

**시스템 프로그래밍(화요일)**

**최상호 교수님**

**Proxy 2-4**

**컴퓨터정보공학부**

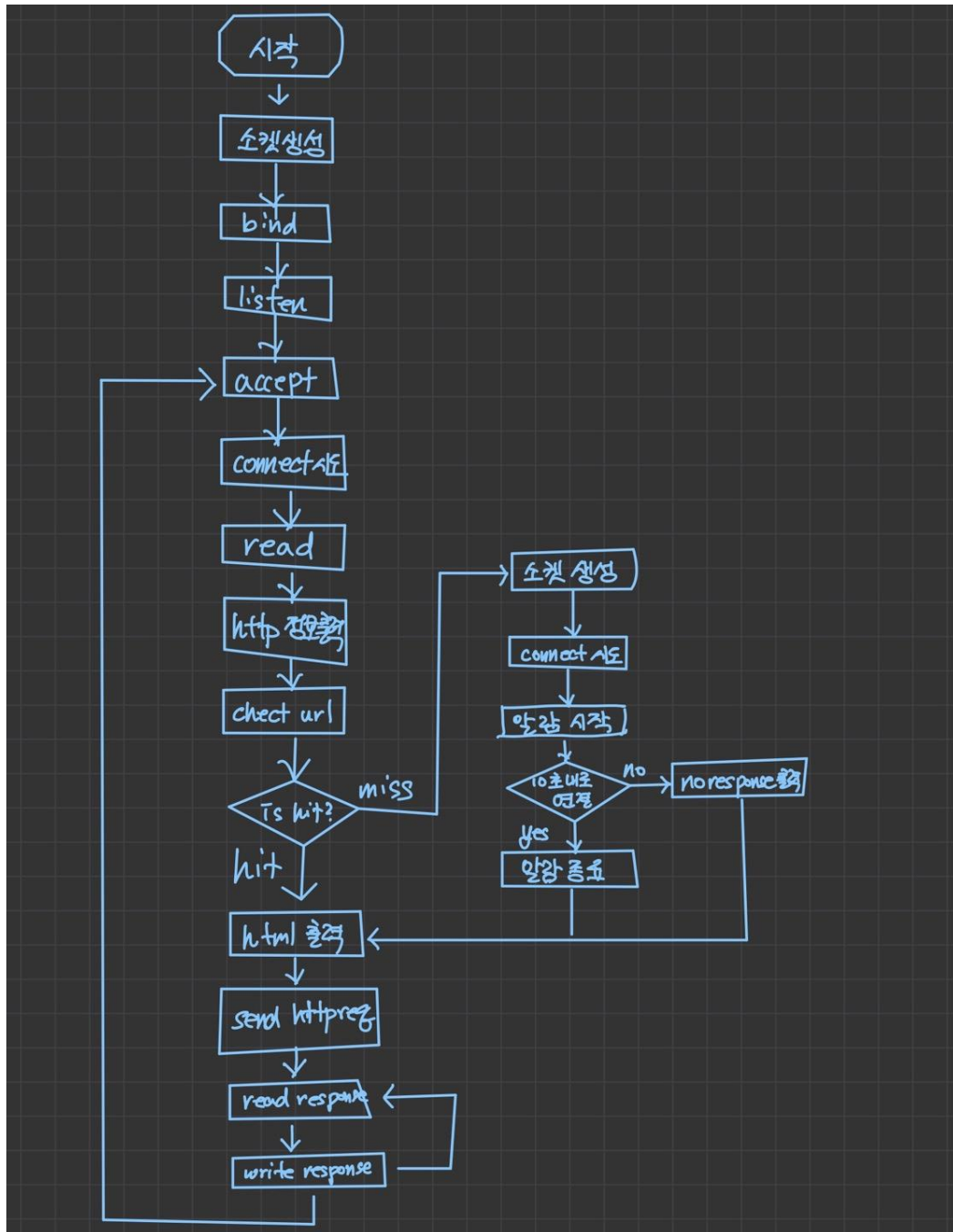
**2018202076**

**이연걸**

- Introduction

이번 과제는 HTTP response를 받아 그것을 사용자의 화면에 띄워주어야 하는 과제이다. 이를 위해서는 우선 request를 보내는 과정이 필요한데 send함수의 이해가 필수적으로 필요했다. Send를 사용해 req를 보내고 recv를 사용한다. 이때 read함수를 사용하여도 무방하지만 socket을 사용할 때는 recv가 더 일반적인 방식이라고 한다. recv함수를 사용해 socket fd의 값을 읽고 나면 화면에 response를 띄워줄 준비가 끝난 것이다. 이제 write함수를 사용해 값을 보내주면 된다. 이 밖에도 logfile 그리고 cache파일 수정, SIGINT를 사용해 ctrl + c를 통한 강제 종료시에 활용되는 signal함수를 추가하는 과제가 있었지만 핵심은 서버, 클라이언트, 프록시 간의 연결을 할 때 socket를 잘 활용할 수 있는가 이다.

