Design Document - Assignment 1

Derrick DeBose

1 Goal

The goal of this assignment is to create a HTTP protocol server in which we can use clients like curl in order to make get and put requests.

2 Assumptions

We are assuming that this code will be run on Ubuntu 18.04 VM. So, this code may not work if it were to run on a Mac or Windows system. We are going to assume that we are using an IPv4 address. Assume, no bad arguments will be given on the server side when starting up the server.

3 Design

I will start the socket program by creating a server socket. I will then assign attributes of the address from the command line arguments like ip address and port number. I can then bind the address to server file descriptor. Then we will listen for a client to connect to the server.

Then we will start a while loop where we can accept the client's connection. Read the request from the client that we can parse the request on newline in order to figure out the command, the filename, the HTTP/1.1 header for the response header and the content length if the request was a put request.

If not a get or put request, send a 500 status code.

We will check the see if the filename used is invalid, if invalid then return a 400 error. If we have a valid filename we can determine if request was Get, Put, or other.

If a get request then first get the content length of the file and send the response header. Then we can actually send over the data to the client of the file they want from the server.

If a put request then we want to know the difference between a 200 or 201 status conformation but send a 403 if we can not truncate the file that we are trying to overwrite. Then we will use the content length number we found when scanning the put request in, to determine how much data we need to recv from the client and write to the file.

Close the client socket, and continue to the top of the while loop in order to accept another connection from a client.

Lastly we need to close the server socket.

4 Pseudocode

We will use the Server algorithm as our main method for the program and the get and put method are called in the main method.

```
if argc ==1 then
    error("Not Enough Arguments");
end
create socket file descriptor;
address.family = IPv4 address type;
if argc == 2 then
    if argv[1] == "localhost" then
        address = localhost;
    end
    else
        address = argv[1];
    end
end
if argc == 3 then
    address.port = argv[2];
end
else
    address.port = 80 ;
end
bind the server file descriptor to the address;
listen for a client;
while 1 do
    Accept client's connect;
    read client's request;
    split request on new line;
    scan lines for command, filename, and content length;
    if not get or put request then
        error(500);
    end
    if invalid filename then
        error(400);
        close client socket;
        continue;
    end
    if command is a get request then
        get content length;
        call get function;
    end
    else if command is a put request then
        call put function;
        print(content length);
    end
    close client socket;
end
close server;
return 0;
```

Algorithm 1: Server

```
open the file;
while content length >0 do
    read the file;
    sent count = send the file to client socket;
    content length -= sent count
end
```

Algorithm 2: Get

```
open/create the file;
while content length >0 do
    receive the data from the client socket;
    write count = write to the file on the server;
    content length -= write count;
end
```

Algorithm 3: Put