Assignment 3: Caching for HTTP server - Design Document

Goal:
The goal of this program is to create a cache that will store files from PUT and GET requests so doing a GET request on the file in the cache will return a faster output

Handling arguments:
Arguments will be the same as assignment 1 with PUT and GET requests in the form of headers.

This design documentation will only cover the caching implementation as the rest of the implementation is in the assignment 1 design documentation.

-Caching will be turned on or off with the -c flag
-If the -c flag is not specified as an argument when starting the httpserver, then the program will work as intended in Assignment 1

-Else caching will be implemented in the GET and PUT functions.

-First define a struct for the files being stored, as well as the buffer (vector)

```
----------------------------------------------------------------------
| struct page {                                                      |
|    std::string name;                                               |
|    char* buffer;                                                   |
|    int dirty;                                                      |
| }                                                                  |
|                                                                    |
| std::vector<page> cache;                                           |
----------------------------------------------------------------------
```

-Now implement caching for PUT and GET
PUT:

```
----------------------------------------------------------------------
| //Define the buffer, and malloc it to give it the same amount of   |
| //space as the contentLength                                       |
| char* buffer = (char*)malloc(contentLength * sizeof(char));        |
| buffer[contentLength] = '\0';                                      |
|                                                                    |
| //Assuming that the caching flag is turned on                      |
| read(fd, buffer, contentLength);                                   |
| page myPage;                                                       |
| myPage.name = fileName;                                            |
| myPage.buffer = buffer;                                            |
|                                                                    |
| //First check if the item being called PUT on is in the buffer     |
| if(inCache) {                                                      |
|   //Compare if the page in the cache vs the page in header is same |
|   //If the pages are different, then replace the old page with the |
|   //new page.                                                      |
|   if(page is not same){                                            |
|     free(oldPage.buffer);                                          |
|     cache.erase(oldPage);                                          |
|     myPage.dirty = 1;                                              |
|     cache.push_back(myPage);                                       |
|   }                                                                |
|   //Else set the dirty bit to that of the oldPage and push new page|
|   else {                                                           |
|     myPage.dirty = oldPage.dirty;                                  |
|     cache.push_back(myPage);                                       |
|   }                                                                |
| }                                                                  |
--------------------------------------------------------------------
```

If the file is not in the cache:

```
----------------------------------------------------------------------
| else if(!inCache) {                                                |
|    //Check if cache size is equal to 4. If it else, pop the first  |
|    //file in the cache                                             |
|    if(cache.size() == 4) {                                         |
|      //If the first file in the cache is dirty, do a writeback to  |
|      //the disk.                                                   |
|      if(cache.begin()->dirty == 1) {                               |
|        writeFD = open(cache.begin()->name, O_RDWR,O_CREAT,O_TRUNC);|
|        write(writeFD, cache.begin()->buf, strlen(cache.begin->buf);|
|        close(writeFD);                                             |
|      }                                                             |
|      free(cache.begin()->buffer);                                  |
|      cache.erase(cache.begin());                                   |
|    }                                                               |
|    //Now check if the file exists. If not, mark the dirty bit      |
|    if(!file_exists) {                                              |
|      myPage.dirty = 1;                                             |
|      cache.push_back(myPage);                                      |
|    }                                                               |
|    //Else if the file exists, first chck if the file content is the|
|    //same as that of the page. If not, mark the dirty bit          |
|    else {                                                          |
|       int openFD = open(filename, O_RDWR);                         |
|       if(!myPage.buffer == filename.data) {                        |
|         myPage.dirty = 1;                                          |
|       }                                                            |
|       else {                                                       |
|         myPage.dirty = 0;                                          |
|       }                                                            |
|       close(openFD);                                               |
|       cache.push_back(myPage);                                     |
|    }                                                                 |
| }                                                                    |
------------------------------------------------------------------------
```

-This concludes caching for PUT requests

-Now for the implementation of caching for GET requests

```
------------------------------------------------------------------------
| //Assuming that the caching flag is on, first check if the file     |
| //is in the cache                                                    |
| auto it = cache.begin();                                             |
| for(; it != cache.end(); ++it) {                                     |
|    //If the file has been found, return the data from the cache     |
|    if(trimmedFileName = it->name) {                                  |
|      write(fd, it->buffer, strlen(it->buffer);                      |
|    }                                                                 |
| }                                                                    |
| //If the file is not found in the cache, open the file in the disk |
| //Assuming that Assignment 1 will use errno to check if file in     |
| //the disk will exist or not, and will take countermeasures if it  |
| //does not exist.                                                    |
|                                                                      |
| readFD = open(trimmedFileName, O_RDWR);                              |
| //Check if the cache is full. If it is, delete first file in the    |
| //cache. If the first file is dirty, do a writeback.                |
| if(cache.size() == 4) {                                              |
|   if(cache.begin()->dirty == 1) {                                    |
|     int writeFD = open(cache.begin()->name,O_RDWR|O_CREAT|O_TRUNC);|
|     write(writeFD, cache.begin()->buff, strlen(cache.begin()->buf);|
|     close(writeFD);                                                  |
|   }                                                                  |
|   free(cache.begin()->buffer);                                       |
|   cache.erase(cache.begin());                                        |
| }                                                                    |
| //Now define the page struct to push the file into disk to cache    |
| //And return that data to the client                                |
| char* buffer = (char *)malloc(fileSize * sizeof(char));              |
| buffer[fileSize] = '\0';                                             |
| int bytes_read = read(readFD, buffer, fileseize);                    |
| page myPage;                                                         |
| myPage.name = trimmedFileName;                                       |
| myPage.buffer = buffer;                                              |
| myPage.dirty = 0;                                                    |
| //Now push the page into the cache and return the data to client    |
| cache.push_back(myPage);                                             |
| write(fd, buffer, bytes_read);                                       |
| close(readFD);                                                       |
------------------------------------------------------------------------
```

With the implementation of caching done for both PUT and GET requests, the documentation for this program has now concluded.