ISL

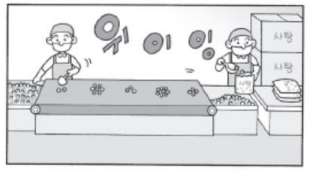
Network Programming #1

ISL (IoT Standard Lab)

실습 예제 소개

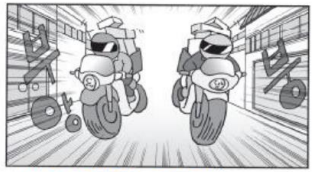
두 타입의 소켓

- 연결지향형 소켓(SOCK_STREAM)의 데이터 전송특성 (TCP 소켓)
  - 중간에 데이터 소멸되지 않는다.
  - 전송 순서대로 데이터가 수신된다.
  - 데이터의 경계가 존재하지 않는다.
  - 소켓 대 소켓의 연결은 반드시 1대 1의 구조.



TCP 데이터 전송특성

- 비 연결지향형 소켓(SOCK_DGRAM)의 데이터 전송특성 (UDP 소켓)
  - 전송순서 상관없이 빠른 속도의 전송을 지향
  - 데이터 손실 및 파손의 우려 있다.
  - 데이터의 경계가 존재한다.
  - 한번에 전송할 수 있는 데이터의 크기가 제한된다.



UDP 데이터 전송특성

리눅스 기반에서의 실행결과

- 예제의 실행결과
  - ❖ 실행결과: hello_server.c

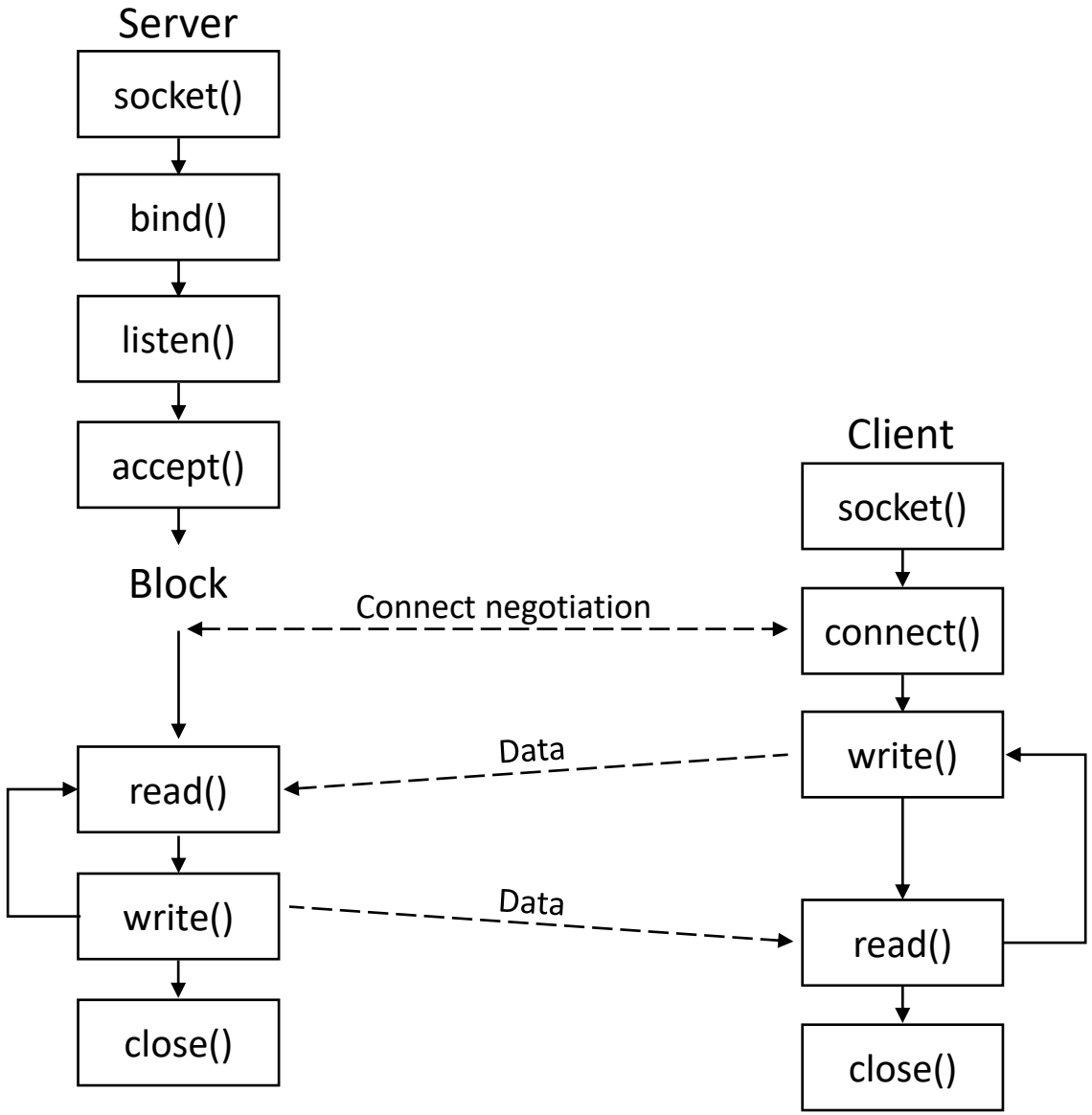
```
root@my_linux
root@my_linux:/tcpip# gcc hello_server.c -o hserver
root@my_linux:/tcpip# ./hserver 9190
```

