

# 시스템 프로그래밍 실습 1-1 과제

이름 : 이준휘

학번 : 2018202046

교수 : 최상호 교수님

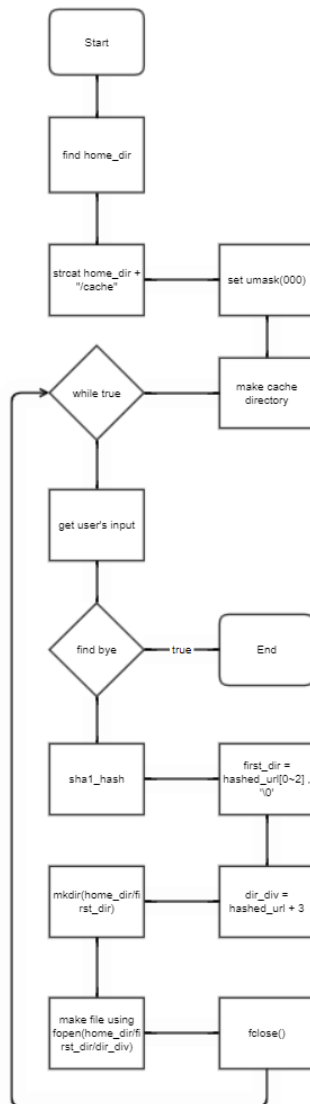
강의 시간 : 화

실습 분반 : 목 7, 8

## 1. Introduction

첫 번째 과제에서 mkdir()을 사용하였을 때 기본 설정된 권한으로 인해 rwxrwxrwx 권한이 부여되지 문제가 발생한다. 이를 umask()를 통해 기본 설정된 권한을 변경하는 문제가 주어져있다. 두 번째 과제로는 웹사이트를 입력 받아 이를 주어진 hash 변환 함수를 이용하여 변환한다. 저장되는 주소는 기본적으로 현재 사용중인 user의 home directory에 cache directory에 저장된다. 만약 cache directory가 없을 경우 새로 생성한다. 변환된 데이터에서 처음 일부를 directory 명으로 사용하고, 일부는 해당 directory 안에 파일로 저장된다.

