시스템프로그래밍
Proxy #1-1

담당교수 : 최상호 교수님

2021202003 강준우

mkdir()

Introduction

Ubuntu에서 디렉토리를 제작하는 코드를 수행해봅니다. 해당 과정에서 권한을 0777로 부여한 후 해당 부여한 권한을 확인해봅니다. 코드와 결과를 비교해보고 이후 해당 코드가 의도한대로 수행되기 위해서 어떻게 해야하는지 알아봅니다. 또한 리눅스 환경에서의 unmask에 대한 기능을 확인합니다.

결과화면

```
kw2021202003@ubuntu:~/work$ ./mkdir_test testdir1
kw2021202003@ubuntu:~/work$ ls -ld testdir1
drwxrwxr-x 2 kw2021202003 kw2021202003 4096 Mar 30 00:13 testdir1
kw2021202003@ubuntu:~/work$ cat mkdir_test.c
#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);
```

vi를 통해서 해당 코드를 작성합니다.

해당 코드는 입력받은 문자열을 이름으로 하는 디렉토리를 rwxrwxrwx 0777 권한으로 만들고자 합니다.

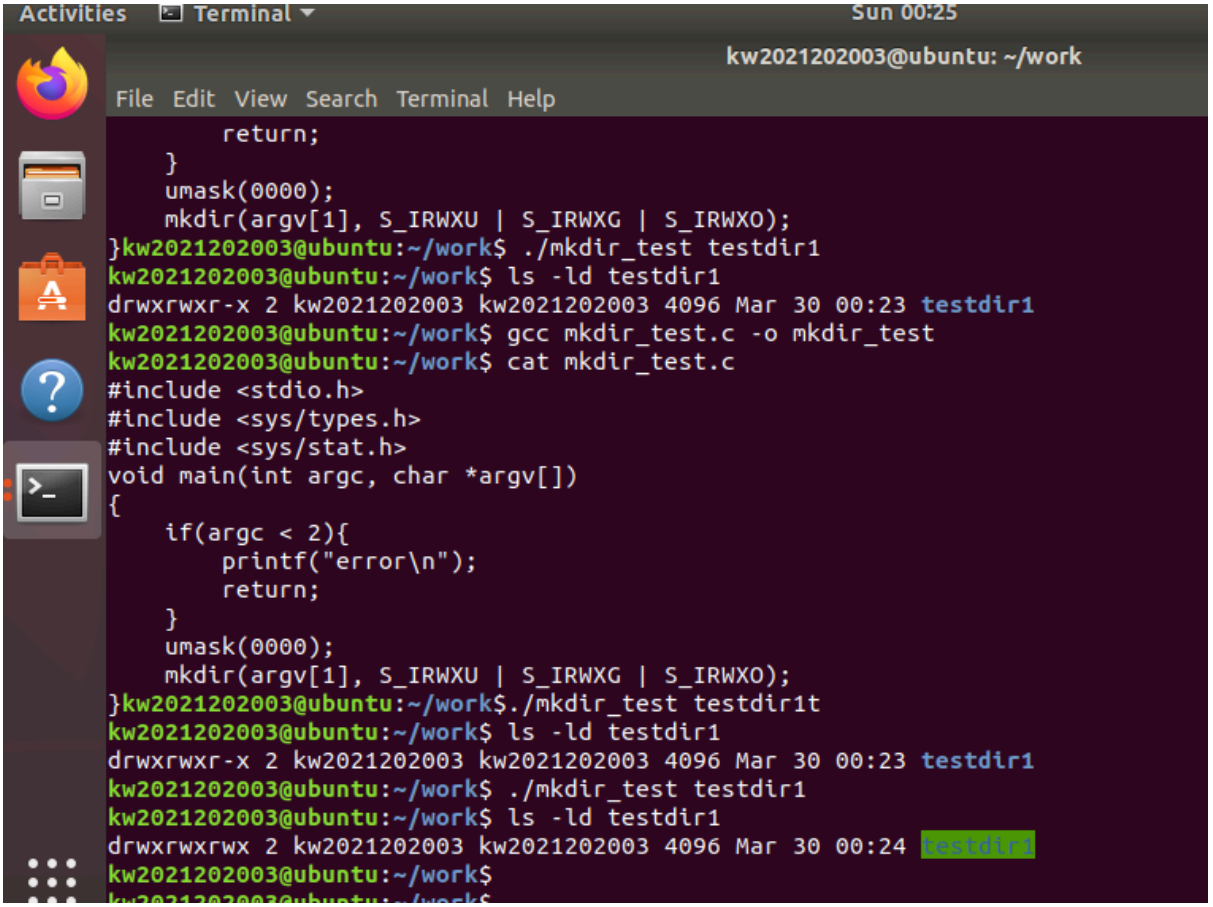
해당 코드 컴파일 후 testdir경로를 제작합니다.

