System Programming

Assignment#1-2

03_Proxy 1-2

Professor	목 3 4 황호영 교수님
Department	컴퓨터공학과
Student ID	2012722028
Name	장 한 별
Date	2018. 04. 05

Introduction.

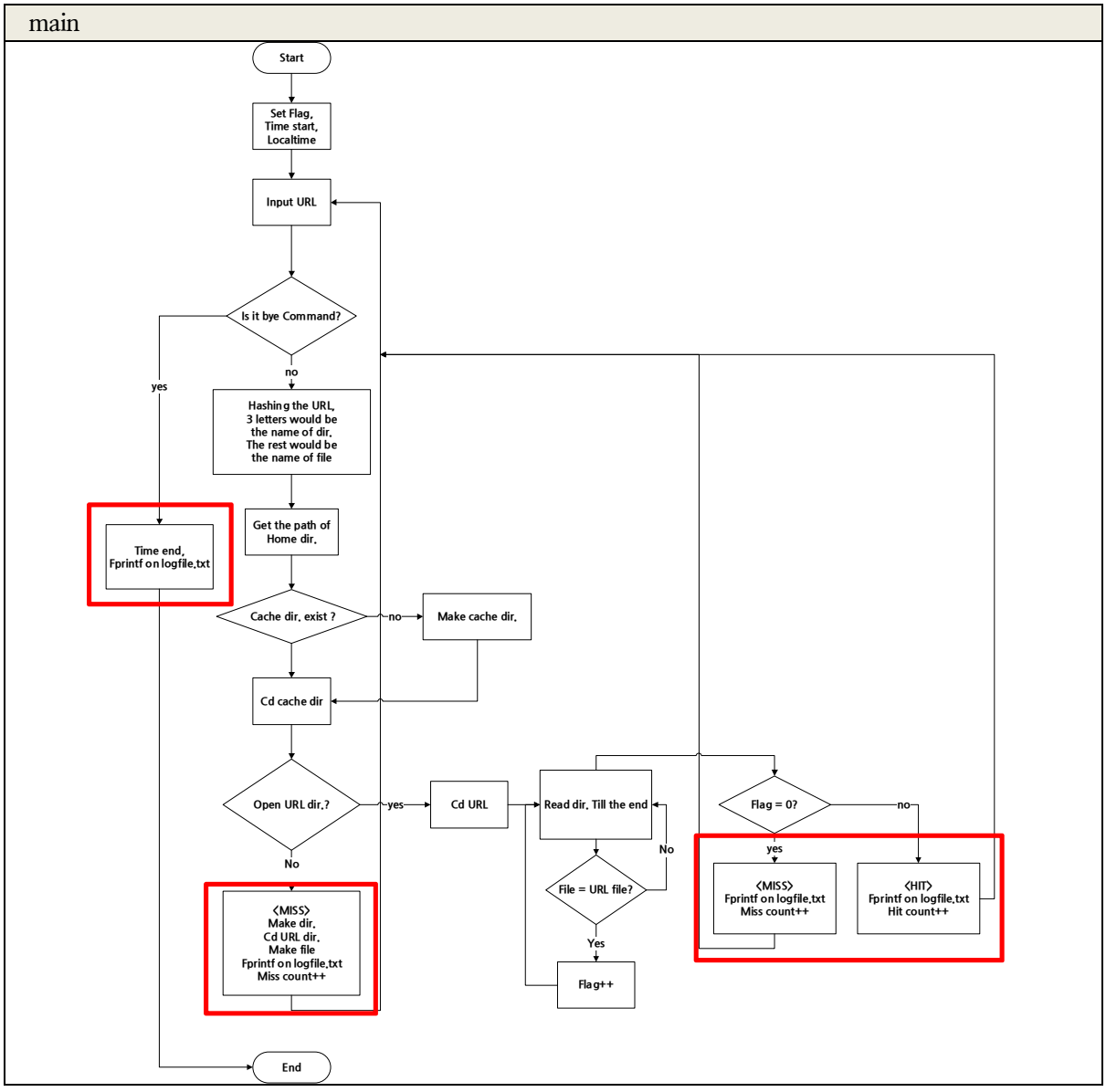
시스템 프로그래밍 강의 시간에 배운 proxy server 를 구현하는 것을 목표로 한다.

