

# 시스템프로그래밍

## Proxy #1-2

담당교수 : 최상호 교수님

2021202003 강준우

## Proxy #1-2

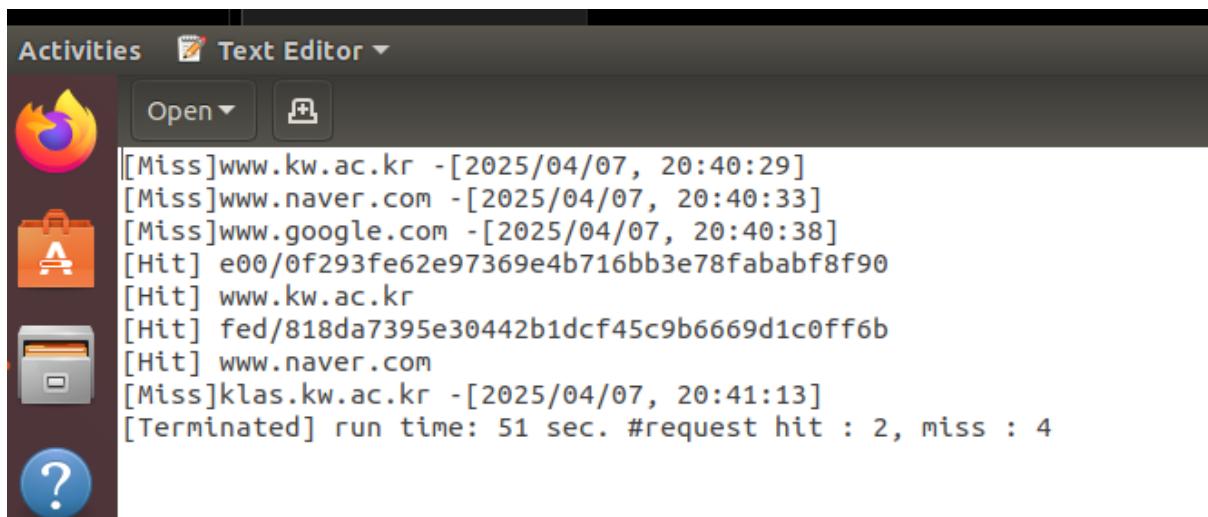
### Introduction

이번 문제는 이전의 proxy 1-1에서의 환경을 기반으로 Hit와 Miss를 확인 및 구현하는 문제입니다. Hit란 경로에 파일이 있을 경우를 의미하며, Miss는 반대로 없는 경우를 의미합니다. 각각의 경우에 현재 시간을 기록하며, 프로그램 종료시 총 hit와 miss의 수를 기록하고, 프로그램 수행시간을 기록합니다.

### 결과화면

```
File Edit View Search Terminal Help
kw2021202003@ubuntu:~$ cd work/Proxy1-2_B_2021202003_강준우/
kw2021202003@ubuntu:~/work/Proxy1-2_B_2021202003_강준우$ ./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
kw2021202003@ubuntu:~/work/Proxy1-2_B_2021202003_강준우$ █
```

프로그램 실행 하는 모습입니다. kw.ac.kr naver를 1번씩 중복해서 입력해 보았습니다.



The screenshot shows a Linux desktop environment with a terminal window and a text editor window. The terminal window displays the command-line interaction with the proxy cache program. The text editor window shows the log file with entries for each request and the final statistics.

```
[Activities] Text Editor
[Open] [New]
Mozilla Firefox
[Open] [New]
[Activities] Text Editor
[Open] [New]
[Miss]www.kw.ac.kr -[2025/04/07, 20:40:29]
[Miss]www.naver.com -[2025/04/07, 20:40:33]
[Miss]www.google.com -[2025/04/07, 20:40:38]
[Hit] e00/0f293fe62e97369e4b716bb3e78fababf8f90
[Hit] www.kw.ac.kr
[Hit] fed/818da7395e30442b1dcf45c9b6669d1c0ff6b
[Hit] www.naver.com
[Miss]klas.kw.ac.kr -[2025/04/07, 20:41:13]
[Terminated] run time: 51 sec. #request hit : 2, miss : 4
```

