수업 명 : 시스템프로그래밍

과제 이름 : Proxy 1-2

학 과: 컴퓨터정보공학부

담당 교수님: 김태석 교수님

분 반: 월5, 수6

학 번: 2023202043

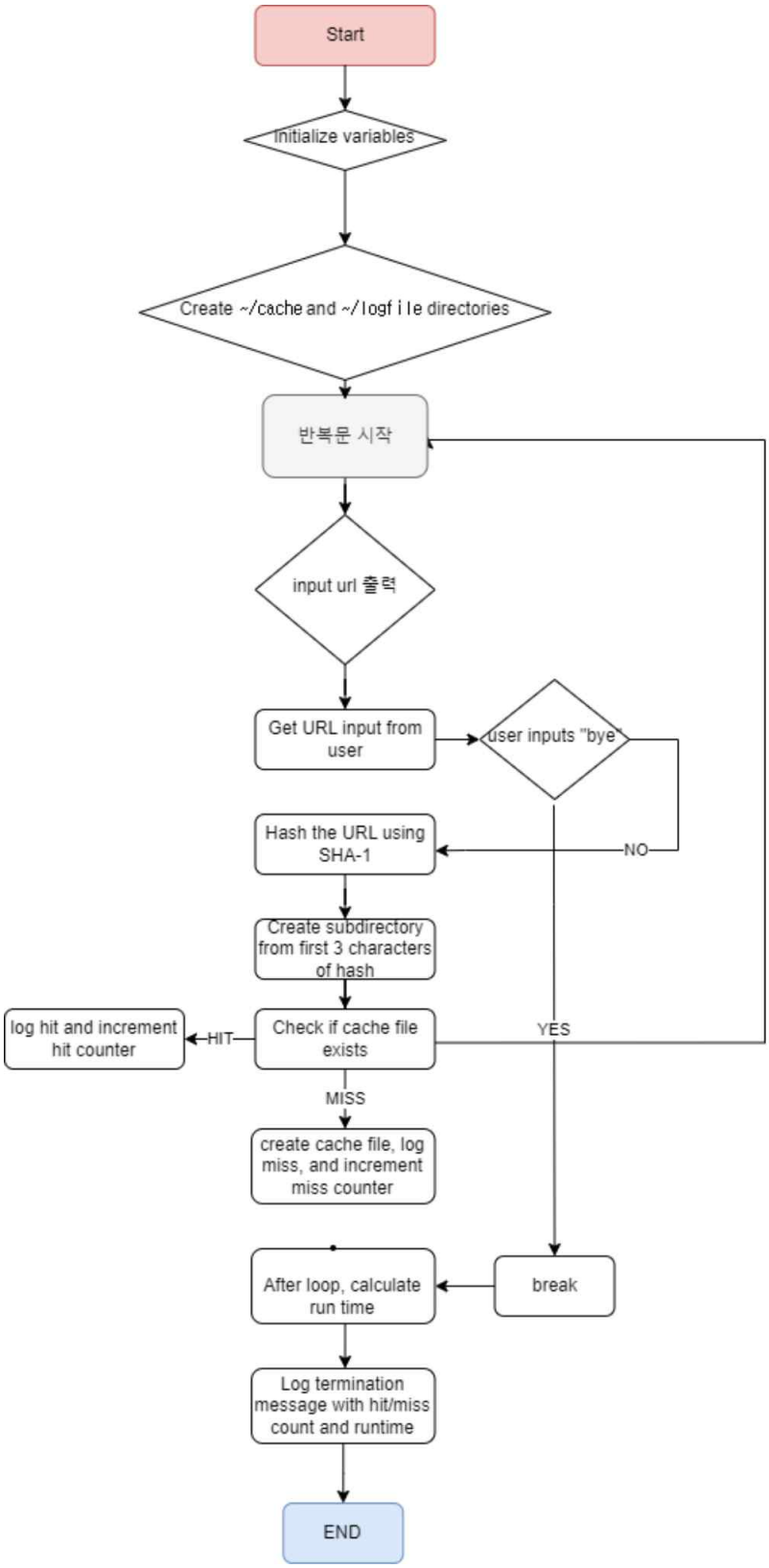
성 명: 최은준

0. Introduction

이 과제에서는 URL 요청을 처리하는 프록시 서버의 캐시 시스템을 구현하는 것을 목표로 한다. 입력된 URL에 대해 SHA-1 해시 함수를 적용하여 고유한 캐시 경로를 생성하고, 이를 기반으로 캐시 HIT 또는 MISS를 판단하여 로그에 기록한다. 캐시된 데이터를 파일로 저장되도록 하여 이 파일에는 요청 시각, HIT/MISS 여부, URL 또는 해시 경로 등이 포함되도록 구현한다.