이번 과제는 Assignment 1-1 에서 구현한 코드를 바탕으로 정해진 format (URL을 검색했을 때의 시간, Hit 의 갯수, Miss 의 개수, Hit 일 때는 'directory name / filename'을, Miss 일 때는 URL 을 출력, 프로그램 실행 시간)을 logfile.txt 에 저장하도록 한다.

이때 logfile.txt 는 프로그램이 재실행 되어도 이전 내용의 내용을 유지해야한다.

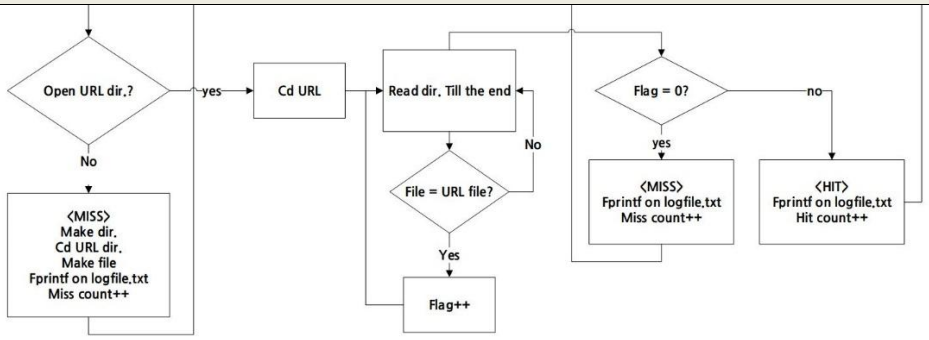
Flow chart.

main



위 그림은 이번 과제 전체의 Flow chart 이다. 알고리즘의 흐름이 어떻게 흘러가는지 시각적으로 한눈에 확인할 수 있다. Assignmnet 1-1 에 실행시간, count, 파일 입출력이 추가되었다.

Hit, Miss



위 그림은 Hit와 Miss 시의 Flow Chart 이다. 애매모호할 수 있는 부분이니 상세하게 알아야 하는 부분이다. Dir 와 File 을 생성하는 부분은 같은 Directory가 있을 경우와 같은 Directory가 없을 경우(Miss), 전자의 경우에 같은 file 이 있을 경우(Hit)와 같은 file이 없을 경우(Miss)로 크게 3가지로 나눌 수 있다. 이 Flow chart 를 이용해 명확하게 구분할 수 있다.

Pseudo code.

main

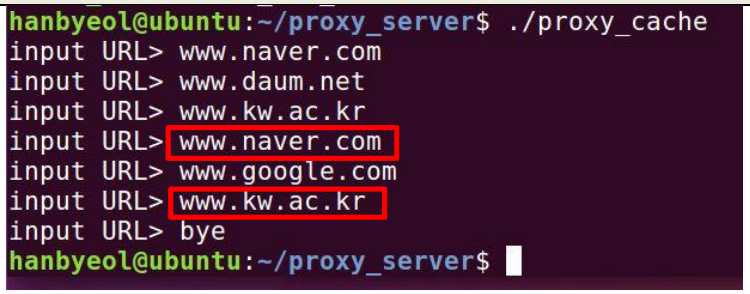
```
Int main(int argc, char* argc[])
{
    Time start;
    While(1)
    {
        Set Flag;
        Localtime;

        Input URL;
        If(input == "bye")
        {
            Time end;
            Fprintf(logfile, "Terminated", runtime, hit, miss);
            break;
        }
        Hashing;
        String copy(directory name, hashed URL, 3);
        String copy(file name, heshed URL +3);
        Make directory (cache);
        Cd dir.(cache);

        If(directory name==NULL)
        {
            Make directory(Hashed URL);
```

```
            Change directory(Hashed URL);
            Make file(Hashed URL);
            Fprintf(logfile, “Miss”, input, year/mon/mday/hour/min/sec);
            Miss count++;
        }
        Else
        {
            Cd dir.(cache);
            While( read dir. )
            {
                If(same filename , hashed filename)
                {
                    Flag++;
                }
            }
            If( flag==0)
            {
                Fprintf(logfile, “Miss”, input, year/mon/mday/hour/min/sec);
                Miss count++;
            }
            Else
            {
                Fprintf(logfile, “Hit”, dir/file, year/mon/mday/hour/min/sec);
                Hit count++;
             }
        }
    }
}
```

위 그림은 이번 과제 main 함수의 pseudo code 이다. 실제 c 언어로 어떻게 작성되었는지 간략하게 설명하여 코드에 대한 이해도를 높일 수 있다.

