시스템프로그래밍 과제 보고서

Proxy 1-2

수업 명: 시스템프로그래밍 월5수6

담당 교수: 김태석

학과: 컴퓨터정보공학부

학번: 2023202070

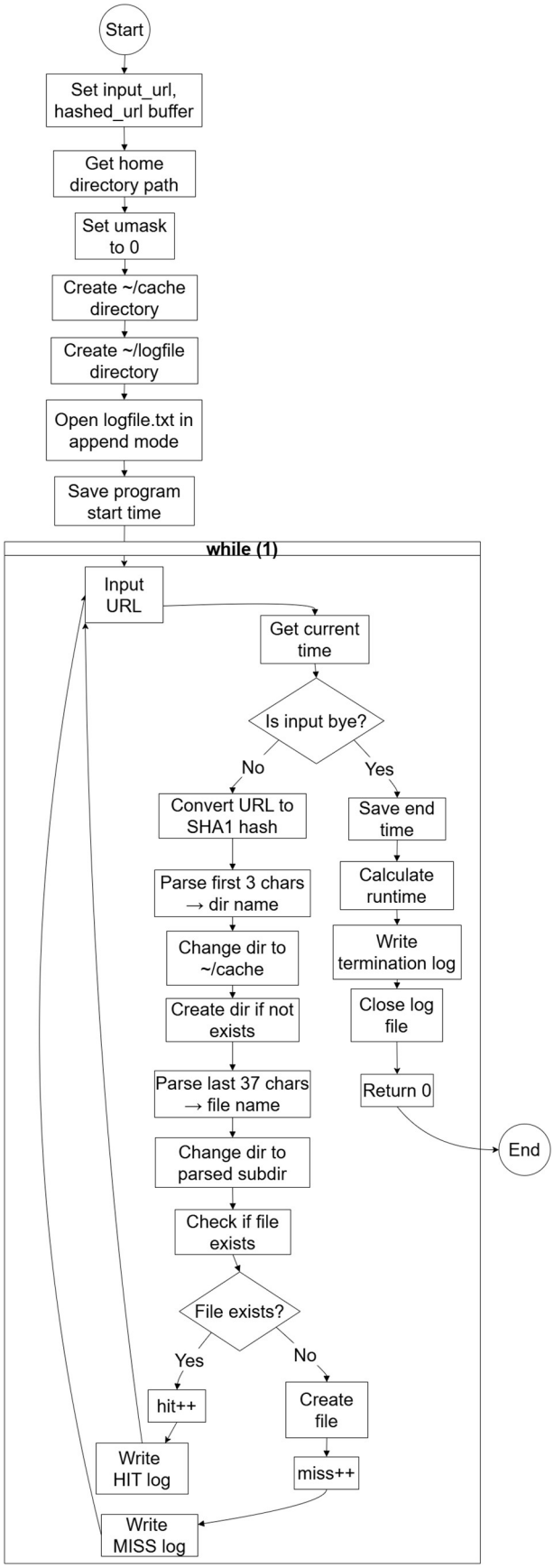
이름: 최현진

제출일: 2025.04.09

Introduction

이번 Proxy 1-2 과제는 Proxy 1-1 과제에서 구현한 기능에, HIT/MISS 판별 및 로그 기록 기능을 추가하는 시스템 프로그래밍 실습 과제이다. 과제는 입력 받은 URL을 해시한 후 해당 캐시 파일의 존재 여부를 확인하여 HIT 또는 MISS 여부를 판별하고, 그 결과를 로그 파일에 작성하는 기능 구현을 포함한다. 이를 위해 opendir()과 readdir() 함수를 활용하여 생성된 모든 캐시 디렉토리와 파일 목록을 확인하고, 해시된 URL과 일치하는 파일이 존재하는지 비교한다. 비교 결과에 따라 MISS 시에는 해당 해시된 URL을 이용해 캐시 파일을 생성하고, HIT 시에는 파일 생성 없이 로그만 기록한다. 로그는 사용자 홈 디렉토리 하위의 logfile 디렉토리에 logfile.txt 파일로 저장되며, 각 요청 입력에 대해 URL과 시간 정보와 함께 기록된다. 프로그램 종료 시에는 총 실행 시간과 HIT/MISS요청 횟수에 대한 정보를 로그에 추가한다. 이 과제를 통해 파일 시스템 함수의 활용과 함께 시간 불러오기, 문자열 처리, 파일 쓰기 등 다양한 시스템 프로그래밍 기술을 실습할 수 있다.