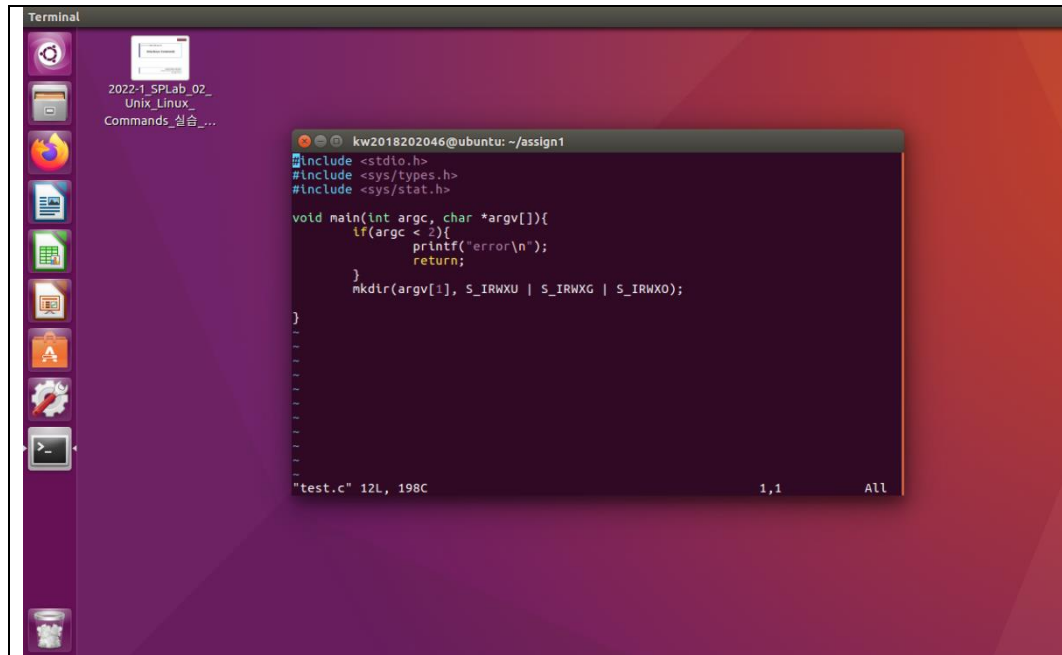
## 2. Flow Chart



### 3. Pseudo Code

```
main(void){  
  
    char[100] input, home_dir, temp_dir;  
  
    char[60] hashed_url;  
  
    char[4] first_dir;  
  
    char *dir_div;  
  
    FILE *temp_file  
  
  
    home_dir = ~/cache;  
  
    set umask 000;  
  
    make home_dir directory(permission = drwxrwxrwx);  
  
    while(true){  
  
        input = User's input;  
  
        if(input is 'bye')        end_program;  
  
        hashed_url = hashed input using sha1;  
  
        first_dir = { hashed_url[0~3], '\0' };  
  
        dir_div = hashed_url + 3 address  
  
        temp_dir = ~/cache/first_dir;  
  
        make temp_dir directory(permission = drwxrwxrwx);  
  
        temp_dir = ~/cache/first_dir/dir_div  
  
        temp_file = make temp_dir file and open file;  
  
        temp_file close;  
  
    }  
  
}
```

### 4. 결과 화면



```

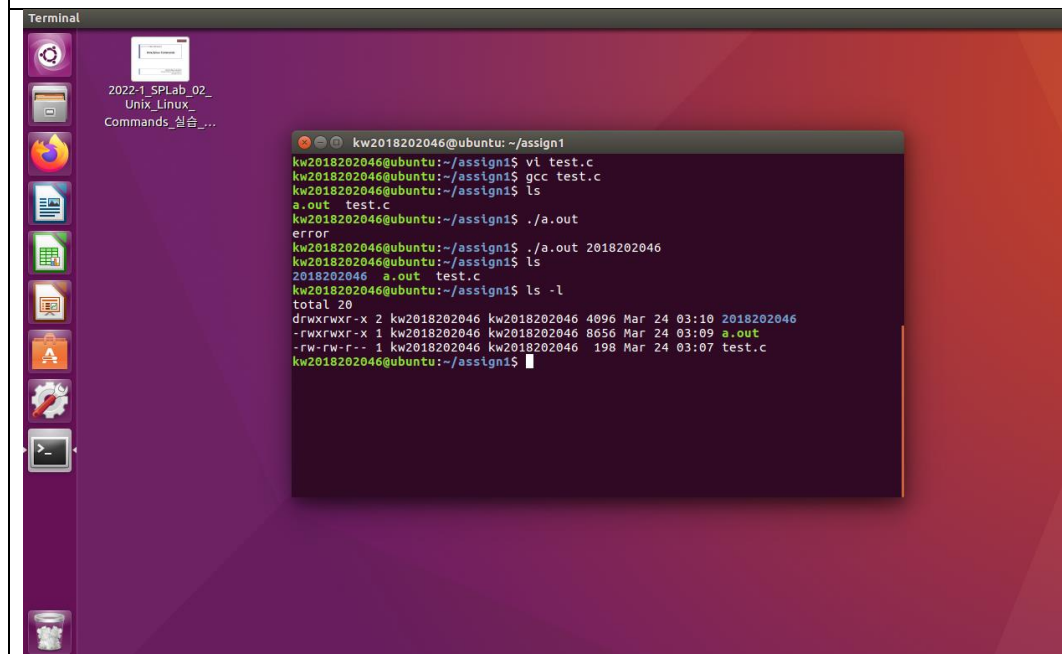
kw2018202046@ubuntu: ~/assign1
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>

void main(int argc, char *argv[]){
    if(argc < 2){
        printf("error\n");
        return;
    }
    mkdir(argv[1], S_IRWXU | S_IRWXG | S_IRWXO);
}

"test.c" 12L, 198C                                1,1      All

```

해당 코드는 기본적으로 주어진 코드다. 해당 코드를 실행하였을 때의 결과는 아래와 같다.