Result.

1-2.(1)



위 그림은 이번 과제에서 구현한 실행파일을 실행시켰을 때의 사진이다. Input URL에 www.naver.com, www.daum.net, www.kw.ac.kr, www.naver.com, www.google.com, www.kw.ac.kr

을 차례대로 입력시켰고, 그 후 bye command 로 프로그램을 종료시켰다.

1-2.(2)

```
hanbyeol@ubuntu:~$ ls
c&cpp      Desktop    Downloads         Music  Pictures      Public     Videos
CProjects  Documents  examples.desktop  my     proxy_server  Templates
```

```
hanbyeol@ubuntu:~$ ls
cache      Desktop    examples.desktop  my            Public
c&cpp      Documents  logfile           Pictures      Templates
CProjects  Downloads  Music             proxy_server  Videos
hanbyeol@ubuntu:~$
```

위 그림은 1-2.(1) 의 실행 결과 cache directory 와 logfile directory 가 올바르게 생성되었음을 확인할 수 있다.

1-2.(3)

```
hanbyeol@ubuntu:~/proxy_server$ ls -R ~/cache
/home/hanbyeol/cache:
025  d8b  e00  fed

/home/hanbyeol/cache/025:
15aba49d232baf9567495aa44eb8d20b2b764

/home/hanbyeol/cache/d8b:
99f68b208b5453b391cb0c6c3d6a9824f3c3a

/home/hanbyeol/cache/e00:
0f293fe62e97369e4b716bb3e78fababf8f90

/home/hanbyeol/cache/fed:
818da7395e30442b1dcf45c9b6669d1c0ff6b
```

```
hanbyeol@ubuntu:~/proxy_server$ tree ~/cache
/home/hanbyeol/cache
├── 025
│   └── 15aba49d232baf9567495aa44eb8d20b2b764
├── d8b
│   └── 99f68b208b5453b391cb0c6c3d6a9824f3c3a
├── e00
│   └── 0f293fe62e97369e4b716bb3e78fababf8f90
└── fed
    └── 818da7395e30442b1dcf45c9b6669d1c0ff6b

4 directories, 4 files
hanbyeol@ubuntu:~/proxy_server$
```

위 그림은 1-2.(1) 의 실행 결과, 생성된 directory 와 file 들을 'ls -R ~/cache' 명령어와 'tree ~/cache' 명령어를 이용해 확인한 결과 화면이다. 입력한 URL 들이 올바르게 Hash 된 후 정확한 위치에 directory 와 file 들을 생성했음을 확인할 수 있다.

1-2.(4)

```
hanbyeol@ubuntu:~$ cat ~/logfile/logfile.txt
[Miss]www.naver.com-[2018/4/5, 20:49:05]
[Miss]www.daum.net-[2018/4/5, 20:49:10]
[Miss]www.kw.ac.kr-[2018/4/5, 20:49:13]
[Hit]fed/818da7395e30442b1dcf45c9b6669d1c0ff6b-[2018/4/5, 20:49:19]
[Hit]www.naver.com
[Miss]www.google.com-[2018/4/5, 20:49:22]
[Hit]e00/0f293fe62e97369e4b716bb3e78fababf8f90-[2018/4/5, 20:49:27]
[Hit]www.kw.ac.kr
[Terminated] run time: 30 sec. #request hit: 2, miss: 4
hanbyeol@ubuntu:~$
```

