System programming

2차 과제 - System Programming Assignment #1-1 (proxy server)

학 과 : 컴퓨터 공학과

담당교수 : 황호영

분 반 : 목34

학 번 : 2016722092

성 명 : 정동호

**목차**

1 Introduction 3

2 Flowchart 3

3 Pseudo code 5

4 결과화면 9

5 결론 및 고찰 11

6 참고 레퍼런스 11

# Introduction

본 과제에서 다룰 프로그램은 캐싱 기능을 수행하는 프록시 서버이다. 이번 1번째 구현 단계에선,

* 표준입력으로부터 URL 입력 받기
* SHA1 알고리즘으로 text URL을 hashed URL로 변환하기
* hashed URL을 이용하여 directory와 file 생성하기

이 3개를 구현한다.

# Flowchart

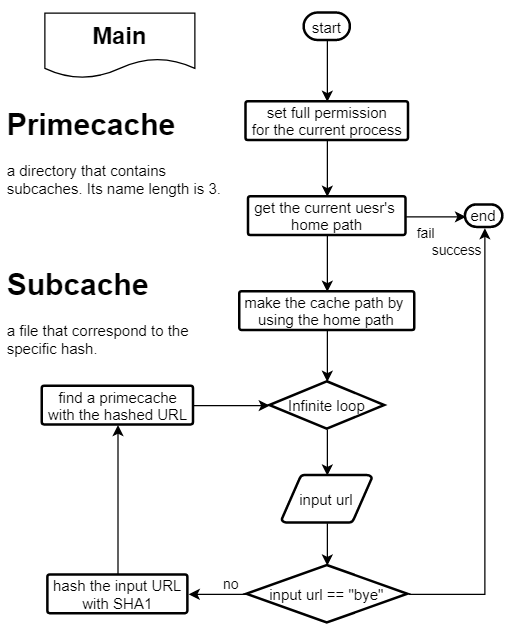
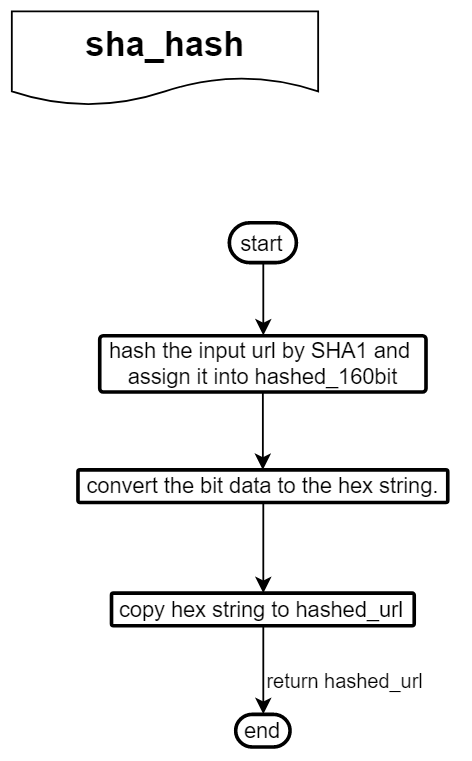
본 과제에서 사용된 주요 함수 4개의 순서도는 다음과 같다:

Figure 2.1 main 함수 순서도

Figure 2.2 sha\_hash 함수 순서도

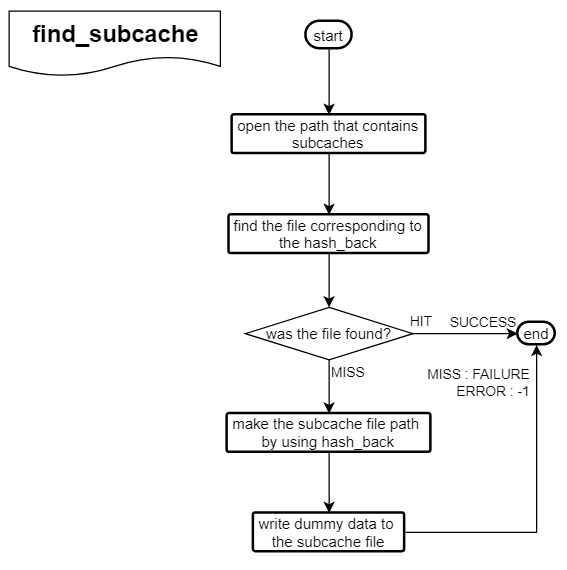
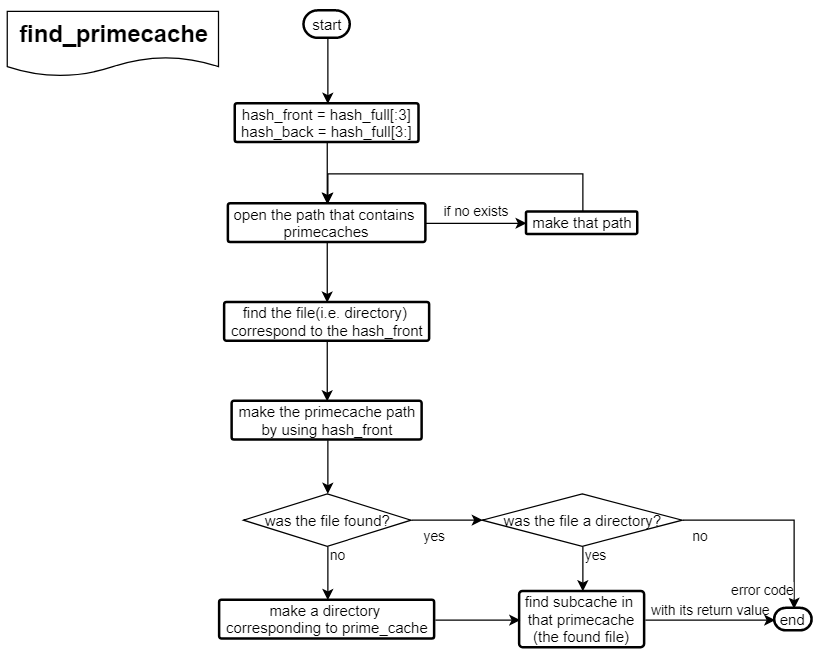


