Design Document
Mamon Alsalihy
CruzID:
malsalih
CSE 130-01, Fall 2019

1.Goal

The goal of Assignment1 is to get us familiar with the HTTP protocol. Things like the recv(), send(),bind(),accept(),socket() were all necessary to use. In Assignment1 in particular the goal was to handle PUT and GET requests from the client and send response headers with the appropriate status codes.

2.Assumptions

I am currently using a list of commands from cURL to handle the client side. These are provided in the README. I assumed we are allowed to use fstat() to get the data associated with the file like file size and I am using sprintf() to convert integer to string. For status codes, I am writing them as a response header to the client and closing the connection when necessary. If there is a PUT request of an empty file, I overwrite the file with nothing but I send a 500 internal server error and close connection. When and which status code I used was based on this Piazza post: @436.

3. Design

So part of the design includes the above assumptions. The approach I am taking is to check for valid arguments and if those arguments are valid for opening a server connection. Then I want to make sure that the file requested exists and is also a valid name. Once I get all those checks I parse the HTTP header based on the request identifier and send/recv the relevant data along with a response header indicating the current status of the server connection.

4. Pseudocode

```
Int main(){
    if(invalid number of arguments{
        Print out invalid number of arguments
    }
    Else{
        Open socket
        Gethostname()
        (Returns the address from struct in_addr and converts into network byte order if
        hostname is invalid it will return error and exit)
        Getport()
        (Validate input to see if it's a number and return either default port or user's port if
        number is invalid will return error and exit)
```

```
        Open connection and handle errors to stderr
        while(1){
            Read HTTP header and tokenize it to get request-identifier
            if(request-identifier is GET){
                (checks  if file name is 27 characters long and 64 ASCII)
                if(Validate Filename()){
                    Tell client it's a 400 Bad Request
                 }
                Else{
                    responseGET();
                }

            }
            Else if(request-identifier is PUT){
                 (checks  if file name is 27 characters long and 64 ASCII)
                if(Validate Filename()){
                    Tell client it's a 400 Bad Request
                 }
                Else{
                    responsePUT();
                }
            Else{
                HTTP1.1 500 Internal Server Error sent to client
            }
        }
}
gethostname(){
    if(if valid hostname){
        Send error and close connection
    }
    Else{
        Return address of hostname
    }
}
gettport(){
    if(invalid argument){
         Send error and close connection
    }
    Else{
        Return default port or user's port if arg count is 2
    }
}
validatefile(){
```

```
        If 27characters {
                For whole filename{
                        If character is between 0 and 9 or between a and z or between A and Z or
        - or _
                                Do nothing
                        Else{
                                Return 0
                        }
                Return 1
                }
        }
        Else{
                Return 0
        }
}
responseGET(){
        if(open file){
                if(error is because server doesn't have permission){
                        Send 403 forbidden
                }
                if(error is because server cannot find file){
                        Send 404 file not found
                }
        }
        Else{
                Read file and write content to the client with a response header indicating 200
        OK and ContentLength
        }
}
responsePUT(){
        if(file exists){
                handlefile(file exists indicator)
        }
        Else{
                handlefile(file exists indicator)
        }
}
handlefile(){
        if(open file){
                if(error is because server doesn't have permission){
                        Send 403 forbidden
                }
        }
```

```
        Else{
                Parse file for Content length
                if(content-length is zero){
                        Overwrite file,send 500 internal service error and close connection
                }
                Else{
                        write to file in server directory upto content-length from http header
                }
                if(cannot find content-length){
                        Write to file in server directory upto EOF from http header
                }
        }
}
```