System Programming

Assignment#2-1

05_Proxy 2-1

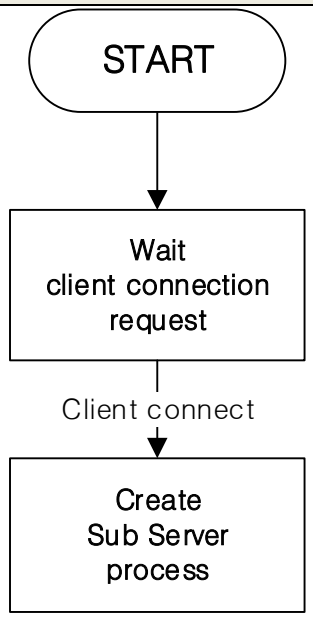
Professor	목 3 4 황호영 교수님
Department	컴퓨터공학과
Student ID	2012722028
Name	장 한 별
Date	2018. 04. 27

Introduction.

시스템 프로그래밍 강의 시간에 배운 proxy server 를 구현하는 것을 목표로 한다.
이번 과제는 Client 와 Server 가 통신하는 것(connection)을 구현한다. Socket을 생성하고 주소를 부여받고, 연결 요청을 대기하다가 수락받으면 Child process를 생성한다. 이후 Resquest(전송) 와 Response(수신)를 적절히 이용하여 정보를 서로 주고 받는 Socket 을 Programming 을 하는 것이 목표이다. 이때, Socket 관련 함수들을 적절하게 사용해야 한다.

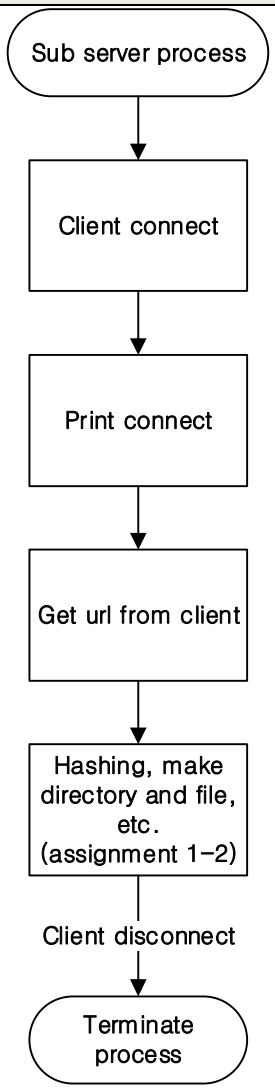
Flow chart.

Main Server process



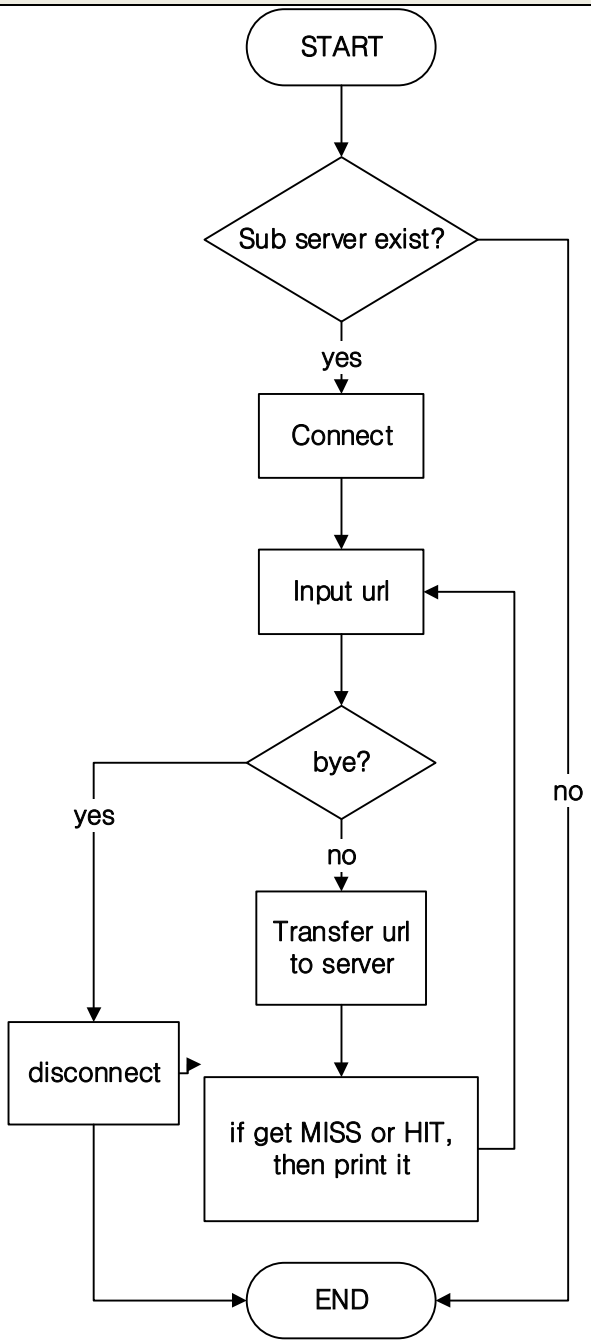
위 그림은 이번 과제의 Main server process 이다. Main server process 는 실행이 되면 Client 가 Connection request message 를 전송할 때까지 대기한다. 만약 0Connection request 가 들어오면 Sub server process 를 생성한다.

