

수업 명 : 시스템프로그래밍

과제 이름 : Proxy 1-1

학 과: 컴퓨터정보공학부

담당 교수님: 김태석 교수님

분 반: 월5, 수6

학 번: 2023202043

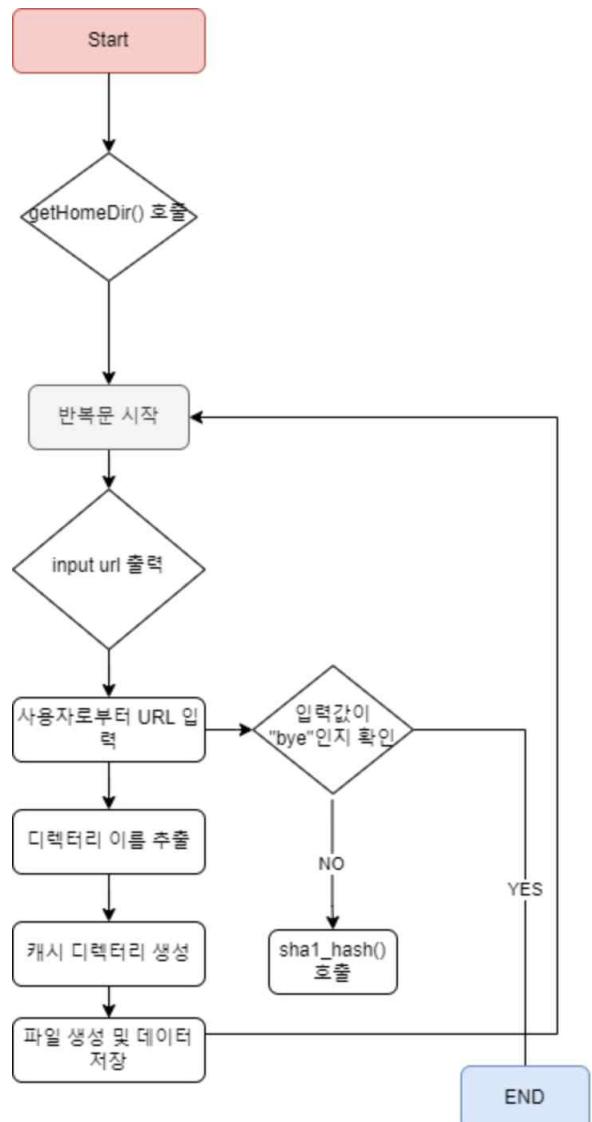
성 명: 최은준

0. Introduction

이 과제에서는 SHA-1 해싱을 활용한 URL 캐시 관리 프로그램을 구현하는 것이다. 입력한 URL을 SHA-1 알고리즘을 사용하여 해시 함수로 변환하여 해시 값을 생성하고, 이를 기반으로 파일 시스템에 캐시를 생성한다. 프로그램은 홈 디렉터리에 cache폴더를 생성하고, 해시된 값의 일부를 디렉터리 구조로 활용하여 캐시를 저장하는 방식으로 동작한다. 사용자가 URL을 입력하면 해당 URL의 캐시 파일(캐시된 데이터를 저장하고 관리하는 파일)을 생성하고 bye를 입력하면 프로그램을 종료한다.

1. Flow Chart

1-1. proxy server #1-1 코드 작성 순서도



위 코드 순서도의 동작 방식은 다음과 같다.

1. 사용자 입력

사용자는 프로그램 실행 후 URL을 입력한다.

"bye"를 입력하면 프로그램이 종료된다.

2. SHA-1 해싱

입력된 URL을 SHA-1 해시 함수로 변환하여 40자리 해시 값을 생성한다.

3. 캐시 디렉터리 구조

사용자의 홈 디렉터리 내 cache폴더를 생성한다.

해시 값의 첫 3자리를 디렉터리 이름으로 사용하여 서브 폴더를 만든다.

해당 서브 폴더에 나머지 해시 값을 이름으로 하는 파일을 생성한다.

4. 캐시 파일 저장

생성된 파일에 해당 URL의 캐시 데이터를 저장한다.

파일 생성 여부를 출력하여 사용자에게 알린다.