Flow Chart


Start
Set input_url, hashed_url buffer
Get home directory path
Set umask to 0
Create ~/cache directory
Create ~/logfile directory
Open logfile.txt in append mode
Save program start time
while (1)
Input URL
Get current time
Is input bye?
No
Yes
Convert URL to SHA1 hash
Parse first 3 chars → dir name
Change dir to ~/cache
Create dir if not exists
Parse last 37 chars → file name
Change dir to parsed subdir
Check if file exists
File exists?
Yes
No
hit++
Write HIT log
Create file
miss++
Write MISS log
Save end time
Calculate runtime
Write termination log
Close log file
Return 0
End

Pseudo code

main:

```
input_url, hashed_url 버퍼 설정
home 디렉토리 경로 가져오기
umask 0 설정
루트 캐시 디렉토리(~/cache) 생성
로그 디렉토리(~/logfile) 생성
로그 파일(logfile.txt) 열기
프로그램 시작 시간 저장

무한 반복
    URL 입력 받기
    현재 시간 불러오기

    입력이 "bye"면
        프로그램 종료 시간 저장
        실행 시간 계산
        종료 로그 작성
        로그 파일 닫기
        반복 종료

    입력된 URL을 SHA1 해싱

    해시 앞 3글자 파싱
    현재 디렉토리를 ~/cache로 이동
    앞 3글자를 이름으로 디렉토리 생성

    나머지 37글자 파싱
    현재 디렉토리를 앞 3글자로 디렉토리로 이동
    나머지 37글자를 이름으로 파일 존재 여부 확인

    파일이 없으면 (MISS)
        해당 이름으로 파일 생성
        miss++
        miss 로그 작성
    파일이 있으면
```

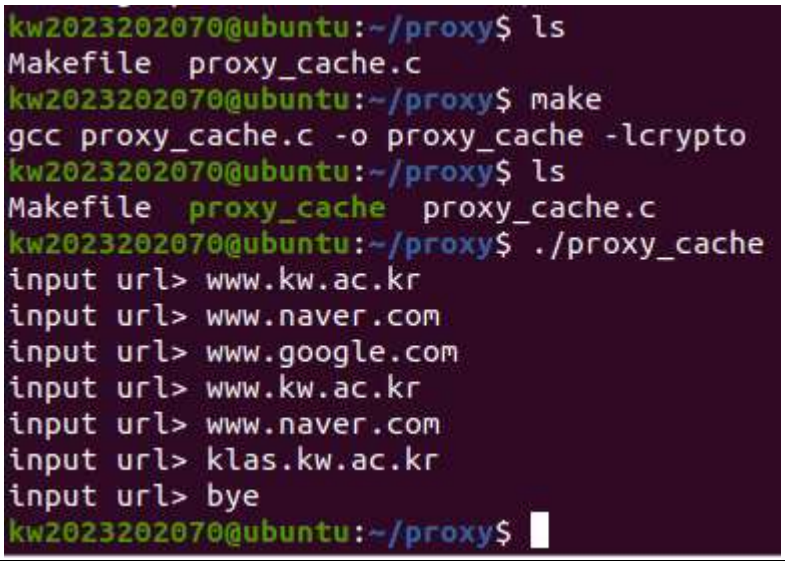
```
        hit++
        hit로그 작성
반복 종료

0 반환
```

결과 화면

1. proxy 1-2

1. Input



$ ls: Makefile과 소스 코드(proxy_cache.c)를 작성하였다.

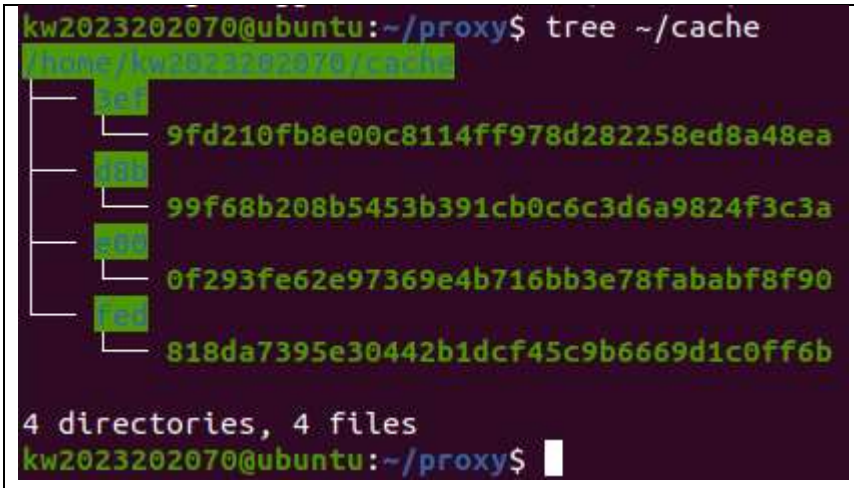
$ make: make를 실행하여 컴파일을 수행하였고, gcc 컴파일 명령어가 정상적으로 실행되었다.

