4th Group Homework Report

Simple HTTP Proxy

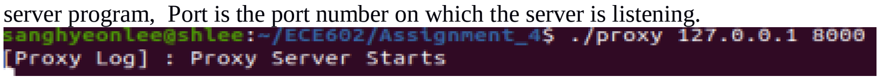
ECEN 602 Network Programming Assignment 4

Mainly, we worked together, but here is the role what we did.
Minhwan Oh : Developed client programming, tested for integration
Sanghyeon Lee : Developed server programming, debugged for integration

File info
===============
For this program, you can see how HTTP Proxy works.
1. proxy.cpp
Main feature: Server application for proxy server.
2. client.cpp
Main feature: proxy client
3. utility.h
Main feature: header file
4. Makefile
Main feature: Compile setting description

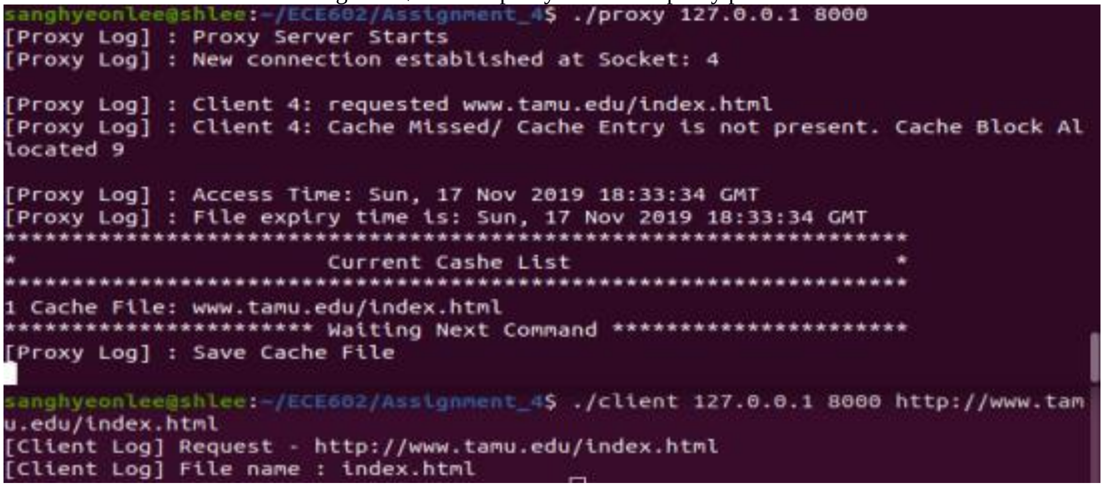
Build info
===============
Command $make

Program scenario
===============
1. Start the server first with the commanding line: $./proxy <ip to bind> <port to bind>, where “proxy” is the name of the server program, Port is the port number on which the server is listening.



```
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./proxy 127.0.0.1 8000
[Proxy Log] : Proxy Server Starts
```

2. Start client with the commanding line: $./client <proxy address> <proxy port> <URL to retrieve>

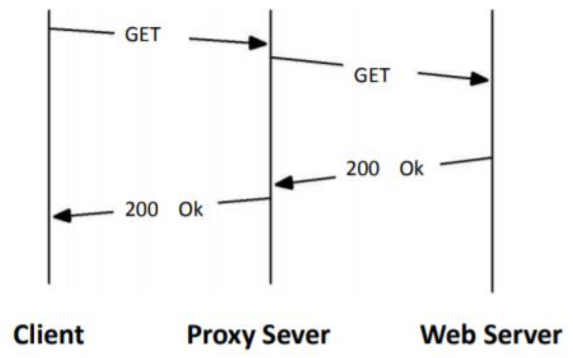


```
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./proxy 127.0.0.1 8000
[Proxy Log] : Proxy Server Starts
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested www.tamu.edu/index.html
[Proxy Log] : Client 4: Cache Missed/ Cache Entry is not present. Cache Block Al
located 9

[Proxy Log] : Access Time: Sun, 17 Nov 2019 18:33:34 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 18:33:34 GMT
*************************************************************************
*                        Current Cashe List                             *
*************************************************************************
1 Cache File: www.tamu.edu/index.html
*********************** Waiting Next Command ****************************
[Proxy Log] : Save Cache File

sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.tam
u.edu/index.html
[Client Log] Request - http://www.tamu.edu/index.html
[Client Log] File name : index.html
```

3. There are several test cases as below, and we comply with all test cases.
(1) A cache hit returns the saved data to the requester.
(2) A request that is not in the cache is proxied, saved in the cache, and returned to the requester.
(3) A cache miss with 10 items already in the cache is proxied, saved in the LRU location in cache, and the data is returned to the requester.
(4) A stale Expires header in the cache is accessed, the cache entry is replaced with a fresh copy, and the fresh data is delivered to the requester.
(5) A stale entry in the cache without an Expires header is determined based on the last Web server access time and last modification time, the stale cache entry is replaced with fresh data, and the fresh data is delivered to the requester.
(6) A cache entry without an Expires header that has been previously accessed from the Web server in the last 24 hours and was last modified more than one month ago is returned to the requester.
(7) Three clients can simultaneously access the proxy server and get the correct data (you will need to access large

documents to show this).
(8) If you implement the Bonus feature, develop test cases to show that it works properly; you should modify (5) – (6) above to match the operation of the Bonus

Program details & File architecture
===============
1. Proxy Server Design



- Server will start on a user specified port and listen for incoming HTTP requests.
- Proxy server checks the received client's request in its cache.
- If the request is new (i.e. no valid cache entry), the request will be proxied to the intended destination, the response will be sent by the proxy to the client, and be cached by the proxy for later use.
- Proxy cache can maintain at least 10 document entries in the cache.
- Entries followed the Least Recently Used (LRU) fashion.

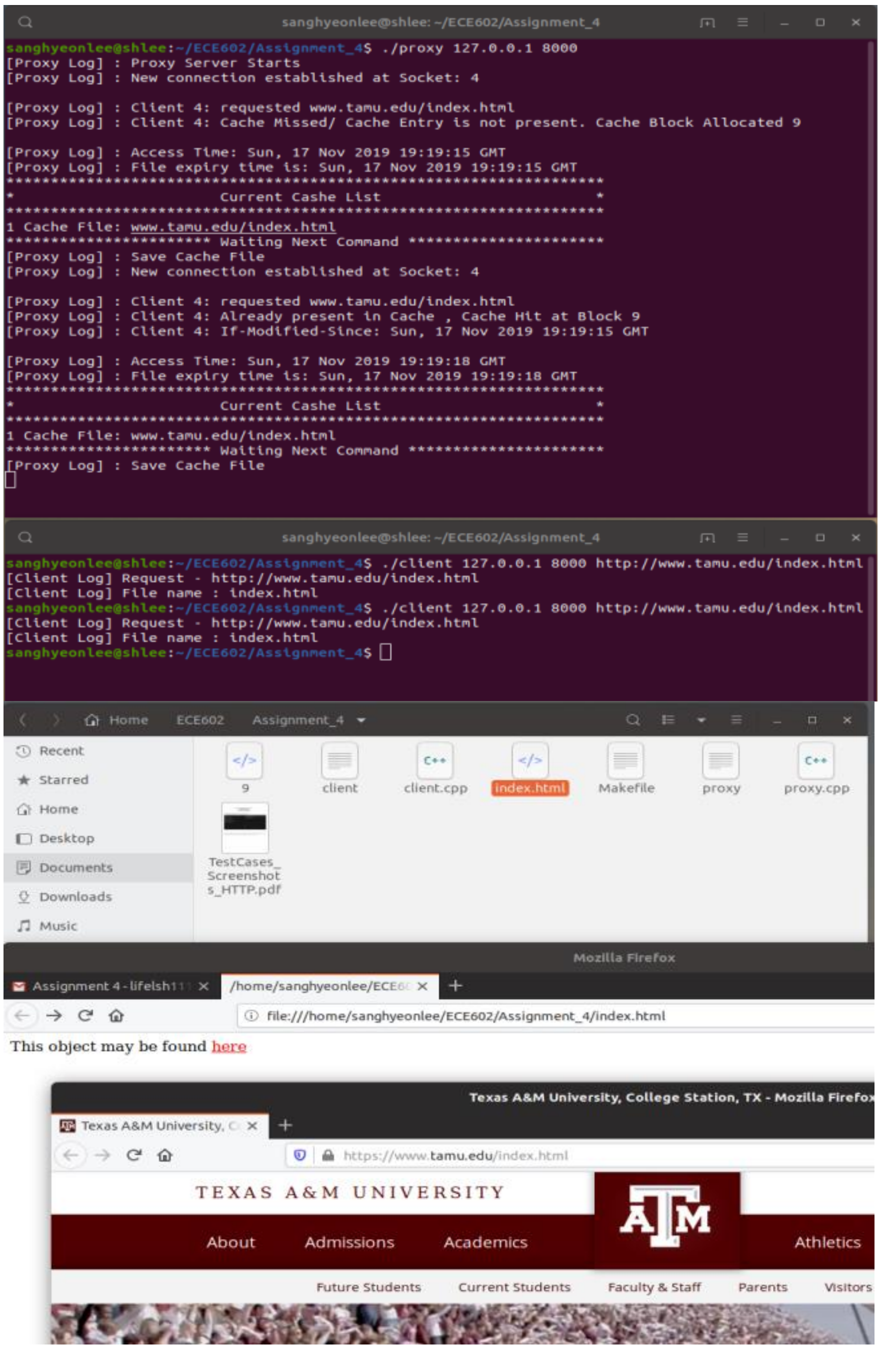
2. Proxy command client
- Client will take input from the command line as to the web proxy address and port as well as the URL to retrieve.
- The client will store the retrieved document in the directory where the client is executed.