위 그림은 1-2.(1) 의 실행 결과 생성된 logfile directory 안의 logfile.txt 의 내용을 출력한 결과 화면이다. 입력한 URL 이 Miss 상태인지 Hit 상태인지 출력하고, 입력한 날짜와 시간도 출력한다. Miss 시 입력한 URL 만 출력하지만 Hit 시에는 입력한 URL 의 directory 이름, '/', file 의 이름 순으로 출력하고, 입력한 URL 도 추가로 출력한다.

www.naver.com, www.daum.net, www.kw.ac.kr, www.google.com 은 처음으로 입력했으니 Miss, 추가로 www.naver.com 와 www.kw.ac.kr 를 입력 했을 시에는 Hit 이다. 마지막으로 프로그램의 실행 시간과, Hit 의 갯수 miss의 갯수 를 출력해준다. 완벽하게 출력되었음을 위 사진에서 확인할 수 있다.

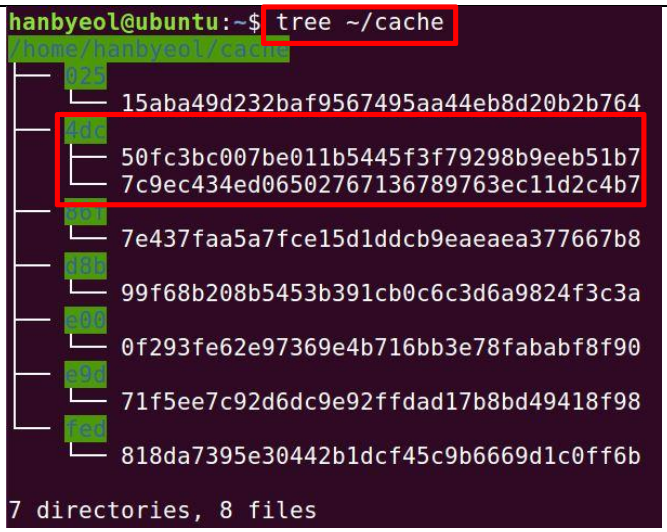
1-2.(5)

```
hanbyeol@ubuntu:~/proxy_server$ ./proxy_cache
input URL> a
input URL> b
input URL> a
input URL> r
input URL> stu
input URL> bye
hanbyeol@ubuntu:~/proxy_server$
```

1-2.(1) ~ 1-2.(4) 까지를 확인해 보면 Miss 와 Hit 가 제대로 판별된 것 같이 생각할 수 있다. 하지만 우연히 Directory 이름이 같지만, file 이름이 다른 Miss 상황이 아주 드물게 발생할 수 있다. 위 그림은 그에 대한 예로 r 과 stu 를 입력했을 때의 화면이다.

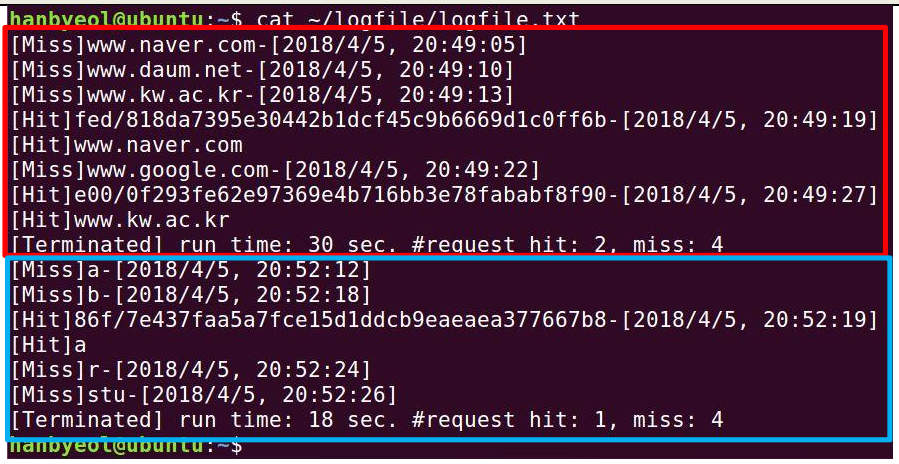
1-2.(6)

```
hanbyeol@ubuntu:~$ ls -R ~/cache
/home/hanbyeol/cache:
025  4dc  86f  d8b  e00  e9d  fed

/home/hanbyeol/cache/025:
15aba49d232baf9567495aa44eb8d20b2b764

/home/hanbyeol/cache/4dc:
50fc3bc007be011b5445f3f79298b9eeb51b7  7c9ec434ed06502767136789763ec11d2c4b7

/home/hanbyeol/cache/86f:
7e437faa5a7fce15d1ddcb9eaeaea377667b8

/home/hanbyeol/cache/d8b:
99f68b208b5453b391cb0c6c3d6a9824f3c3a

/home/hanbyeol/cache/e00:
0f293fe62e97369e4b716bb3e78fababf8f90

/home/hanbyeol/cache/e9d:
71f5ee7c92d6dc9e92ffdad17b8bd49418f98

/home/hanbyeol/cache/fed:
818da7395e30442b1dcf45c9b6669d1c0ff6b
hanbyeol@ubuntu:~$
```



