

2019년 1학기 시스템프로그래밍실습 9주차

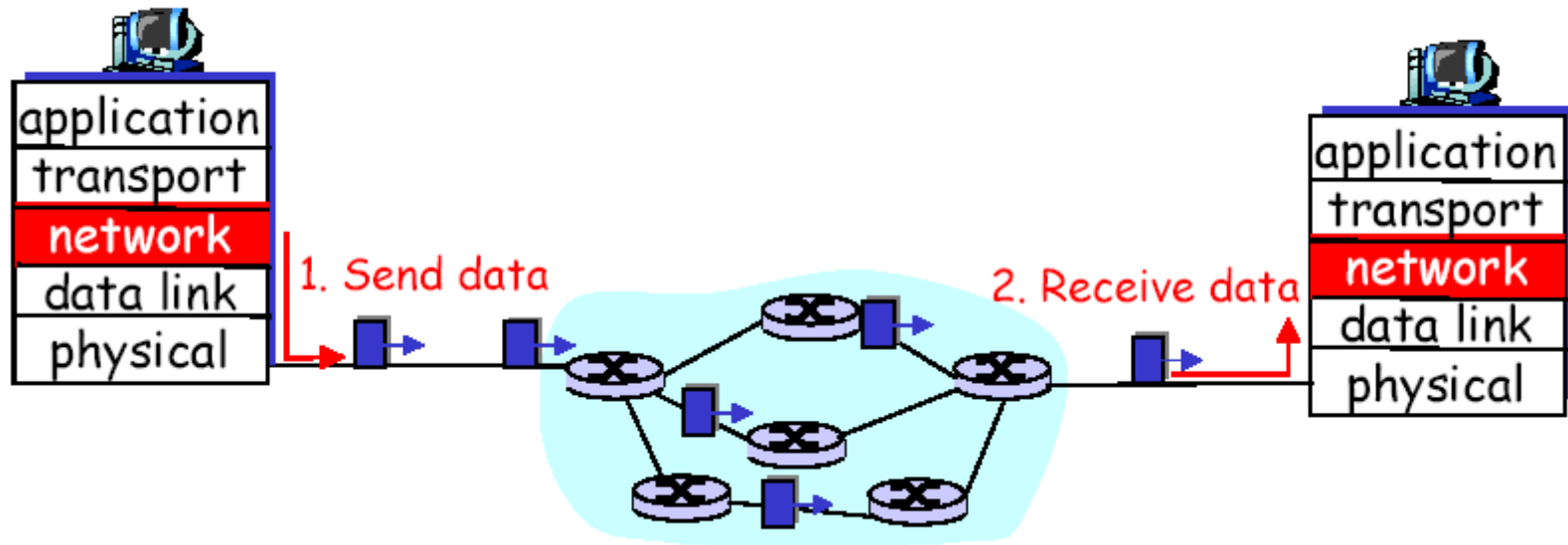
Basic Web Server

System Software Laboratory
College of Software and Convergence
Kwangwoon Univ.

Protocol

- **Protocol**

- An agreement between the communicating parties on how communication is to proceed
- List of protocols used by a particular machine is called the protocol stack

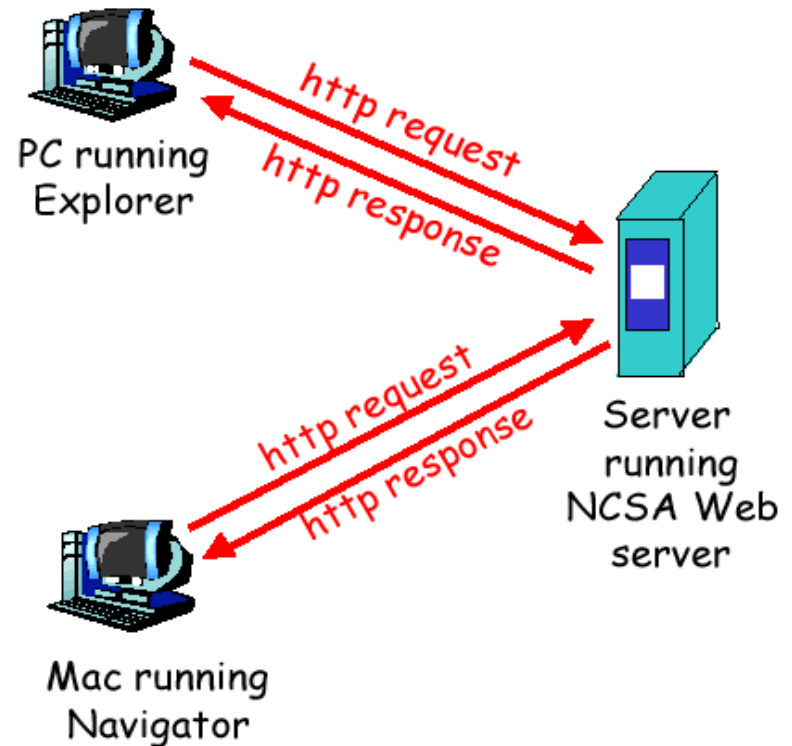


HTTP Protocol

- **Hypertext Transfer Protocol**
- **The standard Web transfer protocol**
- **WWW's(World Wide Web) application layer protocol**
- **Using on-demand method**
- **Client/Server model**
 - Client
 - Browser that requests, receives, display WWW objects
 - Server
 - WWW server sends objects in response to request

HTTP Protocol (cont'd)

- Client requests to HTTP using 80 port in Web Server
- Server is accepting TCP connection
- Exchange HTTP message between web browser and web server
- TCP connection termination



The format of an HTTP message

- Two types of HTTP message
 - Request, response
- The format of an HTTP request

```
GET / HTTP/1.1
```

```
Accept: text/html, application/xhtml+xml, */*
```

```
Accept-Language: ko-KR
```

```
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64;  
Trident/7.0; rv:11.0) like Gecko
```

```
Accept-Encoding: gzip, deflate
```

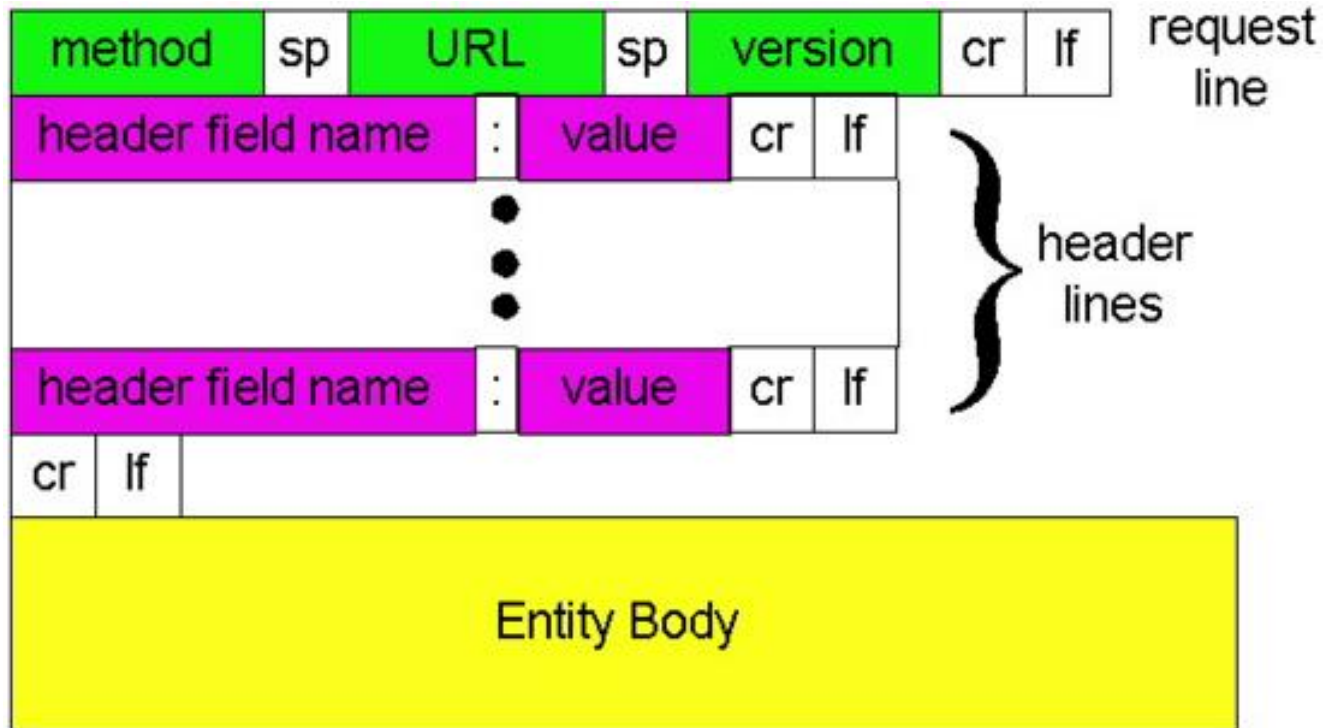
```
Host: sswlab.kw.ac.kr
```

```
Connection: Keep-Alive
```

```
(blank)
```

```
(entity body - POST method의 인자값을 포함)
```

HTTP request format



HTTP request format (cont'd)

- **Method field**
 - Browser가 서버로 데이터를 전달하는 방법 (GET, POST, HEAD 등)
- **URL field**
 - Client(browser)가 request한 URL information
- **Accept field**
 - Client(browser)가 실행할 수 있는 application format
- **Accept-Language field**
 - Language
- **User-Agent field**
 - Client의 operation system과 browser의 정보
- **Host field**
 - Request를 요청 받은 host의 URL

HTTP Response

- The format of an HTTP response

```
HTTP/1.1 200 OK
Date: Thu, 30 Oct 2003 17:50:32 GMT
Server: Apache/1.3.19 (Unix) PHP/4.0.6
Last-Modified: Thu, 02 Oct 2003 04:55:29 GMT
ETag: "5b042-2957-3f7bafc1"
Accept-Ranges: bytes
Content-Length: 10583
Connection: close
Content-Type: text/html
```