  - ❖ 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#
```

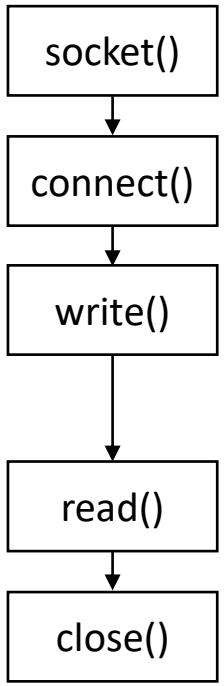
127.0.0.1은 예제를 실행하는 로컬 컴퓨터를 의미함

TCP 소켓 프로그래밍 흐름



클라이언트

Client



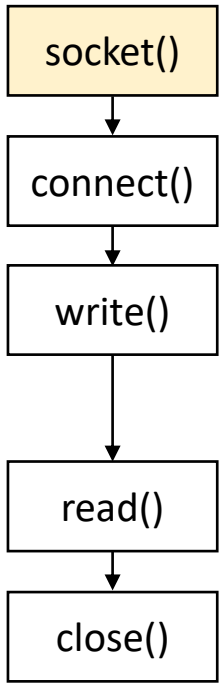

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>

int main(int argc, char** argv) {
    int sockfd;
    char buf[1024];
    char *hello = "Hello from client";
    struct sockaddr_in servaddr;

    if(argc < 3) {
        printf("usage:./client remoteAddress remotePort\n");
        return -1;
    }

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket creation failed");
        return -1;
    }

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr(argv[1]);
    servaddr.sin_port = htons(atoi(argv[2]));

    if (connect(sockfd, (const struct sockaddr*)&servaddr, sizeof(servaddr)) < 0) {
        perror("connect error");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    write(sockfd, hello, strlen(hello));
    read(sockfd, buf, sizeof(buf));
    printf("Message from server: %s\n", buf);

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

```
int argc    : arguments count
char **argv : arguments vector
     *argv[]
```