3. fresh check (Bonus Feature)
- to determine if a cache page is "fresh" is to use the date in the Expires, Last-Modified, or Date header in conjunction with an If-Modified-Since message header line in a GET request, which is known as a Conditional Get, to ask the Web server if the page has changed.

File Function Explanation
===============
1. proxy server
a) Architecture
    - Scenario :
        (1) Setup server_info/bind port/listen functions
        (2) Then it is going to infinite loop until the exit
    - Main function :
        (1) `int main (int argc, char* argv[])` : Including main scenario
        (2) `struct http * parse_http_request (char * burf, int bytes_num)` : Decoding the host name and the file name by parsing through the HTTP request
        (3) `int checkCache (string req) :` Check for the existing request in the cache
        (4) `bool calc_expire_bool (int cacheBlock_index)`: Check for the stale cache

Test cases
===============

(1) A cache hit returns the saved data to the requester.



```
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./proxy 127.0.0.1 8000
[Proxy Log] : Proxy Server Starts
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested www.tamu.edu/index.html
[Proxy Log] : Client 4: Cache Missed/ Cache Entry is not present. Cache Block Allocated 9

[Proxy Log] : Access Time: Sun, 17 Nov 2019 19:19:15 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 19:19:15 GMT
*******************************************************************
*                       Current Cashe List                        *
*******************************************************************
1 Cache File: www.tamu.edu/index.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested www.tamu.edu/index.html
[Proxy Log] : Client 4: Already present in Cache , Cache Hit at Block 9
[Proxy Log] : Client 4: If-Modified-Since: Sun, 17 Nov 2019 19:19:15 GMT

[Proxy Log] : Access Time: Sun, 17 Nov 2019 19:19:18 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 19:19:18 GMT
*******************************************************************
*                       Current Cashe List                        *
*******************************************************************
1 Cache File: www.tamu.edu/index.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
```

```
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.tamu.edu/index.html
[Client Log] Request - http://www.tamu.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.tamu.edu/index.html
[Client Log] Request - http://www.tamu.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$
```

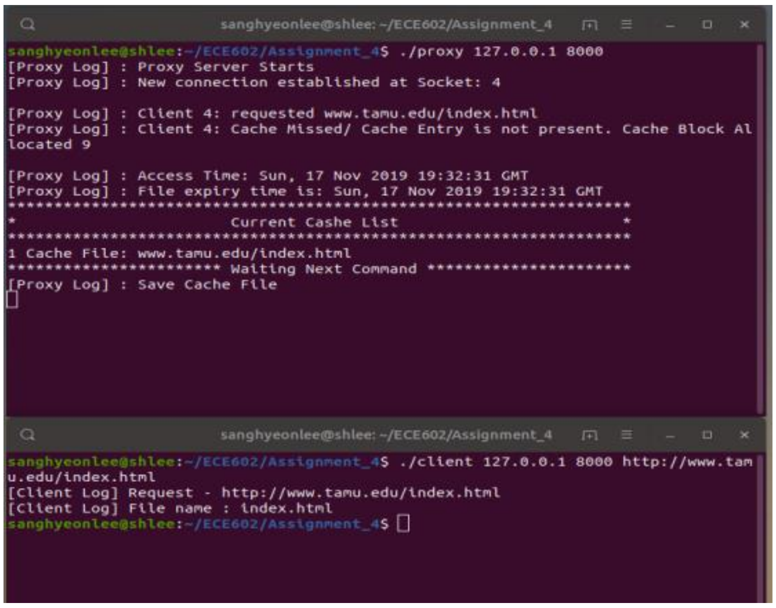
file:///home/sanghyeonlee/ECE602/Assignment_4/index.html

This object may be found here

https://www.tamu.edu/index.html

First, URL: http://www.tamu.edu/index.html was saved to cache.
Then, when it was requested again, Cache hit appears as [Cache Hit at Block 9] on the proxy log.

(2) A request that is not in the cache is proxied, saved in the cache, and returned to the requester.

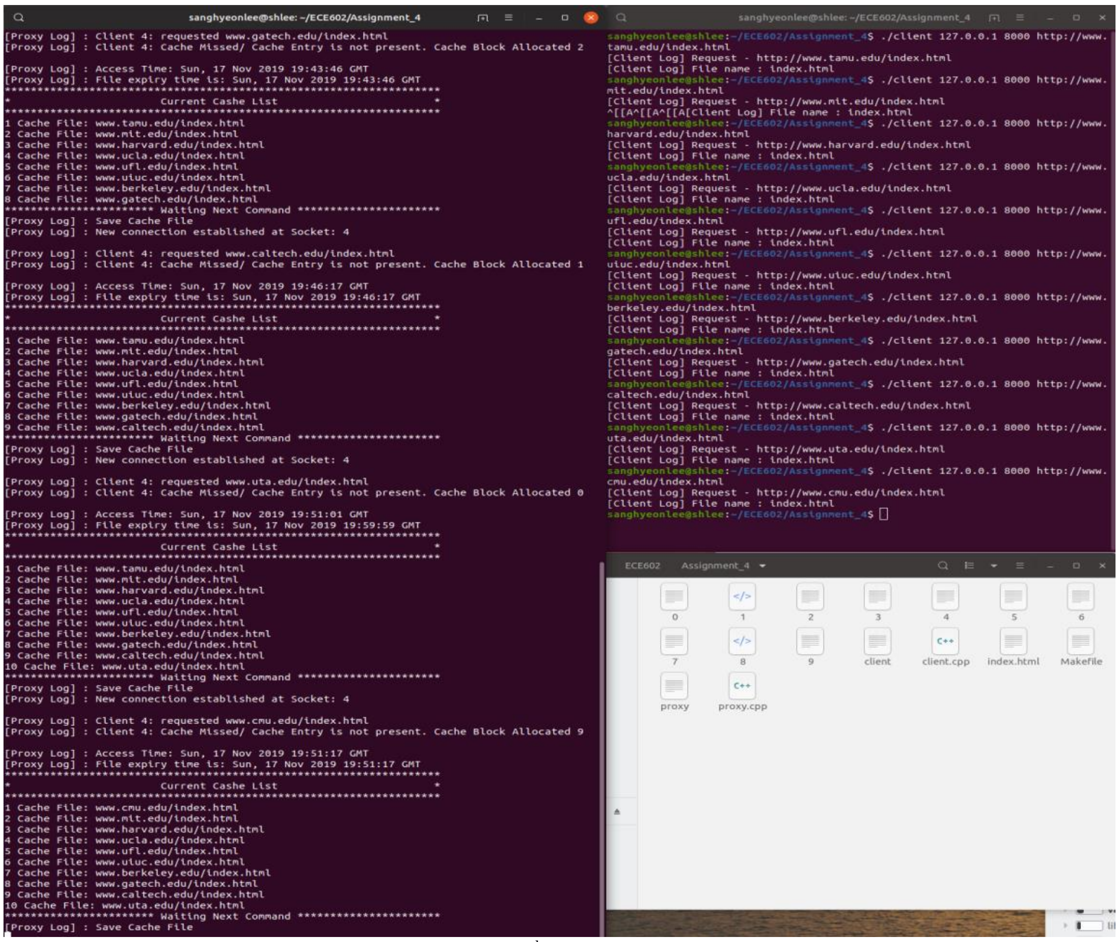


```
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./proxy 127.0.0.1 8000
[Proxy Log] : Proxy Server Starts
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested www.tamu.edu/index.html
[Proxy Log] : Client 4: Cache Missed/ Cache Entry is not present. Cache Block Al
located 9

[Proxy Log] : Access Time: Sun, 17 Nov 2019 19:32:31 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 19:32:31 GMT
*******************************************************************
*                       Current Cashe List                        *
*******************************************************************
1 Cache File: www.tamu.edu/index.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
```

```
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.tam
u.edu/index.html
[Client Log] Request - http://www.tamu.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$
```

-The URL http://www.tamu.edu/index.html is not present in the cache.
- The server proxies the request and saves it in the cache and return to the client.

