

# 네트워크 프로그래밍

## 2. 소켓 통신 과정

# 클라이언트와 서버의 통신

2

- 클라이언트 : 연결을 초기화 하는 주체

Client: Bob

Server: Jane

“Hi. I’ m Bob.” →

← “Hi, Bob. I’ m Jane”

“Nice to meet you, Jane.” →

- 서버 : 수동적으로 연결을 기다림

# TCP 상의 서버/클라이언트 통신

3

## 클라이언트

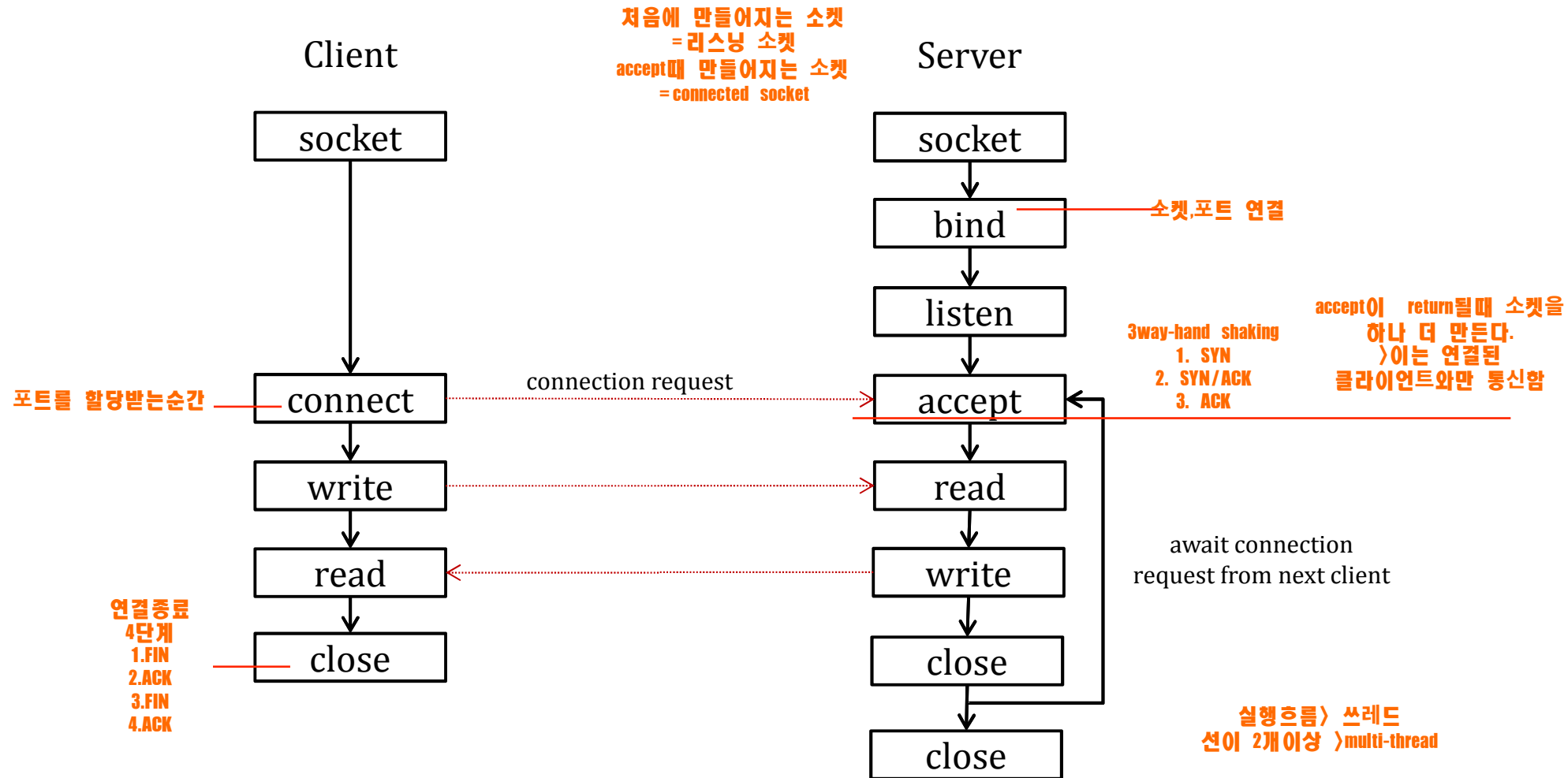
1. TCP 소켓의 생성 `socket()`
2. 연결의 요청
3. 데이터 송수신
4. 연결 종료

## 서버

1. TCP 소켓의 생성 `socket()`
2. IP와 PORT번호의 할당 `bind()`
3. 연결요청 가능상태로 변경
4. 다음을 반복적으로 수행
  - a. 연결요청에 대한 수락
  - b. 데이터 송수신
  - c. 연결 종료
5. 연결 종료