Sub Server process



위 그림은 Client 의 connection request 로 생성된 sub server process 의 flow chart 이다. Connection 이 이루어지면 Terminal 에 connection 이 되었다는 문자열을 해당 client 의 port number 와 함께 출력한다. 그 후, client 에서 url 을 입력하면 그 url 을 넘겨받아 Hashing 후 hashed url 로 directory 와 file 을 만든다. 자세한 내용은 Assignment #1-2 를 참고한다. Client 가 "bye" 를 입력해 process 를 종료할 경우 sub server process 역시 Terminal 에 disconnection 이 되었다는 문자열을 해당 client 의 port number 와 함께 출력한 후, 종료한다.

Client



위 그림은 client 의 flow chart 이다. 먼저 server 가 존재하는지 확인한다. 존재 하지않으면 연결할 수 없다는 문자열을 출력한 뒤, 바로 종료한다. Server 가 존재하면 Sub server process 와의 connection을 request 한다. 문제없이 connection이 되었으면, url 을 입력하고, 그 url 을 connection 된 sub server process 에 보낸다. 이미 입력된 url 이면 HIT 아니면 MISS 를 출력하고, 다시 입력할 수 있다. "bye"를 입력 시, sub server process 와의 disconnection request 를 보내고, 종료한다.

Pseudo code.

client.c

```
Int main(void)
{
    Socket();
    If( socket < 0)
        Printf("can't create socket) ;

    Bzero(buf) ;
    Bzero(server address) ;
    Setting sin_family, sin_addr, sin_port ;

    Connect ();
    If(connect() <0);
        Printf("can't connect");

    write("input url");

    While( buf )
    {
        If( buf == bye)
            Return 0;

        Write buf on socket;
        Printf("MISS or HIT");
        Bzero(buf);
        write("input ulr");
    }
}
```

위 그림은 client.c 의 pseudo code 이다. Socket 을 생성하고, buf 와 server adddress를 초기화 한다. Server 와의 connection을 request 한 후 connection 되었으면 url 을 입력하고 해당 url 을 server 로 보낸다. 그 후 존재하는 url 이면 HIT 아니면 MISS 를 출력한다. "bye"를 입력하면 해당 process를 종료한다.

server.c

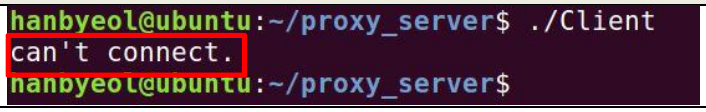
```
Int main(void)
{
    Socket();
    Bind(socket);
    Listen(socket, 5);
    Signal(signal handler);
```

```
    While(1)
    {
        Accept();
        Read(client);

        Assignment #1-2 ;

        Close(client);
    }
    Close(socket);
    Return 0;
}
```

위 그림은 server.c 의 pseudo code 이다. Socket 과 관련된 함수, socket(), bind(), listen(), signal(), accept(), read(), close() 함수를 적절히 잘 이용해야한다. Connection request 를 보낸 client 와 Accept 한 후, Assginment #1-2(url 을 hashing, hashed url로 directory 와 file 생성, MISS,HIT count, logfile에 정보 write 등)를 수행한다. Clinet 에서 "bye" 를 입력시, sub server process 역시 종료한다.

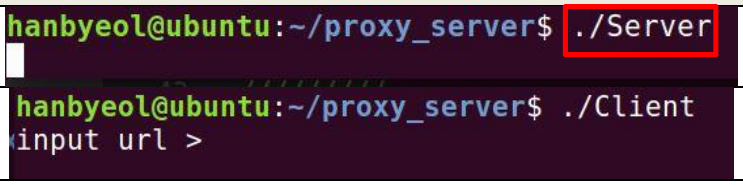
Result.

2-1.(1)

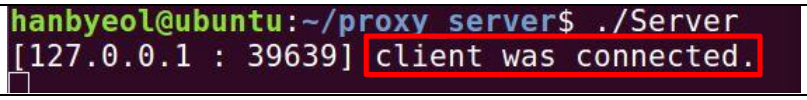


위 그림은 Server 를 실행시키지 않은 상태에서 Client 를 실행시켰을 때의 그림이다. Client는 Server 가 존재해야 그곳으로 connection request 를 보내 connection 이 이루어 진 후 작업이 가능하다. 위 그림과 같이 Server 가 존재 하지않으면 Error message 가 출력된다.