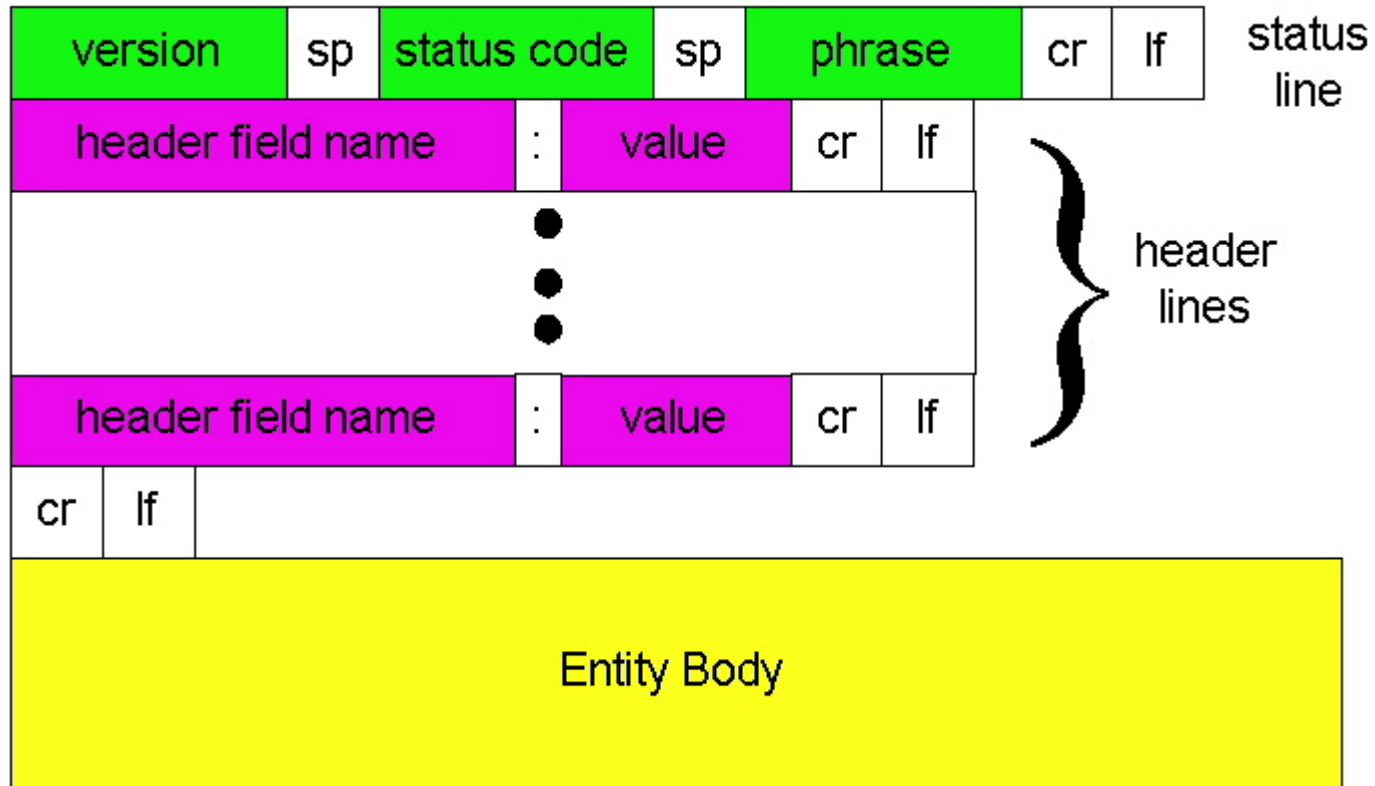
HTTP Response
Header

Html Data

```
<html>
<head>
<title>System Software Laboratory</title>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
<link rel="stylesheet" href="form.css">
```


HTTP Response (cont'd)

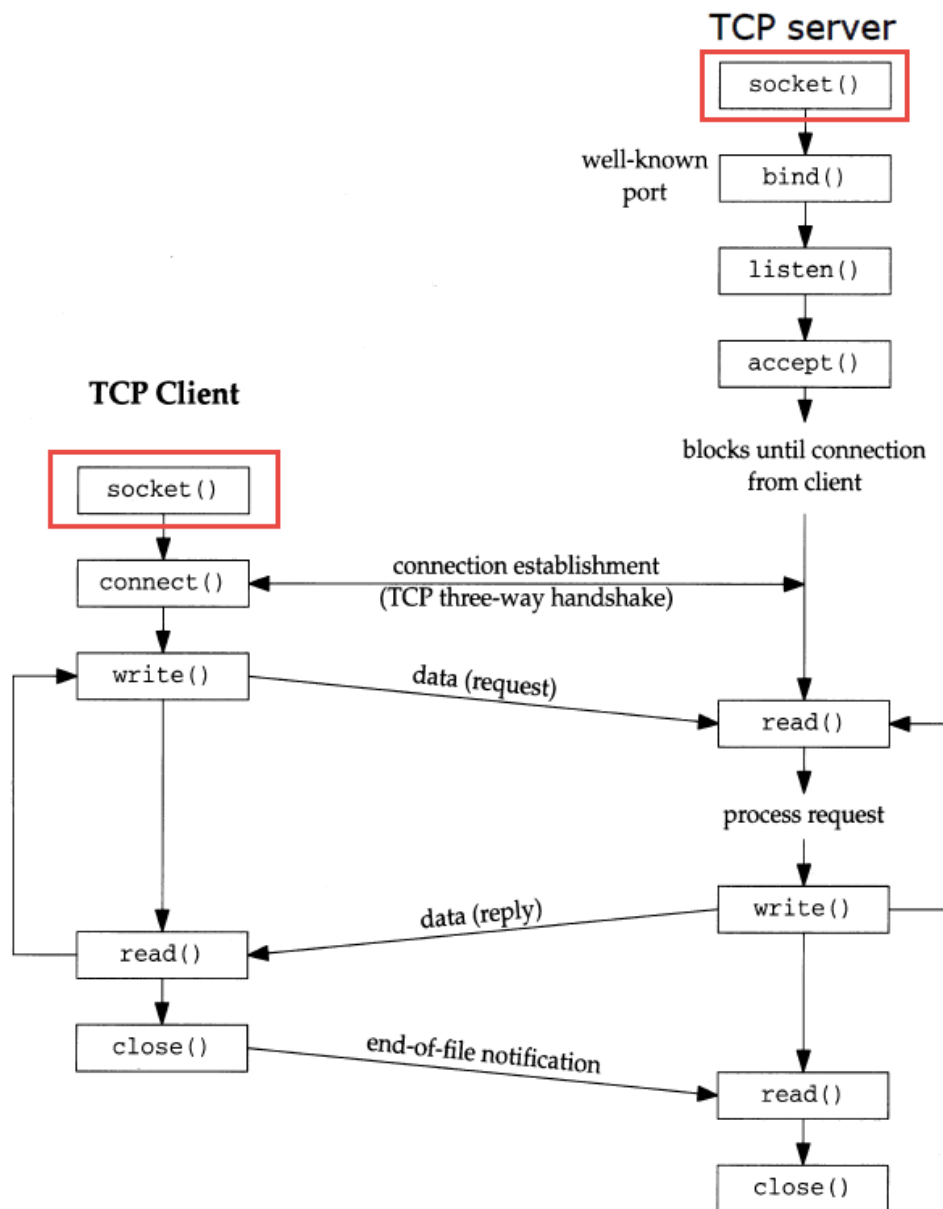
- The format of an HTTP response



HTTP Response (cont'd)

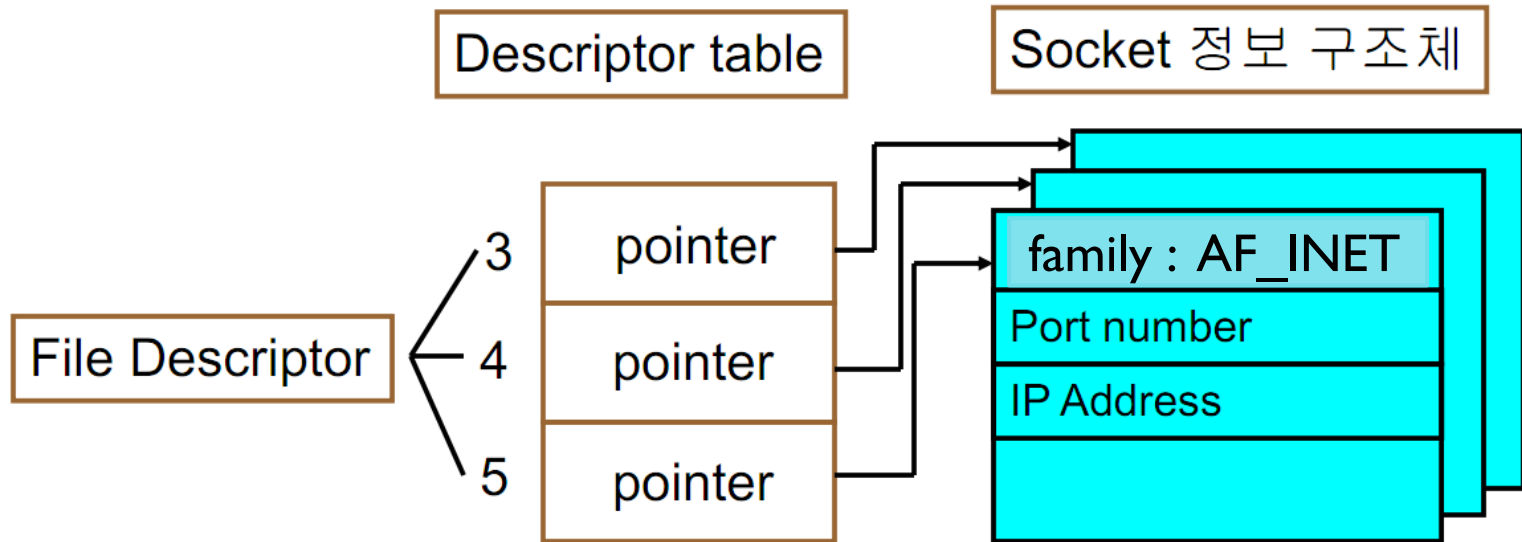
- **HTTP/1.1 200 OK**
 - HTTP version, Server response code
- **Date field**
 - Date, time
- **Last-modified field**
 - Last modified date of html page
- **Content-Length field**
 - Data size
- **Content-Type field**
 - Data format

Socket Programming



Socket

- 소켓은 통신 객체로 네트워크 상에서 데이터 교환을 가능하게 해줌
- API using TCP/IP
- **Socket number**
 - Socket 개설 시 return 되는 값



socket()

```
#include <sys/socket.h>
```

```
int socket(int domain, int type, int protocol);
```

- Connection을 위한 시스템의 socket을 설정
- Return values
 - 정상 동작 시, nonnegative descriptor
 - 에러 발생 시, -1
- Parameters
 - domain: protocol 형태 (Protocol Family)
 - e.g. PF_INET : IPv4 인터넷 프로토콜 체계
 - AF_INET과 동일하나 protocol family에서는 PF_INET을 권장
 - type: service type
 - Ex) SOCK_STREAM : TCP 방식
 - Ex) SOCK_DGRAM : UDP 방식
 - protocol: 프로토콜
 - e.g. 0 : type 설정 값에 따라 자동으로 프로토콜 선택

bind()

```
#include <sys/socket.h>
```

```
int bind(int sockfd, const struct sockaddr * soaddr, socklen_t addrlen);
```

