arguments set the second to address
    if there are a third argument set the the third argument to port
  listen()
while accept()
  recv the header
  parse_header function
  parse_header filename
  parse_protocol
  if filename contains 27 character and is valid
    if get_function
      if file does not exist
         send code 404 to client
      else if permission denied
         send code 403 to client
      else
        read filename → buffer
        send buffer to client
    else if put_fuction
      if file does not exist
         create file
      else if permission denied
         send code 403 to client
      else
        parse content length→length
        recv the file of size length from client→buffer
        write buffer to the requested filename
    else
       send client code 500
  Else
```

```
    Send client code 400
Close socket
```