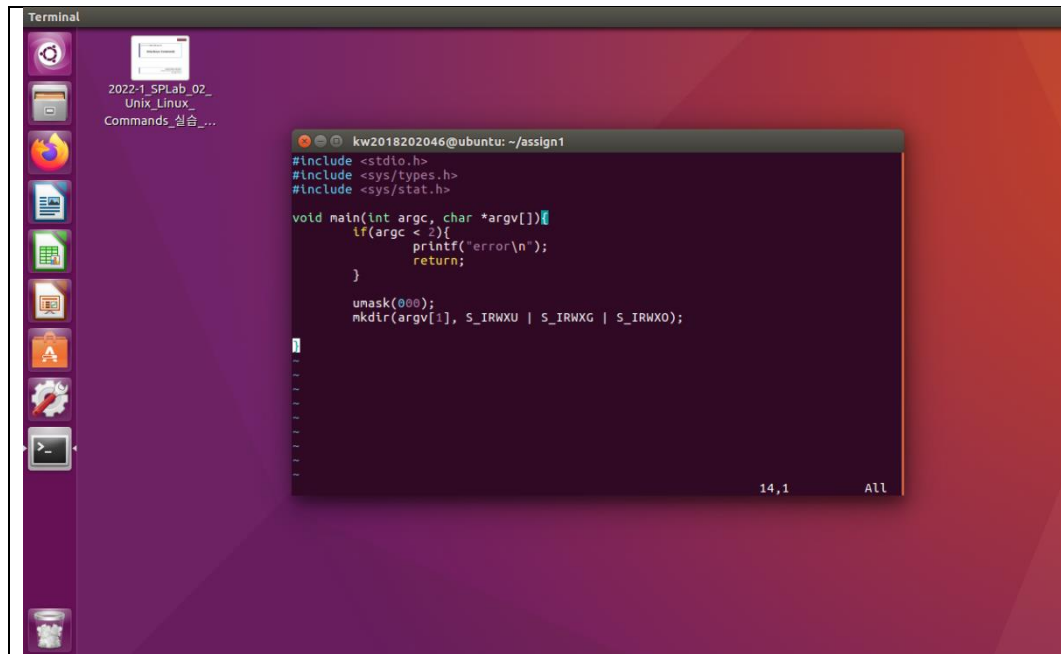


```

kw2018202046@ubuntu: ~/assign1
kw2018202046@ubuntu:~/assign1$ vi test.c
kw2018202046@ubuntu:~/assign1$ gcc test.c
kw2018202046@ubuntu:~/assign1$ ls
a.out test.c
kw2018202046@ubuntu:~/assign1$ ./a.out
error
kw2018202046@ubuntu:~/assign1$ ./a.out 2018202046
kw2018202046@ubuntu:~/assign1$ ls
2018202046 a.out test.c
kw2018202046@ubuntu:~/assign1$ ls -l
total 20
drwxrwxr-x 2 kw2018202046 kw2018202046 4096 Mar 24 03:10 2018202046
-rwxrwxr-x 1 kw2018202046 kw2018202046 8656 Mar 24 03:09 a.out
-rw-rw-r-- 1 kw2018202046 kw2018202046 198 Mar 24 03:07 test.c
kw2018202046@ubuntu:~/assign1$

```

결과를 보았을 때 기존에 mkdir()를 통해 권한을 777로 설정하였지만 실제 권한은 775로 설정된 것을 볼 수 있다. 이는 기본적으로 존재하는 권한 umask()가 002로 설정되어 있기 때문에  $777 - 002 = 775$ (rwxrwxr-x) 권한이 설정되는 것이다.