- 소켓과 sockaddr_in 구조체 정보를 binding한다
- 즉, 생성한 소켓을 사용할 수 있도록 ip, 포트 번호를 할당
- Return values
 - 정상 동작 시, 0
 - 에러 발생 시, -1
- Parameters
 - sockfd : socket() 호출 시 리턴값 (소켓번호)
 - soaddr : sockaddr struct (서버 자신의 소켓 주소 구조체 포인터)
 - addrlen : sockaddr struct의 size (* soaddr 구조체의 크기)

struct sockaddr_in

- **소켓 주소의 구조체**

- Connection 설정 시 필요한 socket 정보로 구성

```
struct sockaddr_in
{
    short sin_family;
    u_short sin_port;
    struct in_addr sin_addr;
    char sin_zero[8];
}
```

- **sin_family**

- 주소체계

- Ex) AF_INET : IPv4 인터넷 프로토콜 체계
 - PF_INET과 동일하나 address family에서는 AF_INET을 권장

struct sockaddr_in (cont'd)

- **sin_port**

- 16비트 포트 번호

- **sin_addr**

- 32비트 IP 주소

- **sin_zero[8]**

- struct sockaddr 과의 호환성을 위한 dummy
- 실제 사용되지는 않으며 sockaddr과 사이즈를 맞추기 위함

```
struct sockaddr
{
    sa_family_t sa_family;
    char sa_data[14];
}

typedef unsigned short int sa_family_t;
```


listen()

```
#include <sys/socket.h>

int listen(int sockfd, int backlog);
```

- Server에서만 사용하며, client로 부터 connection을 위해 대기상태(listen)로 변경
- 연결요청이 들어오면 대기 queue에 들어감
- Return values
 - 정상 동작 시, 0
 - 에러 발생 시, -1
- Parameters
 - sockfd : socket() 호출 시 리턴값 (소켓 번호)
 - backlog : 대기열 queue의 길이 (연결을 기다리는 클라이언트의 최대수)

accept()

```
#include <sys/socket.h>
```

```
int accept(int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen);
```

- 서버가 client의 접속을 받아들인다
- listen() 함수에 의해 대기 queue에 있는 연결요청들을 순서대로 수락
- Return values
 - 정상 동작 시, nonnegative descriptor
 - 에러 발생 시, -1
- Parameters
 - sockfd : socket() 호출시 리턴값 (소켓번호)
 - cliaddr : sockaddr struct (연결요청을 한 상대방의 소켓주소 구조체)
 - addrlen : sockaddr struct의 size (addr 구조체 크기의 포인터)

connect()

```
#include <sys/socket.h>
```

```
int connect(int sockfd, const struct sockaddr *servaddr, socklen_t addrlen);
```

- Client가 server와의 connection을 설정하기 위해 사용
- Return values
 - 정상 동작 시, nonnegative descriptor
 - 에러 발생 시, -1
- Parameters
 - sockfd : socket() 호출 시 리턴값 (서버와 연결시킬 소켓번호)
 - servaddr : sockaddr struct (상대방 서버의 소켓주소 구조체)
 - addrlen : sockaddr struct의 size (구조체 *servaddr의 크기)

close()

```
#include <unistd.h>

int close(int filedес)
```

- File descriptor를 close
- Return values
 - 정상 동작 시, 0
 - 에러 발생 시, -1
- Parameters
 - filedес : File descriptor

write()

```
#include <unistd.h>
```

```
ssize_t write(int fildes, const void * buf, size_t nbytes);
```

- 파일에 데이터를 출력(전송)하는 함수 (system call)
- Return values
 - 성공 시, 전달 한 바이트 수
 - 에러 발생 시, -1
- Parameters
 - fildes : 데이터 전송 영역을 나타내는 파일 디스크립터
 - buf : 전송할 데이터를 가지고 있는 버퍼의 포인터
 - nbytes : 전송할 데이터의 바이트 수

read()

```
#include <unistd.h>
```

```
ssize_t read(int fildes, void *buf, size_t nbytes);
```

- Read 함수는 데이터를 입력(수신)받는 함수 (system call)
- Return values
 - 성공 시, 수신 한 바이트 수(단, EOF 만나면 0)
 - 에러 발생 시, -1
- Parameters
 - fildes : 데이터를 전송해 주는 대상을 가리키는 파일 디스크립터
 - buf : 수신 한 데이터를 저장할 버퍼를 가리키는 포인터
 - nbytes : 수신할 최대 바이트 수
- ※ close(), write(), read() 함수는 소켓 통신에만 한정되어 사용되지 않음

htonl(), htons(), ntohl(), ntohs()

```
#include <netinet/in.h>
```

```
unsigned long int htonl(unsigned long int hostlong);
```

```
unsigned short int htons(unsigned short int hostshort);
```

```
Unsigned long int ntohl(unsigned long int netlong);
```

```
unsigned short int ntohs(unsigned short int netshort);
```

- CPU마다 호스트 바이트 순서가 다르기 때문에, 네트워크를 통한 데이터 전송 시 네트워크 바이트 순서(big endian)로 통일
- 호스트와 네트워크 간 바이트 순서를 바꿈
 - htonl : long data(IP addr.) to network byte order
 - htons : short data(port no.) to network byte order
 - ntohl : long data(IP addr.) to host byte order
 - ntohs : short data(port no.) to host byte order
- i80x86에서,
 - 호스트 바이트 순서 – 하위 바이트가 앞 (little-endian) Ex) 192.168.0.1 저장 → 1.0.168.192
 - 네트워크 바이트 순서 – 상위 바이트가 앞 (big-endian) Ex) 192.168.0.1 저장 → 192.168.0.1

inet_addr()

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

unsigned long int inet_addr(const char *cp);
```

- Dotted decimal 형태인 인터넷 호스트 주소를 네트워크 바이트 순서인 이진 데이터(32bit big-endian)로 바꿈
 - e.g. 192.168.1.102 → 6601a8c0
- Return values
 - 정상 동작 시, 32bit 이진 데이터 반환
 - 입력이 올바르지 않을 경우, INADDR_NONE (usually -1) 반환
- Parameter
 - cp : Dotted decimal notation

setsockopt() Function

```
#include <sys/types.h>
#include <sys/socket.h>

int setsockopt
    (int s, int level, int optname, const void* optval, socklen_t optlen);
```

- **소켓의 옵션을 설정**
- Return values
 - On success, zero is returned. On error, -1 is returned
- Parameters
 - s : socket fd
 - level : 프로토콜의 단계, 소켓-레벨은 SOL_SOCKET 사용
 - optname : option name
 - SO_KEEPALIVE : 주기적인 전송에서 접속 유지
 - SO_REUSEADDR : 포트가 busy상태일 지라도 그것을 계속해서 사용
 - optval, optlen : 옵션 값, optval 길이
 - 대부분 소켓-라벨 옵션은 integer사용

setsockopt() Function (cont'd)

- bind() error
 - 프로그램을 종료 후에 다시 실행하면 bind()에서 error가 생기는 현상
 - TIME_WAIT state
- setsockopt()를 이용해 bind()에 의해 생기는 TIME_WAIT 현상을 아래의 코드로 막을 수 있음

```
int opt = 1;
server_fd = socket(PF_INET, SOCK_STREAM, 0);
Setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));
```

Socket programming APIs

기능	관련 API
자식 프로세스 생성	fork()
서버 역할 소켓 통신	socket(), setsockopt(), htonl(), htons(), bind(), listen(), accept(), close()
요청 문자열에서 HOST 부분 추출	strtok(), strcpy()
클라이언트 역할 소켓 통신	gethostbyname(), socket(), htons(), connect(), read(), write(), close()
웹 서버 “응답 없음” 처리	signal(), alarm(), raise()

Practice1. Echo Server

- **srv.c**

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>

#define BUFFSIZE 1024
#define PORTNO 40000

int main() {

    struct sockaddr_in server_addr, client_addr;
    int socket_fd, client_fd;
    int len, len_out;
    char buf[BUFFSIZE];

    if ((socket_fd = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        printf("Server: Can't open stream socket.");
        return 0;
    }

    int opt = 1;
    setsockopt(socket_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));

    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    server_addr.sin_port = htons(PORTNO);

    if (bind(socket_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        printf("Server: can't bind local address.\n");
        return 0;
    }

    listen(socket_fd, 5);
```

Practice1. Echo Server (cont'd)

```
while (1) {
    len = sizeof(client_addr);
    client_fd = accept(socket_fd, (struct sockaddr*)&client_addr, &len);
    if (client_fd < 0) {
        printf("Server: accept failed.\n");
        return 0;
    }

    printf("[%d:%d] client was connected.\n",
           client_addr.sin_addr.s_addr, client_addr.sin_port);
    while ((len_out = read(client_fd, buf, BUFSIZE)) > 0) {
        write(STDOUT_FILENO, "  - Message : ", 15);
        write(STDOUT_FILENO, buf, len_out);
        write(client_fd, buf, len_out);
        write(STDOUT_FILENO, "\n", 1);
    }
    printf("[%d:%d] client was disconnected.\n",
           client_addr.sin_addr.s_addr, client_addr.sin_port);

    close(client_fd);
}
close(socket_fd);
return 0;
```

```
}|
```

Practice1. Echo Client

- cli.c

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define BUFFSIZE 1024
#define PORTNO 40000

int main() {
    int socket_fd, len;
    struct sockaddr_in server_addr;
    char haddr[] = "127.0.0.1";
    char buf[BUFFSIZE];

    if ((socket_fd = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        printf("can't create socket.\n");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr(haddr);
    server_addr.sin_port = htons(PORTNO);

    if (connect(socket_fd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
        printf("can't connect.\n");
        return -1;
    }

    write(STDOUT_FILENO, "> ", 2);
    while ((len = read(STDIN_FILENO, buf, sizeof(buf))) > 0) {
        if (write(socket_fd, buf, strlen(buf)) > 0) {
            if ((len = read(socket_fd, buf, sizeof(buf))) > 0) {
                write(STDOUT_FILENO, buf, len);
                memset(buf, 0, sizeof(buf));
            }
        }
        write(STDOUT_FILENO, "> ", 2);
    }
    close(socket_fd);
    return 0;
}
```

Practice1. Execution of Echo Server and Client

■ 실행

- Command 창은 두 개를 이용
- Server를 실행한 상태에서 client를 실행