```
$ ./client 127.0.0.1 9190
 int argc    = 2
 char **argv = { "./hello", "9190" }

 argv[0] = { "./hello" }
 argv[1] = { "127.0.0.1" }
 argv[2] = { "9190" }
```

클라이언트

Client

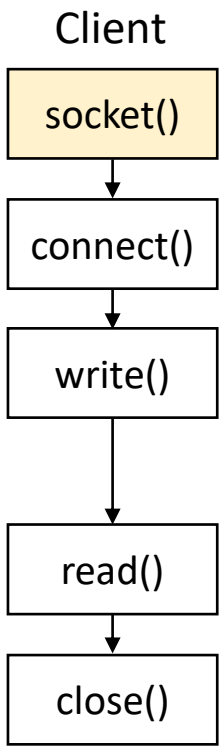



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>

int main(int argc, char** argv) {
    int sockfd;
    char buf[1024];
    char *hello = "Hello from client";
    struct sockaddr_in servaddr;

    if(argc < 3) {
        printf("usage:./client remoteAddress remotePort\n");
        return -1;
    }

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket creation failed");
        return -1;
    }

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr(argv[1]);
    servaddr.sin_port = htons(atoi(argv[2]));

    if (connect(sockfd, (const struct sockaddr*)&servaddr, sizeof(servaddr)) < 0) {
        perror("connect error");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    write(sockfd, hello, strlen(hello));
    read(sockfd, buf, sizeof(buf));
    printf("Message from server: %s\n", buf);

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

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

```c
int domain : Protocol Family
int type : Socket Type
int protocol : protocol
```

클라이언트




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

int main(int argc, char** argv) {
    int sockfd;
    char buf[1024];
    char *hello = "Hello from client";
    struct sockaddr_in servaddr;

    if(argc < 3) {
        printf("usage:./client remoteAddress remotePort\n");
        return -1;
    }

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket creation failed");
        return -1;
    }

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr(argv[1]);
    servaddr.sin_port = htons(atoi(argv[2]));

    if (connect(sockfd, (const struct sockaddr*)&servaddr, sizeof(servaddr)) < 0) {
        perror("connect error");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    write(sockfd, hello, strlen(hello));
    read(sockfd, buf, sizeof(buf));
    printf("Message from server: %s\n", buf);

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

AF_INET : IPv4 Internet Protocol Family

AF_INET vs PF_INET

AF_INET : Address Family

PF_INET : Protocol Family

*뭘 넣든 상관없이 동작함

SOCK_STREAM : 연결 지향형 타입(커넥션 생성)

AF_INET + SOCK_STREAM
: IPv4 + 연결 지향형 타입 = TCP

AF_INET + SOCK_DGRAM
: IPv4 + 비연결 지향형 타입 = UDP

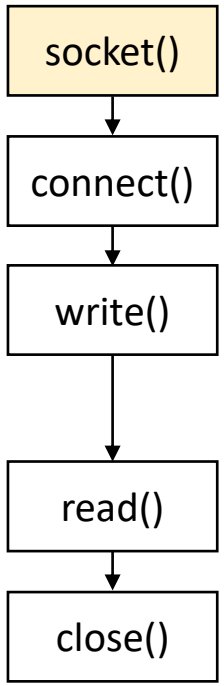
AF_INET + SOCK_STREAM : IPPROTO_TCP

AF_INET + SOCK_DGRAM : IPPROTO_UDP

0 : 자동 지정

클라이언트

Client




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

int main(int argc, char** argv) {
    int sockfd;
    char buf[1024];
    char *hello = "Hello from client";
    struct sockaddr_in servaddr;

    if(argc < 3) {
        printf("usage:./client remoteAddress remotePort\n");
        return -1;
    }

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket creation failed");
        return -1;
    }

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr(argv[1]);
    servaddr.sin_port = htons(atoi(argv[2]));

    if (connect(sockfd, (const struct sockaddr*)&servaddr, sizeof(servaddr)) < 0) {
        perror("connect error");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    write(sockfd, hello, strlen(hello));
    read(sockfd, buf, sizeof(buf));
    printf("Message from server: %s\n", buf);

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

