수업 명 : 시스템프로그래밍

과제 이름 : Proxy 1-3

학 과: 컴퓨터정보공학부

담당 교수님: 김태석 교수님

분 반: 월5, 수6

학 번: 2023202043

성 명: 최은준

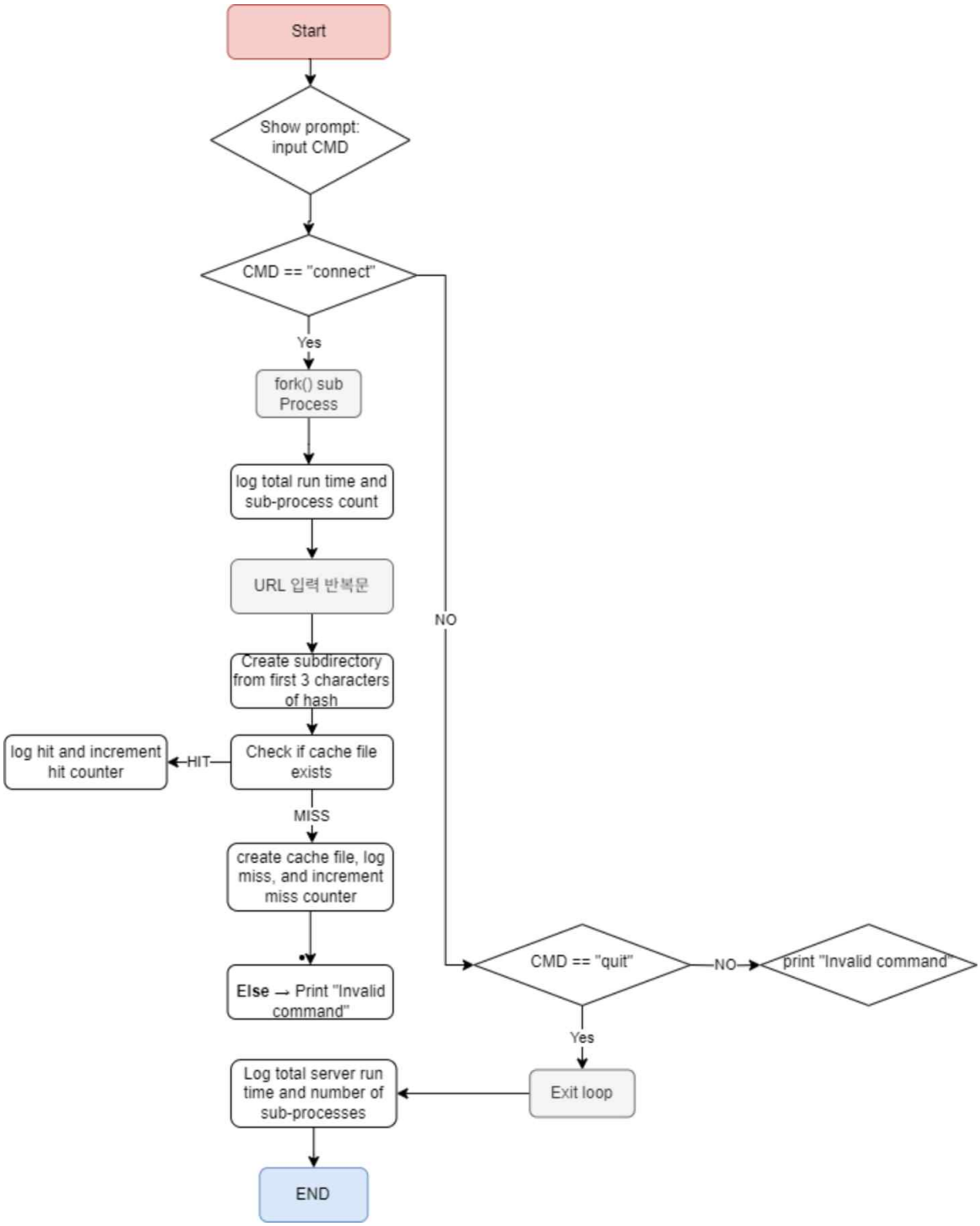
0. Introduction

이 과제에서는 Process를 활용한 Proxy cache server를 구현하는 것이다.