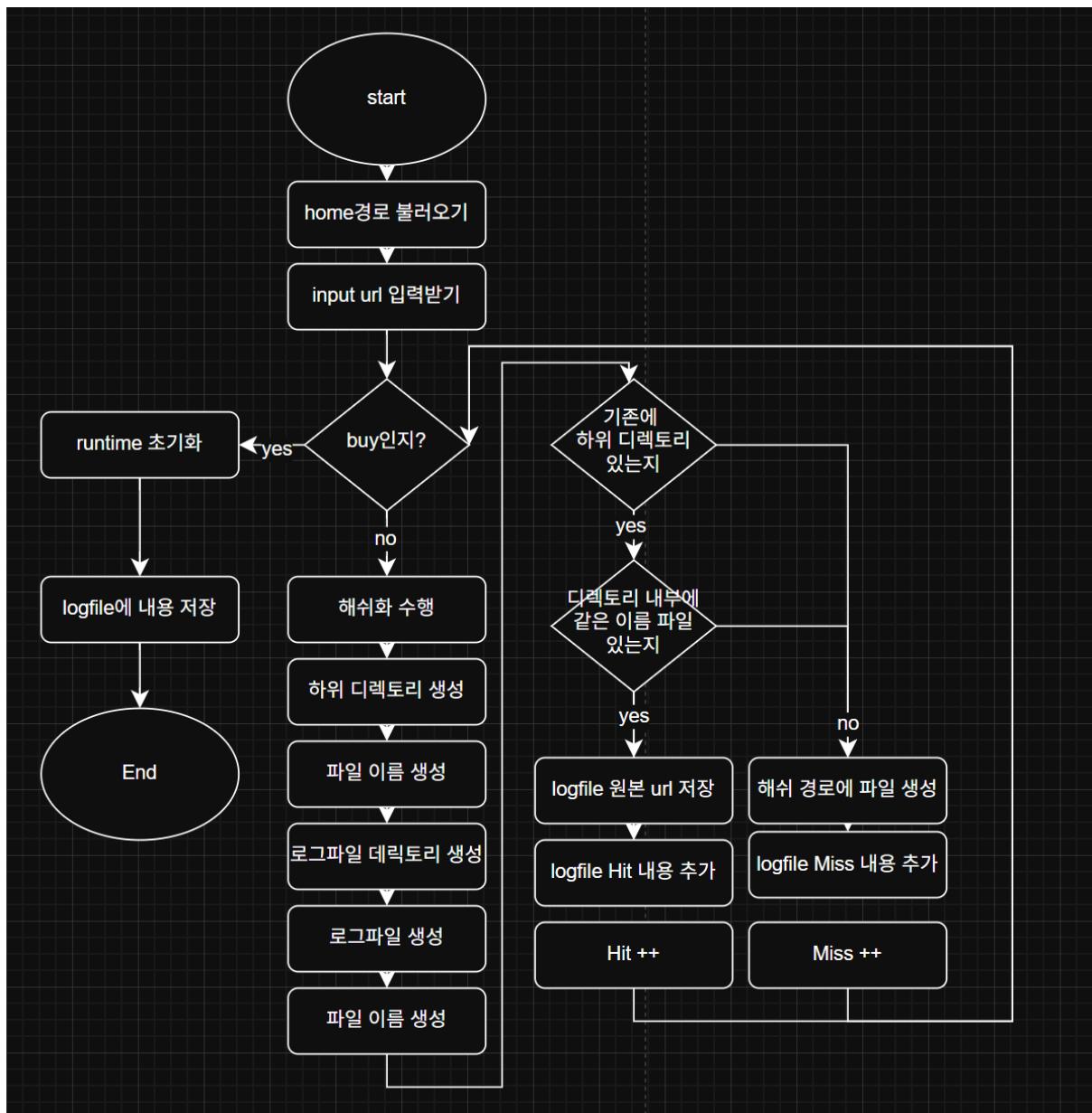
기록에서 miss와 hit가 정상적으로 기록되어 있는 모습입니다. hit의 경우 해시화 데이터와 원본 데이터를 모두 기록해줍니다. 마지막에 종료와 함께 총 실행 시간과 hit miss의 수를 기록하여 줍니다.

```
kw2021202003@ubuntu:~$ cd cache/
kw2021202003@ubuntu:~/cache$ tree
.
├── 3ef
│   └── 9fd210fb8e00c8114ff978d282258ed8a48ea
├── d8b
│   └── 99f68b208b5453b391cb0c6c3d6a9824f3c3a
├── e00
│   └── 0f293fe62e97369e4b716bb3e78fababf8f90
└── fed
    └── 818da7395e30442b1dcf45c9b6669d1c0ff6b

4 directories, 4 files
```

miss를 기반으로 디렉토리와 파일이 생성된 모습입니다.

Algorithm – Flow Chart



Pseudo Code

시작

변수 초기화:

- input\_url: 사용자로부터 입력 받을 URL
- hashed\_url: SHA1 해시 결과 저장
- sub\_dir, full\_dir, full\_file: 캐시 디렉토리 및 파일 경로
- logfile\_path: 로그파일 경로
- hit, miss: 카운터 (0으로 초기화)
- start: 프로그램 시작 시간 저장

반복문 시작 (무한 루프):

사용자에게 "input url>" 출력

사용자 입력 → input\_url에 저장

만약 **input\_url**이 "bye"라면:

    현재 시간을 **end**에 저장

    실행 시간 계산

**logfile.txt** <run time, hit , miss 로그에 기록

    루프 종료

입력된 URL을 SHA1 해시하여 **hashed\_url**에 저장

해시값의 앞 3자리 **sub\_dir**에 저장

나머지 해시값 파일 이름으로 사용

**~/cache** 및 **~/logfile** 디렉토리 생성

**found** 변수 초기화 (**false**)

**~/cache** 디렉토리 열기

디렉토리 내 항목들 반복:

    만약 항목 이름이 **sub\_dir**와 같다면:

        해당 디렉토리 (**/cache/sub\_dir**) 열기

        내부 파일들 반복:

            파일 이름이 해시 나머지값과 같다면:

**found = true**

            내부 디렉토리 닫기

**logfile.txt** 파일을 append 모드로 열기

만약 **found**가 **true**라면

    Hit로그에 기록

**input\_url** 로그에 기록

    Hit 카운터 증가

아니라면

**input\_url** 로그에 기록

    Miss 카운터 증가

    해시된 파일 생성

로그파일 닫기

프로그램 종료

## 고찰

이번 과제를 통해서 해싱화된 데이터를 매칭해보는 경험을 해보았습니다. 전래 로그인 기능을 구현할 때와 유사한 경험을 느꼈습니다. 또한 프로젝트를 구현하는 과정에서 경로와 파일 내용을 모두 검토하여 중복을 검사하는 과정에서 추가적인 중복 알고리즘을 고안할 수 있었습니다. 개인적으로 프로그램을 매번 실행할 때 경로와 로그파일을 삭제하거나

초기화해주는 과정이 번거롭다고 느꼈습니다. 제안서에 실행시 경로에 파일이 있다면 초기화 하는 기능들이 있다면 좀더 편하게 테스트 및 검증을 할 수 있을것이라고 생각합니다.

## Reference

2025-1\_SPLab\_proxy\_Assginment1-2