위 그림에서 확인할 수 있듯이, r 과 stu 는 hash된 URL 의 앞 3글자가 4dc 인 것을 확인할 수 있다. 하지만 우연히 3글자가 같을 지라도, 언뜻 보기에도 r 과 stu 가 다른 문자열 이듯, 앞 3글자 이후의 문자열은 완전히 다르다. 이럴 경우 같은 directory 가 있으면 그 안의 file 들을 모두 read 한 후 파일이 다르면 그 directory 안에 file을 생성(Miss)하고, 같으면 directory 와 file 이 모두 같은 경우이므로 Hit 로 구현했다.

1-2.(7)



이 후 logfile.txt 의 내용을 확인한 화면이다. 빨간색 박스 안에 있는 내용은 1-2.(1) 의 결과이고, 파란색 박스 안에 있는 내용은 1-2.(5)의 결과이다. 프로그램이 재실행 되어도 이전 내용이 올바르게 유지되는 것을 확인할 수 있다.

Conclusion.

Assignment 1-1 에서 구현한 코드를 바탕으로 정해진 Format 에 맞춰 logfile에 저장하는 것이 이번 과제의 목표였다. Assignment 1-1 에서 세분화 시켜 정확하게 구현 했었다면, 이번 과제는 크게 어렵지 않았을 것이다. 그 전 과제에선 단순히 directory 가 cache 에 있는 지만 확인하고 Miss 와 Hit 를 나눴었기 때문에 좀 더 세밀한 형태인, Directory 이름이 같지만 file 이름이 다른 경우를 해결하지못해 코드를 대폭 수정해야했다. Readdir 함수의 사용법과 무엇을 반환하는지 정확하게 안 후에야 구현할 수 있었고, 그 후 그

구분을 flag 를 넣어서 Miss 와 Hit를 판별했다. 결과 화면에서 볼 수 있듯이 올바르게 해결할 수 있었다.

프로그램 실행 시간을 출력하는 부분도 생각보다 쉽게 해결되지 않았다. Time 함수를 써야하는지 clock 함수를 써야하는지 많이 헷갈렸고, 시간을 측정하는 부분인 start 부분과 시간 측정이 끝나는 부분인 end 부분을 각각 어디에 넣어줘야 할 지 헷갈렸고, 실제로 구현을 한 후 에도 스톱워치를 이용해 시간을 직접 재서 비교해 완벽하게 구현되었음을 확인했다.

이번 과제는 코드의 실행 결과를 terminal 에 출력하는 것이 아닌 파일에 저장하는 것이기 때문에 파일 입출력에 대한 기본적인 문법과 활용도를 이해할 필요가 있었다. 오랜만에 사용했기 때문에 처음에는 헷갈렸지만 알고리즘에 대해 크게 영향을 주는 부분이 아니기 때문에 어렵지 않게 구현할 수 있었다. 이때, 파일이 없으면 새로 생성하고, 새로운 프로그램이 실행 되었을 때 기존의 내용이 유지되어야 하기 때문에 'a' 모드를 입력해서 파일을 open 했다.

기본적인 proxy server 의 원리를 구현했으니 이 프로세스에 대해 관련된 작업은 Assignment 1-3 에서 구현하도록 한다.