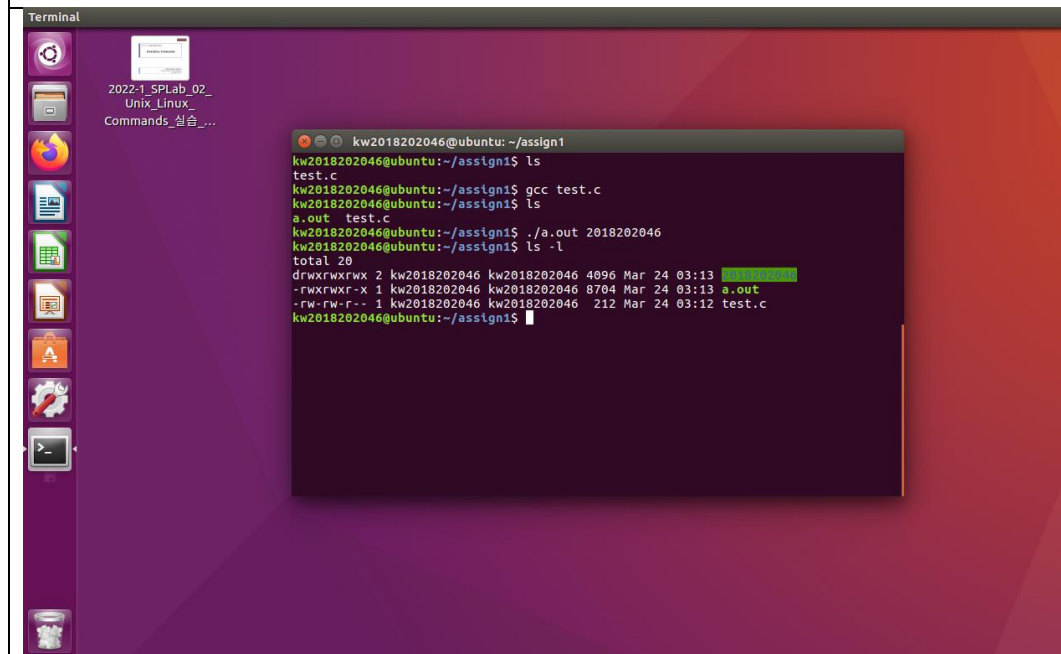


```
kw2018202046@ubuntu: ~/assign1
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>

void main(int argc, char *argv[]) {
    if(argc < 2){
        printf("error\n");
        return;
    }

    umask(000);
    mkdir(argv[1], S_IRWXU | S_IRWXG | S_IRWXO);
}
```

이를 해결하기 위해 mkdir를 하기에 앞서 umask() 함수를 통해 기본 권한을 000으로 설정을 한 모습이다. 이를 실행하면 다음과 같다.



```
kw2018202046@ubuntu: ~/assign1
kw2018202046@ubuntu:~/assign1$ ls
test.c
kw2018202046@ubuntu:~/assign1$ gcc test.c
kw2018202046@ubuntu:~/assign1$ ls
a.out test.c
kw2018202046@ubuntu:~/assign1$ ./a.out 2018202046
kw2018202046@ubuntu:~/assign1$ ls -l
total 20
drwxrwxrwx 2 kw2018202046 kw2018202046 4096 Mar 24 03:13 2018202046
-rwxrwxr-x 1 kw2018202046 kw2018202046 8704 Mar 24 03:13 a.out
-rw-rw-r-- 1 kw2018202046 kw2018202046 212 Mar 24 03:12 test.c
kw2018202046@ubuntu:~/assign1$
```