```
sslab@ubuntu: ~/work/practice
sslab@ubuntu:~/work/practice$ gcc -o cli cli.c
sslab@ubuntu:~/work/practice$ ./cli ②
> How are you? ④
How are you? ⑥
> I'm fine. Thank you. And you? ⑦
I'm fine. Thank you. And you? ⑨
> ^C < Ctrl + D > ⑩
sslab@ubuntu:~/work/practice$ █

sslab@ubuntu: ~/work/practice
sslab@ubuntu:~/work/practice$ gcc -o srv srv.c
sslab@ubuntu:~/work/practice$ ./srv ①
[16777343:34951] client was connected. ③
- Message : How are you? ⑤
- Message : I'm fine. Thank you. And you? ⑧
[16777343:34951] client was disconnected. ⑪
```

■ 결과

- Client에 메시지를 입력하면 Server로 전달 Server에선 받은 메시지를 출력하고 다시 Client에게 전달
- Client는 자신이 보냈던 메시지를 받아서 출력

Practice2. Request URL Parsing

- `srv_request.c`

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>

#define URL_LEN 256
#define BUFSIZE 1024
#define PORTNO 40000

int main() {

    struct sockaddr_in server_addr, client_addr;
    int socket_fd, client_fd;
    int len, len_out;
    int opt = 1;

    if ((socket_fd = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        printf("Server : Can't open stream socket\n");
        return 0;
    }

    setsockopt(socket_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));

    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    server_addr.sin_port = htons(PORTNO);

    if (bind(socket_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        printf("Server : Can't bind local address\n");
        return 0;
    }

    listen(socket_fd, 5);

    while (1)
    {

        struct in_addr inet_client_address;
        char buf[BUFSIZE] = {0, };
        char tmp[BUFSIZE] = {0, };
        char response_header[BUFSIZE] = {0, };
        char response_message[BUFSIZE] = {0, };
        char url[URL_LEN] = {0, };
        char method[20] = {0, };
        char * tok=NULL;
```


Practice2. Request URL Parsing (cont'd)

```
len = sizeof(client_addr);
client_fd = accept(socket_fd, (struct sockaddr*)&client_addr, &len);
if (client_fd < 0) {
    printf("Server : accept failed\n");
    return 0;
}
inet_client_address.s_addr = client_addr.sin_addr.s_addr;
printf("[%s : %d] client was connected\n", inet_ntoa(inet_client_address), client_addr.sin_port);
read(client_fd, buf, BUFSIZE);
strcpy(tmp, buf);
puts("=====");
printf("Request from [%s : %d]\n", inet_ntoa(inet_client_address), client_addr.sin_port);
puts(buf);
puts("=====");

tok = strtok(tmp, " ");
strcpy(method, tok);
if(strcmp(method, "GET") == 0)
{
    tok = strtok(NULL, " ");
    strcpy(url, tok);
}

sprintf(response_message,
        "<h1>RESPONSE</h1><br>"
        "Hello %s:%d<br>"
        "%s", inet_ntoa(inet_client_address), client_addr.sin_port, url);
sprintf(response_header,
        "HTTP/1.0 200 OK\r\n"
        "Server:2019 simple web server\r\n"
        "Content-length:%lu\r\n"
        "Content-type:text/html\r\n\r\n", strlen(response_message));

write(client_fd, response_header, strlen(response_header));
write(client_fd, response_message, strlen(response_message));

printf("[%s : %d] client was disconnected\n", inet_ntoa(inet_client_address), client_addr.sin_port);
close(client_fd);
}

close(socket_fd);
return 0;
}
```

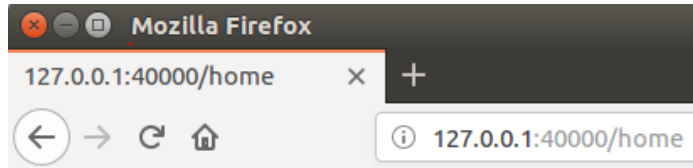
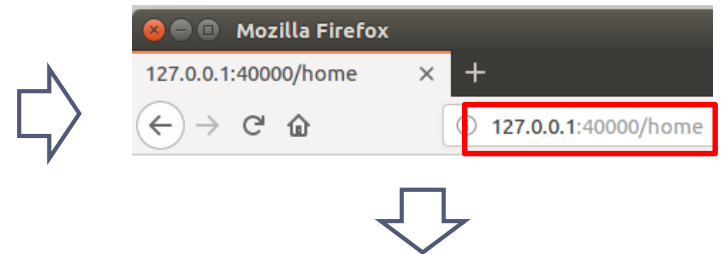
Practice2. Request URL Parsing (cont'd)

■ 실행

- Command 창에서 server 실행
- Server 는 client의 연결을 대기
- Web browser(client)가 server에 HTTP Request 전송
- Server는 HTTP request로부터 URL 추출

```
sslab@ubuntu: ~/work/practice
sslab@ubuntu:~/work/practice$ gcc -o srv_request srv_request.c
sslab@ubuntu:~/work/practice$ ls
srv_request  srv_request.c
sslab@ubuntu:~/work/practice$ ./srv_request
```

Web Browser를 열고 URL 입력



RESPONSE

Hello 127.0.0.1:25779
/home

```
sslab@ubuntu:~/work/practice$ ./srv_request
[127.0.0.1 : 25779] client was connected
=====
Request from [127.0.0.1 : 25779]
GET /home HTTP/1.1
Host: 127.0.0.1:40000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
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

=====
[127.0.0.1 : 25779] client was disconnected
```

Practice2. Request URL Parsing (cont'd)

- 이전 페이지 이어서 브라우저에서 F12(관리자 모드) 진입
- Network 탭 클릭
- F5 눌러서 새로 고침

The image shows a terminal window on the left and a Mozilla Firefox browser window on the right. The terminal displays two HTTP requests from 127.0.0.1:27827 and 127.0.0.1:28339. The browser window shows the response 'Hello 127.0.0.1:27827 /home'. The Network tab in the browser's developer tools is open, showing a list of requests. The first request is a GET request to 127.0.0.1:40000/home, and the second is a GET request to 127.0.0.1:40000/favicon.ico. The status of both requests is 200.

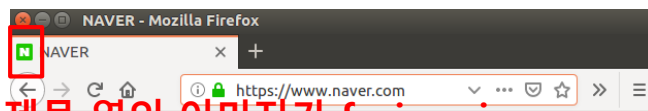
```
[127.0.0.1 : 25779] client was disconnected
[127.0.0.1 : 27827] client was connected
=====
Request from [127.0.0.1 : 27827]
GET /home HTTP/1.1
Host: 127.0.0.1:40000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache
=====
[127.0.0.1 : 27827] client was disconnected
[127.0.0.1 : 28339] client was connected
=====
Request from [127.0.0.1 : 28339]
GET /favicon.ico HTTP/1.1
Host: 127.0.0.1:40000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
=====
[127.0.0.1 : 28339] client was disconnected
```

RESPONSE

Hello 127.0.0.1:27827 /home

Status	Method	Domain	File	Cause	Type	Transferred	Size
200	GET	127.0.0.1:40000	home	document	html	147 B	51 B
200	GET	127.0.0.1:40000	favicon...	img	html	154 B	58 B

요청하는 URL 뿐만 아니라 favicon 등
기타 request도 존재할 수 있음



제목 옆의 이미지가 favicon.ico

2019년 1학기 시스템프로그래밍실습 9주차

Assignment 3-2

System Software Laboratory
College of Software and Convergence
Kwangwoon Univ.

Assignment 3-2

- HTML_1s의 결과를 다른 장치의 웹 브라우저에서 확인할 수 있도록 지원하는 서버 프로그램 작성
 - Steps
 - (1) (**web_server**) Socket 생성
 - 각자 할당 받은 포트 번호 사용
 - (2) (**사용자**) 웹 브라우저로 사용자가 접속
 - ex) http://128.134.52.61:port_number
 - (3) (**web_server**) 웹 브라우저가 보낸 HTTP request message를 처리
 - (4) (**web_server**) HTML-1s의 결과를 HTTP response message에 담아 웹 브라우저에 전송
 - (5) (**사용자**) 웹 브라우저에서 실행 결과를 확인

Assignment 3-2 (cont'd)

- HTML_ls의 결과를 다른 장치의 웹 브라우저에서 확인할 수 있도록 지원하는 서버 프로그램 작성
 - Code Requirements
 - a 옵션 및 -l 옵션을 적용한 결과 출력
 - Hint : getopt 함수에서 optind는 전역 변수이므로 재사용 시 리셋 필요
 - ./web_server의 실행위치를 root path(default directory)로 하고, root path인 경우 -l 옵션 적용
 - root path 밑의 하위 디렉토리인 경우 -a 옵션 적용
 - ex. 사용자 포트: 1111, ./web_server 실행 위치가 /home/user/sp인 경우
 - http:// 223.194.46.163:1111로 접속 시, /home/user/sp의 ls -al결과를 출력
 - http:// 223.194.46.163:1111/tmp로 접속 시, /home/user/sp/tmp의 ls -al결과를 출력
 - 존재하지 않는 디렉토리의 URL 입력 시 404 Error
 - 추가적인 자세한 내용은 실행 예제를 참고
 - 실행 예제에 대한 것만 채점에 반영
 - 콘솔로 출력되는 내용은 채점에 반영하지 않음

Assignment 3-2 (cont'd)

- HTML_ls의 결과를 다른 장치의 웹브라우저에서 확인할 수 있도록 지원하는 서버 프로그램 작성
 - Parsing HTTP Request

GET / HTTP/1.1

- Initial state of request
- List information of files **in default directory**

GET /a.out HTTP/1.1

- (하이퍼링크가 클릭되었을 때) 해당 request가 binary 파일(실행 파일 및 이미지)을 요청하는 경우에는 다운로드 혹은 오픈 (예제 참고)

GET /temp HTTP/1.1

- (하이퍼링크가 클릭되었을 때) Request of directory information
- List information of files **in specified directory**

Assignment 3-2 (cont'd)

- 1. 하이퍼링크는 상대경로로만 생성
- 2. Server application을 실행한 디렉토리의 상위 디렉토리로 이동 불가
 - ex) Server application을 실행한 디렉토리의 path가 /home/sslslab/splab 이라면,
 - 다음의 디렉토리는 **이동 불가 (구현 시 고려하지 않도록 함)**
 - /home/sslslab/bonobono (* 상위 디렉토리 내용물을 알 수 없으므로 이동 불가)
 - /home/sslslab
 - /home
 - /
 - ...
 - 다음의 디렉토리는 **이동 가능**
 - /home/sslslab/splab
 - /home/sslslab/splab/quiz3
 - ...

Assignment 3-2 (cont'd)

■ HTTP Response Header 중..