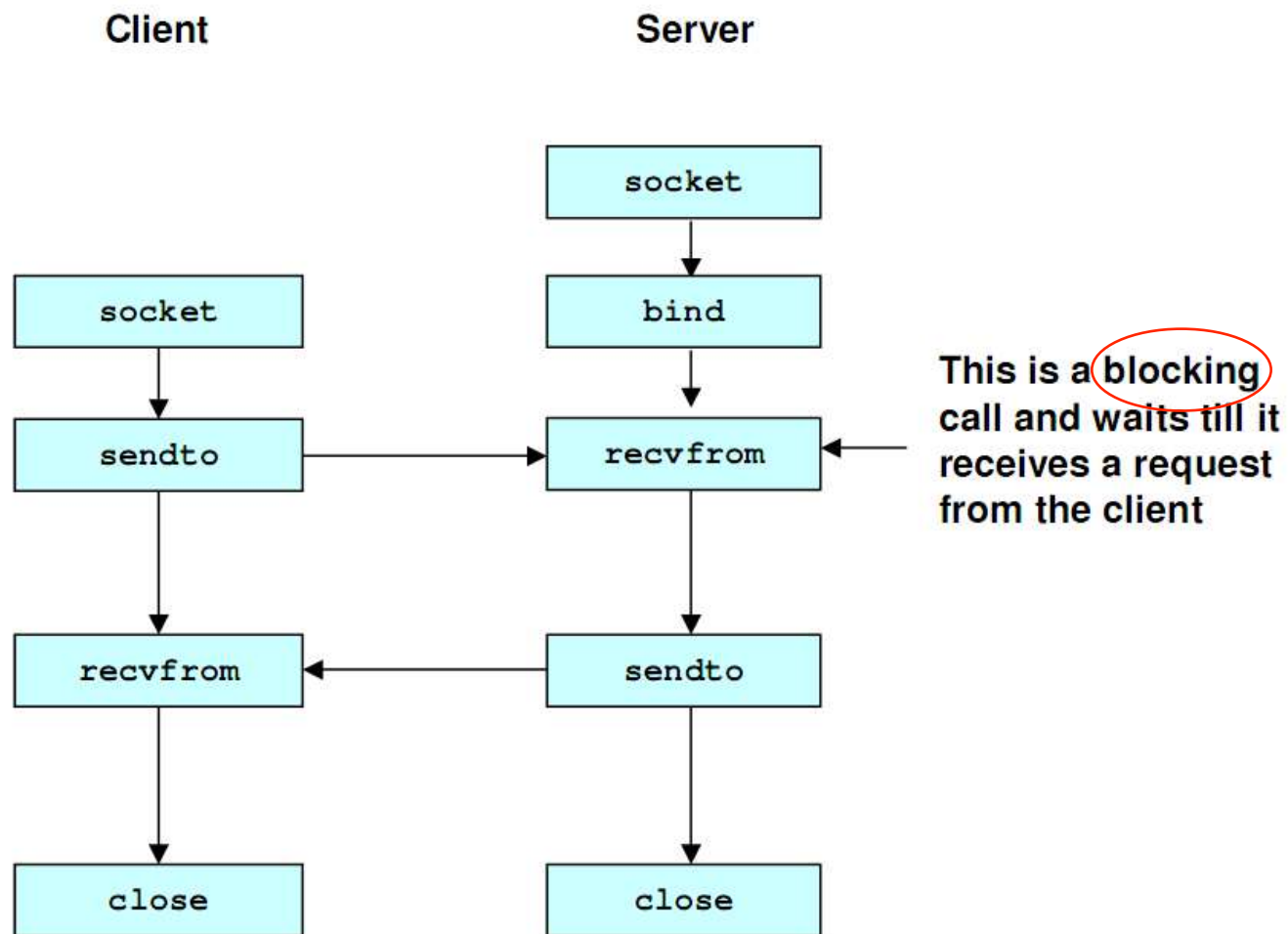
# TCP 클라이언트/서버 프로그램의 흐름

4



## UDP 클라이언트/서버 프로그램의 흐름 - 참고

5



## hello\_server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

void error_handling(char *message)
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}

int main(int argc, char *argv[])
{
    int serv_sock;
    int clnt_sock;

    struct sockaddr_in serv_addr;
    struct sockaddr_in clnt_addr;
    socklen_t clnt_addr_size;

    char message[]="Hello World!";

    if(argc!=2){
        printf("Usage : %s <port>\n", argv[0]);
        exit(1);
    }

    serv_sock=socket(PF_INET, SOCK_STREAM, 0);
    if(serv_sock == -1)
        error_handling("socket() error");

