Design

Zhijun Yang
CruzID: zyang100

CSE130, Fall 2019

1 Goal

The goal of this program is to implement a single-threaded HTTP server, and the server is able to respond GET and PUT commands to read and write “files”. PUT header consists a single line of string and content length. GET header only consists a single line of string

2 Assumptions

I think in order to complete this program, it is necessary to use socket, bind, listen, accept, connect, send, recv for network protocol. Also, I need to use read, write, close for file descriptor.
I assume the server starts with socket, then bind, listen to the socket. When it is accepted, the connection between client and server are established, so the server can receive the request from the client and send the data to the client.

3 Design

My approach is to set the port as 8888, use asgn0 code for file descriptors, dprintf() can also print to a file descriptor. The first step is to create socket and binds the socket to the address and port number. The next step is to listen to the ports, and make the connection between client and the server. The next step is to implement the program that can parse HTTP header, which is either PUT or GET from the client. Also, each HTTP header should consist a string using sscanf() for parsing data. If the string is not recognized, then it will print 404 error code or other codes like 403, 404, etc., otherwise it is 200(means ok) or 201(created).

Pseudocode:

```
Define port number
Define buffer size
Int main(){
    Bind()
    Listen()
While(1){
    Allocate space for request header and response header
    Accept
    processHttpRequest()
    returnHttpResponse()
    close()
}
}
Int processHttpRequest(){
    Use strtok and sscanf
If (!isvalidRequestPath(filename)){
    getHttpStatusHeader
    return-1;
}
while(token !=NULL){
    check for content length;
strcmp “GET” and “PUT”
}

Int isvalidRquestPath(){
    Return strcmp(path, compare)
Int get(){
    Open the file
    Use fstat
    getHttpStatusHeader
}
```

```
Int putInit{
        If(uploadfile <0){
                Open the file
                getHttpStatusHeader
        }

Int putdatahandler{
        If there is not contentlength
                Read and write
        }
        Else{
        Write the content into buffer
}

Void returnHttpResponse{
        Write the client socket
        If(responseFD <=1){
                Return;
        }
        While loop for read and write
}

Void getHttpStatusString{
        100, 200, 201,400,404, 500 status code
}

Void getHttpStatusHeader{
        Check for contetnlength
        If(contentlengtth){
        Print(header, "HTTP/1.1 contentlength"
        }
        Else{
        Print(header, "HTTP/1.1, httpstatus)
        }
```