Figure 2.3 find\_primecache 함수 순서도

그림 2.1 find\_subcache 순서도

Figure 2.4 find\_subcache 함수 순서도

2.2 find\_primecache 함수 순서도

Figure 2.5

먼저 main 함수 순서도에서 Primecache와 Subcache에 대해 정의했다. 실제로 있는 말은 아니지만 본 과제에서 SHA1 해시값이 두 부분으로 나뉘어 편의상 이렇게 구분하기로 하였다. 또 분기문이 아닌 블록에서 가끔 화살표가 두개로 나뉘는 데 이 부분은 예외 처리를 나타낸것이다. 따라서 예외가 없으면 일반 화살표 방향대로 가고 예외가 생길 경우에만 해당 블럭으로 넘어간다. 그리고 굳이 Infinite loop와 prime\_cache의 반복을 구분한 이유는 실제 코드 상에서 유의미한 차이가 있기 때문이었다.

자세한 설명은 다음 절의 Pseudo code 부분과 상당 부분 겹치므로 일단 생략한다.

# Pseudo code

본 report에서 사용된 Psuedo code style은 다음과 같다.

1. 코드를 그대로 갖고온다.
2. 문장으로 치환할 수 있는 부분은 치환하고 굵은 노란색으로 표시한다.
3. 2번을 거친 뒤 나머지 1번에서 가져온 부분 중 언급되지 않은 것들은 제거한다.
4. 코드 블럭 밑에서 덧붙일 부가 설명과 중복된 부분도 제거한다.

그리고 2.의 경우 슬라이싱이나 집합등 기타 부분에서 python style을 사용할 것이다.

Figure 3.1 상수값들



Figure 3.2 sha1\_hash pseudo code



Figure 3.3 main pseudo code



Figure 3.4 find\_primecache pseudo code

|  |
| --- |
| \* **@param** path\_primecache A const char pointer to the path containing primecaches.  \* **@param** hash\_full A const char pointer to be used as a part of primecache name.  \* **@return** [int] HIT:0, MISS:1, FAIL:-1  int find\_primecache(const char \*path\_primecache, const char \*hash\_full){  char hash\_front[PROXY\_LEN\_PREFIX + 1] = {0};  char hash\_back[PROXY\_LEN\_HASH - PROXY\_LEN\_PREFIX + 1] = {0};  int result = 0;  **hash\_front = hash\_full[PROXY\_LEN\_PREFIX:]**  **hash\_back = hash\_full[:PROXY\_LEN\_PREFIX]**  **If the path of primecache dosen't exist, create that path**    **Find the primecache while traversing the path {**  **If the primecache was found {**  **If it is a directory {**  **the primecache was found**  **} else {**  **exception handling**  **}**  **}**  **}**    **If there isn't the path of subcache, then create that path**    **Find the subcache in the path of the current primecache,**  **and assign its return value into result**  **return** result;  }  find\_primecache는 인자로 주어진 path\_primecache에서 hash\_full의 앞 세글자 짜리 폴더(primecache)를 찾는 역할을 수행한다. 따라서 hash\_front는 primecache의 이름으로 쓰이고 hash\_back은 primecache아래 subcache의 이름으로 쓰이게 된다.  만약 primecache가 존재할 경로가 없다면(여기선 home path 밑의 cache directory) full permission으로 해당 경로를 생성한다.  그리고 그 경로를 순회하며 hash\_front와 일치하는 primecache를 탐색한다. 일치하는 primecache를 발견했는데 directory가 아니라면 cache 폴더 구조에 문제가 생긴 것이므로 유저에게 이를 알려주고 예외 처리를 수행한다.  만약 primecache(the path of subcache)가 없다면 MISS인 경우로 이때는 해당 directory를 생성한다.  마지막으로 해당 primecache를 경로로하여 subcache를 탐색하는 함수를 호출하고 그 반환값을 다시 결과로써 반환한다. |

Figure 3.5 find\_subcache pseudo code



# 결과화면

Figure 4.1 초기 상태

맨 처음 초기 상태는 위와 같다.