2. Pseudo code

2-1. proxy server #1-1 알고리즘

```
sha1_hash Function
Function sha1_hash(input_url, hashed_url)
    Declare hashed_160bits as an array of 20 unsigned characters
    Declare hashed_hex as an array of 41 characters
    Perform SHA-1 hashing on input_url and store the result in hashed_160bits

    For each byte in hashed_160bits
        Convert byte to a 2-character hexadecimal string and append to hashed_hex
    End For

    Copy hashed_hex to hashed_url
    Return hashed_url
```

End Function

getHomeDir Function

Function getHomeDir(home)

 Get current user information

 Copy user home directory path to home

 Return home

End Function

main Function

Function main()

 Declare input_url as a string of 256 characters

 Declare hashed_url as a string of 41 characters

 Declare home_dir as a string of 256 characters

 Declare full_path as a string of 512 characters

 Declare dir_name as a string of 4 characters

 Get home directory and store in home_dir

 Construct full_path as home_dir + "/cache"

 Create directory at full_path

Loop Forever

 Print "input url> "

 Read input_url from user

 If input_url is "bye" Then

 Exit Loop

 End If

 Call sha1_hash(input_url, hashed_url)

 Copy first 3 characters of hashed_url to dir_name

 Append null terminator to dir_name

 Construct cache_path as full_path + "/" + dir_name

 Create directory at cache_path

 Construct file_path as cache_path + "/" + hashed_url[3:]

 Open file at file_path for writing

```

If file opened successfully Then
    Write "Cached data for input_url" to file
    Close file
    Print "Cache created: file_path"
Else
    Print "File creation failed"
End If
End Loop

Return 0
End Function

```

3. 결과화면

3-1. mkdir()

```

● kw2023202043@ubuntu:~/HW_1$ make
gcc -Wall -g -o hw1 hw1.c
hw1.c:4:6: warning: return type of 'main' is not 'int' [-Wmain]
  4 | void main(int argc, char *argv[])
     | ^~~~
● kw2023202043@ubuntu:~/HW_1$ ls
hw1  hw1.c  hw1.o  Makefile
● kw2023202043@ubuntu:~/HW_1$ ls -ld hw1
-rwxrwxr-x 1 kw2023202043 kw2023202043 19288 Apr  1 05:32 hw1
● kw2023202043@ubuntu:~/HW_1$ cd ..

```

이 코드에서 directory의 권한이 mkdir의 인자로 준 것과 일치하지 않다는 문제가 발생한다. 리

눅스의 기본 umask 값이 002 또는 022이기 때문에 umask 설정값에 따라 directory의 권한이 자동으로 조정된 것이 이유이다. 여기서 umask란 file 및 directory를 생성할 때 적용되는 기본 제한 값으로 umask 값이 022라면 directory는 (777-022)755로 생성된다. 따라서 umask(0)으로 설정하여 의도대로 권한을 적용해야 한다. 이를 해결하기 위한 소스코드는 아래와 같다.

1. mode_t old_mask = umask(0) -> 기존 umask 값을 저장한다.
2. mkdir("cache", 0777) -> mkdir()를 실행한다.
3. umask(old_mask) -> umask를 원래대로 복구한다.
4. 0777 권한을 가진 directory가 생성된다.

```
hw1.c: In function 'main':  
11     mode_t old_mask = umask(0);  
12     if (mkdir("cache", 0777) == -1) {  
13         perror("mkdir error");  
14         return 1;  
15     }  
16     umask(old_mask);  
  
make: Nothing to be done for 'all'.  
ls  
cache  hw1  hw1.c  hw1.o  Makefile  
ls -ld cache  
drwxrwxrwx 2 kw2023202043 kw2023202043 4096 Apr  1 05:42 cache
```

5. 위의 사진처럼 777 권한을 가진 directory cache가 생성이 된 모습이다.

3-2. proxy server #1-1

3-2-1) Download package

```
kw2023202043@ubuntu:~$ sudo apt-get install libssl-dev  
[sudo] password for kw2023202043:
```