Main Process는 connect 명령어를 받아 Sub Process를 생성하며, Sub Process는 URL을 입력받아 Proxy 1-2에서의 연산 기능을 수행한다. quit 명령어를 받으면 input cmd프롬프트를 출력하여 입력받는 것을 중지한다. buy 명령어 입력 시 Sub Process는 실행 시간과 Hit/Miss 횟수를 log에 남기고 해당 Process를 종료한다. 각 요청 결과는 log.txt 파일에 기록되며, 서버 종료 시 전체 실행 시간 및 Sub Process 수가 출력된다.

1. Flow Chart

1-1. proxy server #1-3 코드 작성 순서도



위 코드 순서도의 동작 방식은 다음과 같다.

1. 명령어 입력 대기

메인 프로세스는 input CMD>프롬프트를 출력하며 사용자 입력을 기다린다.

connect명령어가 입력되면 sub 프로세스를 생성하고, quit입력 시 프로그램을 종료한다.

2. sub 프로세스 동작

sub 프로세스는 input URL>프롬프트를 출력하며 URL 입력을 기다린다.

입력된 URL은 SHA1 해시 함수를 통해 고유한 해시값으로 변환된다.

해당 해시값을 기반으로 캐시 디렉터리 및 파일 경로를 생성한다.

동일한 캐시 파일이 존재하면 Hit, 존재하지 않으면 캐시 파일을 생성하고 Miss로 처리한다.

처리 결과는 모두 log.txt 파일에 기록된다.

사용자가 bye를 입력하면 자식 프로세스는 실행 시간 및 Hit/Miss 횟수를 log에 남기고 종료된다.

3. 서버 종료 처리

사용자가 quit명령어를 입력하면 메인 프로세스는 서버 전체 실행 시간과 생성된 자식 프로세스 수를 로그 파일에 기록한 후 종료된다.

2. Pseudo code

2-1. proxy server #1-3 알고리즘

```
Function: run_subprocess

Function run_subprocess()
    Initialize hit and miss counters
    Get home directory and set up cache/log directories

    Loop
        Prompt user for input_url
        If input_url is "bye", break loop

        Compute SHA-1 hash into hashed_url
        Extract first 3 characters into dir_name

        Build full cache path and file path
        If cache file exists:
            Increment hit
            Write 2 log entries:
                - [Hit]hashed path - timestamp
                - [Hit]original URL
        Else:
            Increment miss
            Create file with dummy content
            Write [Miss] log entry with original URL
    End Loop

    Write process termination log with hit/miss stats
    Exit
End Function
```

```
Function: main

Function main()
    Initialize subproc_count and start_time

    Loop
        Prompt user for command
```

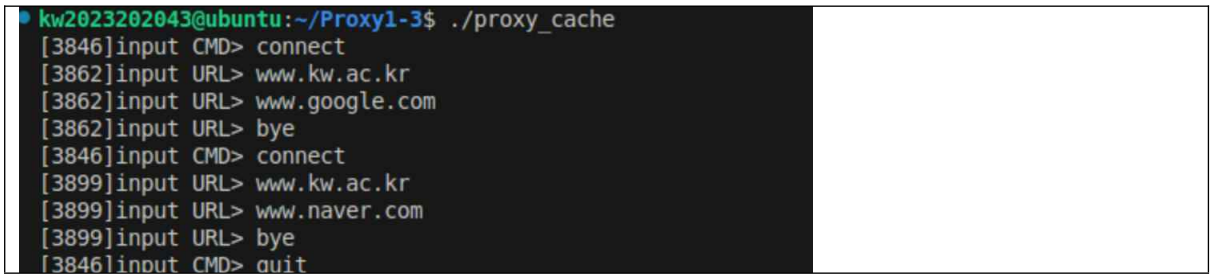
```
        If command is "connect":
            Fork new subprocess
            Child: run_subprocess
            Parent: wait, increment subproc_count
        Else if command is "quit":
            break
        Else:
            print invalid command message
    End Loop

    Write server termination log
End Function
```