2-1.(2)



위 그림은 server 를 실행시킨 후 Client 를 실행시켰을 때의 그림이다. 2-1.(1) 과 달리 정상적으로 connection 이 이루어졌고, client 의 작업이 올바르게 이루어진 것을 확인할 수 있다.



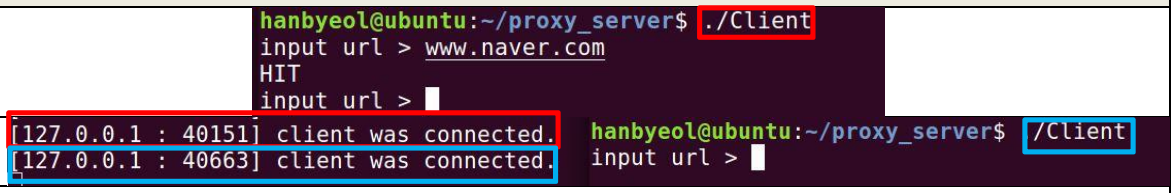
이때, Server를 실행시킨 Terminal 을 확인해보면 위와 같이 연결된 client 의 정보와 connection 되었다는 message 가 함께 출력됨을 확인할 수 있다.

그 후, Client 가 실행된 terminal 에서 url 을 입력해 해당 url 이 없으면 MISS, 있으면 HIT 가 올바르게 출력됨을 확인할 수 있다.

2-1.(3)

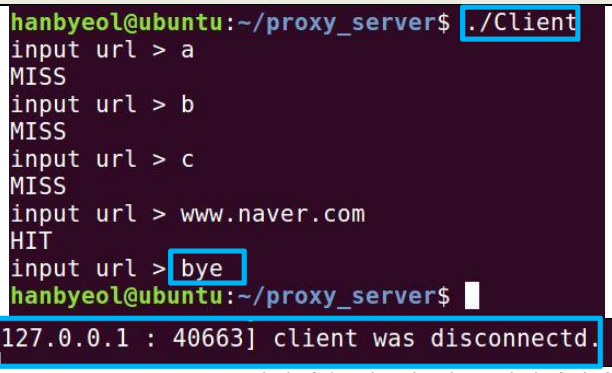
위 그림은 client 에서 bye 를 입력했을 때의 그림이다. Server 에서 해당 client 정보와 disconnection 이 되었다는 문자열을 함께 출력한다. 올바르게 출력되었음을 확인할 수 있다.

2-1.(4)



위 그림은 하나의 client 를 connection 한 후, 또 하나의 client를 connection 했을 때의 그림이다. 처음 client 를 client 1, 그 다음으로 connection 한 client를 client 2 라고 할 때, client 1 가 connection 되었다는 message 와 client 2 가 connection 되었다는 message 가 server 에 올바르게 출력됨을 확인할 수 있다. Client 1 에서 www.naver.com 을 입력한다.

2-1.(5)



그 후 client 2 에서 a, b, c, www.naver.com 입력했을 시, 한 번도 입력되지않은 a, b ,c 는 MISS 가 출력되지만 www.naver.com 는 client 1 에서 입력했으므로, HIT 가 출력되는 것을 확인할 수 있다. 그 후, bye 를 입력시켜 process 를 종료시켰다. Server terminal를 확인해보니 client 2 가 disconnection 되었음을 확인할 수 있다.

```
input url > a
HIT
input url > bye
hanbyeol@ubuntu:~/proxy_server$
[127.0.0.1 : 40151] client was disconnectd.
```

Client 1 에서도 a를 입력 시켰을 때, client 2 에서 입력했으므로, HIT 가 출력되는 것을 확인할 수 있다. 그 후 , bye 를 입력시켜 process를 종료시키고, server 를 확인하면 server와 client 가 disconnection 되었다는 것을 알 수 있다. 이 때, 어떤 client 가 종료되었는지 확인하기 어려울 수 있으므로 해당 port number 를 확인하면 쉽게 구별할 수 있다.

