Design

Zhijun Yang
CruzID: zyang100

CSE130, Fall 2019

1 Goal

The goal of this program is to modify the HTTP server, which is based on assignment 2, but there is additional feature which is caching. Caching means that you are going to maintain a buffer in your server that contains a subset of the pages. When a request is received, if the requested page is in the cache, then it is read from the cache (if it is a GET request) or updated in the cache (if it is a PUT request). Otherwise, the page is first read from disk into the cache.

2 Assumptions

I think in order to complete this program, it is necessary to implement base on assignment 2. First of all, since I am going to implement caching. It is helpful to use LRU and write through. The cache is going to be initially empty, and I feel like it should be dynamic allocation。

3 Design

My approach to this assignment is to use the code for assignment 2. I am probably going to implement caching using LRU. When requests are received, if the requested page is not in the cache, then it brings it to cache from disk and place it in the cache. The cache has a size limit. If the program is bringing a page from disk to cache and the cache is full, then the program decides what page to replace using the page replacement algorithm you chose. When a page is replaced, it will write it back if needed.

Pseudocode:
Define port number
Define BUF_SIZE
Define Cache_size

Typedef{
    Set the cache to the empty
}

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)
        }
```