(3) A cache miss with 10 items already in the cache is proxied, saved in the LRU location in cache, and the data is returned to the requester.

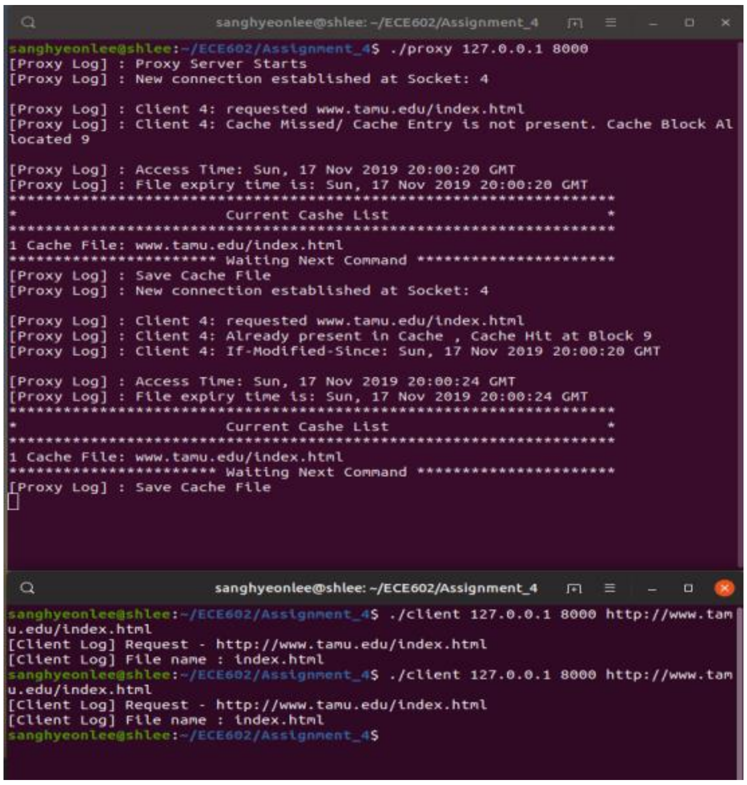


```
[Proxy Log] : Client 4: requested www.gatech.edu/index.html
[Proxy Log] : Client 4: Cache Missed/ Cache Entry is not present. Cache Block Allocated 2

[Proxy Log] : Access Time: Sun, 17 Nov 2019 19:43:46 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 19:43:46 GMT
*******************************************************************
*                       Current Cashe List                        *
*******************************************************************
1 Cache File: www.tamu.edu/index.html
2 Cache File: www.mit.edu/index.html
3 Cache File: www.harvard.edu/index.html
4 Cache File: www.ucla.edu/index.html
5 Cache File: www.ufl.edu/index.html
6 Cache File: www.uiuc.edu/index.html
7 Cache File: www.berkeley.edu/index.html
8 Cache File: www.gatech.edu/index.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested www.caltech.edu/index.html
[Proxy Log] : Client 4: Cache Missed/ Cache Entry is not present. Cache Block Allocated 1

[Proxy Log] : Access Time: Sun, 17 Nov 2019 19:46:17 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 19:46:17 GMT
*******************************************************************
*                       Current Cashe List                        *
*******************************************************************
1 Cache File: www.tamu.edu/index.html
2 Cache File: www.mit.edu/index.html
3 Cache File: www.harvard.edu/index.html
4 Cache File: www.ucla.edu/index.html
5 Cache File: www.ufl.edu/index.html
6 Cache File: www.uiuc.edu/index.html
7 Cache File: www.berkeley.edu/index.html
8 Cache File: www.gatech.edu/index.html
9 Cache File: www.caltech.edu/index.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested www.uta.edu/index.html
[Proxy Log] : Client 4: Cache Missed/ Cache Entry is not present. Cache Block Allocated 0

[Proxy Log] : Access Time: Sun, 17 Nov 2019 19:51:01 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 19:59:59 GMT
*******************************************************************
*                       Current Cashe List                        *
*******************************************************************
1 Cache File: www.tamu.edu/index.html
2 Cache File: www.mit.edu/index.html
3 Cache File: www.harvard.edu/index.html
4 Cache File: www.ucla.edu/index.html
5 Cache File: www.ufl.edu/index.html
6 Cache File: www.uiuc.edu/index.html
7 Cache File: www.berkeley.edu/index.html
8 Cache File: www.gatech.edu/index.html
9 Cache File: www.caltech.edu/index.html
10 Cache File: www.uta.edu/index.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested www.cmu.edu/index.html
[Proxy Log] : Client 4: Cache Missed/ Cache Entry is not present. Cache Block Allocated 9

[Proxy Log] : Access Time: Sun, 17 Nov 2019 19:51:17 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 19:51:17 GMT
*******************************************************************
*                       Current Cashe List                        *
*******************************************************************
1 Cache File: www.cmu.edu/index.html
2 Cache File: www.mit.edu/index.html
3 Cache File: www.harvard.edu/index.html
4 Cache File: www.ucla.edu/index.html
5 Cache File: www.ufl.edu/index.html
6 Cache File: www.uiuc.edu/index.html
7 Cache File: www.berkeley.edu/index.html
8 Cache File: www.gatech.edu/index.html
9 Cache File: www.caltech.edu/index.html
10 Cache File: www.uta.edu/index.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
```

```
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.
tamu.edu/index.html
[Client Log] Request - http://www.tamu.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.
mit.edu/index.html
[Client Log] Request - http://www.mit.edu/index.html
^[[A^[[A^[[A[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.
harvard.edu/index.html
[Client Log] Request - http://www.harvard.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.
ucla.edu/index.html
[Client Log] Request - http://www.ucla.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.
ufl.edu/index.html
[Client Log] Request - http://www.ufl.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.
uiuc.edu/index.html
[Client Log] Request - http://www.uiuc.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.
berkeley.edu/index.html
[Client Log] Request - http://www.berkeley.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.
gatech.edu/index.html
[Client Log] Request - http://www.gatech.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.
caltech.edu/index.html
[Client Log] Request - http://www.caltech.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.
uta.edu/index.html
[Client Log] Request - http://www.uta.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.
cmu.edu/index.html
[Client Log] Request - http://www.cmu.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$
```

- The requests are saved in the cache, also when the 11th request is stored in block 9 which is indication the LRU.

(4) A stale Expires header in the cache is accessed, the cache entry is replaced with a fresh copy, and the fresh data is delivered to the requester.

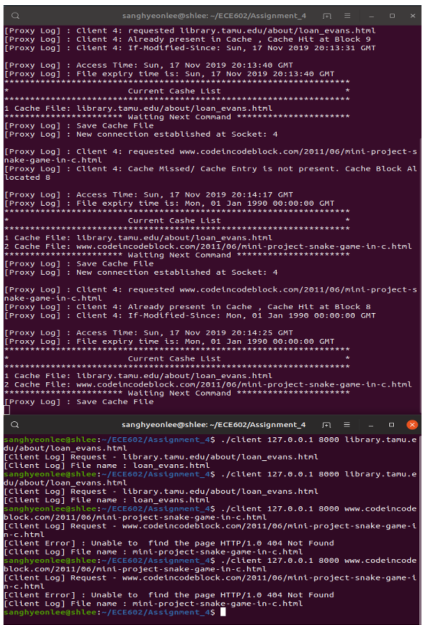


```
sanghyeonlee@shlee: ~/ECE602/Assignment_4
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./proxy 127.0.0.1 8000
[Proxy Log] : Proxy Server Starts
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested www.tamu.edu/index.html
[Proxy Log] : Client 4: Cache Missed/ Cache Entry is not present. Cache Block Al
located 9

[Proxy Log] : Access Time: Sun, 17 Nov 2019 20:00:20 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 20:00:20 GMT
*******************************************************************
*                        Current Cashe List                       *
*******************************************************************
1 Cache File: www.tamu.edu/index.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested www.tamu.edu/index.html
[Proxy Log] : Client 4: Already present in Cache , Cache Hit at Block 9
[Proxy Log] : Client 4: If-Modified-Since: Sun, 17 Nov 2019 20:00:20 GMT

[Proxy Log] : Access Time: Sun, 17 Nov 2019 20:00:24 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 20:00:24 GMT
*******************************************************************
*                        Current Cashe List                       *
*******************************************************************
1 Cache File: www.tamu.edu/index.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
```

```
sanghyeonlee@shlee: ~/ECE602/Assignment_4
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.tam
u.edu/index.html
[Client Log] Request - http://www.tamu.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.tam
u.edu/index.html
[Client Log] Request - http://www.tamu.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$
```

- The expiry time is passed, the entry is fetched using a Conditional GET and the cache entry is replaced to the new copy and fresh data is delivered to the requester.

