**Design**

Zhijun Yang

CruzID: zyang100

CSE130, Fall 2019

**1 Goal**

The goal of this program is to modify assignment 1(HTTP server) and add two additional features: multi-threading and logging. Multi-threading is to let server hand multiple requests. Logging means to write the record for each request.

**2 Assumptions**

I think in order to complete this program, it is necessary to implement base on assignment 1. First of all, since I am going to implement multi-threading, it is necessary to improve the throughput, so I have to use a “pool” of “worker” threads available for use. For the logging requests, which is basically getting a record from the client. I would say it is similar to the header in assignment 1.

**3 Design**

My approach to this assignment is to use the code for assignment 1. I am probably going to implement multithreading using POSIX threads library, such as pthread\_create(), pthread\_mutex\_init(), etc. Also, I am going to set a “pool” of “worker” threads as default, which is 4. Since the server will never exit, so I am not going to put pthread\_exit. Instead, it is necessary to put a while loop to let the server keep receiving the requests from the client. For logging request, I think I am going to put a specific method to convert data to hex for log record. On the other hand, I need to consider the failure of a situation. For example, if the server returns an error response code, the log record should look like something different instead of hex.

*Pseudocode:*

*Define port number*

*Define BUF\_SIZE*

*void writelog(){*

*pthread\_mutex\_lock();*

*boolean writing*

*pthread\_mutex\_unlock();*

*}*

*void enqueue(){*

*pthread\_mutex\_lock();*

*push\_front;*

*pthread\_mutex\_unlock();*

*}*

*Int dequeue(){*

*Check task.queue is not empty;*

*}*

*Void dispatch(){*

*Enqueue();*

*Sem\_post();*

*}*

*Void processor (){*

*Client reads the buffer*

*}*

*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)*

*}*