■ Content-Type

- 콘텐츠의 종류(html, image, ...)를 MIME type(media type) 형태로 기입
- 본 과제에서는, **아래의 MIME type 및 target 만을 고려**
- 이미지의 경우 jpg, png, jpeg 확장자만 고려(대소문자 둘 다 가능)

MIME type	Target
text/html	Web page file (html)
text/plain	Normal text file, source code
image/*	Image file

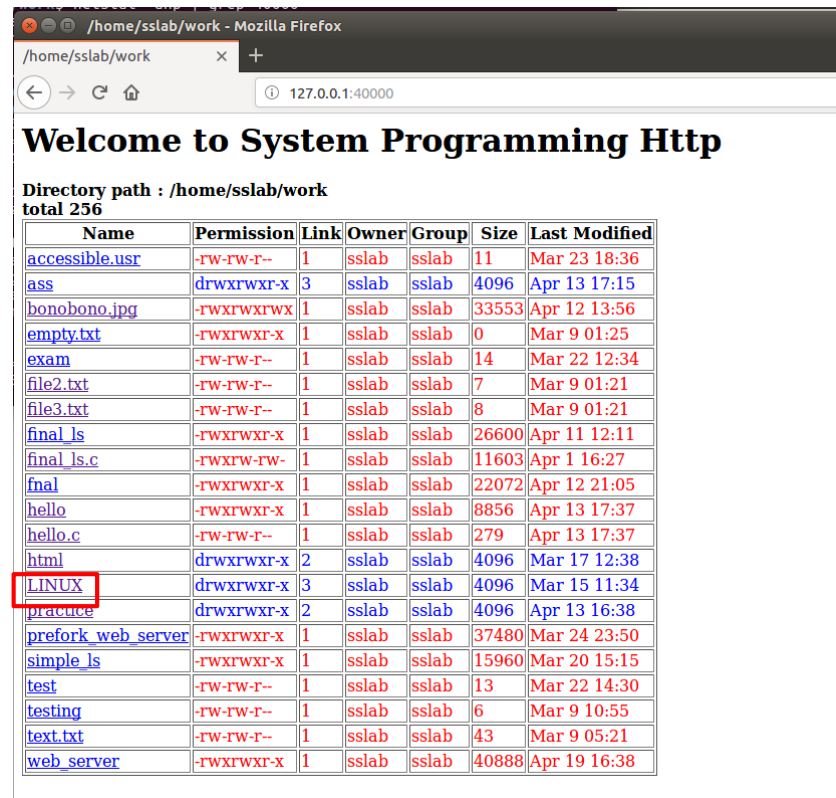
- Hint) _GNU_SOURCE를 #define 하면 fnmatch 사용 시 FNM_CASEFOLD 옵션(대소문자무시) 추가 사용 가능
- 아래와 같은 코드 처리로 이미지 파일 처리

```
#define _GNU_SOURCE
if ( fnmatch("*.jpg", path, FNM_CASEFOLD) == 0 ||
     fnmatch("*.png", path, FNM_CASEFOLD) == 0 ||
     fnmatch("*.jpeg", path, FNM_CASEFOLD) == 0 )
```

Assignment 3-2: Examples

■ Root Path

- ./web_server가 실행되는 root path에서는 -l 옵션의 결과 출력 및 상단에 “Welcome to System Programming Http” 출력
- Root Path의 부모 디렉토리로 이동할 수 없다고 가정
- Root Path의 하위 디렉토리 간에는 자유로운 이동이 가능해야 함



Welcome to System Programming Http

Directory path : /home/sslslab/work
total 256

Name	Permission	Link	Owner	Group	Size	Last Modified
accessible.usr	-rw-rw-r--	1	sslslab	sslslab	11	Mar 23 18:36
ass	drwxrwxr-x	3	sslslab	sslslab	4096	Apr 13 17:15
bonobono.jpg	-rwxrwxrwx	1	sslslab	sslslab	33553	Apr 12 13:56
empty.txt	-rwxrwxr-x	1	sslslab	sslslab	0	Mar 9 01:25
exam	-rw-rw-r--	1	sslslab	sslslab	14	Mar 22 12:34
file2.txt	-rw-rw-r--	1	sslslab	sslslab	7	Mar 9 01:21
file3.txt	-rw-rw-r--	1	sslslab	sslslab	8	Mar 9 01:21
final_ls	-rwxrwxr-x	1	sslslab	sslslab	26600	Apr 11 12:11
final_ls.c	-rwxrw-rw-	1	sslslab	sslslab	11603	Apr 1 16:27
fnal	-rwxrwxr-x	1	sslslab	sslslab	22072	Apr 12 21:05
hello	-rwxrwxr-x	1	sslslab	sslslab	8856	Apr 13 17:37
hello.c	-rw-rw-r--	1	sslslab	sslslab	279	Apr 13 17:37
html	drwxrwxr-x	2	sslslab	sslslab	4096	Mar 17 12:38
LINUX	drwxrwxr-x	3	sslslab	sslslab	4096	Mar 15 11:34
practice	drwxrwxr-x	2	sslslab	sslslab	4096	Apr 13 16:38
prefork_web_server	-rwxrwxr-x	1	sslslab	sslslab	37480	Mar 24 23:50
simple_ls	-rwxrwxr-x	1	sslslab	sslslab	15960	Mar 20 15:15
test	-rw-rw-r--	1	sslslab	sslslab	13	Mar 22 14:30
testing	-rw-rw-r--	1	sslslab	sslslab	6	Mar 9 10:55
text.txt	-rw-rw-r--	1	sslslab	sslslab	43	Mar 9 05:21
web_server	-rwxrwxr-x	1	sslslab	sslslab	40888	Apr 19 16:38

클릭 시 해당
디렉토리로 이동

Assignment 3-2: Examples (cont'd)

- 하위 디렉토리 폴더로 이동시
 - al 옵션의 결과를 출력하고 상단에 “System Programming Http” 출력
 - (단. 폴더명에는 공백이 없다고 가정한다.)
 - ex Untitled Folder (이런 경우는 고려하지 않음)

The screenshot shows a web browser window with the address bar displaying `127.0.0.1:40000/LINUX`. The page title is "System Programming Http". Below the title, the directory path is shown as `/home/sslslab/work/LINUX` with a total size of 32. A table lists the contents of the directory:

Name	Permission	Link	Owner	Group	Size	Last Modified
.	drwxrwxr-x	3	sslslab	sslslab	4096	Mar 15 11:34
..	drwxrwxr-x	6	sslslab	sslslab	4096	Apr 19 16:56
file_a	-rw-rw-r--	2	sslslab	sslslab	33	Mar 9 06:50
file_b	-rw-rw-r--	2	sslslab	sslslab	33	Mar 9 06:50
file_c	-rw-rw-r--	1	sslslab	sslslab	16	Mar 9 06:52
file_d	-rw-rw-r--	1	sslslab	sslslab	33	Mar 9 06:52
test	drwxrwxr-x	2	sslslab	sslslab	4096	Mar 21 00:08
world.txt	-rw-rw-r--	1	sslslab	sslslab	4	Mar 9 06:44

해당 디렉토리로 이동된 URL

Assignment 3-2: Examples (cont'd)

- Text 파일 및 소스 코드 클릭 시
 - 파일의 내용을 display
 - 링크파일 클릭 시 원본 파일내용 display

```
127.0.0.1:40000/practice/cli x +
127.0.0.1:40000/practice/cli.c

#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define BUFFSIZE 1024
#define PORTNO 40000

int main() {
    int socket_fd, len;
    struct sockaddr_in server_addr;
    char haddr[] = "127.0.0.1";
    char buf[BUFFSIZE];

    if ((socket_fd = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        printf("can't create socket.\n");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr(haddr);
    server_addr.sin_port = htons(PORTNO);

    if (connect(socket_fd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
        printf("can't connect.\n");
        return -1;
    }
}
```

```
Mozilla Firefox
127.0.0.1:40000/practice/IPSL x +
127.0.0.1:40000/practice/IPSL

this is text file WOW!
```

```
Mozilla Firefox
127.0.0.1:40000/html/exam x +
127.0.0.1:40000/html/example_link

<html>
<head>
<title> Our first document </title>
</head>
<body>
Hello World
</body>
</html>
```

/home/sslslab/work - Mozilla Firefox

/home/sslslab/work x +

127.0.0.1:40000/practice

System Programming Http

Directory path : /home/sslslab/work/practice
total 48

Name	Permission	Link	Owner	Group	Size	Last Modified
.	drwxrwxr-x	2	sslslab	sslslab	4096	Apr 19 17:03
..	drwxrwxr-x	6	sslslab	sslslab	4096	Apr 19 16:56
cli	-rwxrwxr-x	1	sslslab	sslslab	9128	Apr 13 16:38
cli.c	-rw-rw-r--	1	sslslab	sslslab	1148	Apr 12 03:48
IPSL	-rw-rw-r--	1	sslslab	sslslab	23	Apr 19 17:03
srv	-rwxrwxr-x	1	sslslab	sslslab	9280	Apr 13 16:38
srv.c	-rw-rw-r--	1	sslslab	sslslab	1602	Apr 12 03:42
srv_request.c	-rw-rw-r--	1	sslslab	sslslab	2716	Apr 12 03:43

/home/sslslab/work - Mozilla Firefox

/home/sslslab/work x +

127.0.0.1:40000/html

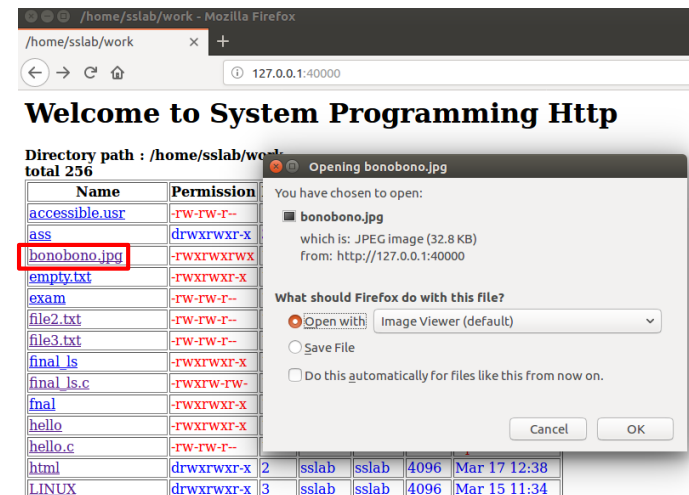
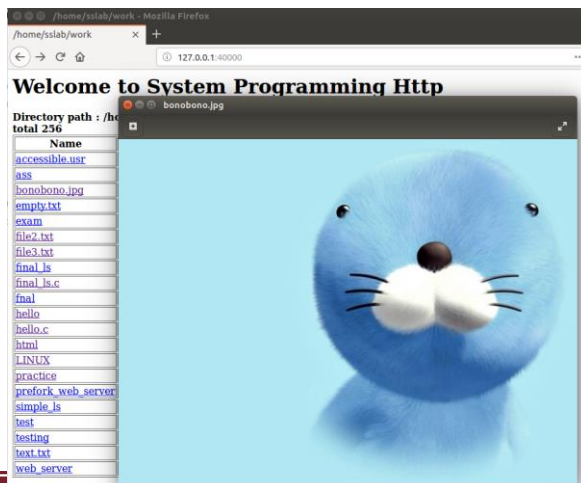
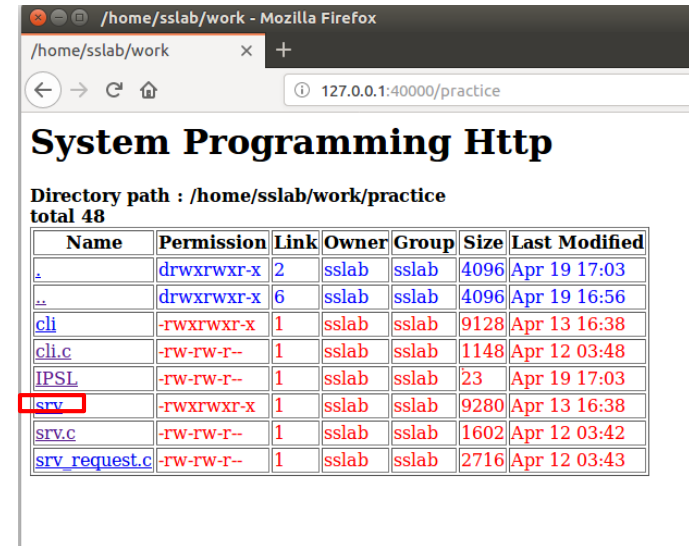
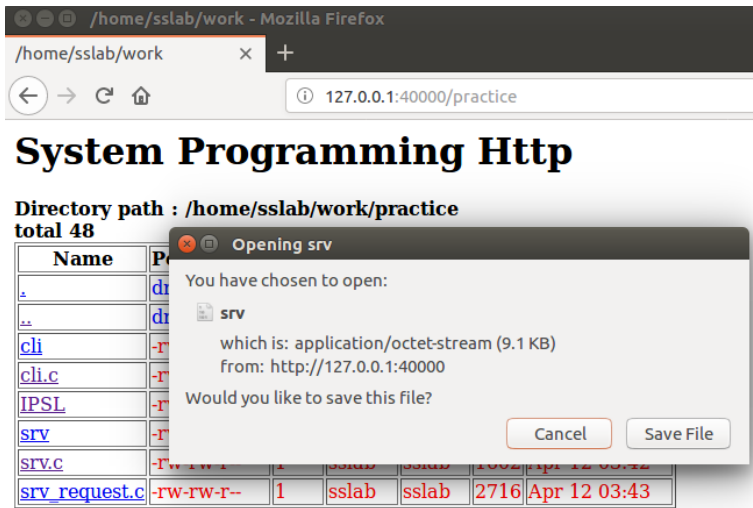
System Programming Http