(5) A stale entry in the cache without an Expires header is determined based on the last Web server access time and last modification time, the stale cache entry is replaced with fresh data, and the fresh data is delivered to the requester.

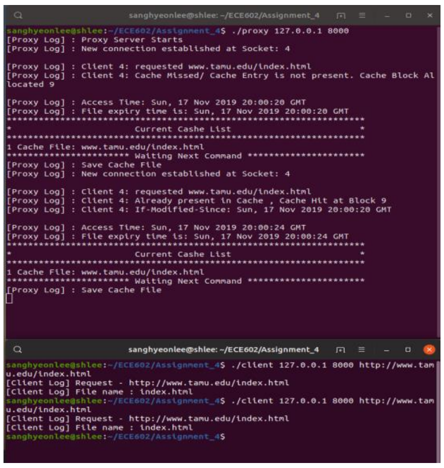


```
sanghyeonlee@shlee: ~/ECE602/Assignment_4
[Proxy Log] : Client 4: requested library.tamu.edu/about/loan_evans.html
[Proxy Log] : Client 4: Already present in Cache , Cache Hit at Block 9
[Proxy Log] : Client 4: If-Modified-Since: Sun, 17 Nov 2019 20:13:31 GMT

[Proxy Log] : Access Time: Sun, 17 Nov 2019 20:13:40 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 20:13:40 GMT
*******************************************************************
*                        Current Cashe List                       *
*******************************************************************
1 Cache File: library.tamu.edu/about/loan_evans.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested www.codeincodeblock.com/2011/06/mini-project-s
nake-game-in-c.html
[Proxy Log] : Client 4: Cache Missed/ Cache Entry is not present. Cache Block Al
located 8

[Proxy Log] : Access Time: Sun, 17 Nov 2019 20:14:17 GMT
[Proxy Log] : File expiry time is: Mon, 01 Jan 1990 00:00:00 GMT
*******************************************************************
*                        Current Cashe List                       *
*******************************************************************
1 Cache File: library.tamu.edu/about/loan_evans.html
2 Cache File: www.codeincodeblock.com/2011/06/mini-project-snake-game-in-c.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested www.codeincodeblock.com/2011/06/mini-project-s
nake-game-in-c.html
[Proxy Log] : Client 4: Already present in Cache , Cache Hit at Block 8
[Proxy Log] : Client 4: If-Modified-Since: Mon, 01 Jan 1990 00:00:00 GMT

[Proxy Log] : Access Time: Sun, 17 Nov 2019 20:14:25 GMT
[Proxy Log] : File expiry time is: Mon, 01 Jan 1990 00:00:00 GMT
*******************************************************************
*                        Current Cashe List                       *
*******************************************************************
1 Cache File: library.tamu.edu/about/loan_evans.html
2 Cache File: www.codeincodeblock.com/2011/06/mini-project-snake-game-in-c.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
```

```
sanghyeonlee@shlee: ~/ECE602/Assignment_4
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 library.tamu.e
du/about/loan_evans.html
[Client Log] Request - library.tamu.edu/about/loan_evans.html
[Client Log] File name : loan_evans.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 library.tamu.e
du/about/loan_evans.html
[Client Log] Request - library.tamu.edu/about/loan_evans.html
[Client Log] File name : loan_evans.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 www.codeincode
block.com/2011/06/mini-project-snake-game-in-c.html
[Client Log] Request - www.codeincodeblock.com/2011/06/mini-project-snake-game-i
n-c.html
[Client Error] : Unable to  find the page HTTP/1.0 404 Not Found
[Client Log] File name : mini-project-snake-game-in-c.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 www.codeincode
block.com/2011/06/mini-project-snake-game-in-c.html
[Client Log] Request - www.codeincodeblock.com/2011/06/mini-project-snake-game-i
n-c.html
[Client Error] : Unable to  find the page HTTP/1.0 404 Not Found
[Client Log] File name : mini-project-snake-game-in-c.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$
```

- Html could be seen there is no expires header and the stale entry is determined based on the last web server access time and the modified time.

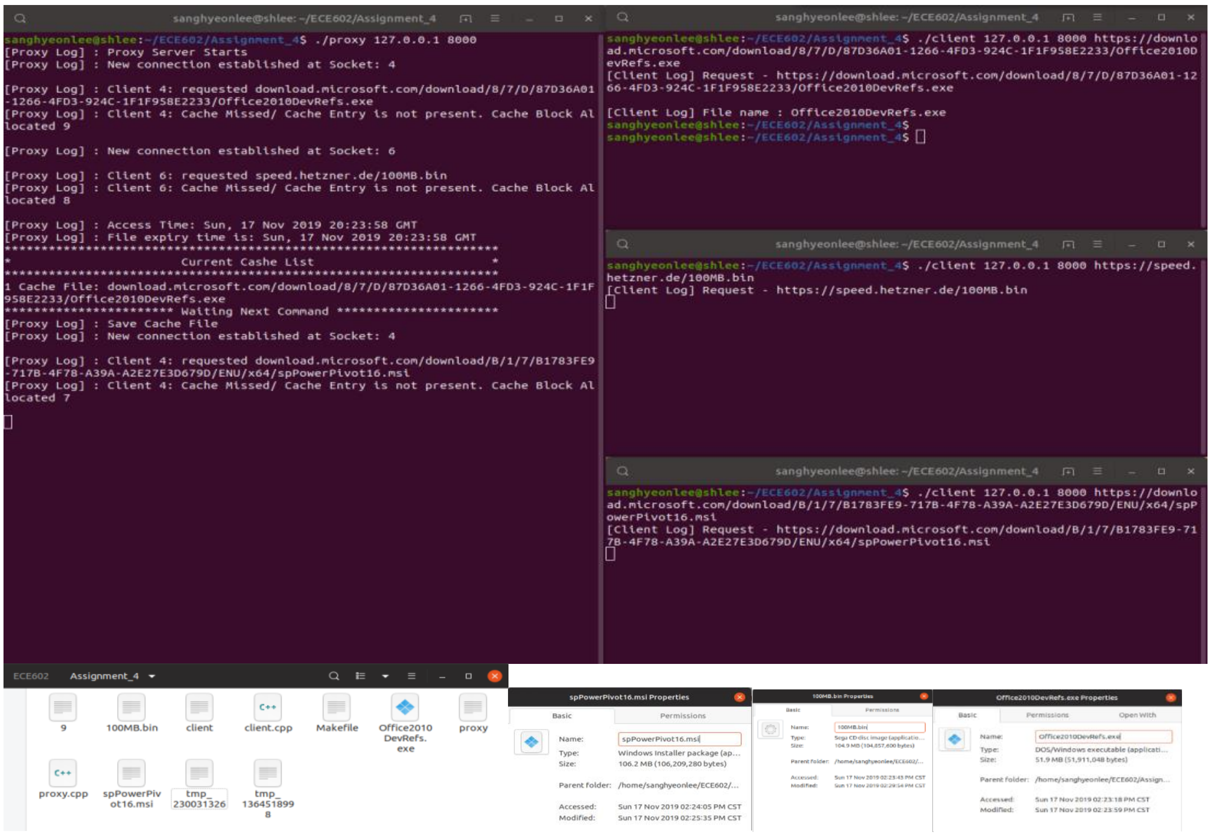
(6) A cache entry without an Expires header that has been previously accessed from the Web server in the last 24 hours and was last modified more than one month ago is returned to the requester.



```
sanghyeonlee@shlee: ~/ECE602/Assignment_4
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./proxy 127.0.0.1 8000
[Proxy Log] : Proxy Server Starts
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested www.tamu.edu/index.html
[Proxy Log] : Client 4: Cache Missed/ Cache Entry is not present. Cache Block Al
located 9

[Proxy Log] : Access Time: Sun, 17 Nov 2019 20:00:20 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 20:00:20 GMT
*******************************************************************
*                     Current Cashe List                          *
*******************************************************************
1 Cache File: www.tamu.edu/index.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested www.tamu.edu/index.html
[Proxy Log] : Client 4: Already present in Cache , Cache Hit at Block 9
[Proxy Log] : Client 4: If-Modified-Since: Sun, 17 Nov 2019 20:00:20 GMT

[Proxy Log] : Access Time: Sun, 17 Nov 2019 20:00:24 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 20:00:24 GMT
*******************************************************************
*                     Current Cashe List                          *
*******************************************************************
1 Cache File: www.tamu.edu/index.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
```

```
sanghyeonlee@shlee: ~/ECE602/Assignment_4
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.tam
u.edu/index.html
[Client Log] Request - http://www.tamu.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.tam
u.edu/index.html
[Client Log] Request - http://www.tamu.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$
```