- Flow chart



- Pseudo Code

프로그램 시작

소켓 연결

Bind()

Listen()

While

Accept()

connect시도

read()

http정보 출력

소켓 연결

connect시도

url 확인

로그, 캐시 기록

if hit then continue

else

알람 10초 시작

If 10초내로 연결 then 알람 종료

Else

Alarm handler 호출

End if

Send http request

While (모든 response를 읽을 때 까지)

읽은 response write로 소켓에 전달

End while

End while

프로그램 종료

- 결과화면

BSAI Lab - Mozilla Firefox


System Programming

sslslab.practice.s3-website.ap-northeast-2.amazonaws.com/bsai.html

Connection Is Not Secure

Permissions  
You have not granted this site any special permissions.

# Biomedical Intelligence Laboratory



**Professor**

**Sang Ho Choi**

School of Computer and Information Engineering,  
College of Software and Convergence,  
Kwangwoon University  
#705, Saebit building  
shchoi@kw.ac.kr +2-940-8450

## Research Area

Healthcare System  
Artificial Intelligence  
Biomedical Engineering

## Ongoing Projects

Development of an unrestrained patient monitoring system using wireless power transmission  
Development of artificial intelligence-based vital signal and sleep monitoring technology using UWB radar

**Lab Location : #713, Saebit Building**

System Programming - Mozilla Firefox

BSAI Lab

System Programming

sslslab.practice.s3-website.ap-northeast-2.amazonaws.com

# System Programming

## Professor

- Sangho Choi
- Lab : BSAI
- Office : Saebit 705
- shchoi@kw.ac.kr

## System Programming TA List

| Name       | Course | Lab   | e-mail               |
|------------|--------|-------|----------------------|
| Hyejin Cha | Master | SSLAB | hj.cha@kw.ac.kr      |
| Jaewon An  | Master | BSAI  | jaewonan95@gmail.com |

## System Programming Lab. TA List

| Name        | Course | Lab   | e-mail               | Lecture Time            |
|-------------|--------|-------|----------------------|-------------------------|
| Hyejin Cha  | Master | SSLAB | hj.cha@kw.ac.kr      | Friday 1,2              |
| Taehyun Eom | Ph.D   | CINe. | crackscendo@kw.ac.kr | Friday 1,2 & Friday 5,6 |
| Dongju Kim  | Master | CINe. | gggg8657@gmail.com   | Friday 5,6              |
| Hyunjin Kim | Master | CINe. | zx8635@naver.com     | Thursday 7,8            |
| Ubin Jin    | Master | CINe. | ubinjin2@naver.com   | Thursday 7,8            |

HTTP RESPONSE와 http경고화면 확인

```
182 void int_handler(void)
183 {
184     printf("\n\nTEST\n\n");
185     exit(0);
186 }
187
188 char *get_ip_handler(char *addr)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - Proxy 2-4 + ~ [ ] [ ] ^ X

kw2018202076@ubuntu:~/workspace/KW-Univ/시스템 프로그래밍 /Proxy 2-4$ make
make: Nothing to be done for 'all'.
kw2018202076@ubuntu:~/workspace/KW-Univ/시스템 프로그래밍 /Proxy 2-4$ ./proxy_cache
^C

TEST

kw2018202076@ubuntu:~/workspace/KW-Univ/시스템 프로그래밍 /Proxy 2-4$
```