이러한 과정은 아래의 server terminal 에 출력된 결과 화면을 보고 쉽게 파악할 수 있다.

```
hanbyeol@ubuntu:~/proxy_server$ ./Server
[127.0.0.1 : 39639] client was connected.
[127.0.0.1 : 39639] client was disconnectd.
[127.0.0.1 : 40151] client was connected.
[127.0.0.1 : 40663] client was connected.
[127.0.0.1 : 40663] client was disconnectd.
[127.0.0.1 : 40151] client was disconnectd.
```

2-1.(6)

```
hanbyeol@ubuntu:~$ tree ~/cache
/home/hanbyeol/cache
├── 84a
│   └── 516841ba77a5b4648de2cd0dfcb30ea46dbb4
├── 86f
│   └── 7e437faa5a7fce15d1ddcb9eaeaea377667b8
├── d8b
│   └── 99f68b208b5453b391cb0c6c3d6a9824f3c3a
├── e00
│   └── 0f293fe62e97369e4b716bb3e78fababf8f90
├── e9d
│   └── 71f5ee7c92d6dc9e92ffdad17b8bd49418f98
└── fed
    └── 818da7395e30442b1dcf45c9b6669d1c0ff6b

6 directories, 6 files
```

위 그림은 2-1.(4) ~ 2-1.(5) 의 결과로 생성된 directory 와 file 들을 tree로 출력한 결과 화면이다. 올바르게 출력되었음을 확인할 수 있다.

2-1.(7)

```
hanbyeol@ubuntu:~$ cat ~/logfile/logfile.txt
[Miss] ServerPID : 2547 | www.naver.com-[2018/4/27, 22:47:26]
[Miss] ServerPID : 2547 | www.google.com-[2018/4/27, 22:47:26]
[Miss] ServerPID : 2547 | www.kw.ac.kr-[2018/4/27, 22:47:26]
[Hit] ServerPID : 2547 | fed/818da7395e30442b1dcf45c9b6669d1c0ff6b-[2018/4/27, 22:47:26]
[Hit]www.naver.com
[Terminated] ServerPID: 2547 | run time: 87 sec. #request hit: 1, miss: 3
[Miss] ServerPID : 2563 | a-[2018/4/27, 22:50:00]
[Miss] ServerPID : 2563 | b-[2018/4/27, 22:50:00]
[Miss] ServerPID : 2563 | c-[2018/4/27, 22:50:00]
[Hit] ServerPID : 2563 | fed/818da7395e30442b1dcf45c9b6669d1c0ff6b-[2018/4/27, 22:50:00]
[Hit]www.naver.com
[Terminated] ServerPID: 2563 | run time: 42 sec. #request hit: 1, miss: 3
```

위 그림은 2-1.(4) ~ 2-1.(5) 의 결과로 생성된 logfile.txt 이다. 올바르게 출력되었음을 확인할 수 있다.

Conclusion.

이번 과제의 핵심은 socket 이다. Socket 은 application layer 와 transport layer 를 오갈 수 있는 문 역할을 하는 것으로 알고 있었는데 이것을 막상 구현할 때는 쉽지않았다. Socket 과 관련된 각종 함수들인 socket(), bind(), listen(), accept(), read(), write(), connect(), close() 들을 모두 정확하게 파악하고 있어야 했고, 어떤 인터넷 프로토콜을 사용하는지 type은 무엇인지 등등 고려할 사항이 많았다. 이번 과제에선 AF_INET, 즉, IPv4 인터넷 프로토콜과, TCP/IP 프로토콜을 이용했다. 컴퓨터 네트워킹 강의를 수강하며 이론적으로 습득했던 부분을 직접 구현해보니 어려웠지만 원리를 이해하는 것에 많은 도움이 되었던 것 같다.

이번 과제에서 가장 어려웠던 부분은 read() 와 write() 를 이용해 client 와 server 가 서로 통신하는 부분인 것 같다. Read( ) 나 write( ) 의 첫 번째 인자에 STDIN, STDOUT 을 넣어주거나 socket_fd, client_fd 를 넣어 주는 것이 완전히 다른 의미라는 것을 깨닫기에는 많은 시간이 걸렸고, client 입장에서, server 입장에서 각각 read() 와 write() 가 수신, 전송이 다르다는 것을 파악하는 것도 많이 헷갈려 프로그래밍 하는데 있어 많은 어려움을 겪었다. Server 에서는 Assignment #1-2 까지 구현한 부분을 바로 쓰면 될 줄 알았지만 buf 를 이용해 수정하는 부분이 깔끔하게 되지않았다. 이 역시 read() 와 write() 용도와 쓰임새를 정확하게 파악한 후에는, 간단히 해결할 수 있는 부분이었다.

Buf 에 url 을 입력시, enter 로 입력을 마쳐 항상 buf 에는 입력한 url 과 개행 문자인 '₩n' 이 포함되어 있어 원하는 결과가 나오지 않아 어떻게 처리해야할 지 고민할 때 13학번 권영상 군의 도움을 받아 개행 문자 '₩n' 가 있는 자리에 NULL 문자 '₩0' 을 넣어주는 것으로 해결했다.

기존의 과제들은 문자열 관련 함수들을 이용해 프로그래밍 하는 것이어서 난이도도 쉽고, 흥미도가 떨어졌지만 socket programming 을 함으로서 컴퓨터 공학도로서의 실력을 한층 더 업그레이드한 느낌이 든다. 앞으로의 실습이 기대된다.