(7) Three clients can simultaneously access the proxy server and get the correct data (you will need to access large documents to show this).



```
sanghyeonlee@shlee: ~/ECE602/Assignment_4
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./proxy 127.0.0.1 8000
[Proxy Log] : Proxy Server Starts
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested download.microsoft.com/download/8/7/D/87D36A01
-1266-4FD3-924C-1F1F958E2233/Office2010DevRefs.exe
[Proxy Log] : Client 4: Cache Missed/ Cache Entry is not present. Cache Block Al
located 9

[Proxy Log] : New connection established at Socket: 6

[Proxy Log] : Client 6: requested speed.hetzner.de/100MB.bin
[Proxy Log] : Client 6: Cache Missed/ Cache Entry is not present. Cache Block Al
located 8

[Proxy Log] : Access Time: Sun, 17 Nov 2019 20:23:58 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 20:23:58 GMT
*******************************************************************
*                     Current Cashe List                          *
*******************************************************************
1 Cache File: download.microsoft.com/download/8/7/D/87D36A01-1266-4FD3-924C-1F1F
958E2233/Office2010DevRefs.exe
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested download.microsoft.com/download/B/1/7/B1783FE9
-717B-4F78-A39A-A2E27E3D679D/ENU/x64/spPowerPivot16.msi
[Proxy Log] : Client 4: Cache Missed/ Cache Entry is not present. Cache Block Al
located 7
```

```
sanghyeonlee@shlee: ~/ECE602/Assignment_4
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 https://downlo
ad.microsoft.com/download/8/7/D/87D36A01-1266-4FD3-924C-1F1F958E2233/Office2010D
evRefs.exe
[Client Log] Request - https://download.microsoft.com/download/8/7/D/87D36A01-12
66-4FD3-924C-1F1F958E2233/Office2010DevRefs.exe

[Client Log] File name : Office2010DevRefs.exe
sanghyeonlee@shlee:~/ECE602/Assignment_4$
sanghyeonlee@shlee:~/ECE602/Assignment_4$
```

```
sanghyeonlee@shlee: ~/ECE602/Assignment_4
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 https://speed.
hetzner.de/100MB.bin
[Client Log] Request - https://speed.hetzner.de/100MB.bin
```

```
sanghyeonlee@shlee: ~/ECE602/Assignment_4
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 https://downlo
ad.microsoft.com/download/B/1/7/B1783FE9-717B-4F78-A39A-A2E27E3D679D/ENU/x64/spP
owerPivot16.msi
[Client Log] Request - https://download.microsoft.com/download/B/1/7/B1783FE9-71
7B-4F78-A39A-A2E27E3D679D/ENU/x64/spPowerPivot16.msi
```

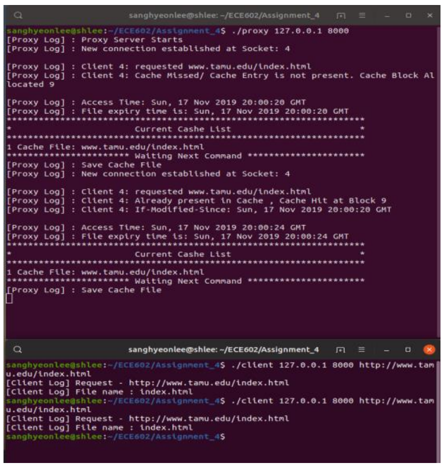
spPowerPivot16.msi Properties
Basic | Permissions
Name: spPowerPivot16.msi
Type: Windows Installer package (ap…
Size: 106.2 MB (106,209,280 bytes)
Parent folder: /home/sanghyeonlee/ECE602/…
Accessed: Sun 17 Nov 2019 02:24:05 PM CST
Modified: Sun 17 Nov 2019 02:25:35 PM CST

100MB.bin Properties
Basic | Permissions
Name: 100MB.bin
Type: Sega CD disc image (applicatio…
Size: 104.9 MB (104,857,600 bytes)
Parent folder: /home/sanghyeonlee/ECE602/…
Accessed: Sun 17 Nov 2019 02:23:43 PM CST
Modified: Sun 17 Nov 2019 02:29:54 PM CST

Office2010DevRefs.exe Properties
Basic | Permissions | Open With
Name: Office2010DevRefs.exe
Type: DOS/Windows executable (applicati…
Size: 51.9 MB (51,911,048 bytes)
Parent folder: /home/sanghyeonlee/ECE602/Assign…
Accessed: Sun 17 Nov 2019 02:23:18 PM CST
Modified: Sun 17 Nov 2019 02:23:59 PM CST

Three clients can be simultaneously connected when large files were being accessed.
In this case, we requested 50MB exe file, 100MB bin file, and 100MB msi file.

(8) If you implement the Bonus feature, develop test cases to show that it works properly; you should modify (5) – (6) above to match the operation of the Bonus



```
sanghyeonlee@shlee: ~/ECE602/Assignment_4
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./proxy 127.0.0.1 8000
[Proxy Log] : Proxy Server Starts
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested www.tamu.edu/index.html
[Proxy Log] : Client 4: Cache Missed/ Cache Entry is not present. Cache Block Al
located 9

[Proxy Log] : Access Time: Sun, 17 Nov 2019 20:00:20 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 20:00:20 GMT
*******************************************************************
*                        Current Cashe List                       *
*******************************************************************
1 Cache File: www.tamu.edu/index.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
[Proxy Log] : New connection established at Socket: 4

[Proxy Log] : Client 4: requested www.tamu.edu/index.html
[Proxy Log] : Client 4: Already present in Cache , Cache Hit at Block 9
[Proxy Log] : Client 4: If-Modified-Since: Sun, 17 Nov 2019 20:00:20 GMT

[Proxy Log] : Access Time: Sun, 17 Nov 2019 20:00:24 GMT
[Proxy Log] : File expiry time is: Sun, 17 Nov 2019 20:00:24 GMT
*******************************************************************
*                        Current Cashe List                       *
*******************************************************************
1 Cache File: www.tamu.edu/index.html
*********************** Waiting Next Command **********************
[Proxy Log] : Save Cache File
```

```
sanghyeonlee@shlee: ~/ECE602/Assignment_4
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.tam
u.edu/index.html
[Client Log] Request - http://www.tamu.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$ ./client 127.0.0.1 8000 http://www.tam
u.edu/index.html
[Client Log] Request - http://www.tamu.edu/index.html
[Client Log] File name : index.html
sanghyeonlee@shlee:~/ECE602/Assignment_4$
```

If-Modified-Since is implemented in all test cases.
If the html file exists in the cache and it is stale, the proxy issues a Conditional GET to the web server using If-Modified-Since header in the request. It contains the date from the expiry. If the expiry header is absent, the Last-Modified or the Data timestamps is used to check the html file's expiry.