1. Flow Chart

1-1. proxy server #1-2 코드 작성 순서도



위 코드 순서도의 동작 방식은 다음과 같다.

1. 사용자로부터 URL 입력을 받아 SHA-1 해시를 계산한다.

2. 해시값의 앞 세 자리를 디렉터리 이름으로 사용하고 나머지 값은 파일 이름으로 사용한다.

3. 해당 경로에 캐시 파일이 존재하면 HIT, 존재하지 않으면 MISS로 판별한다.

4. MISS일 경우 해당 경로에 캐시 파일을 생성하고, 로그 파일에 결과를 기록한다.

5. bye 입력 시 프로그램이 종료되며, 총 실행 시간과 HIT/MISS 횟수가 로그 파일에 기록된다.

2. Pseudo code

2-1. proxy server #1-2 알고리즘

Function: sha1_hash(input_url, hashed_url) Convert the input URL to a SHA-1 hash Store the 160-bit hash as a 40-character hexadecimal string Copy the hexadecimal string into hashed_url Return the hashed_url
Function: getHomeDir(home) Get the current user's ID Retrieve the user's password and home directory info using the ID Copy the home directory path into home Return the home directory path
Function: is_cache_hit(path, filename) Combine the path and filename into a full file path Check if the file exists using the access() system call If the file exists, return 1 (cache hit) Otherwise, return 0 (cache miss)
Function: write_log(status, info) Get the current user's home directory Construct the full path to the log file Get the current time and format it into readable date and time Open the log file in append mode If file opens successfully: Write a log entry in the format: [status]info-[yyyy/mm/dd, hh:mm:ss]

Close the log file

Function: main()
Initialize variables for URL input, hashed value, and home directory
Start timer to record the program start time
Initialize hit and miss counters to 0

Get the user's home directory
Create a "cache" directory under the home directory
Create a "logfile" directory under the home directory
Build the full path to the log file

Loop infinitely:
    Prompt the user for a URL
    Read the input URL

    If the input is "bye":
        Exit the loop

    Call sha1_hash() to hash the URL
    Extract the first 3 characters of the hash to determine the cache subdirectory
    Build the full cache directory path
    Create the cache subdirectory if it doesn't exist
    Build the full file path for the cache file (subdirectory + remaining hash)

    Get the current time

    Call is_cache_hit() to check if the file already exists:
        If the file exists (cache hit):
            Increase hit count
            Format log info including the hash and URL
            Write a [Hit] log entry with details
        Else (cache miss):
            Increase miss count
            Create and write dummy data to the new cache file
            Write a [Miss] log entry with URL and timestamp