```c
struct sockaddr_in servaddr
```

```c
struct sockaddr_in {
    sa_family_t     sin_fmaily;
    uint16_t        sin_port; // 2Bytes
    struct in_addr  sin_addr; // 4Bytes
    char            sin_zero[8];
}

struct sockaddr {
    u_short     sa_fmaily;
    char        sa_data[14];
}
```

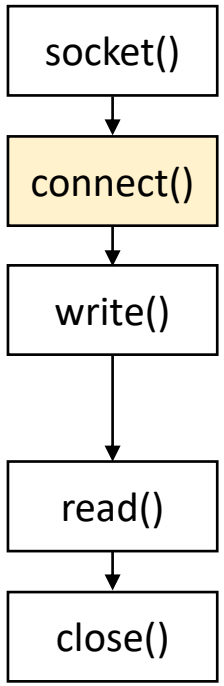
`htons()` : **Host to Network Short**
Network Byte order
(Big Endian)

`atoi()` : **Ascii(char) to Integer**

`inet_addr()` = “192.168.0.1”
→ IPv4 String to Network Byte order

클라이언트

Client




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

int main(int argc, char** argv) {
    int sockfd;
    char buf[1024];
    char *hello = "Hello from client";
    struct sockaddr_in servaddr;

    if(argc < 3) {
        printf("usage:./client remoteAddress remotePort\n");
        return -1;
    }

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket creation failed");
        return -1;
    }

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr(argv[1]);
    servaddr.sin_port = htons(atoi(argv[2]));

    if (connect(sockfd, (const struct sockaddr*)&servaddr, sizeof(servaddr)) < 0) {
        perror("connect error");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    write(sockfd, hello, strlen(hello));
    read(sockfd, buf, sizeof(buf));
    printf("Message from server: %s\n", buf);

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

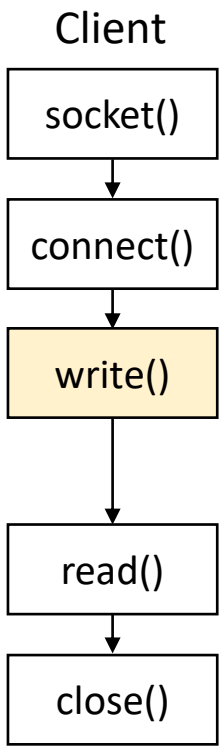
```c
connect(int fd, const struct sockaddr* addr, socklen_t len)
```

fd : **소켓 파일 디스크립터**

addr: **소켓 주소 체계 구조체**

len: **구조체 길이**

클라이언트




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

int main(int argc, char** argv) {
    int sockfd;
    char buf[1024];
    char *hello = "Hello from client";
    struct sockaddr_in servaddr;

    if(argc < 3) {
        printf("usage:./client remoteAddress remotePort\n");
        return -1;
    }

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket creation failed");
        return -1;
    }

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr(argv[1]);
    servaddr.sin_port = htons(atoi(argv[2]));

    if (connect(sockfd, (const struct sockaddr*)&servaddr, sizeof(servaddr)) < 0) {
        perror("connect error");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    write(sockfd, hello, strlen(hello));
    read(sockfd, buf, sizeof(buf));
    printf("Message from server: %s\n", buf);

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

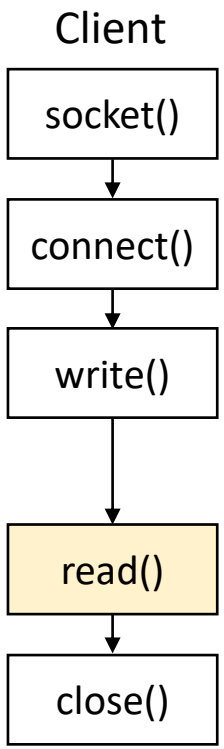
```c
write(int fd, const void* buf, size_t count)
```

fd : **소켓 파일 디스크립터**

buf : **데이터 버퍼**

count : **소켓에 쓸 데이터 크기**

클라이언트




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

int main(int argc, char** argv) {
    int sockfd;
    char buf[1024];
    char *hello = "Hello from client";
    struct sockaddr_in servaddr;

    if(argc < 3) {
        printf("usage:./client remoteAddress remotePort\n");
        return -1;
    }

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket creation failed");
        return -1;
    }

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr(argv[1]);
    servaddr.sin_port = htons(atoi(argv[2]));

    if (connect(sockfd, (const struct sockaddr*)&servaddr, sizeof(servaddr)) < 0) {
        perror("connect error");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    write(sockfd, hello, strlen(hello));
    read(sockfd, buf, sizeof(buf));
    printf("Message from server: %s\n", buf);

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

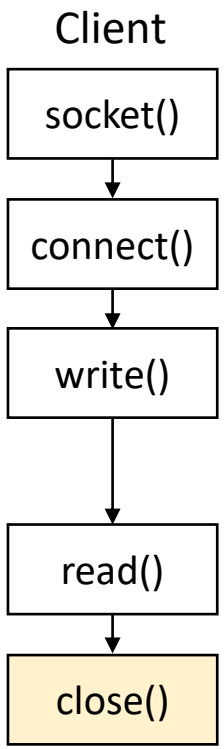
```c
read(int fd, void* buf, size_t count)
```

fd : **소켓 파일 디스크립터**

buf : **데이터 버퍼**

count : **소켓에서 읽어올 데이터 크기**

클라이언트

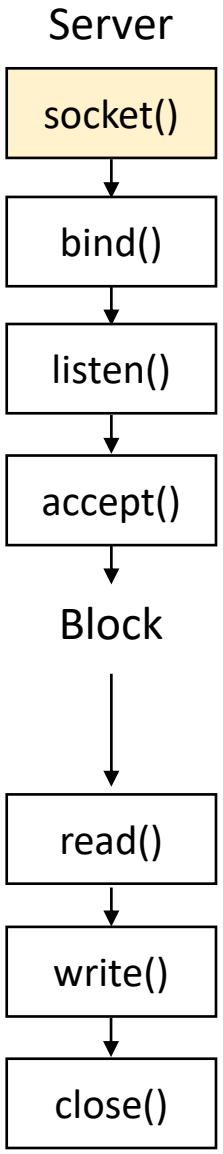



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>

int main(int argc, char** argv) {
    int sockfd;
    char buf[1024];
    char *hello = "Hello from client";
    struct sockaddr_in servaddr;

    if(argc < 3) {
        printf("usage:./client remoteAddress remotePort\n");
        return -1;
    }

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket creation failed");
        return -1;
    }

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr(argv[1]);
    servaddr.sin_port = htons(atoi(argv[2]));

    if (connect(sockfd, (const struct sockaddr*)&servaddr, sizeof(servaddr)) < 0) {
        perror("connect error");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    write(sockfd, hello, strlen(hello));
    read(sockfd, buf, sizeof(buf));
    printf("Message from server: %s\n", buf);

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