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=htonl(INADDR_ANY);
    serv_addr.sin_port=htons(atoi(argv[1]));

    if(bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))== -1 )
        error_handling("bind() error");

    if(listen(serv_sock, 5)==-1)
        error_handling("listen() error");

    clnt_addr_size=sizeof(clnt_addr);
    clnt_sock=accept(serv_sock, (struct sockaddr*)&clnt_addr,&clnt_addr_size);
    if(clnt_sock==-1)
        error_handling("accept() error");

    write(clnt_sock, message, sizeof(message));
    close(clnt_sock);
    close(serv_sock);
    return 0;
}
```

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

void error_handling(char *message);

int main(int argc, char* argv[])
{
    int sock;
    struct sockaddr_in serv_addr;
    char message[30];
    int str_len;

    if(argc!=3){
        printf("Usage : %s <IP> <port>\n", argv[0]);
        exit(1);
    }

    sock=socket(PF_INET, SOCK_STREAM, 0);
    if(sock == -1)
        error_handling("socket() error");

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=inet_addr(argv[1]);
    serv_addr.sin_port=htons(atoi(argv[2]));

    if(connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))== -1)
        error_handling("connect() error!");

    str_len=read(sock, message, sizeof(message)-1);
    if(str_len== -1)
        error_handling("read() error!");

    printf("Message from server: %s\n", message);
    close(sock);
    return 0;
}

```

## hello\_client.c

```

void error_handling(char *message)
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}

```

# 리눅스 기반에서의 실행방법

8

## □ 컴파일 및 실행방법

### 컴파일 방법

```
gcc hello_server.c -o hserver
```

➔ hello\_server.c 파일을 컴파일해서 hserver라는 이름의 실행파일을 만드는 문장이다.

### 실행방법

```
./hserver
```

➔ 현재 디렉터리에 있는 hserver라는 이름의 파일을 실행시키라는 의미이다.

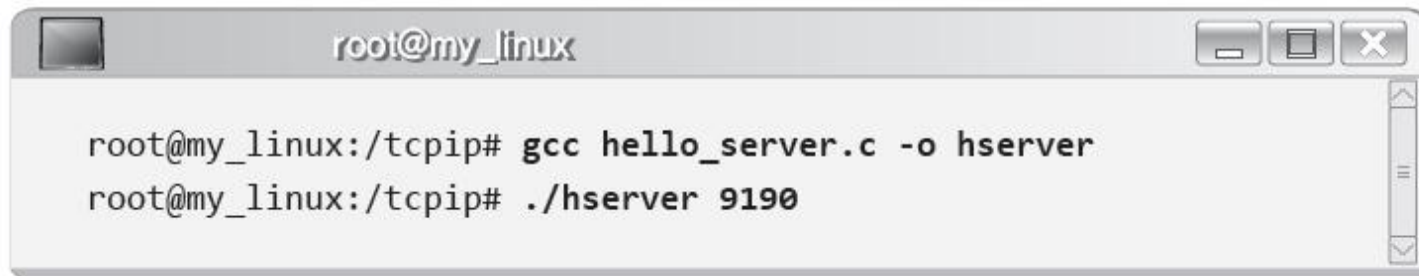


# 리눅스 기반에서의 실행결과

9

## □ 예제의 실행결과

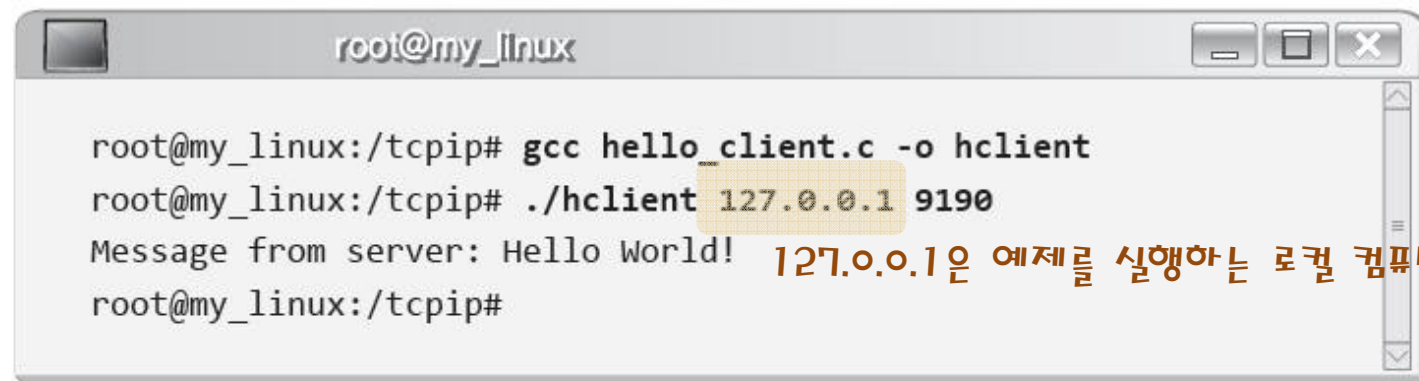
❖ 실행결과: hello\_server.c



```
root@my_linux

root@my_linux:/tcpip# gcc hello_server.c -o hserver
root@my_linux:/tcpip# ./hserver 9190
```

❖ 실행결과: hello\_client.c



```
root@my_linux

root@my_linux:/tcpip# gcc hello_client.c -o hclient
root@my_linux:/tcpip# ./hclient 127.0.0.1 9190
Message from server: Hello World!
root@my_linux:/tcpip#
```

127.0.0.1은 예제를 실행하는 로컬 컴퓨터를 의미함