After exiting the loop:
    Get the current time as the end time

Calculate total run time by subtracting start from end

Write a final [Terminated] log entry

Include runtime, hit count, and miss count

Return 0 to indicate successful execution

3. 결과화면

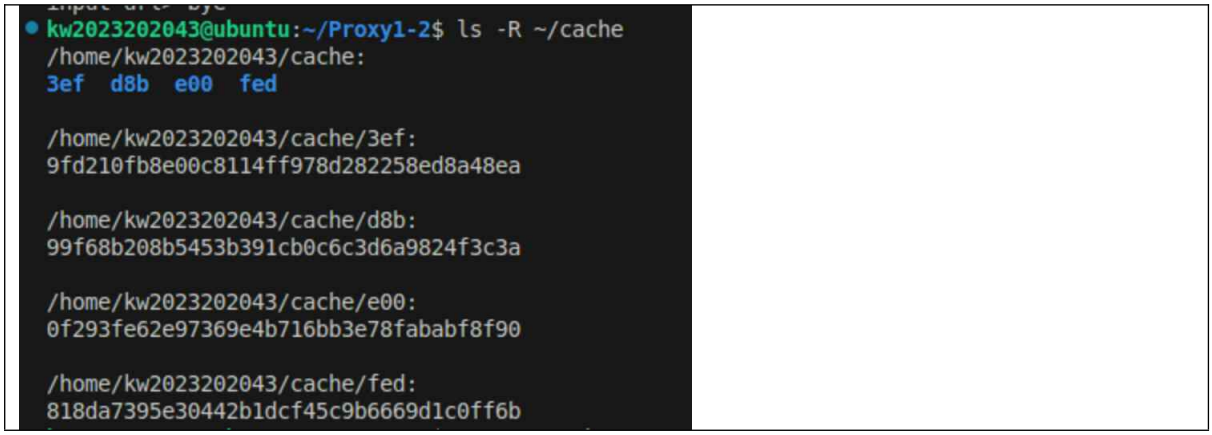
```
kw2023202043@ubuntu:~/Proxy1-2$ ls
Makefile  proxy_cache.c
kw2023202043@ubuntu:~/Proxy1-2$ make
gcc -Wall -Wextra -g -o proxy_cache proxy_cache.c -lcrypto
proxy_cache.c: In function 'sha1_hash':
proxy_cache.c:35:19: warning: comparison of integer expressions of different s
ignedness: 'int' and 'long unsigned int' [-Wsign-compare]
   35 |     for (i = 0; i < sizeof(hashed_160bits); i++)
      |                   ^
proxy_cache.c: In function 'main':
proxy_cache.c:112:33: warning: '%s' directive writing up to 3 bytes into a reg
ion of size between 0 and 511 [-Wformat-overflow=]
  112 |         sprintf(cache_path, "%s/%s", full_path, dir_name);
      |                                 ^~              ~~~~~~~~
proxy_cache.c:112:9: note: 'sprintf' output between 2 and 516 bytes into a des
tination of size 512
  112 |         sprintf(cache_path, "%s/%s", full_path, dir_name);
      |         ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
proxy_cache.c:116:34: warning: 'sprintf' may write a terminating nul past the
end of the destination [-Wformat-overflow=]
  116 |         sprintf(file_path, "%s/%s", cache_path, hashed_url + 3);
      |                                  ^
proxy_cache.c:116:9: note: 'sprintf' output 2 or more bytes (assuming 513) int
o a destination of size 512
  116 |         sprintf(file_path, "%s/%s", cache_path, hashed_url + 3);
      |         ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
kw2023202043@ubuntu:~/Proxy1-2$ ls
Makefile  proxy_cache  proxy_cache.c
```

코드 작성 후 make 한 뒤 proxy_cache가 생성된 것을 확인할 수 있다.

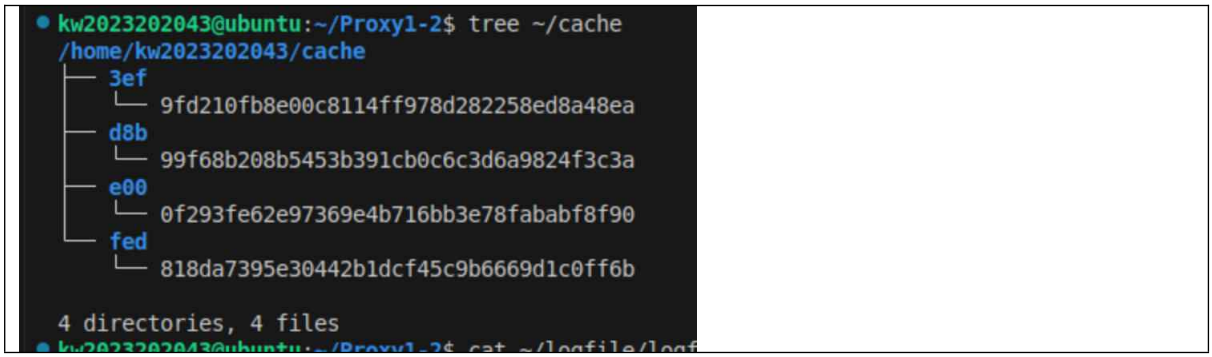
```
kw2023202043@ubuntu:~/Proxy1-2$ ./proxy_cache
input url> www.kw.ac.kr
input url> www.naver.com
input url> www.google.com
input url> www.kw.ac.kr
input url> www.naver.com
input url> klas.kw.ac.kr
input url> bye
```

./proxy_cache는 작성한 캐시 서버 프로그램을 실행하는 명령이다.