```
close(int fd)
```

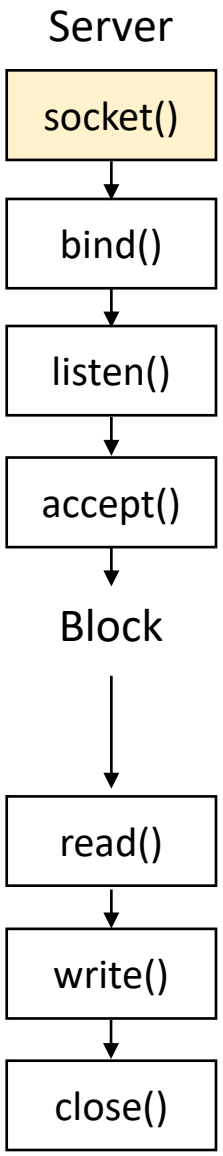
fd : **소켓 파일 디스크립터**

서버




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

int main(int argc, char** argv) {
    int sockfd, cSockfd;
    struct sockaddr_in servaddr, cliaddr;
    char buf[1024];
    socklen_t len;

    if(argc < 2) {
        printf("usage:./server localPort\n");
        return -1;
    }

    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket creation failed");
        return -1;
    }

    int enable = 1;
    setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &enable, sizeof(int));

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    //servaddr.sin_addr.s_addr = 0;
    //servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(atoi(argv[1]));

    if(bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
        perror("bind failed");
        return -1;
    }
```