Directory path : /home/sslslab/work/html
total 28

Name	Permission	Link	Owner	Group	Size	Last Modified
.	drwxrwxr-x	2	sslslab	sslslab	4096	Apr 19 17:42
..	drwxrwxr-x	6	sslslab	sslslab	4096	Apr 19 16:56
ex.html	-rw-rw-r--	1	sslslab	sslslab	92	Mar 17 12:38
example_link	lrwxrwxrwx	1	sslslab	sslslab	7	Apr 19 17:42
file1.txt	-rw-rw-r--	1	sslslab	sslslab	4	Mar 9 01:00
file2.txt	-rw-rw-r--	1	sslslab	sslslab	5	Mar 9 01:00
file3.txt	-rw-rw-r--	1	sslslab	sslslab	4	Mar 9 01:00
hello_copy.txt	-rw-rw-r--	1	sslslab	sslslab	5	Mar 9 01:00

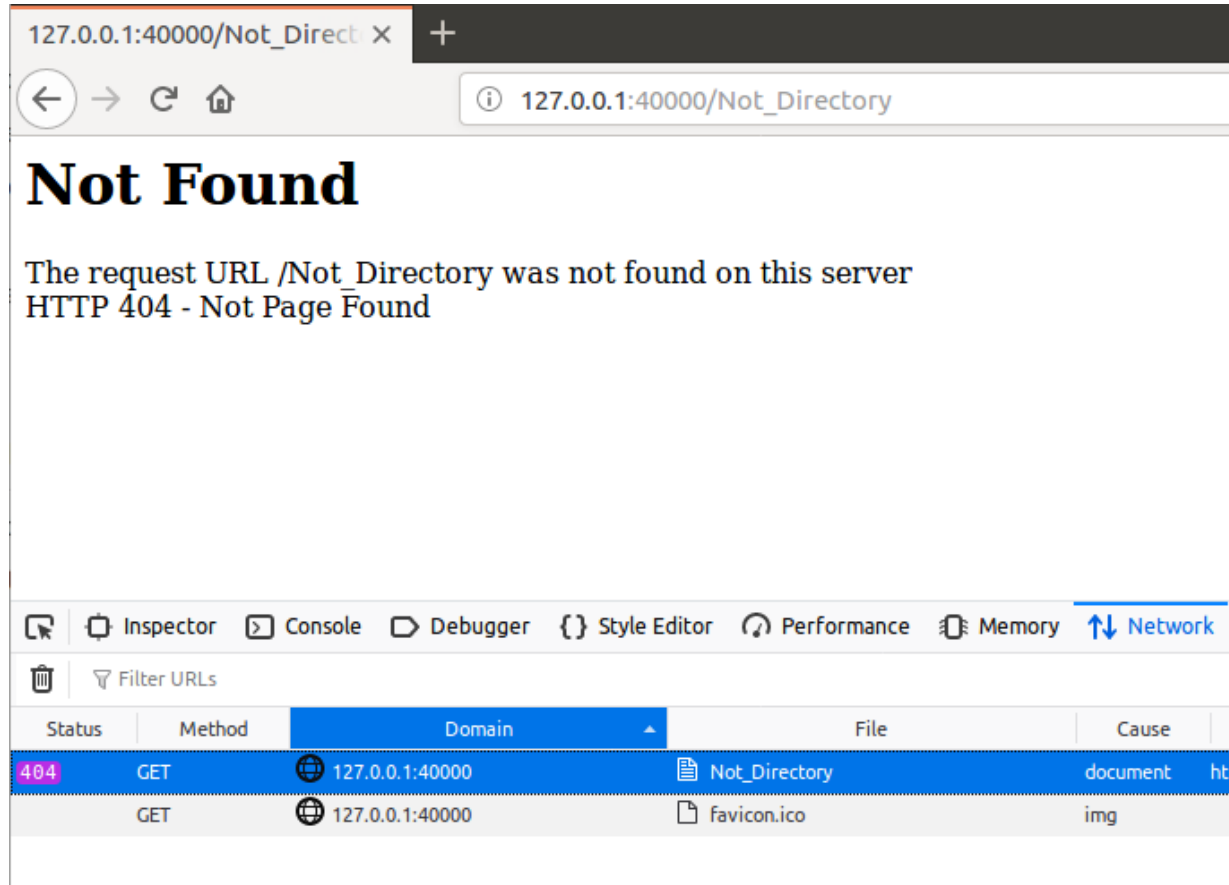
Assignment 3-2: Examples (cont'd)

- 실행 파일(binary file, image file) 클릭 시
 - 파일 Download가 정상적으로 되어야 함



Assignment 3-2: Examples (cont'd)

- 존재하지 않는 디렉토리 URL 입력 시



- 과제 진행 시 favicon.ico Request는 무시

Assignment 3-2

▪ Softcopy Upload

- 제출 파일
 - 보고서 (파일명 : 요일_3-2_학번.pdf)
 - *.c , *.h, Makefile (실행 파일명 : web_server)
- **Soruce code copy 적발 시 예외 없이 0점 처리**
- 위 파일들을 압축해서 제출 (파일명: 실습 요일_3-2_학번.tar.gz)
 - e.g. 월1,2 → **mon_3-2_2017202000.tar.gz**
 - e.g. 화3,4 → **tue_3-2_2017202000.tar.gz**
 - e.g. 금5,6 → **fri_3-2_2017202000.tar.gz**
- U-Campus의 과제 제출에 **5월 10일(금) 23:59:59까지** 제출
 - U-Campus에 올린 후 다시 다운로드 받아서 **파일이 정확한지 확인**
- 미리 공지한 바와 같이, **delay 받지 않음 (예외 없음)**
- **Ubuntu 16.04 64bits 환경**에서 채점

▪ 과제 질문 관련

- 해당 과제 출제 담당 조교에게 이메일로 문의 → **남건욱 조교 (ngotic@kw.ac.kr)**
- 과제 제출 마감 당일에는 **오후 4시까지 도착한 질문 메일에만 답변**

Assignment 3-2 (cont'd)

■ 표지

- 다음의 내용은 **필히** 기록
 - 과제 이름 (e.g. Assignment#3-2)
 - 분반 (요일, 담당 교수님)
 - 본인 인적 사항 (학번, 이름)

■ 과제 내용

- Introduction : 5줄 이하
- Flow chart : 4주차 자료의 Appendix 참고
- Pseudo code : 4주차 자료의 Appendix 참고
- Result : 수행한 내용을 캡처 이미지와 함께 설명
- Conclusion : 결론 및 고찰

■ 보고서 이름은 “실습 요일_과제명_학번”으로 수정

- e.g. 월1,2 → **mon_3-2_2017202000.pdf**
- e.g. 화3,4 → **tue_3-2_2017202000.pdf**
- e.g. 금5,6 → **fri_3-2_2017202000.pdf**

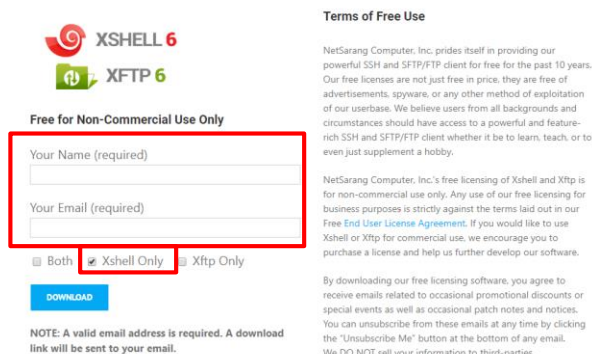
Appendix. Remote Access Terminal

■ Xshell 6

- OpenSSH에 접속 가능한 터미널 프로그램

■ Installation

- Download
 - <https://www.netsarang.com/en/free-for-home-school/>
 - **Non-Commercial** 버전 선택 후 다운로드



The screenshot shows the Xshell 6 download page. At the top, there are logos for XSHELL 6 and XFTP 6. Below them is a section titled "Free for Non-Commercial Use Only". This section contains two input fields: "Your Name (required)" and "Your Email (required)". Below these fields are three radio buttons: "Both", "Xshell Only" (which is selected and highlighted with a red box), and "Xftp Only". A blue "DOWNLOAD" button is located below the radio buttons. To the right of the form, there is a "Terms of Free Use" section with text explaining the free licensing. At the bottom, a note states: "NOTE: A valid email address is required. A download link will be sent to your email."