이후 해당 경로의 권한을 확인해보면 drwxrwxr-x 0775 로 확인할 수 있습니다.

해당 코드에서의 문제점 원인은 Linux 시스템에서 unmask가 해당 코드에 권한에 제한을 걸은 모습입니다.

unmask로 0002가 적용되어 0777>0775로 변경되어있습니다.

해결방법은 unmask가 실행되지 못하게 코드에서 unmask를 0000으로 지정해둡니다.



이후 정상적으로 해당 0777의 권한을 가져간 모습입니다.

고찰

이번 과제에서는 디렉토리를 지정함에 불구하고 unmask가 0002로 적용되어 권한이 제한되는 모습을 확인했습니다. 개인적으로 왜 unmsk가 있서 매번 unmask를 0000으로 초기화 해야할까 그냥 디폴트로 000이 들어가 있으면 되는것이 아닌가? 라고 생각했습니다. 하지만 마지막 부분에서 others를 우연히 오픈해버리면 코드에 다른 사람이 영향을 끼칠 수 있지 않을까 라는 생각을 하기도 했습니다. 보안상의 문제가 이런 자동 제한을 걸었다고 생각하게 되었습니다.

Reference

Proxy #1-1

Introduction

해당 과제는 입력받은 URL데이터를 SHA-1로 변환하고 해당 결과를 기반으로 디렉토리와 파일을 생성하는 코드를 제작한다. 문제에서는 다음과 같은 내용을 제시한다.

Input url은 웹사이트 명으로 입력

Bye command를 입력으로 받았을 때만 프로그램 종료

제공된 sha1_hash 함수를 사용하여 입력 받은 URL을 Hashed URL로 변환

getHomedir 함수를 사용하여 Home directory path를 얻고, Home directory에 cache 디렉토리를 생성

다음과 같은 조건과 요구사항을 수행해본다.

결과화면

```
kw2021202003@ubuntu:~/work$ sudo apt-get install libssl-dev
[sudo] password for kw2021202003:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  libssl-doc
The following NEW packages will be installed:
  libssl-dev
0 upgraded, 1 newly installed, 0 to remove and 334 not upgraded.
Need to get 1,568 kB of archives.
After this operation, 7,854 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libssl-dev
 [1,568 kB]
Fetched 1,568 kB in 3s (494 kB/s)
Selecting previously unselected package libssl-dev:amd64.
(Reading database ... 157277 files and directories currently installed.)
Preparing to unpack .../libssl-dev_1.1.1-1ubuntu2.1~18.04.23_amd64.deb ...
Unpacking libssl-dev:amd64 (1.1.1-1ubuntu2.1~18.04.23) ...
```