utility.h

```
/*
*HTTP Proxy Utility Header
*/

#ifndef __HTTP_PROXY_H__
#define __HTTP_PROXY_H__

#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sstream>
#include <fstream>
#include <algorithm>
#include <iostream>
#include <string>
#include <map>
#include <time.h>

using namespace std;
int max_cache_capacity;
unsigned short int client_num;

map<int,struct condition> req_parameters;
map<int,int> client_request;
map<int,int> get_request;
map<int,int> type;
```

```
map<string,int> block_number;
map<int,string> URL;
map<int,bool> Random_number;


struct http{
		char file_name[1024];
		char host_name[1024];
};

struct cache_block{
		string expire_date;
		string host_file;
		int lastused;
		int current;
		int expire;
};
struct cache_block cache[10];

/* Structure to represent the request parameters */
struct condition{
		int expires;
		int length;
		int cb_index;
		int index;

		char filename[200];
		bool boolean;
		bool conditional;
};

char * http(char *url)
{
		int length = strlen(url);
		char *out = (char *)malloc((length+50)*sizeof(char));
		int i=0, j=4;

		out[0]='G';out[1]='E';out[2]='T';out[3]=' ';

		if(length>7)
		{
				if(url[0]=='h' && url[1]=='t' && url[2]=='t' && url[3]=='p' && url[4]==':' && url[5]=='/' && url[6]=='/')
						i = 7;
				if(url[0]=='h' && url[1]=='t' && url[2]=='t' && url[3]=='p' && url[4]=='s' && url[5]==':' && url[6]=='/' &&
url[7]=='/')
						i = 8;
		}

		int temp = i;

		while(i<length && url[temp]!='/')
				temp++;

		out[j++]='/';
		temp++;
		while(temp<length)
			out[j++]=url[temp++];

		out[j++]=' ';out[j++]='H';out[j++]='T';out[j++]='T';out[j++]='P';out[j++]='/';
		out[j++]='1';out[j++]='.';out[j++]='0';out[j++]='₩r';out[j++]='₩n';
		out[j++]='H';out[j++]='o';out[j++]='s';out[j++]='t';out[j++]=':';out[j++]=' ';

		while(i<length && url[i]!='/')
			out[j++] = url[i++];

		out[j++]='₩r';out[j++]='₩n';out[j++]='₩r';out[j++]='₩n';out[j]='₩0';

		return out;
}

char *get_filename(char *file)
{
		char *file_name = (char *)malloc(512*sizeof(char));
		int j=0;
		int last = 0;
		int i=0;
		int length = strlen(file);

		while(i < length)
		{
			if(file[i]=='/')
				last = i;
			i++;
		}
		if(last!=0)
		{
			memcpy(file_name,&file[last+1],(length-last-1));
			file_name[length-last]='₩0';
		}
		if(strlen(file_name)==0)
		{
			file_name[j++]='i';file_name[j++]='n';file_name[j++]='d';file_name[j++]='e';file_name[j++]='x';file_name[j++]='.';

			file_name[j++]='h';file_name[j++]='t';file_name[j++]='m';file_name[j++]='l';file_name[j]='₩0';
			return file_name;
		}
		return file_name;
}


// Updating Cache objects
void update(int block)
{
		if(cache[block].lastused==1)
			return;
		cache[block].lastused = 1;
		for(int i=0; i<10; i++)
		{
			if(i!=block)
			{
				cache[i].lastused++;
```

```
                        if(cache[i].lastused>10)
                                cache[i].lastused = 10;
                }
        }
        return;
}

// Updating LRU Cache
int update_lru()
{
        int end = 10;
        int ans = -1;
        int i;
        while(end > 0)
        {
                for(i=0; i<10; i++)
                {
                        if(cache[i].lastused==end && cache[i].current==0)
                        {

                                ans = i;
                                cache[ans].current=-2;
                                end = 0;
                                break;
                        }
                }
                end--;
        }
        return ans;
}

void show_cache_list(int nList)
{
        int i,j, num;
        num = 10 - nList;
        printf("**********************************************************************₩n");
        printf("*                    Current Cashe List                    *₩n");
        printf("**********************************************************************₩n");

        for( i = 0; i < 10; i++)
        {
                j = 9 - i;
                if(cache[j].host_file.empty()==true)
                        break;
                cout << i+1 << " Cache File: " << cache[j].host_file  << endl;
        }
        printf("*********************** Waiting Next Command **********************₩n");
}

// Below we shall check for the stale cachee
bool calc_expire_bool(int cacheBlock_index)
{
        time_t raw_time;
        time(&raw_time);
        struct tm * utc;
        utc = gmtime(&raw_time);
        raw_time = mktime(utc);
        if(cache[cacheBlock_index].expire-raw_time > 0)
        {
                return false;
        }
        return true;
}

// Check for the exisiting request in the cache
int checkCache(string req)
{
        if(block_number.find(req)==block_number.end())
        {
                return -1;
        }

        int cacheBlock_index = block_number[req];

        if(cache[cacheBlock_index].host_file == req)
        {
                return cacheBlock_index;
        }
        return -1;
}

// Decoding buffer into the host name and the file name
struct http * parse_http_request(char *buf, int bytes_num)
{
        struct http* out = (struct http *)malloc(sizeof(struct http));
        char* strtmp = (char *)malloc(64);
        int temp;

        for(temp=0; temp<bytes_num; temp++)
        {
                if(buf[temp]==' ')
                {
                        temp++;
                        break;
                }
        }
        int j=0;
        out->file_name[0]='/';

        if(buf[temp+8]=='₩r' && buf[temp+9]=='₩n')
        {
                out->file_name[1]='₩0';
        }
        else
        {
                while(buf[temp]!=' ')
                        out->file_name[j++]=buf[temp++];
                out->file_name[j]='₩0';
        }
        temp+=1;
        while(buf[temp++]!=' ');
```

```
		j=0;

		while(buf[temp]!='₩r')
				out->host_name[j++]=buf[temp++];

		out->host_name[j]='₩0';

		return out;
}


#endif
```

Makefile

```
PROGS = proxy client
CLEANFILES = *.o
CC = g++

all:		${PROGS}

proxy:		proxy.o
				${CC} -o $@ proxy.o

client:		client.o
				${CC} -o $@ client.o



clean:
				rm -f ${PROGS} ${CLEANFILES}
```