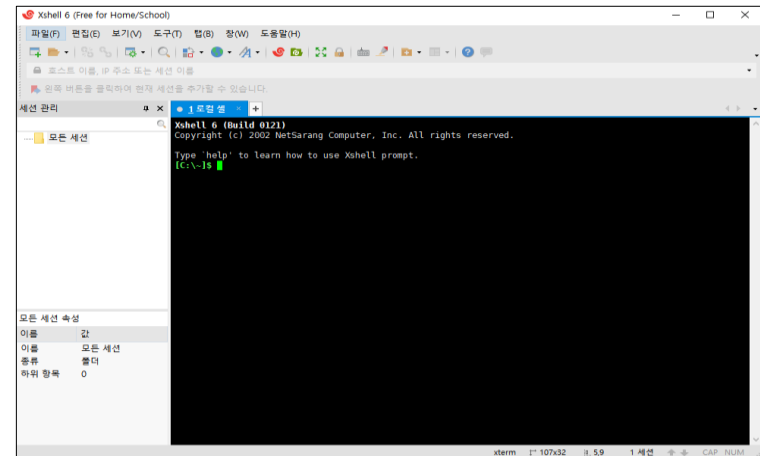
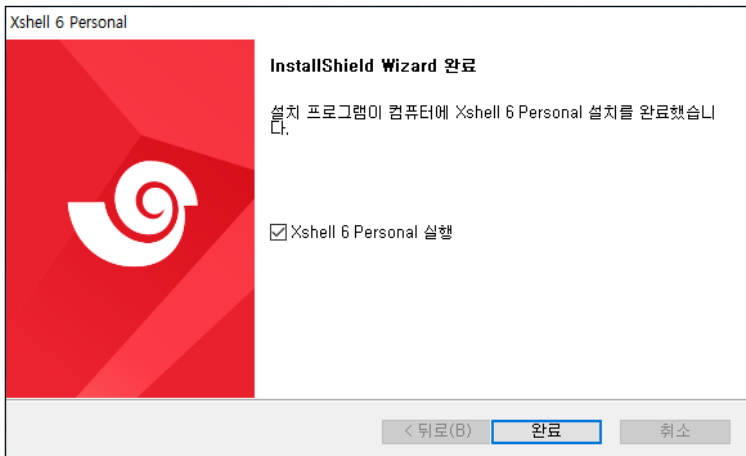
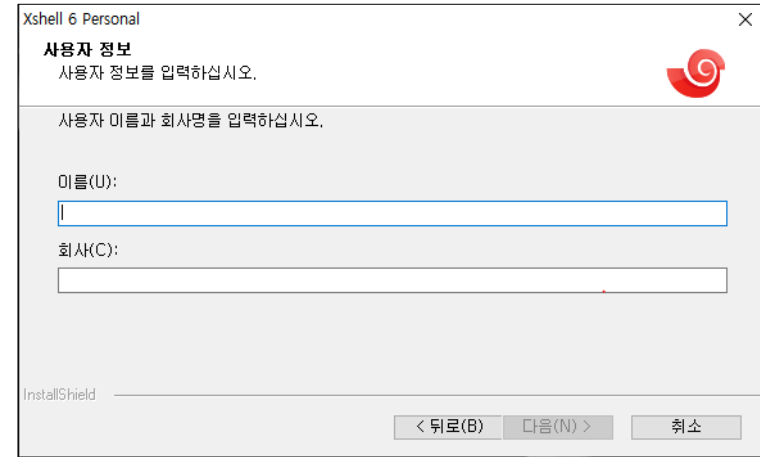
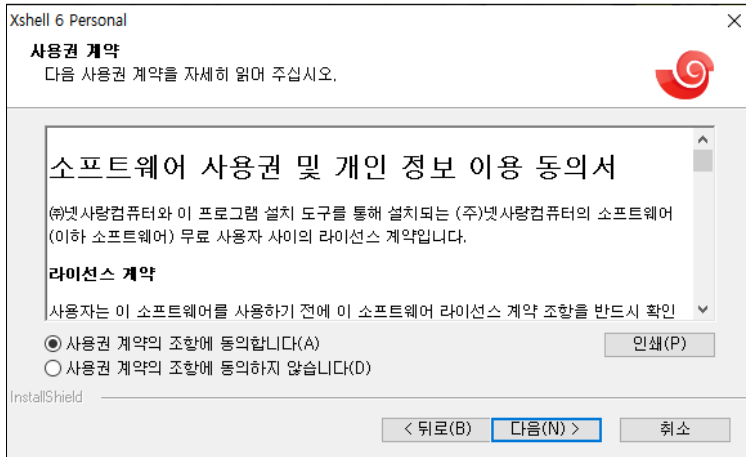
Thank you for requesting a download of our software.

A download link will be emailed to the email address you have inputted.

If you are unable to locate it, please make sure our email was not filtered to your Spam folder.

Appendix. Remote Access Terminal (cont'd)

■ Installation

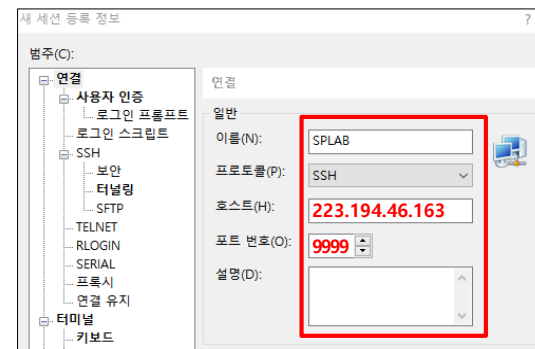
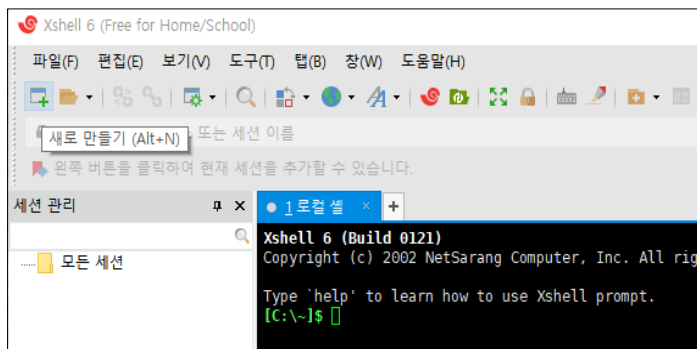


Appendix. Remote Access Terminal (cont'd)

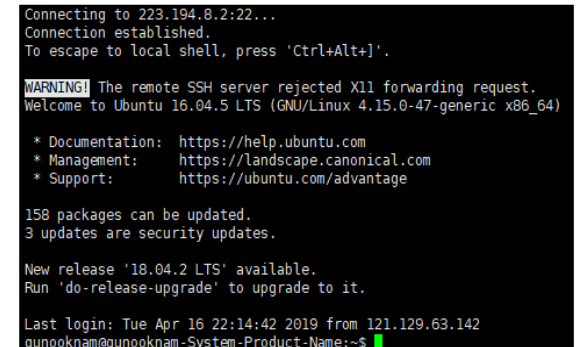
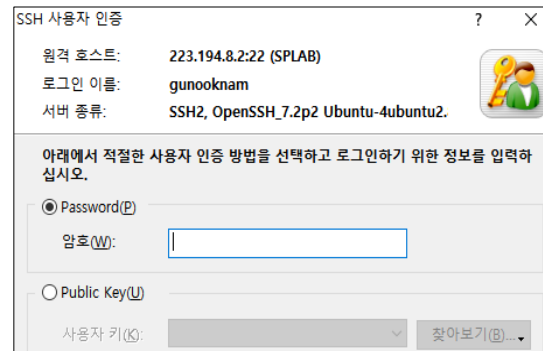
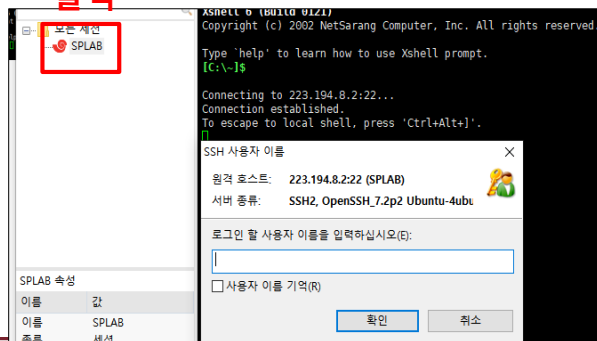
Log-in

Set-up

- 새 세션 등록 ➔ 이름 및 비밀번호 입력하여 접속
 - IP: 223.194.46.163
 - 포트번호: 9999
- ID는 본인 학번, 초기 비밀번호는 ID와 동일
- 첫 로그인 후, \$ passwd 명령어로 필히 비밀번호를 변경할 것**



클릭



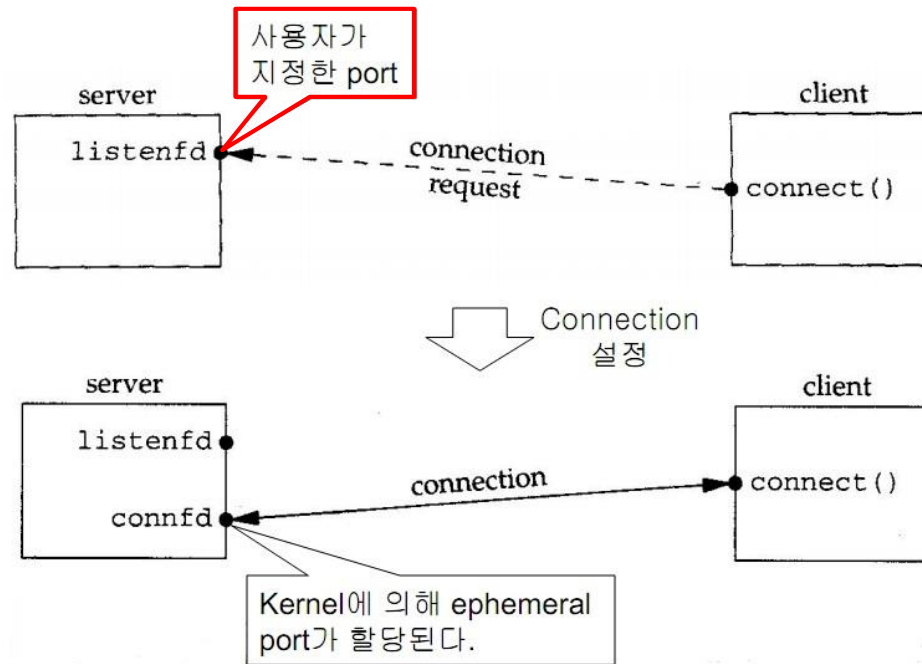
Appendix. Remote Access Terminal (cont'd)

- 시스템프로그래밍실습 공용 서버 사용 관련 안내 사항
 - 다음의 내용을 위반할 경우, 해당 주차 과제 감점 처리
 - 포트 번호는 본인에게 부여된 것만 사용
 - 본 수업에 관련된 프로그램 테스트 목적으로만 사용할 것
 - 동작시킨 프로그램이 무한 fork를 발생시키지 않도록 주의 깊게 관리
 - 다음의 내용을 위반할 경우, 해당 주차 과제 0점 처리
 - 본인 계정만 사용
 - 다른 계정에 접근 시도하는 것을 과제 카피 시도로 간주
 - 기타
 - 처음 부여한 비밀번호는 필히 바꾸어 사용할 것
 - 본인이 작성한 소스 코드를 서버에 두지 말 것
 - 서버를 코드 보관용으로 사용하지 말 것
 - 본 서버는 테스트 용으로만 운영되며, 언제든지 데이터가 유실될 수 있음
 - 서버 동작 관련 문의
 - 김규식 조교(kks@kw.ac.kr / 김태석 교수님 연구실)