우선 필요한 SHA-1 라이브러리를 설치해줍니다. 또한 앞으로 해당 라이브러리를 사용하기위해 컴파일 때 -lcrypto를 추가해줄것 입니다.

```
Open
#include <sys/types.h>
#include <unistd.h>
#include <pwd.h>
#include <string.h>


char *getHomeDir(char *home) {
    struct passwd *usr_info = getpwuid(getuid());
    strcpy(home, usr_info->pw_dir);
    return home;
}
```

home 디렉토리 경로를 반환하는 코드를 먼제 제작합니다.
이후 make파일을 실행할 때 해당 코드를 반영하여 실행파일을 제작해줍니다.

```c
char *getHomeDir(char *home);

char* sha1_hash(char* input_url, char* hashed_url) {
    unsigned char hashed_160bits[20];
    char hashed_hex[41];
    int i;

    SHA1((unsigned char*)input_url, strlen(input_url), hashed_160bits);
    for (i = 0; i < sizeof(hashed_160bits); i++) {
        sprintf(hashed_hex + i * 2, "%02x", hashed_160bits[i]);
    }

    strcpy(hashed_url, hashed_hex);
    return hashed_url;
}
```

빈칸에는 각각 다음과 같은 코드가 들어갑니다.

(unsigned char*)input_url, strlen(input_url), hashed_160bits

: input_url 값을 SHA1알고리즘으로 헤쉬화 하고 hashed_160bits에 넣습니다.

hashed_url, hashed_hex

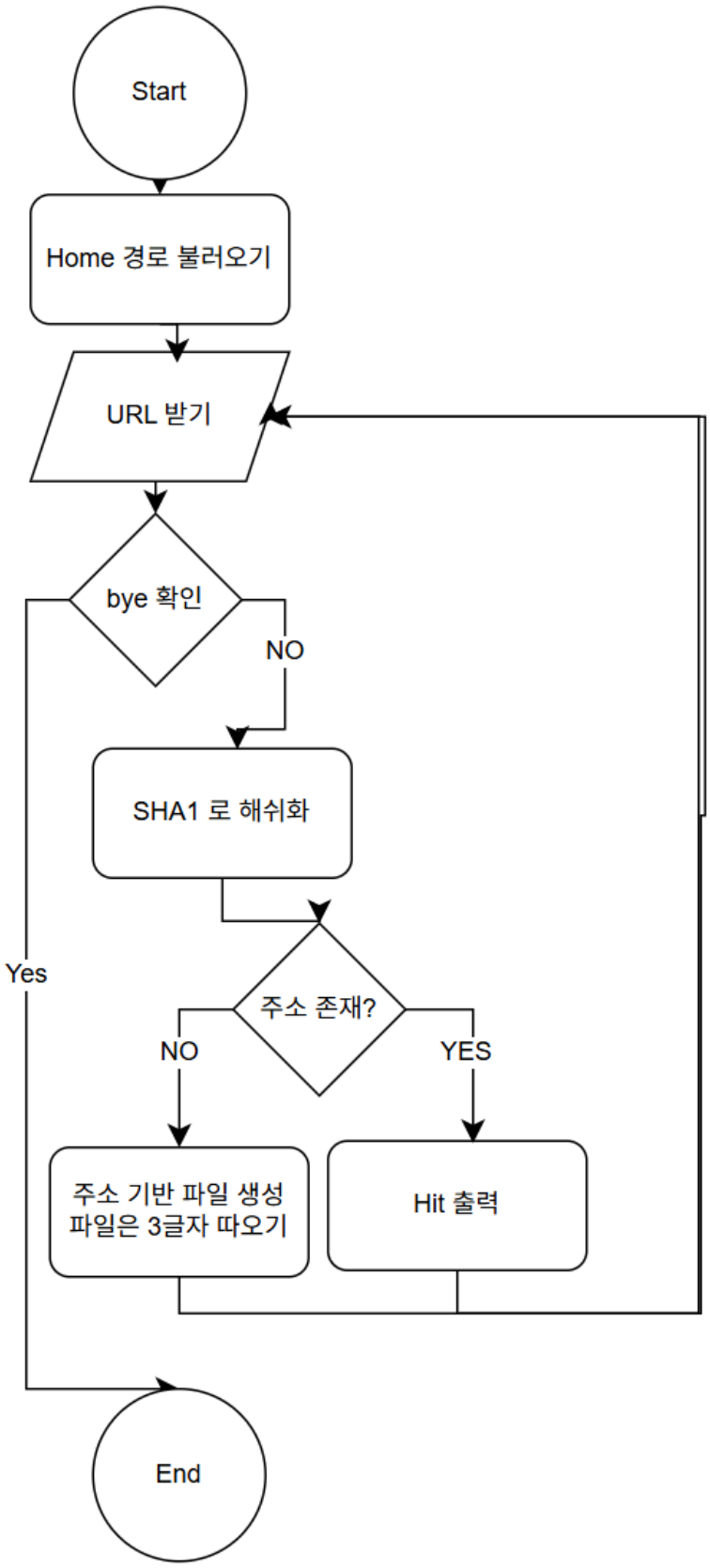
: hashed_hex 데이터를 hashed_url에 복사하고 반환합니다.

```
kw2021202003@ubuntu:~/work$ make proxy_cache
gcc -o proxy_cache proxy_cache.c homedir.c -lcrypto
kw2021202003@ubuntu:~/work$ ./proxy_cache
input url> www.kw.ac.kr
input url> www.google.com
input url> bye
kw2021202003@ubuntu:~/work$ tree ~/cache/
/home/kw2021202003/cache/
├── d8b
│   └── 99f68b208b5453b391cb0c6c3d6a9824f3c3a
└── e00
    └── 0f293fe62e97369e4b716bb3e78fababf8f90

2 directories, 2 files
kw2021202003@ubuntu:~/work$ cd ..
kw2021202003@ubuntu:~$ cd cache/
kw2021202003@ubuntu:~/cache$ ls -l
total 8
drwxrwxrwx 2 kw2021202003 kw2021202003 4096 Mar 30 02:19 d8b
drwxrwxrwx 2 kw2021202003 kw2021202003 4096 Mar 30 02:19 e00
```