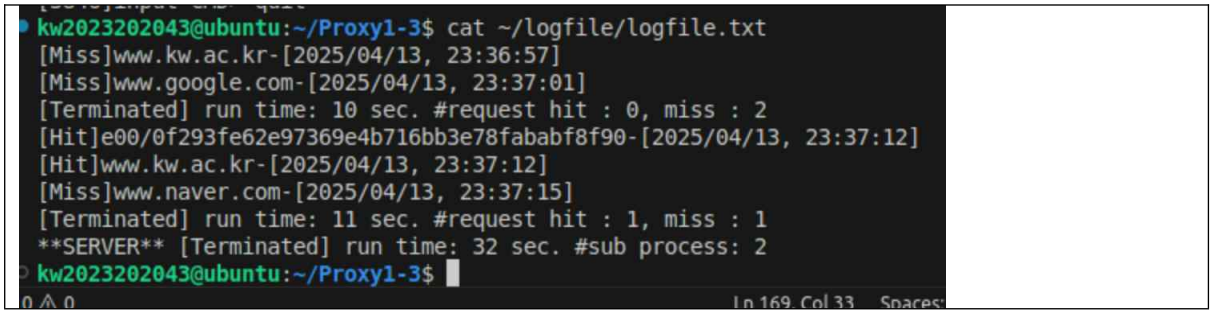
3. 결과화면

```
kw2023202043@ubuntu:~/Proxy1-3$ ls
Makefile  proxy_cache.c
kw2023202043@ubuntu:~/Proxy1-3$ make
gcc -Wall -Wextra -g -o proxy_cache proxy_cache.c -lcrypto
proxy_cache.c: In function 'sha1_hash':
proxy_cache.c:47:19: warning: comparison of integer expressions of different signedness: 'int' and
'long unsigned int' [-Wsign-compare]
   47 |     for (i = 0; i < sizeof(hashed_160bits); i++)
      |                   ^
proxy_cache.c: In function 'run_subprocess':
proxy_cache.c:163:30: warning: unused variable 'info2' [-Wunused-variable]
  163 |             char info1[512], info2[256];
      |                              ^~~~~
proxy_cache.c:159:20: warning: unused variable 't' [-Wunused-variable]
  159 |         struct tm *t = localtime(&now);
      |                    ^
proxy_cache.c:154:33: warning: '%s' directive writing up to 3 bytes into a region of size between 0
 and 511 [-Wformat-overflow=]
  154 |         sprintf(cache_path, "%s/%s", full_path, dir_name);
      |                                 ^~              ~~~~~~~~
proxy_cache.c:154:9: note: 'sprintf' output between 2 and 516 bytes into a destination of size 512
  154 |         sprintf(cache_path, "%s/%s", full_path, dir_name);
      |         ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
proxy_cache.c:156:34: warning: 'sprintf' may write a terminating nul past the end of the destinatio
n [-Wformat-overflow=]
  156 |         sprintf(file_path, "%s/%s", cache_path, hashed_url + 3);
      |                                  ^
proxy_cache.c:156:9: note: 'sprintf' output 2 or more bytes (assuming 513) into a destination of si
ze 512
  156 |         sprintf(file_path, "%s/%s", cache_path, hashed_url + 3);
      |         ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
kw2023202043@ubuntu:~/Proxy1-3$ ls
Makefile  proxy_cache  proxy_cache.c
```

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



./proxy_cache는 작성한 캐시 서버 프로그램을 실행하는 명령이다. Main Process는 connect명령을 입력받아 Sub Process생성하고, quit입력 시 종료한다. 종료 시 생성한 Sub Process 수와 총 실행 시간을 기록한다. Sub Process는 URL 캐싱 로직을 실행하여 bye입력 시 종료한다. 종료 시 HIT과 MISS 요청 수 로그를 기록한다.



cat ~/logfile/logfile.txt는 로그 파일의 내용을 출력하는 명령어다. 발생한 캐시 HIT, MISS, 프로그램 종료 등이 시간 정보와 함께 기록된 로그를 확인할 수 있다.

위와 같이 logfile에 잘 기록된 것을 확인할 수 있다.

4. 고찰

이번 과제를 통해 Proess 생성, File 입출력, Directory 관리, SHA1, 그리고 Cach System의 기본 개념을 실제 코드로 구현해볼 수 있었다. fork()함수를 활용한 sub process, 작업 종료 후 로그를 남기는 구조를 통해 프로세스 제어를 이해할 수 있었다. 파일 시스템의 구조를 기반으로 캐시 디렉터리를 자동으로 생성하고, 요청 결과에 따라 Hit과 Miss를 판단하는 로직을 구현하면서, 파일 시스템과 프로세스 등을 알 수 있었다.

마지막으로, 이번 과제를 통해 시스템 프로그래밍의 핵심 주제들을 실용적인 방식으로 익힐 수 있었으며, 차후 캐시 최적화, 네트워크 기반 서버 구현 등의 심화 학습에 좋은 기초가 될 수 있었다.