Add aliases to the HTTP server

Name: Zhihang Zhou
ID: zzhou37

Goal:

To add the aliases to the HTTP server, so the server can support patch and alias. Patch can put the name of a file into hash table and aliases can use a new name to reference the name we just stored in the hash table. So, basically what we trying to do here is create a hash table which could mapping the key and value, which allowed the user to reference the name of the file without mention the original name.

Design:

System design:

Before we handle the request, the first thing we need to do is create a file as an array which has 8000 bucket and 125 bytes for each bucket. The we will use the magic function to check if the file is used to implement hash table. What we do here is we use a fixed number and put into hash function, then we find the index of the bucket and check if what we want is there. Our program could support the preexist hash table, if we don't have it, we will create one. Then, we should modify the parse message function to get the patch and aliases correctly. After that we need to search the key in hash table until we find the real 27-character file name, if we can't find it, we need to respond 404 file not found to client. But, if the 27-charcter file name itself is also valid, then store the key and value into the hash table.

Interconnection between different functions:

Most of the code is from assignment 2, all we need to add are some functions and modify parse header.
In modify parse header, if we received PUT or GET request, first we need to call read hash table to find if the file exist in hash table or not. If the file doesn't exist in hash table, just handle it like a normal request. If it does, replace the name with the real name in harsh table.
If the request is Patch and aliases, just call write function to write the file name into a hash table. The write function can handle different cases, if the key doesn't exist then respond 404

Design:

openHashTable:

open a hash table and confirm if it is the hash table we are looking for, if the file exists open the file and return the file descript. If the file doesn't exist, crate a hash table, write the magic file into the file and return the file descriptor.

Code:

```
if file doesn't exist
        create and open a magic file
        write the magic number in the last bucket
        return the FD
else if the file exists
        check the last bucket
        if the last bucket is magic number
                open the file
                return the FD
        else
                crate and open a hash Table file
```

write into hash table:

this function can write the key and value into hash table, if the key doesn't exist write 1 into the first bucket and write key and value into buckety.

code:

```
writeIntoHashTable (key, value)
hashSem.wait()
index = hash(value)
if the value exists
        mark the value as 0
        index = hash(key)
        write key and value into the index
else
        if the file name is valid
                index = hash(key)
                write key and value into the index
        else
                return 404
hashSem.post()
```

readHashTable:

This function take key as the input and trace all the way to the original and return the pointer of the real file name. If the key doesn't exist, then return NULL.

```
value readHashTable(key)
hashSem.wait()
```

```
keyCP = key
mark = the first byte in the bucket
while the mark is not 1
        index = hash(keyCP)
        keyCP = value
```

Synchronism:

Like what we implement in the assignment 2, we need to prevent each thread read and write the same file at the same time. So, I will create a semaphore as the lock of hash table, when use the hash table, the function will call wait on the semaphore and call post on it after it finished access it.