homedir를 참고한 실행파일을 수행 후 주소를 입력해주고 bye로 프로그램을 종료합니다.
이후 tree를 통해서 해당 코드가 해쉬화 되어 만들어져 있는 모습을 확인할 수 있습니다.
이후 권한들이 777로 들어가 있는 모습을 확인할 수 있습니다.

Algorithm – Flow Chart



Pseudo Code

1. 프로그램 시작 시
먼저 사용자의 홈 디렉토리 경로를 가져옵니다.
이 경로는 이후에 캐시를 저장할 ~/cache 폴더를 만들기 위해 사용됩니다.

umask(0000)을 설정해서, 나중에 만들 파일과 디렉토리의 권한을 제한 없이 777로 설정할 수 있도록 해둡니다.

2. 사용자 입력 반복
프로그램은 계속해서 사용자로부터 URL을 입력받습니다.
사용자가 bye라고 입력하면 프로그램은 종료됩니다.

3. URL 해싱
입력된 URL을 SHA1 알고리즘을 이용해 해시 문자열로 변환합니다.
이 해시값은 파일 이름과 디렉토리 구조에 활용됩니다.

4.해시값의 앞 세 글자를 따서 서브 디렉토리 이름으로 사용합니다.
~/cache/a1b 같은 디렉토리도 생성합니다.
이 구조는 해시 충돌을 줄이고 검색 속도를 높이기 위한 설계입니다.

5. 캐시 파일 존재 여부 확인
~/cache/a1b 디렉토리를 엽니다.
해당 파일이 이미 있다면 → "Hit" 메시지를 출력합니다.

없다면 해당 이름의 파일을 빈 파일로 생성합니다.

고찰

이번 프로그램을 제작하면서 컴파일 시 다른 코드를 추가하여 실행파일을 만들 수 있는 법을 알게 되었습니다. 이전에 객제지향 프로그래밍에서 배운 방법은 부모 자식 관계로 짓거나 친구 관계로 불러오는 방법을 배웠는데 이렇게 컴파일 방법도 있음을 배웠습니다. 또한 해당 코드를 제작하면서 추가적으로 #include <sys/stat.h>를 사용하였는데 이전에 mkdir() 과제에서 해당 선언을 통해서 권한과 unmask등을 사용한 경험을 바탕으로 사용해보았습니다. 코드를 작성할 때 dir함수를 사용하여 문제를 해결할수 있었습니다.

Reference