proxy.cpp

```

#include "utility.h"

using namespace std;

int main(int argc, char *argv[])
{


		max_cache_capacity = 10;
		client_num = 0;
		int sock_fd, error;
		int i, bytes_num;
		char buffer[2048],TB[256],tmp[512];

		struct addrinfo hints, *server_info, *p;



		if(argc!=3)
		{
				fprintf(stderr,"[Proxy Error] : Invalid number of arguments : Server Ip Port URL₩n");
				return 1;
		}
```

```
memset(&hints,0,sizeof(hints));
hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;


// Accessing the server
if((error = getaddrinfo(argv[1], argv[2], &hints, &server_info)) != 0)
{
        fprintf(stderr, "[Proxy Error] : getaddrinfo error: %s₩n", gai_strerror(error));
        exit(1);
}

for(p=server_info; p!=NULL; p=p->ai_next)
{

        if((sock_fd = socket(p->ai_family, p->ai_socktype, p->ai_protocol))== -1){
                perror("[Proxy Error] : Socket error");
                continue;
        }

        if((error = bind(sock_fd, p->ai_addr, p->ai_addrlen))== -1){
                perror("[Proxy Error] : Bind error₩n");
                continue;
        }
        break;
}

if(p == NULL)
{
        fprintf(stderr, "[Proxy Error] : Bind failed₩n");
        return 2;
}


if( (error = listen(sock_fd, max_cache_capacity)) == -1)
{
        perror("[Proxy Error] : Listen failed");
}


freeaddrinfo(server_info);

fd_set master, read_fds, write_fds, master_write;
FD_ZERO(&master);
FD_ZERO(&read_fds);
FD_ZERO(&write_fds);
FD_ZERO(&master_write);
FD_SET(sock_fd,&master);

int fdmax = sock_fd, c_sockfd;
struct sockaddr_storage client_addr;
socklen_t addr_length;

// Initialize cache
```

```
            for(int j=0; j<10; j++)
            {
                        cache[j].lastused=j+1;
                        cache[j].current=0;
            }


            struct tm tm,*utc;
            time_t rawtime;

            printf("[Proxy Log] : Proxy Server Starts ₩n");


            while(1)
            {
              read_fds =  master;
      write_fds = master_write;

              /*Performing select for multiclient operations*/
                        if( (error = select(fdmax+1, &read_fds, &write_fds, NULL, NULL)==-1)){
                                    perror("[Proxy Error] : Select");
                                    exit(4);
                        }


                        for(i=0; i<=fdmax; i++)
                        {
                          if(FD_ISSET(i,&write_fds))
                           {
                                                if(FD_ISSET(i,&master)==false)
                                                {
                                                  FD_CLR(i,&master);
                                                            FD_CLR(i,&master_write);
                                                            close(i);


                                                            if(req_parameters[i].cb_index==-1)
                                                            {
                                                                        if(req_parameters[i].boolean==true)
                                                                                                remove(req_parameters[i].filename);
                                                            }

                                                            else if(req_parameters[i].cb_index != -1)
                                                            {
                                                                        update(req_parameters[i].cb_index);
                                                                        cache[req_parameters[i].cb_index].current-=1;
                                                            }
                                                            continue;
                                        }

                                                /*Opening the file and putting the contents*/
                                                ifstream first_file;

                                                first_file.open(req_parameters[i].filename,ios::in | ios::binary);

                                                if(!first_file.is_open())
                                                {
                                                            cout << req_parameters[i].filename << endl;
                                                            FD_CLR(i,&master_write);
                                                            cout << "[Proxy Error] : Unable to Open Cache File - " <<
            req_parameters[i].filename << endl;
                                                }
                                                else
                                                {
                                                            first_file.seekg((req_parameters[i].index)*2048,ios::beg);
                                                            req_parameters[i].index+=1;
                                                            first_file.read(buffer,2048);

                        bytes_num = first_file.gcount();
                        if(bytes_num!=0)
                                                            {
                                                                        if( (error = send(i,buffer,bytes_num,0)) == -1)
                                                                        {
                                                                                        perror("[Proxy Error] : Client send
            error");
                                                                        }
                                                            }

                                                            // Disconnecting once end of file is reached
                                                            if(bytes_num<2048)
                                                            {
                                                                        FD_CLR(i,&master_write);
                                                                        FD_CLR(i,&master);
                                                                        close(i);

                                                                        if(req_parameters[i].cb_index==-1)
                                                                        {

                                                                                cout << "[Proxy Error] : Unable to save Cache
            File" << endl;
                                                                                if(req_parameters[i].boolean==true)
                                                                                {

                                                                                        cout << "[Proxy Error] : Unable to
            Open Cache File" << endl;

                                                                                        remove(req_parameters[i].filename);

                                                                                }
                                                                        }
                                                                        else if(req_parameters[i].cb_index != -1)
                                                                        {
                                                                                update(req_parameters[i].cb_index);
                                                                                cache[req_parameters[i].cb_index].current-=1;
                                                                                cout << "[Proxy Log] : Save Cache File" << endl;

                                                                        }
                                                            }

                                                }

                                                first_file.close();
                                    }
                        else if(FD_ISSET(i,&read_fds))
                                                {
                                                        /*Handling the new clients getting connected to the server*/

                                                        if(i==sock_fd)
                                                        {

                                                                addr_length = sizeof(client_addr);

                                                                if( (c_sockfd = accept(sock_fd, (struct sockaddr
            *)&client_addr,&addr_length)) == -1)
                                                                {
                                                                        perror("[Proxy Error] : Accept error");
                                                                }
                                                                else
                                                                {
                                                                        FD_SET(c_sockfd,&master);
                                                                        type[c_sockfd]=0;
                                                                        client_num++;
                                                                        if(c_sockfd > fdmax)
                                                                                fdmax = c_sockfd;
                                                                        cout << "[Proxy Log] : New connection established at Socket:
            " << c_sockfd << endl;
                                                                }
                                                        }
                                                        else
                                                        {
                                                                if((bytes_num = recv(i,buffer,sizeof(buffer),0)) <=0)
                                                                {
                                                                                if(bytes_num == 0)
                                                                                {
                                                                          }
                                                                          else
                                                                                {
                                                                                        perror("[Proxy Error] : Recevie error");
                                                                          }

                                                                                close(i);
                                                                          FD_CLR(i,&master);

                                                                                if(type[i]==1)
                                                                                {



                            FD_SET(client_request[i],&master_write);
                                                                                                        int j = client_request[i];
                                                                                                        int a =
            req_parameters[j].cb_index;
                                                                                                        if(a==-1){
                                                                                                        }
                                                                                                        else
                                                                                                        {// now we shall serve
            from the cache block after copying tmp file to cache block

                                                                                                                ifstream
            third_file;
```

```
                third_file.open(req_parameters[j].filename,ios::in | ios::binary);

                                                                                                    string line;
                                                                                                    bool
boo=false;


                if(third_file.is_open())
                                                                                                    {

                getline(third_file,line);

                // now we shall check if the cache is modified or not


                if(line.find("304")!=string::npos)

                boo = true;

                bool flag = false;


                while(!third_file.eof())

                {

                getline(third_file,line);



                if(line.compare("₩r") == 0)

                {

                                break;

                }



                string line_cp = line;

                transform(line.begin(), line.end(), line.begin(), ::tolower);



                //Checking and getting the expiry field

                if(line.find("expires:")==0)

                {

                                flag = true;

                                int pos = 8;
```

```
			if(line[pos]==' ')
				pos++;

			stringstream s2;
			s2 << line.substr(pos);

			getline(s2,line);
			line = line_cp.substr(pos);
			cache[a].expire_date = line;
			memset(&tm, 0, sizeof(struct tm));

			strptime(line.c_str(), "%a, %d %b %Y %H:%M:%S ", &tm);
			cache[a].expire = mktime(&tm);
			break;
		}
		}
		third_file.close();

		if(!flag)
		{

		time(&rawtime);
		utc = gmtime(&rawtime);

		strftime(TB, sizeof(TB), "%a, %d %b %Y %H:%M:%S ",utc);
		cache[a].expire = mktime(utc);
		cache[a].expire_date = string(TB) + string("GMT");
```

```
                }
                                                                                                  }

                time(&rawtime);
                                                                                                  utc =
gmtime(&rawtime);
                                                                                                  strftime(TB,
sizeof(TB), "%a, %d %b %Y %H:%M:%S ",utc);
                                                                                                  cout <<
"[Proxy Log] : Access Time: " << TB << "GMT" << endl;
                                                                                                  cout <<
"[Proxy Log] : File expiry time is: " << cache[a].expire_date << endl;

                                                                                                  if(boo ==
true)
                                                                                                  {

                //Request served from the cache block

                printf("[Proxy Log] : Serving from the cache block ₩n");

                if(req_parameters[j].conditional)

                {

                remove(req_parameters[j].filename);

                stringstream s4;

                s4 << a;

                strcpy(tmp,s4.str().c_str());

                strcpy(req_parameters[j].filename,tmp);

                req_parameters[j].boolean = false;

                }
                                                                                                  }

                if(boo==false)
                                                                                                  {

                if(cache[a].current!=1)

                {

                if(cache[a].current>1)

                {
                    cache[a].current-=1;

                    int new_cb;

                    new_cb = update_lru();



                    if(new_cb==-1)

                    {

                            req_parameters[j].boolean = true;

                            req_parameters[j].cb_index = new_cb;

                            a = new_cb;

                    }

                    else

                    {

                            req_parameters[j].cb_index = new_cb;

                            cache[new_cb].expire = cache[a].expire;

                            cache[new_cb].expire_date = cache[a].expire_date;

                            a = new_cb;

                    }

            }

            }

            else

            {

            cache[a].current = -2;

            }

                                                                                                        }
                                                                                                        /*Updating
the cache block and removing temp*/
                                                                                                        if(a!=-1 &&
cache[a].current==-2)
                                                                                                        {

            stringstream s3;

            s3 << a;
            strcpy(tmp,s3.str().c_str());


            ofstream mf;

            mf.open(tmp,ios::out | ios::binary);

            mf.close();

            remove(tmp);


            rename(req_parameters[j].filename,tmp);

            strcpy(req_parameters[j].filename,tmp);


            cache[a].host_file = string(URL[client_request[i]]);

            block_number[cache[a].host_file]=a;

            req_parameters[j].boolean = false;

            cache[a].current = 1;
                                                                                                   }



            show_cache_list(a);

                                                                                         }

                                                                       }
                                                                       else
                                                                       {
                                                                                 cout << "[Proxy Log] : Client number
" << i << " is disconnected" << endl;
                                                               }

                                                   }

                                                   else if(type[i]==1)
                                                   {
                                                             ofstream tmp_file;

            tmp_file.open(req_parameters[client_request[i]].filename,ios::out | ios::binary | ios::app);

                                                             if(!tmp_file.is_open())
                                                             {
                                                                       cout << "[Proxy Error] : Error opening the file"
<< endl;
                                                             }
                                                             else
                                                             {
                                                                       tmp_file.write(buffer,bytes_num);
```

```
                            tmp_file.close();
                        }
                    }

                    else if(type[i]==0)
                    {
                        struct http* tmp;
                        tmp = parse_http_request(buffer,bytes_num);

                        URL[i] = string(tmp->host_name)+string(tmp->file_name);
                        cout <<endl<< "[Proxy Log] : Client "<< i << ": requested "
<< URL[i] << endl;

                        int cacheBlock_index = checkCache(URL[i]);
                        bool expired = false;
                        req_parameters[i].conditional = false;

                        if(cacheBlock_index!=-1)
                        {
                            if(cache[cacheBlock_index].current>=0)
                            {

                                expired =
calc_expire_bool(cacheBlock_index);

                                if(expired)
                                {

    req_parameters[i].conditional = true;
                                }

                                else
                                {

    req_parameters[i].cb_index = cacheBlock_index;
                                    req_parameters[i].index
= 0;

    req_parameters[i].boolean = false;


    cache[cacheBlock_index].current += 1;

                                    stringstream ss;
                                    ss << cacheBlock_index;


    strcpy(req_parameters[i].filename,ss.str().c_str());
                                    FD_SET(i,&master_write);

                                    cout << "[Proxy Log] :
Cache Block Hit at block number :  "<< cacheBlock_index << endl;
                                    continue;
                                }
                            }
                        }
                        if(cacheBlock_index==-1 || expired)
                        {
                                string tmp_str = "GET "+string(tmp->file_name)+"
HTTP/1.0₩r₩nHost: "+string(tmp->host_name)+"₩r₩n₩r₩n";
                                bytes_num = tmp_str.length();
                                strcpy(buffer,tmp_str.c_str());

                                if(!expired)
                                {
                                        cacheBlock_index = update_lru();

                                        cout << "[Proxy Log] : Client "<< i
<< ": New or erased info. Cache Block Created " << cacheBlock_index << endl;
                                }
                                else
                                {
                                        cache[cacheBlock_index].current+=1;
                                        buffer[bytes_num-2]='₩0';
                                        string req = string(buffer);

                                        // now if the request if stale we shall
send a GET request which shall conatain If-modified-since in header

                                        string new_req = req+"If-Modified-
Since: "+cache[cacheBlock_index].expire_date +"₩r₩n₩r₩n₩0";

                                        bytes_num=new_req.length();
                                        strcpy(buffer,new_req.c_str());
                                        cout << "[Proxy Log] : Client " << i
<< ": Already present in Cache , Cache Hit at Block "<< cacheBlock_index << endl;

                                        cout << "[Proxy Log] : Client " << i
<< ": If-Modified-Since: "<< cache[cacheBlock_index].expire_date << endl;

                                        //cout<<"Status 304 :Served from
proxy because of IF-modified"<<endl;
                                }
                                req_parameters[i].cb_index = cacheBlock_index;


                                int new_socket;
                                memset(&hints,0,sizeof(hints));
                                hints.ai_family = AF_INET;
                                hints.ai_socktype = SOCK_STREAM;
                                // fetching the address
                                if((error = getaddrinfo(tmp->host_name, "80",
&hints, &server_info)) != 0)

                                {
                                        fprintf(stderr, "[Proxy Error] :
getaddrinfo error: %s₩n", gai_strerror(error));

                                        exit(1);
                                }
                                for(p=server_info; p!=NULL; p=p->ai_next)
                                {
                                        if((new_socket = socket(p->ai_family,
p->ai_socktype, p->ai_protocol))== -1)

                                        {               // we shall create socket
for the server

                                                        perror("[Proxy
Error] :Socket fail");
                                        continue;
                        }
                        break;
        }
        if(p==NULL)
        {
                        fprintf(stderr, "[Proxy Error] :bind
error₩n");
                        return 2;
        }
        if( (error = connect(new_socket,p->ai_addr,p-
>ai_addrlen)) == -1)
        {
                        perror("[Proxy Error] :connect
error₩n");
        }
        freeaddrinfo(server_info);

        type[new_socket]=1;
        get_request[i] = new_socket;
        client_request[new_socket] = i;

        req_parameters[i].boolean = true;

        req_parameters[i].index = 0;
        stringstream ss;

        srand(time(NULL));
        int rr = rand();
        while(Random_number.find(rr) !=
Random_number.end() )
        {

                        rr = (rr+1)%1000000007;
        }

        Random_number[rr]=true;
        ss << "tmp_"<< rr;
        strcpy(req_parameters[i].filename,ss.str().c_str());

        ofstream touch;
        touch.open(req_parameters[i].filename,ios::out |
ios::binary);


        if(touch.is_open())
                        touch.close();

        FD_SET(new_socket,&master);
        if(new_socket > fdmax)
                        fdmax = new_socket;
        cout<<endl;

        if( (error = send(new_socket,buffer,bytes_num,0))
== -1)
        {
```

```
								perror("[Proxy Error] : Send error");
							}


						}

						free(tmp);
					}

				}

			}



		}


	}

	return 0;
}
```