해당 코드를 살펴보면 만들어진 directory는 정상적으로 777(rwxrwxrwx)권한이 설정된 것을 확인할 수 있다.

```
2022.1.18. Lab 02.
Unix/Linux
Command: ...

// =====
// File Name : proxy_cache.c
// Date : 2022/01/24
// OS : Ubuntu 18.04 LTS 64bits
// Author : Jee Beol Lee
// Student ID : 2018202040
// Title : System Programming Assignment #1.3 (proxy server)
// Description : Making cache directory with hashed URL
// =====
#include <stdio.h> //printf()
#include <string.h> //strcpy()
#include <openssl/sha.h> //SHA1()
#include <sys/types.h> //gethomeDir()
#include <unistd.h> //gethomeDir()
#include <sys/stat.h> //mkdir()

// sha1 hash
// =====
// Input : char *input_url -> input URL
// Output : char * -> hashed URL using SHA1
// Purpose : hashing URL
// =====
char *sha1_hash(char *input_url, char *hashed_url) {
    unsigned char hashed_160bits[20];
    int i;

    SHA1(input_url, strlen(input_url), hashed_160bits);

    for(i = 0; i < sizeof(hashed_160bits); i++)
        sprintf(hashed_hex + "%x", "%02x", hashed_160bits[i]);

    strcpy(hashed_url, hashed_hex);

    return hashed_url;
}

1,1 Top
```