1. SHA-1 library

-> SHA-1를 이용한 hashing 함수를 사용할 것이기 때문에 SHA-1 라이브러리를 설치한다.

```
● kw2023202043@ubuntu:~/HW_1_2$ gcc proxy_cache.c -lcrypto
proxy_cache.c: In function 'main':
proxy_cache.c:57:33: warning: '%s' directive writing up to 3 bytes into a r
egion of size between 0 and 511 [-Wformat-overflow=]
  57 |         sprintf(cache_path, "%s/%s", full_path, dir_name);
                  ^
proxy_cache.c:57:9: note: 'sprintf' output between 2 and 516 bytes into a d
estination of size 512
  57 |         sprintf(cache_path, "%s/%s", full_path, dir_name);
                  ^
proxy_cache.c:61:34: warning: 'sprintf' may write a terminating nul past th
e end of the destination [-Wformat-overflow=]
  61 |         sprintf(file_path, "%s/%s", cache_path, hashed_url + 3);
                  ^
proxy_cache.c:61:9: note: 'sprintf' output 2 or more bytes (assuming 513) i
nto a destination of size 512
  61 |         sprintf(file_path, "%s/%s", cache_path, hashed_url + 3);
                  ^
○ kw2023202043@ubuntu:~/HW_1_2$
```

2. -lcrypto

-> SHA-1 library를 사용한 코드 컴파일 시 옵션이 필요하므로 -lcrypto를 추가로 설치한다.

```
Input url> bye
kw2023202043@ubuntu:~/HW_1_2$ sudo apt-get install tree
[sudo] password for kw2023202043:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  tree
0 upgraded, 1 newly installed, 0 to remove and 310 not upgraded.
Need to get 43.0 kB of archives.
After this operation, 115 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 tree amd64 1.8.0-1 [43.0 kB]
Fetched 43.0 kB in 1s (42.2 kB/s)
Selecting previously unselected package tree.
(Reading database ... 167432 files and directories currently installed.)
Preparing to unpack .../tree_1.8.0-1_amd64.deb ...
Unpacking tree (1.8.0-1) ...
Setting up tree (1.8.0-1) ...
Processing triggers for man-db (2.9.1-1) ...
kw2023202043@ubuntu:~/HW_1_2$
```

3. tree

-> tree 명령어 사용을 위해 추가로 package를 설치한다.

3-2-1) Create Cache Directory and File

```
kw2023202043@ubuntu:~/HW_1_2$ make
gcc -Wall -Wextra -g -o proxy_cache proxy_cache.c -lcrypto
kw2023202043@ubuntu:~/HW_1_2$ ls
a.out  Makefile  proxy_cache  proxy_cache.c
kw2023202043@ubuntu:~/HW_1_2$ ./proxy_cache
input url> www.kw.ac.kr
Cache created: /home/kw2023202043/cache/e00/0f293fe62e97369e4b716bb3e78fa
b6f0aa
Studio Code | www.google.com
Cache created: /home/kw2023202043/cache/d8b/99f68b208b5453b391cb0c6c3d6a9
4f3c3a
input url> bye
kw2023202043@ubuntu:~/HW_1_2$ sudo apt-get install tree
```

www.kw.ac.kr, www.google.com 두 개의 url을 생성한다.

```
● kw2023202043@ubuntu:~/HW_1_2$ tree ~/cache/
/home/kw2023202043/cache/
└── d8b
    └── e00
        └── 0f293fe62e97369e4b716bb3e78fababf8f90
    2 directories, 2 files
● kw2023202043@ubuntu:~/HW_1_2$
```

Hashed된 URL에 해당하는 directory와 file을 생성한다.

디렉토리 이름이 hashed_ur의 앞 세 글자로 생성된 것을 확인할 수 있다.

4. 고찰

이번 과제를 수행하면서 SHA-1 해싱, 파일 시스템 관리, 예외 처리등의 개념을 이해할 수 있었다. SHA-1 해싱을 활용하여 URL을 변환하고 디렉터리 및 파일을 생성하는 과정에서 해싱의 활용성을 알 수 있었다.

또한, 사용자의 홈 디렉터리를 자동으로 탐색하여 cache폴더를 생성하는 과정에서 사용자 환경을 고려한 프로그래밍의 중요성을 배웠다. getpwuid(getuid())함수를 활용하여 홈 디렉터리를 찾을 수 있으며, 이를 바탕으로 프로그램이 사용자마다 개별적인 캐시 저장 공간을 유지할 수 있다는 것을 알게 됐다. SHA-1 해싱을 활용하여 데이터를 고유한 식별자로 변환하는 방법과 사용자의 홈 디렉터리 및 파일 권한 설정 등 운영체제 환경을 고려한 프로그래밍의 중요성을 깨달을 수 있었다.

이번 과제는 파일 시스템과 해싱을 활용한 캐시 저장이라는 개념을 실습하는 좋은 경험이 되었으며, 이를 통해 데이터 처리 및 성능 최적화에 대한 고민을 해볼 수 있는 계기가 되었다.