Appendix. Remote Access Terminal (cont'd)

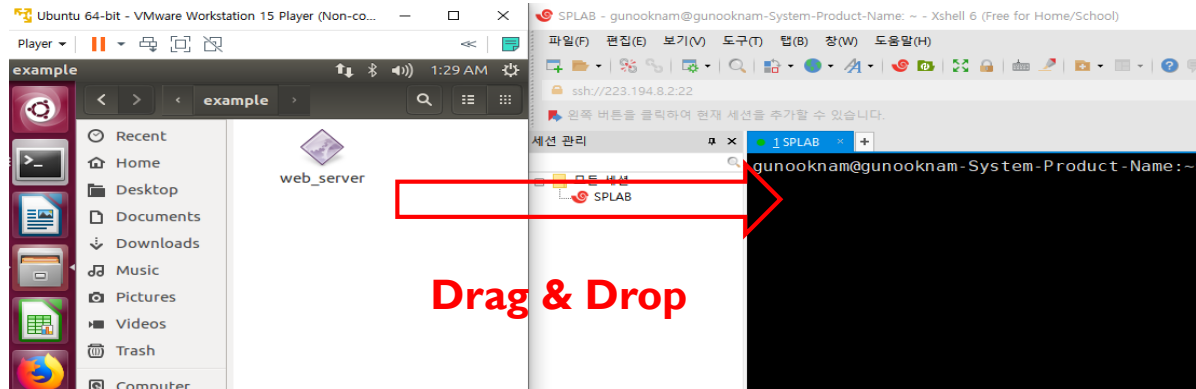
■ 포트 번호

- [U-캠퍼스] → [공지사항] 에 게시된 개인 포트 번호 확인
- 각자 구현된 웹 서버는 **각 학생에게 지정된 포트 번호 만을 사용**해야 함



Appendix. Remote Access Terminal (cont'd)

- Upload files to SPLAB. Server and change permission
 - Upload



- Change Permission
 - 서버에 upload되는 파일은 실행 권한이 OFF 되므로 chmod 명령으로 권한 on

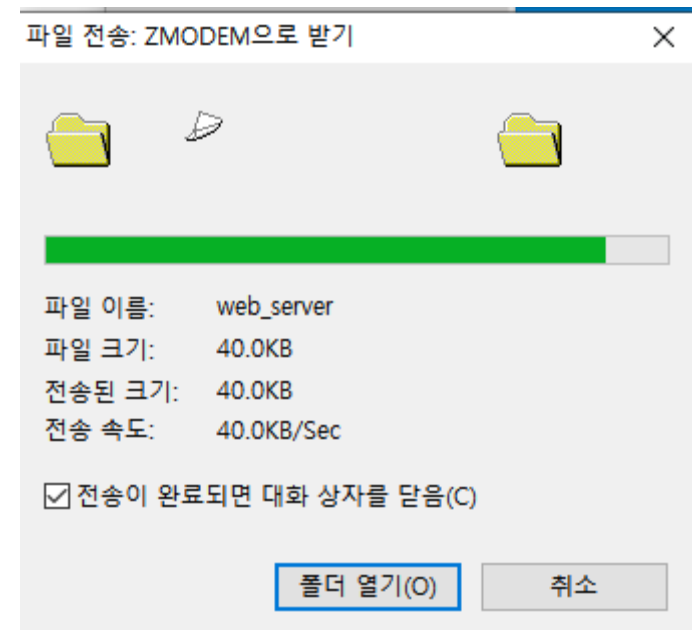
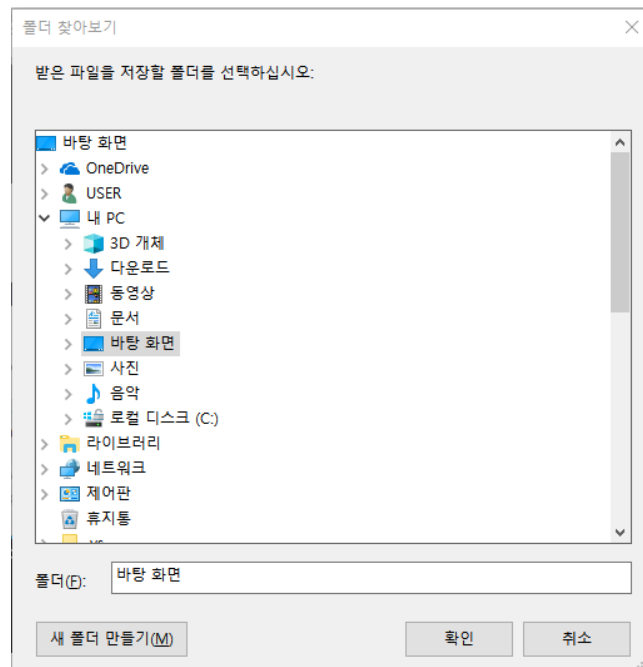
```
gunooknam@gunooknam-System-Product-Name:~$ ls -l web_server
-rw-r--r-- 1 gunooknam gunooknam 40984  4월  17 00:52 web_server
gunooknam@gunooknam-System-Product-Name:~$ chmod 777 web_server
gunooknam@gunooknam-System-Product-Name:~$ ls -l web server
-rwxrwxrwx 1 gunooknam gunooknam 40984  4월  17 00:52 web server
gunooknam@gunooknam-System-Product-Name:~$
```

Appendix. Remote Access Terminal (cont'd)

- Download files from SPLAB. Server

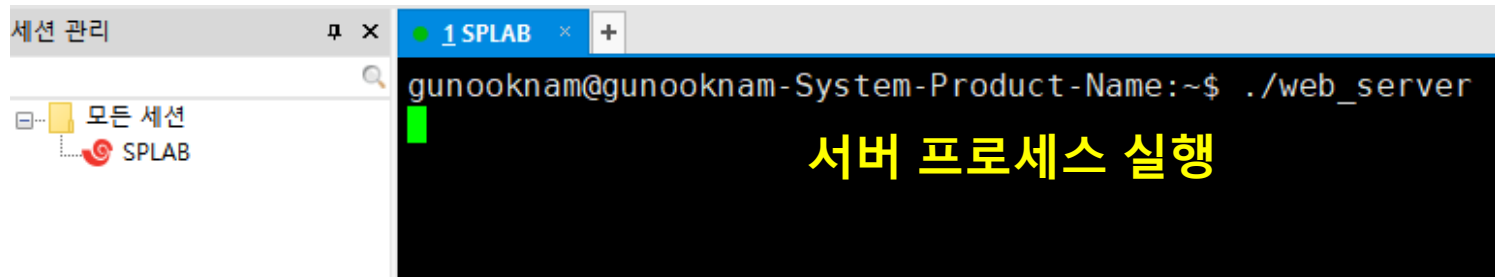
```
gunooknam@gunooknam-System-Product-Name:~$ sz web_server
```

\$ sz 파일_이름

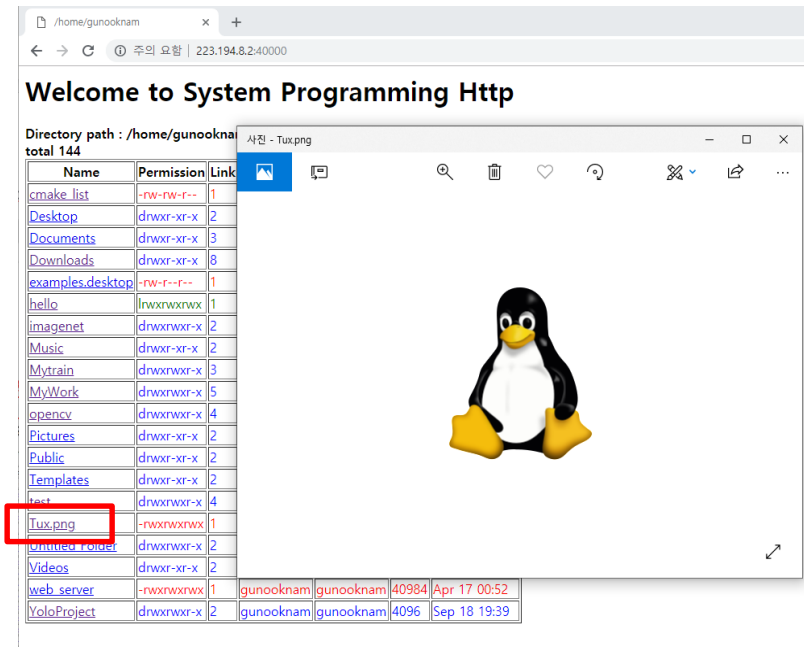


Appendix. Remote Access Terminal (cont'd)

- Run Splab web_server



Chrome or firefox로 실행하여 IP:Port_number 입력, 이 때 Port 번호는 자신에게 할당된 번호 입력



Appendix. 특정 포트번호를 가진 프로세스 종료

- 모든 프로세스를 확인
 - ps 명령어에 ef 옵션을 붙여서 확인

```
sslab@ubuntu:~$ ps -ef | tail
sslab    36678  36677  0 21:25 ?        00:00:00 /usr/lib/openssh/sftp-server
root     36682   1057  0 21:25 ?        00:00:00 sshd: sslab [priv]
root     36686   1057  0 21:25 ?        00:00:00 sshd: sslab [priv]
sslab    36716  36682  0 21:25 ?        00:00:00 sshd: sslab@notty
sslab    36749  36686  0 21:25 ?        00:00:00 sshd: sslab@notty
sslab    36754  36749  0 21:25 ?        00:00:00 /usr/lib/openssh/sftp-server
root     37060     2   0 21:26 ?        00:00:00 [kworker/u256:0]
sslab    37070  35802  0 21:27 pts/18   00:00:00 ./web_server
sslab    37076  35940  0 21:27 pts/20   00:00:00 ps -ef
sslab    37077  35940  0 21:27 pts/20   00:00:00 tail
sslab@ubuntu:~$
```

- 특정 포트 번호를 가진 프로세스 검색을 위해 netstat 명령어 사용
 - netstat -anp | grep “특정 포트번호”
 - 해당 프로세스를 찾아서 kill 명령어 사용

```
sslab@ubuntu:~/work$ ./web_server
Killed
sslab@ubuntu:~/work$

sslab@ubuntu:~$ netstat -anp | grep 40000
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:40000        0.0.0.0:*           LISTEN      37070/web_server
sslab@ubuntu:~$ kill -9 37070
sslab@ubuntu:~$
```