해당 파일은 proxy\_cache.c이다. 우선 sha1\_hash 함수를 살펴보면 input\_url을 받아서 hashed\_url에 저장하고 출력하는 양식을 가지고 있다. 변환의 경우 openssl/sha.h 헤더에서 제공하는 SHA1()함수를 사용하여 변환하고, unsigned char 형 160bit로 변환된 값을 char형으로 변환하기 위한 for문을 도는 모습을 볼 수 있다.

```
2022.1.18. Lab 02.
Unix/Linux
Command: ...

// =====
// gethomeDir
// =====
// Input : char *home -> Store home directory
// Output : char * -> Print home directory
// Purpose : Finding current user's home
// =====
char *gethomeDir(char *home){
    struct passwd *user_info = getpwuid(getuid());
    strcpy(home, user_info->dir);
    return home;
}

void main(void){
    char input_url[100]; //Store input URL(url or ip)
    char hashed_url[100]; //Store hashed URL using SHA1
    char home_dir[100]; //Store user's home directory
    char first_dir[100]; //Directory that will be made in cache directory
    char *dir; //Directory name
    char temp_dir[100]; //Path is used when it makes directory or file
    FILE *temp_file; //Using when make a empty file
    gethomeDir(home_dir); //Store -/cache at home_dir
    strcpy(home_dir, "/cache");

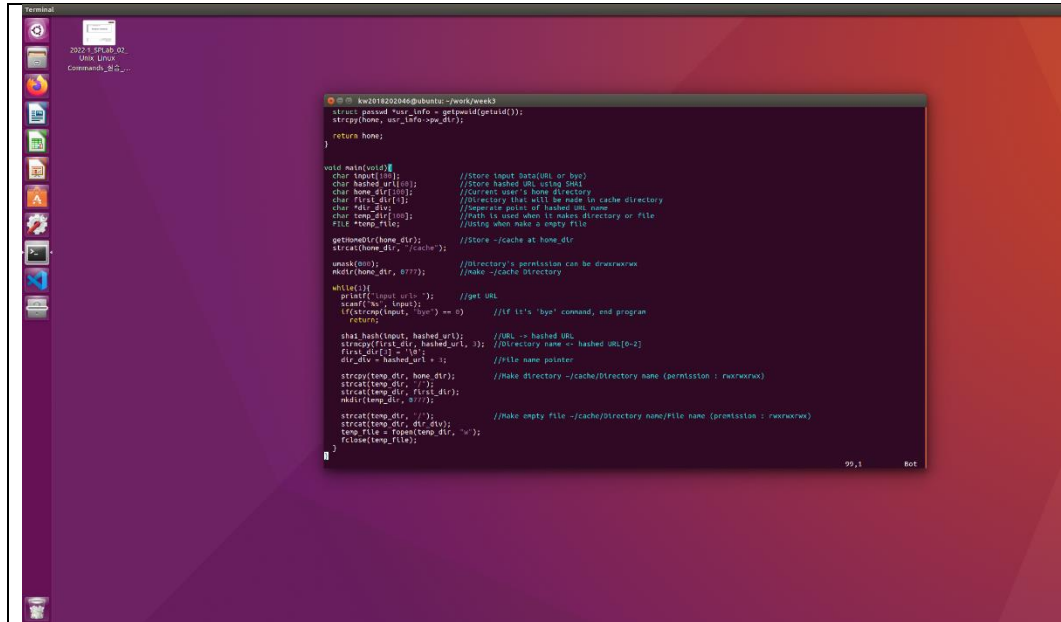
    mkdir(home_dir, 0777); //Directory's permission can be drwxrwxrwx
    //make -/cache directory

    while(1){
        printf("input url: "); //get URL
        scanf("%s", input_url);
        if(strcmp(input_url, "bye") == 0) //if it's "bye" command, end program
            return;

        sha1_hash(input_url, hashed_url); //URL -> hashed URL
        strcpy(first_dir, hashed_url); //Directory name <- hashed URL[0-2]
        first_dir[3] = '\0'; //file name pointer
        strcpy(temp_dir, home_dir); //Make directory -/cache/directory name (permission : rwxrwxrwx)
    }
}

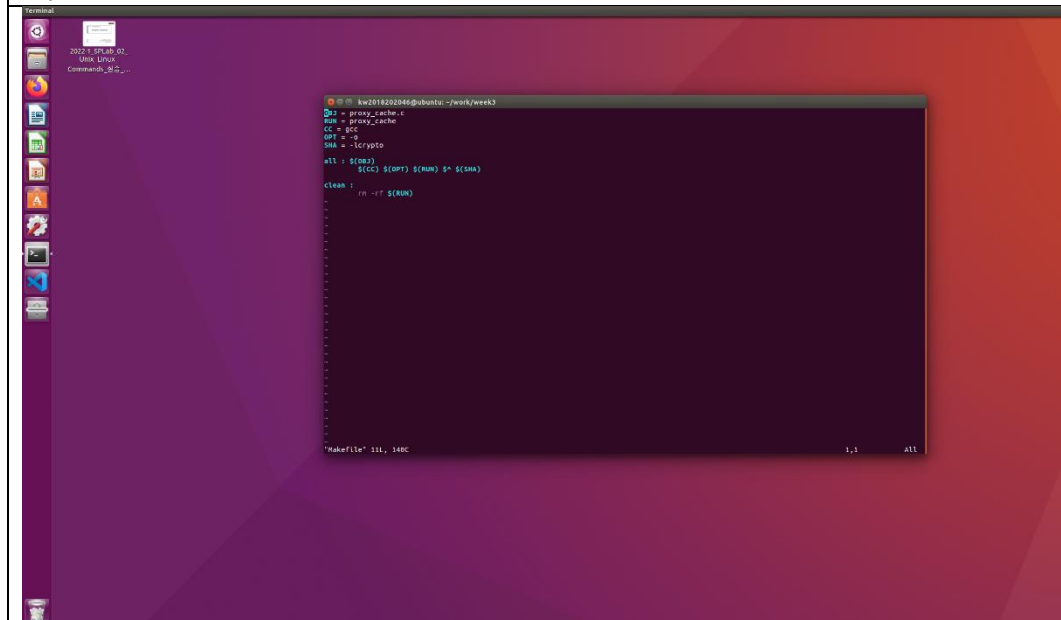
40,0 / 1 81%
```

해당 함수 또한 위의 함수와 마찬가지로 기본적으로 주어졌다. 해당 함수는 getpwuid(getuid())를 통해 현재 사용자의 정보를 가져와 passwd 구조체에 넣으며 여기에서 home directory를 찾아 출력하는 모습을 볼 수 있다.



