

# 시스템프로그래밍



과 제 명 : Proxy 2-1

교 수 님 : 최상호 교수님

강의시간 : 월요일

학    번 : 2019202100

이    름 : 하주영

# Introduction

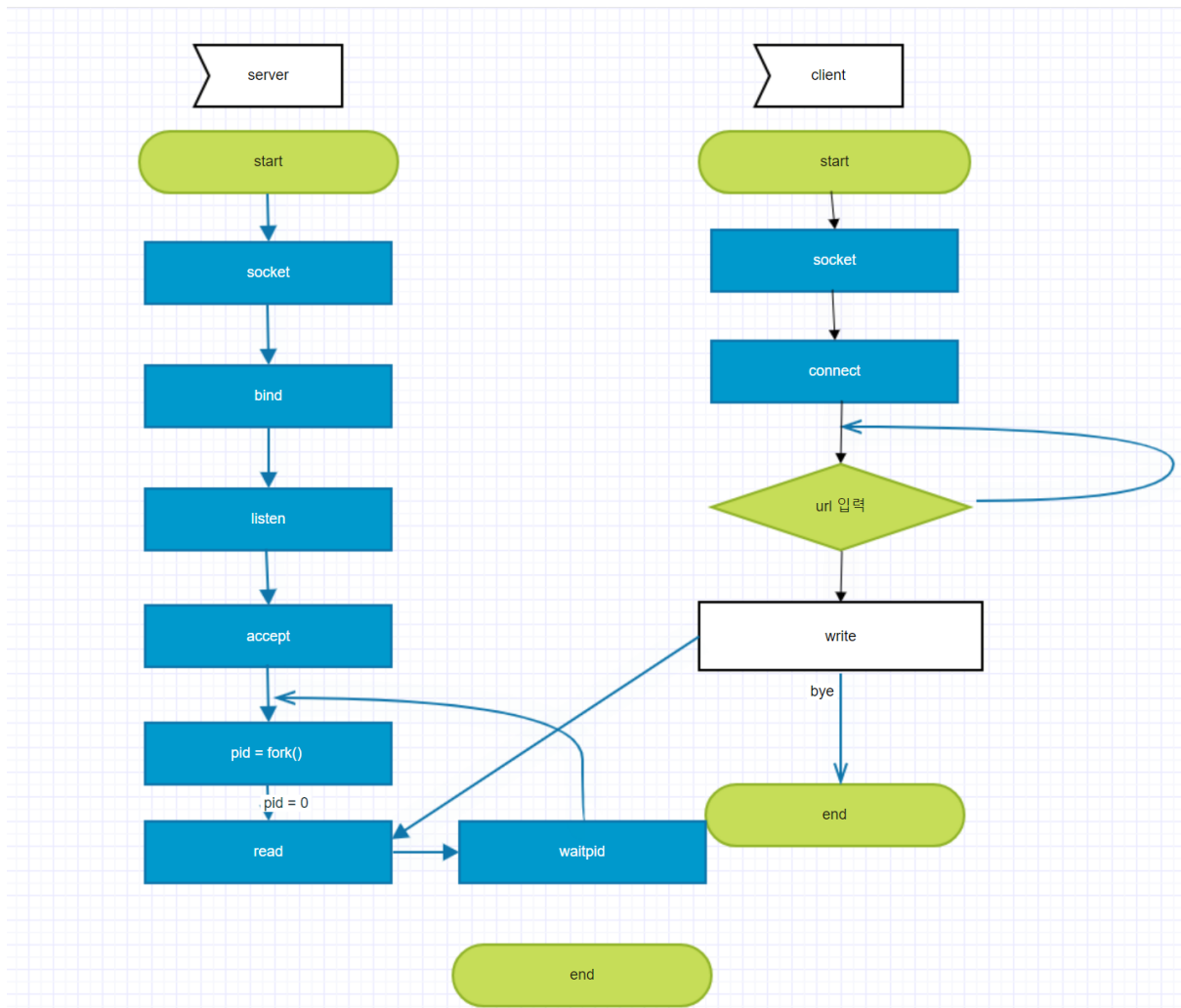
Client와 server를 연결하는 socket API를 사용하여 전체 과제를 수행한다.

다수의 client를 처리하는 server를 만든다. client에서 url을 입력하고 server로 url을 전송한다.

Client가 연결되면 sub server process로 url을 전송한다. Server는 입력한 url이 cache에 저장되어 있는지 확인하고 Hit or Miss 결과를 client로 전송한다.

sslslab@ubuntu:~\$ ./Server	sslslab@ubuntu:~\$ ./Client	sslslab@ubuntu:~\$ ./Client
[127.0.0.1 : 16094] client was connected	input url > www.kw.ac.kr	input url > www.naver.com
[127.0.0.1 : 16606] client was connected	MISS	MISS
[127.0.0.1 : 16606] client was disconnected	input url > www.google.com	input url > www.kw.ac.kr
[127.0.0.1 : 16094] client was disconnected	MISS	HIT
^C	input url > www.naver.com	input url > bye
sslslab@ubuntu:~\$	HIT	sslslab@ubuntu:~\$
	input url > bye	
	sslslab@ubuntu:~\$	
< Server >	< Client [127.0.0.1:16094] >	< Client [127.0.0.1:16606] >