## SIGINT 동작 확인

```
kw2018202076@ubuntu: ~/cache/02f

kw2018202076@ubuntu:~$ ls
cache Desktop Downloads logfile Pictures Templates workspace
core Documents examples.desktop Music Public Videos
kw2018202076@ubuntu:~$ cd cache/
kw2018202076@ubuntu:~/cache$ ls
02f 5d1
kw2018202076@ubuntu:~/cache$ ls -R
.:
02f 5d1

./02f:
e842cce0e046569f08277b585f871b641dff4

./5d1:
87e27aef62ef20e930d70e613f77c22731d2d
kw2018202076@ubuntu:~/cache$ cd e
bash: cd: e: No such file or directory
kw2018202076@ubuntu:~/cache$ cd
02f/ 5d1/
kw2018202076@ubuntu:~/cache$ cd
02f/ 5d1/
kw2018202076@ubuntu:~/cache$ cd 02f/
kw2018202076@ubuntu:~/cache/02f$ ls
e842cce0e046569f08277b585f871b641dff4
kw2018202076@ubuntu:~/cache/02f$ cat e842cce0e046569f08277b585f871b641dff4
GET http://sslslab.practice.s3-website.ap-northeast-2.amazonaws.com/bsai.html HTTP/1.1
Host: sslslab.practice.s3-website.ap-northeast-2.amazonaws.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefox/61.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Sat, 30 Apr 2022 03:30:05 GMT
If-None-Match: "901fc08f1e2a7e2e79b1711dafd2c1dc"
Cache-Control: max-age=0

kw2018202076@ubuntu:~/cache/02f$
```

## Cache 파일에 HTTP request 정보 저장 확인

```
kw2018202076@ubuntu:~$ ls
cache Desktop Downloads logfile Pictures Templates workspace
core Documents examples.desktop Music Public Videos
kw2018202076@ubuntu:~$ cd logfile/
kw2018202076@ubuntu:~/logfile$ ls
logfile.txt
kw2018202076@ubuntu:~/logfile$ cat logfile.txt
[MISS] http://sslslab.practice.s3-website.ap-northeast-2.amazonaws.com/bsai.html - [2022/5/25, 7:16:51]
[MISS] http://sslslab.practice.s3-website.ap-northeast-2.amazonaws.com/ - [2022/5/25, 7:17:6]
[Hit] 02f/ e842cce0e046569f08277b585f871b641dff4 - [2022/5/25, 7:17:14]
[Hit] http://sslslab.practice.s3-website.ap-northeast-2.amazonaws.com/bsai.html
[Hit] 5d1/ 87e27aef62ef20e930d70e613f77c22731d2d - [2022/5/25, 7:17:30]
[Hit] http://sslslab.practice.s3-website.ap-northeast-2.amazonaws.com/
[MISS] http://sslslab.practice.s3-website.ap-northeast-2.amazonaws.com/img/bsai_shchoi.jpg - [2022/5/25, 7:17:53]
[Hit] 2a3/ d2ac4bdeb4453ec13f2f1dc5e67afc8f66de8 - [2022/5/25, 7:18:16]
[Hit] http://sslslab.practice.s3-website.ap-northeast-2.amazonaws.com/img/bsai_shchoi.jpg
kw2018202076@ubuntu:~/logfile$
```

## Log 파일 저장 내용 확인

- 고찰

과제를 수행하면서 우여곡절이 굉장히 많았다. 우선 소켓에 정보를 집어넣는 부분에서 막혔다. 2-3에서 수행했던 것처럼 `write()` 함수를 호출하면 간단히 해결할 수 있었지만 새로운 소켓이 필요한 것을 눈치채지 못했다. 새로운 소켓을 만들어서 웹서버와 연결해야 하는데 프록시가 웹으로부터 request를 받으면 fork를하고, child process에서 새로운 소켓인 `sock_fd`를 만들어서 HOST의 주소, HTTP 포트번호 80을 넣어주어야 했다. 그 이유는 프록시에서 웹서버와 통신하기 때문이다. 결론을 말하자면 기존에 만들어 두었던 `client_fd`(포트번호 39999)가 아닌 `sock_fd`(포트번호 80)를 만들어 `send` 함수에서 본 소켓을 사용해 요청을 보내고 본 소켓으로 응답을 받아야한다. 이렇게 웹서버와 통신하는 소켓을 사용하지 않아서 문제가 있었다.