client.cpp

```
#include "utility.h"

using namespace std;

int main(int argc, char *argv[])
{
	/*regular connection sequence*/
	int error;
	if(argc!=4)
	{
		fprintf(stderr,"[Client Error] : Should type < Server IP > < Port number > < URL >₩n");
		return 1;
	}
	struct addrinfo hints;
	struct addrinfo *servinfo, *p;

	memset(&hints,0,sizeof(hints));
	hints.ai_family = AF_INET;
	hints.ai_socktype = SOCK_STREAM;

	if( (error = getaddrinfo(argv[1], argv[2], &hints, &servinfo)) != 0){
		fprintf(stderr, "[Client Error] : Getaddrinfo error: %s₩n", gai_strerror(error));
		exit(1);
	}

	int sockfd;
	for(p=servinfo; p!=NULL; p=p->ai_next)
	{
		if((sockfd = socket(p->ai_family, p->ai_socktype, p->ai_protocol))== -1){
			perror("[Client Error] : Socket Failed");
```

```
                                        continue;
                                }
                                break;
                }

                if(p==NULL)
                {
                                fprintf(stderr, "[Client Error] : Connect failed₩n");
                                return 2;
                }

                int size = 2048, bytes_num;
                char *buffer = (char *)malloc((size+1)*sizeof(char));
                unsigned short int length;

                if( (error = connect(sockfd,p->ai_addr,p->ai_addrlen)) == -1)
                {
                                perror("[Client Error] : Connect Failed");
                }
                freeaddrinfo(servinfo);

                char *query = http(argv[3]);

length = strlen(query);
cout << "[Client Log] Request - " << argv[3] << endl;
                if( (error = send(sockfd,query,length,0)) == -1)
                {
                                perror("[Client Error] : Send Failed");
                }

                ofstream testFile;
                char *filename = get_filename(argv[3]);
                stringstream ss;
                ss << filename;

                testFile.open(ss.str().c_str(),ios::out | ios::binary);
                if(!testFile.is_open())
                {
                                cout << "[Client Error] : Unable to open file in filesystem" << endl;
                                exit(0);
                }

                bool header = true;
                bool first=true;

                while(1)
                {
                                int i = 0;
                                bytes_num = recv(sockfd,buffer,2048,0);
                                if(first)
                                {
                                                stringstream tempstream;
                                                tempstream << buffer;
                                                string line;
                                                getline(tempstream,line);
                                                if(line.find("404")!=string::npos)
                                        cout << "[Client Error] : Unable to  find the page " << line << endl;
                                first = false;
                        }

                        if(header)
                        {
                                if(bytes_num>=4)
                                {
                                        for(i=3; i<bytes_num; i++)
                                        {
                                                if(buffer[i]=='₩n' && buffer[i-1]=='₩r')
                                                {
                                                        if(buffer[i-2]=='₩n' && buffer[i-3]=='₩r')
                                                        {
                                                                header=false;

                                                                i++;
                                                                break;
                                                        }
                                                }
                                        }
                                }
                        }

                        if(!header)
                                testFile.write(buffer+i,bytes_num-i);

                        if(bytes_num == -1)
                        {
                                perror("[Client Error] : Recv Failed ");
                        }
                        else if(bytes_num == 0)
                        {
                                close(sockfd);
                                cout << "[Client Log] File name : " << filename << endl;
                                testFile.flush();
                                testFile.close();
                                break;
                        }
                        testFile.flush();
                }
                testFile.close();
                return 0;
}
```