## Flow Chart



## Pseudo Code

```
/*server.c*/
```

```
Int main() {  
    struct sockaddr_in server_addr, client_addr;  
    int socket_fd, client_fd;  
    socket_fd = socket(PF_INET, SOCK_STREAM, 0);  
    socket 주소 설정  
    bind(socket_fd, struct sockaddr *)&server_addr, sizeof(server_addr));  
    listen(socketfd, 5);  
    while(1) {  
        client_fd = accept(socket_fd, (struct sockaddr*)&client_addr, &len);  
        printf("client was connected\n");  
        if((pid == fork()) == 0) /*child process*/ {  
            assignment #2-1  
            while ((read(client_fd, buf, BUFSIZE))>0) {  
                write(client_fd, "Hit or False", 3 or 4letters);  
            }  
            Printf("cliens was disconnected\n");  
            Close(client_fd);  
            Return 0;  
        }  
        Close(client_fd);  
    }  
    Close(socket_fd);  
}
```

```
/*client.c*/
```

```
Int main() {  
    Int socket_fd;  
    Struct sockaddr_in server_addr;  
    Char haddr[] = "127.0.0.1";
```

```
Socket_fd = socket(PF_INET, SOCK_STREAM, 0)

Server_addr 주소 설정

Connect(socket_fd, (struct sockaddr*)&server_addr, sizeof(server_addr));

url 입력

write(socket_fd, url, strlen(url);

read(socket_fd, "hit or miss", size);

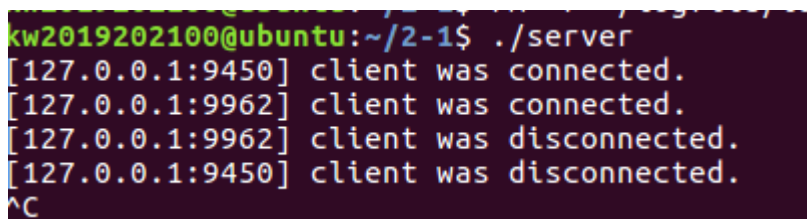
화면에 HIT or MISS 출력

close(socket_fd);

}
```

## Result Screen

### 1. Server 실행

A terminal window with a dark background and light-colored text. The prompt is 'kw2019202100@ubuntu:~/2-1\$'. The user has entered './server'. The output shows four lines of status messages: '[127.0.0.1:9450] client was connected.', '[127.0.0.1:9962] client was connected.', '[127.0.0.1:9962] client was disconnected.', and '[127.0.0.1:9450] client was disconnected.'. The cursor is on the line following the last message, and there is a '^C' character at the bottom left of the terminal area.

```
kw2019202100@ubuntu:~/2-1$ ./server
[127.0.0.1:9450] client was connected.
[127.0.0.1:9962] client was connected.
[127.0.0.1:9962] client was disconnected.
[127.0.0.1:9450] client was disconnected.
^C
```

### 2. Client 실행 (2개)

```
kw2019202100@ubuntu:~/2-1$ ./client
input url > www.naver.com
HIT
input url > bye
kw2019202100@ubuntu:~/2-1$ █

kw2019202100@ubuntu:~/2-1$ ./client
input url > www.naver.com
MISS
input url > www.google.com
MISS
input url > www.google.com
HIT
input url > bye
kw2019202100@ubuntu:~/2-1$ █
```

### 3. logfile

```
kw2019202100@ubuntu:~$ cat ~/logfile/logfile.txt
[Miss] ServerPID : 4260 | www.naver.com-[2022/04/27, 05:12:00]
[Miss] ServerPID : 4260 | www.google.com-[2022/04/27, 05:12:03]
[Hit] ServerPID : 4260 | d8b/99f68b208b5453b391cb0c6c3d6a9824f3c3a -[2022/04/27, 05:12:05]
[Hit]www.google.com
[Terminated] ServerPID : 4260 | run time: 9 sec. #request hit : 1, miss : 2
[Hit] ServerPID : 4257 | fed/818da7395e30442b1dcf45c9b6669d1c0ff6b -[2022/04/27, 05:12:10]
[Hit]www.naver.com
[Terminated] ServerPID : 4257 | run time: 24 sec. #request hit : 1, miss : 0
```

## Consideration

과제를 진행하면서 buf에 문자열1에서 문자열2로 바꿨음에도 불구하고 문자열2의 길이가 더 짧을 경우 뒤에 문자열1의 문자 몇 개가 남아있는 이슈가 있었다.

Ex) 첫번째 주소 입력 : [www.naver.com](http://www.naver.com)

두번째 주소 입력 : [www.kw.ac.kr](http://www.kw.ac.kr)

->주소를 입력받는 변수를 buf라고 했을 때 두번째 주소를 입력받은 후 buf를 출력하면

: [www.kw.ac.krm](http://www.kw.ac.krm)

이유는 buf를 0으로 초기화하지 않아서 이며 bzero(buf, sizeof(buf))로 초기화를 해준다음 주소를 저장했을 때는 올바르게 출력되었다.

bzero의 쓰임과 왜 사용하는 지에 대해 알게 되었다.

## Conclusion

Server와 client를 연결하는 socket API에 대해 알게 되었다. server에서는 socket, bind, listen, accept로 client 접속을 대기하고 client는 socket, connect로 연결요청을 한다. url을 치면 url을 server로 보내 #1-2에서 구현한 cache에서 있는 url인지를 검사하여 hit/miss를 판별한다. Hit or miss 정보를 client에 전송하는 것이 이번과제의 목표이다. 문자열을 저장하는 buf를 초기화하는 방법으로 memset도 있지만 bzero에 대해서도 알게 되었다.

## Reference

[C에서 문자 배열 지우기 | Delft Stack](#)