$ ls: 컴파일 결과, 실행 파일 proxy_cache가 생성된 것을 확인했다.

$ ./proxy_cache: 실행 파일 실행 결과, 표준 입력으로 여러 개의 url들을 입력하였고, "bye"를 입력하여 프로그램을 종료했다.

2. Cache Directory

$ ls -R ~/cache: ls 명령어를 –R 옵션을 통해 재귀적으로 실행하여 ~/cache 디렉토리와 그 하위 디렉토리까지 출력하여 캐시 디렉토리와 파일들이 생성됨을 확인했다.



$ tree ~/cache/: ~/cache 구조 확인 결과, SHA1 해시된 url의 앞 3글자를 이름으로 하여 디렉토리가 생성되었다. 그 디렉토리의 하위에는 나머지 37글자 이름으로 파일이 생성되었다.

3. Log file

```
kw2023202070@ubuntu:~/proxy$ cat ~/logfile/logfile.txt
[Miss]www.kw.ac.kr-[2025/04/08, 07:56:25]
[Miss]www.naver.com-[2025/04/08, 07:56:28]
[Miss]www.google.com-[2025/04/08, 07:56:32]
[Hit]e00/0f293fe62e97369e4b716bb3e78fababf8f90-[2025/04/08, 07:56:35]
[Hit]www.kw.ac.kr
[Hit]fed/818da7395e30442b1dcf45c9b6669d1c0ff6b-[2025/04/08, 07:56:38]
[Hit]www.naver.com
[Miss]klas.kw.ac.kr-[2025/04/08, 07:56:42]
[Terminated] run time: 44 sec. #request hit : 2, miss : 4
kw2023202070@ubuntu:~/proxy$
```

$ cat ~/logfile/logfile.txt: logfile.txt을 출력한 결과 miss 동작 처리가 되어 url과 시간 정보가 저장되었고, hit 동작 처리가 되어 생성된 디렉토리와 파일과 시간 정보, url이 저장된 것을 확인했다. 또한 프로그램 종료 시 실행시간 및 요청 결과 (hit/miss 횟수)가 저장된 것을 확인했다.

고찰

이번 과제를 수행하며 먼저 opendir()과 readdir() 함수가 생소하게 느껴져서 자료를 찾아보며 함수의 구조와 사용 방법을 익혀야 했다. 디렉토리 내부의 파일 목록을 탐색하는 기능은 생각보다 자주 쓰이는 것 같아서 과제를 통해 직접 적용해보며 hit와 miss를 판별하고 실전에서의 사용 예를 자연스럽게 익힐 수 있었다.

또한, 로그를 파일로 작성하는 작업에서 fprintf() 함수의 사용법과 형식 지정 방법에 대해 다시 한 번 공부하게 되었다. 특히 localtime()을 통해 받아온 시간 정보를 과제가 요구하는 로그 포맷에 맞게 출력하는 과정이 처음에는 까다롭게 느껴졌고, 연도나 월을 따로 포맷팅해주는 과정이 번거로웠다. 2자리로 맞추고 남는 자리는 0으로 채우는 등 여러 번 logfile.txt 결과를 확인하며 고쳐야 했다. 최종적으로 포맷이 예시와 일치하도록 맞췄을 때 뿌듯함도 컸다.

로그에 시간을 작성하는 부분에서 다양한 시간 관련 함수를 사용해 보았다. 그러나 각각의 시간을 불러오는 순서와 시점이 프로그램 시작/종료/입력 등으로 다양하기 때문에 그 위치에 유의하여 코드를 작성하는 것에 집중했다.

이번 과제를 전반적으로 구현하며 시스템프로그래밍 이론과 실습 수업 자료를 정독하며 다시 한 번 시스템프로그램 수업에 대한 이해도를 높일 수 있었다. 앞으로의 Proxy 과제를 진행하며 더 복잡한 시스템 프로그래밍을 수행할 기대가 된다.

Reference

시스템프로그래밍 이론 및 실습 자료 참고하였습니다.