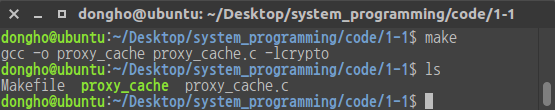


Figure 4.2 make

make를 해주어 실행파일을 생성한다.

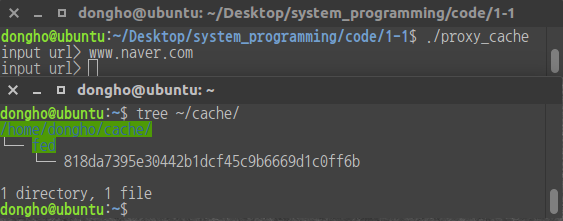


Figure 4.3 1번째 URL 입력

www.naver.com을 입력하자 cache directory가 생성되고 그 아래에 캐시가 생성된것을 확인하였다.

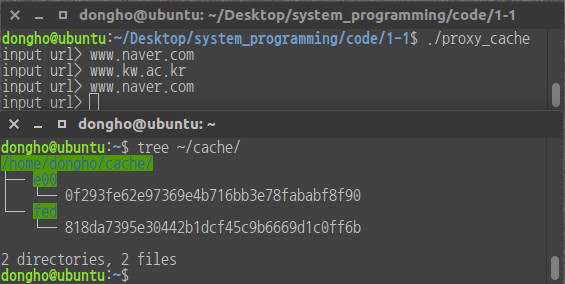
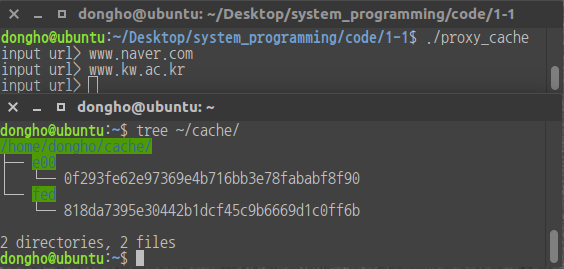
이번엔 www.kw.ac.kr을 입력하였다. 새로운 cache가 생성된것을 확인하였다.

Figure 4.4 중복된 URL 입력

Figure 4.5 2번째 URL 입력

중복된 URL을 입력하여 HIT인 경우를 만들어보았다. 아직 HIT인 경우는 TODO 상태므로 아무일도 일어나지 않았다.

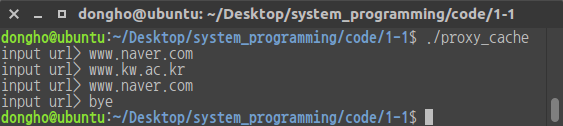


Figure 4.6 bye 입력

bye가 입력되면 종료되는 것 까지 확인하였다.

# 결론 및 고찰

unix에서 C로 무언가를 코딩해보는건 처음이라 왜 안되지(VLA 초기화, 네임스페이스 설정 등등)와 무엇이 더 나은 방법일까(static const vs define vs enum, 함수 관계 등등)를 많이 고민했었다. 확실히 너무 불편하고 이해안되고 힘든 점도 있었지만 이러한 제약덕분에 컴파일 시간과 런타임 시간에서 우위에 있을 수 있던 것 같다.

위와 별개로 프록시니 캐시니 하는 것도 모두 처음이라 흥미롭다. 아직 전체 3단계에서 1단계를 구현하였고 남은 2단계도 기대된다.

사실 더 쓰고 싶은 말이 좀 있긴한데 과제와 별 관련이 없을것 같아 다음으로 참고문헌들을 소개하며 마무리 하겠다.

# 참고 레퍼런스

<http://yhcting.tistory.com/entry/Order-of-file-list-of-the-directory-in-ext4> readdir가 파일 탐색 결과를 반환하는 순서에 대해 상세한 설명이 나와있다.  
Stackoverflow에서도 제대로 된 답변을 못찾았는데 티스토리에서 찾을 수 있었다.

<https://stackoverflow.com/questions/417142/what-is-the-maximum-length-of-a-url-in-different-browsers> 최대 url 버퍼 길이값 선정에 대한 근거

<https://en.wikipedia.org/wiki/Comparison_of_file_systems>, <https://askubuntu.com/a/859953> UNIX에서 PATH의 최대 경로 버퍼 길이값 선정에 대한 근거 (근거들이라고 했지만 공식적인 것들은 아니다.)

<http://pubs.opengroup.org/onlinepubs/7908799/xsh/pwd.h.html> 헤더 요약 참고