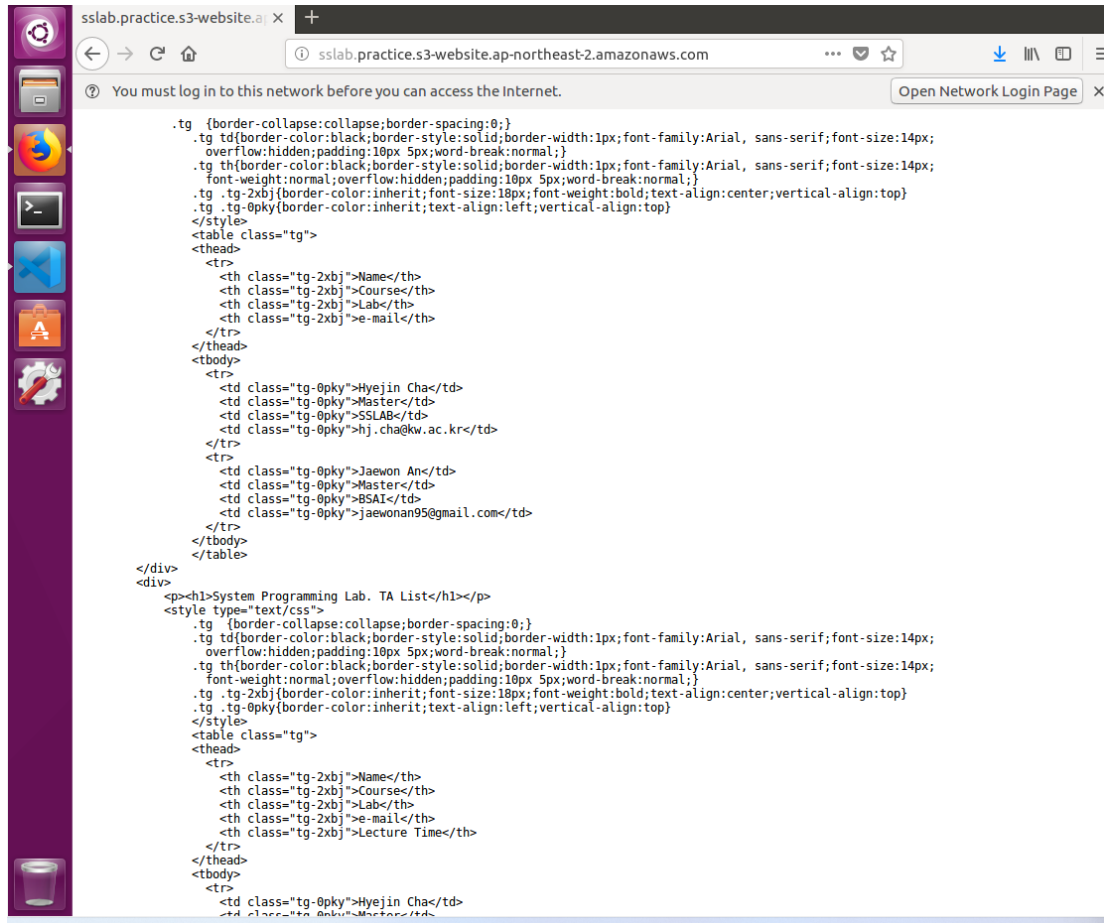
다른 하나는 `write()` 함수에서도 `sock_fd`를 사용한 것이다. `Write()` 함수는 특정 크기의 데이터를 file descriptor에 작성하는 기능을 하는데 이 file descriptor에 필요한 값은 `client_fd`이다. Request와 response를 보내고 받는 것은 프록시와 웹서버이지만 그 response를 받아야 할 대상은 client이기 때문에 `client_fd`를 사용해야 했다.

마지막은 html대로 꾸며진 웹페이지가 아닌 raw html이 나왔던 경우이다. 이 오류는 `read`와 `write`가 순차적으로 이루어지지 않았기 때문에 발생한다. 처음에 코드를 다음과 같이 작성했었다.

```
send(sock_fd, http_req, strlen(http_req), 0);
len = recv(sock_fd, tmp, BUFF_SIZE - 1, 0);

while ((len = recv(sock_fd, tmp, BUFF_SIZE - 1, 0)) > 0) {
    tmp[len] = '\0';
    write(cli_fd, tmp, len);
}
```

Html은 정해진 형식이 있는데 `<html><body> </body></html>` 이런 식으로 태그안에 다른 태그를 품고 있고 만약 해당 태그의 종료를 알리는 `</>`가 나오지 않는다면 오류가 생긴 것이며 그 때 컴퓨터는 해당 파일을 html이 아닌 텍스트로 인식해 raw html이 나오게 되었다. 오류 내용은 다음과 같다.



다행히 html을 공부한 적이 있어서 금방 해결할 수 있었다.