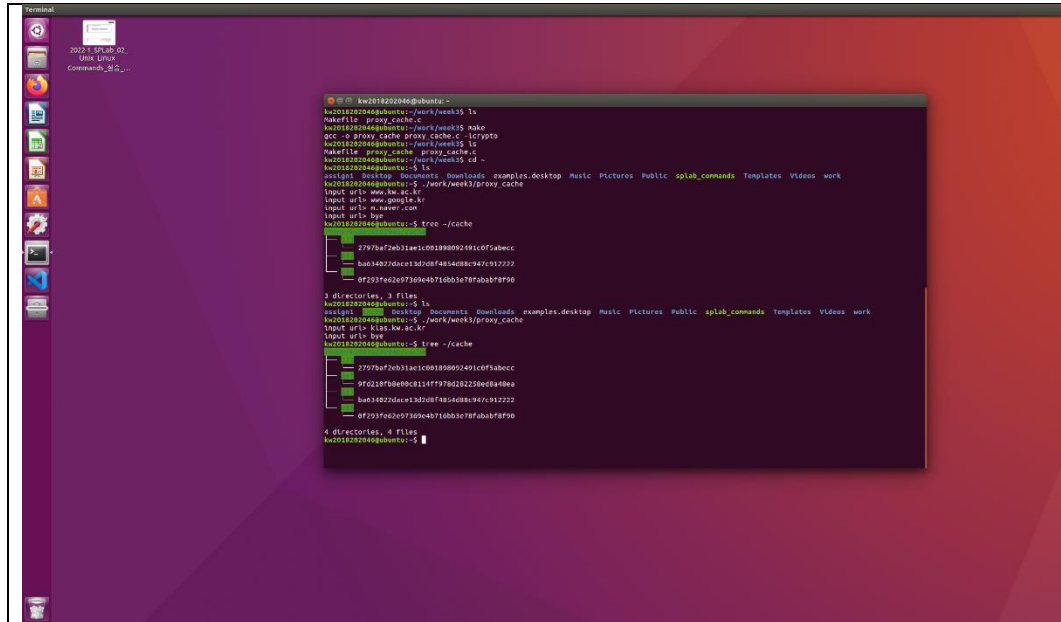
```
1 struct passwd *usr_info = getpwuid(getuid());
2 strcpy(home, usr_info->pw_dir);
3
4 return home;
5 }
6
7 void main(void)
8 {
9     char input[1024]; //Store input data(URL or file)
10    char hashed_url[64]; //Store hashed URL using SHA1
11    char home_dir[1024]; //Store user's home directory
12    char first_dir[128]; //Directory that will be made in cache directory
13    char *dir_ptr; //Pointer to the directory name
14    char temp_dir[1024]; //Path to used when it makes directory or file
15    FILE *temp_file; //File when make a empty file
16
17    gethomeDir(home_dir); //Store -jcache at home_dir
18    strcat(home_dir, "/cache");
19
20    umask(000); //Directory's permission can be rwxrwxrwx
21    mkdir(home_dir, 0777); //Make -jcache directory
22
23    while(1)
24    {
25        printf("Input url: "); //get URL
26        scanf("%s", input);
27        if(strcmp(input, "bye") == 0) //If it's 'bye' command, end program
28            return;
29
30        sha1_hash(input, hashed_url); //URL -> hashed URL
31        strcpy(first_dir, hashed_url); //Directory name -> hashed URL[0-3]
32        first_dir[4] = '\0';
33        dir_ptr = hashed_url + 4; //File name pointer
34
35        strcpy(temp_dir, home_dir); //Make directory -jcache/directory name (permission : rwxrwxrwx)
36        strcat(temp_dir, first_dir);
37        mkdir(temp_dir, 0777);
38
39        strcpy(temp_dir, "/"); //Make empty file -jcache/directory name/file name (permission : rwxrwxrwx)
40        temp_file = fopen(temp_dir, "w");
41        fclose(temp_file);
42    }
43 }
```