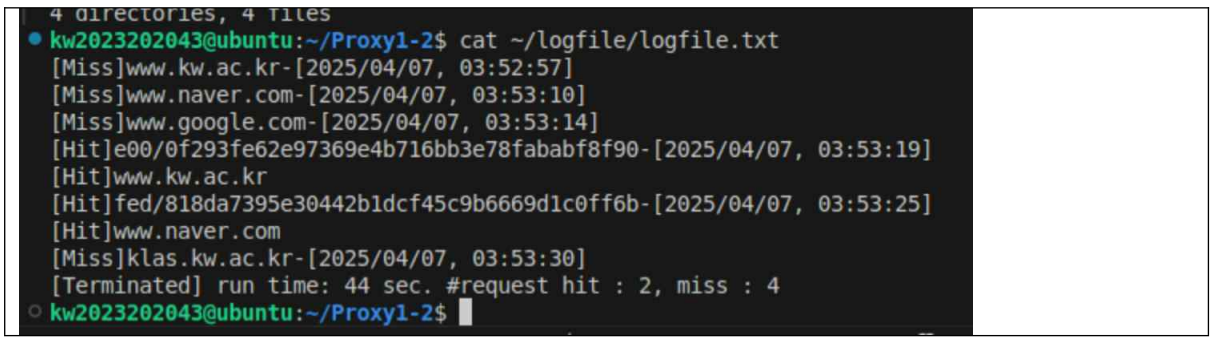
사용자로부터 URL을 입력받아 해당 URL의 SHA-1 해시 값을 생성한다. 이후 캐시 파일을 디렉토리에 저장하거나 조회하며 입력값이 bye라면 종료한다. 종료 시점까지의 실행 시간, 요청 횟수, HIT/MISS 통계를 logfile.txt에 기록한다.



ls -R ~/cache는 ~/cache디렉토리와 그 하위 폴더 및 파일을 재귀적으로 나열하는 명령어다. 생성된 캐시 파일이 지정된 디렉토리 구조에 따라 잘 저장되었는지를 확인할 수 있다.

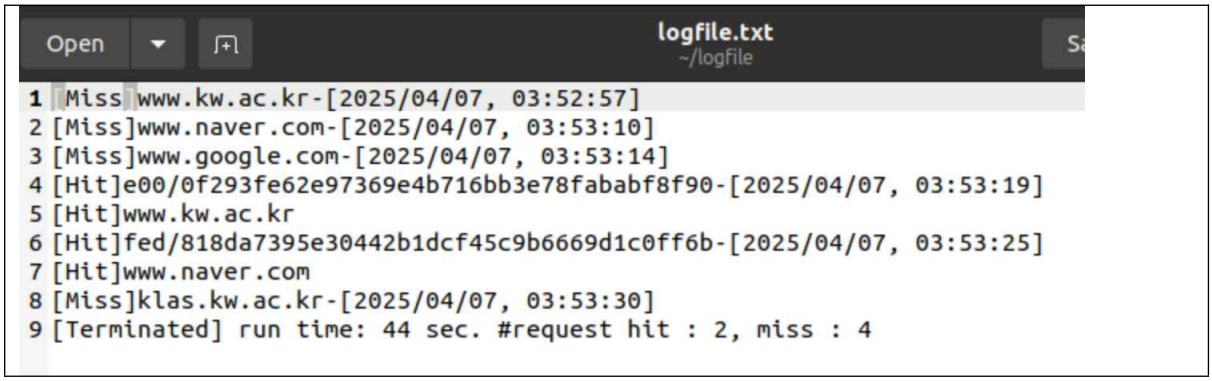


tree ~/cache는 캐시 디렉토리의 전체 구조를 트리 형태로 시각화하여 보여주는 명령어다. URL의 해시 앞 3자리를 기준으로 생성된 하위 디렉토리와 그 안의 파일들을 계층적으로 확인할 수 있다.



cat ~/logfile/logfile.txt는 로그 파일의 내용을 출력하는 명령어다. 발생한 캐시 HIT, MISS, 프로그

램 종료 등이 시간 정보와 함께 기록된 로그를 확인할 수 있다.



위와 같이 logfile에 잘 기록된 것을 확인할 수 있다.

4. 고찰

이번 과제를 수행하면서 파일 시스템, 해시 알고리즘, 캐시 구조 설계, 로그 처리 등을 학습할 수 있었다. 특히, 입력한 URL을 기반으로 SHA-1 해시를 생성하고, 이를 통해 캐시 디렉토리 구조를 동적으로 구성하는 과정을 수행하며, 해시 기반 파일 저장 방식에 대해 알 수 있었다.

또한 시스템 콜을 활용하여 디렉토리와 파일을 생성하거나 검사하는 코드를 작성하며 파일 입출력 시스템 함수의 동작 원리를 배울 수 있었습니다. 캐시의 HIT / MISS 여부를 판단하고, 이에 따라 로그 파일에 정확한 시간 정보와 함께 기록을 남기는 기능을 구현하며 캐시 관리의 중요성을 깨우쳤다.

이렇게 이번 과제를 통해 프록시 서버라는 개념을 실습하는 경험을 얻게 되었다.