위의 사진은 메인 함수를 나타낸 것이다. getHomeDir()함수를 통해 현재 위치를 불러와 home\_dir에 저장하고 strcat()함수를 통해 뒤쪽에 /cache를 붙인다. 그 후 umask를 통해 기본 권한을 000으로 설정, mkdir() 통해 cache폴더가 없을 경우 만든다. while문에서는 URL을 받고 이를 sha1\_hash()함수를 통해 변환 후 일부의 이름은 directory로, 일부는 file명으로 저장하는 역할을 수행한다. 파일의 생성은 fopen()함수를 활용하여 제작되었다.



```
1 prony_cache:
2     RMK = prony_cache
3     CC = gcc
4     OPT = -g
5     SHL = -lcrypto
6
7     all : $(RMK)
8           $(CC) $(OPT) $(RMK) -o $(RMK)
9
10    clean :
11        rm -f $(RMK)
12
13    "Makefile" 311, 140C
```

해당 그림은 Makefile의 내용을 표시한 것이다. 기존의 Makefile과 다른 점이 있다면 SHA 변수를 통해 -lcrypto 옵션이 사용된다는 점인데, 이는 SHA1()함수를 쓰기 위해서 필수적으로 요구된다.



```
kw201820246@ubuntu:~/week3$ ls
Makefile proxy_cache.c
kw201820246@ubuntu:~/week3$ make
gcc -o proxy_cache proxy_cache.c -lcrypto
kw201820246@ubuntu:~/week3$ ls
Makefile proxy_cache proxy_cache.c
kw201820246@ubuntu:~/week3$ cd -
kw201820246@ubuntu:~$ ls
Desktop Documents Downloads examples.desktop Music Pictures Public splab_commands Templates Videos work
kw201820246@ubuntu:~$ ./work/week3/proxy_cache
Input url: www.kw.ac.kr
Input url: www.google.kr
Input url: www.naver.com
Input url: byp
kw201820246@ubuntu:~$ tree -j/cache
tree
├── 2797ba72eb31ae1c051898092491c0f5abccc
├── ba634827dacc13d28f481468bc947c912222
└── 0f293f62ce973d9e4b714bb3c78fab878f96
3 directories, 3 files
kw201820246@ubuntu:~$ ls
Desktop Documents Downloads examples.desktop Music Pictures Public splab_commands Templates Videos work
kw201820246@ubuntu:~$ ./work/week3/proxy_cache
Input url: kls.kw.ac.kr
Input url: byp
kw201820246@ubuntu:~$ tree -j/cache
tree
├── 2797ba72eb31ae1c051898092491c0f5abccc
├── 0f293f62ce973d9e4b714bb3c78fab878f96
├── ba634827dacc13d28f481468bc947c912222
└── 0f293f62ce973d9e4b714bb3c78fab878f96
4 directories, 4 files
kw201820246@ubuntu:~$
```

위의 사진은 실행 결과를 나타낸 것이다. make을 통해 정상적으로 파일이 생성되고, 기존에 cache directory가 없는 상태에서 이를 실행하면 정상적으로 생성되는 것을 볼 수 있다. 또한 url을 입력하면 해당하는 url의 앞 3글자는 directory명으로, 뒤의 글자는 파일 명으로 생성되는 모습을 볼 수 있다. 한 번 더 실행하였을 때에는 cache폴더가 기존에 있는 상황이며 이 상황에서도 마찬가지로 정상적으로 url이 추가가 되는 모습을 볼 수 있다.

## 5. 고찰

해당 과제를 진행하면서 우선 시스템 프로그래밍 과목에서 배웠던 mkdir()와 umask()를 사용하였다. 이로 인하여 해당 함수의 사용을 익히고 이를 활용할 수 있었다. 또한 코드를 작성하면서 vi 편집기를 최대한 활용하여 프로그래밍을 함으로써 3주차에서 익혔던 내용을 이용할 수 있는 과제였다. tree 명령어를 설치하고 사용하면서 파일의 구조를 쉽게 확인하는 방법을 알 수 있었다. 마지막으로 strcat과 strcpy 등의 문자열 함수를 사용하여 문자열을 쉽게 조작하는 법을 터득할 수 있는 과제였다.

## 6. Reference

강의 자료만을 참고