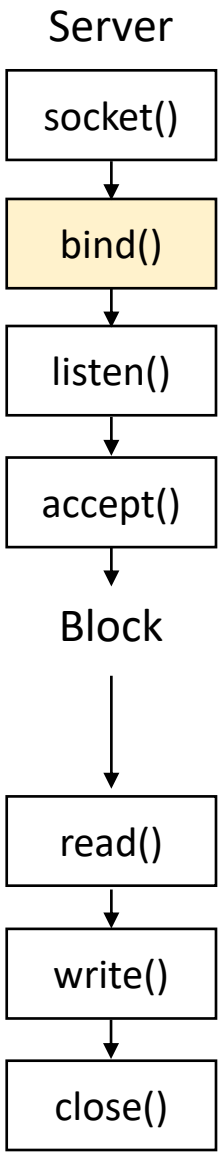
서버




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

int main(int argc, char** argv) {
    int sockfd, cSockfd;
    struct sockaddr_in servaddr, cliaddr;
    char buf[1024];
    socklen_t len;

    if(argc < 2) {
        printf("usage:./server localPort\n");
        return -1;
    }

    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket creation failed");
        return -1;
    }

    int enable = 1;
    setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &enable, sizeof(int));

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    //servaddr.sin_addr.s_addr = 0;
    //servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(atoi(argv[1]));

    if(bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
        perror("bind failed");
        return -1;
    }
```

```
smalldragon@DESKTOP-PMPPMHH:~/Workspace/socket1$ ./server 8080
bind failed: Address already in use
smalldragon@DESKTOP-PMPPMHH:~/Workspace/socket1$ ./server 8080
bind failed: Address already in use
smalldragon@DESKTOP-PMPPMHH:~/Workspace/socket1$ ./server 8080
bind failed: Address already in use
smalldragon@DESKTOP-PMPPMHH:~/Workspace/socket1$ ./server 8080
bind failed: Address already in use
smalldragon@DESKTOP-PMPPMHH:~/Workspace/socket1$ ./server 8080
bind failed: Address already in use
```

서버

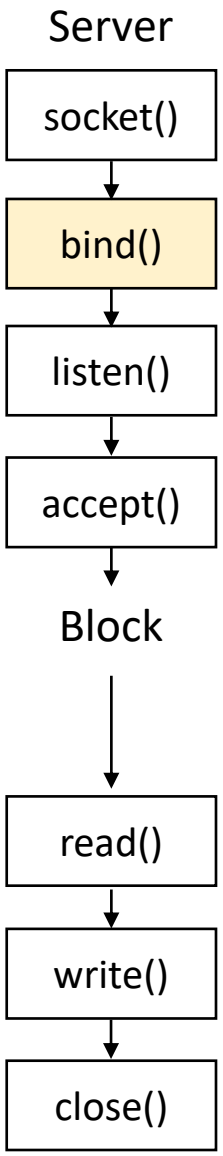



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>

int main(int argc, char** argv) {
    int sockfd, cSockfd;
    struct sockaddr_in servaddr, cliaddr;
    char buf[1024];
    socklen_t len;

    if(argc < 2) {
        printf("usage:./server localPort\n");
        return -1;
    }

    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket creation failed");
        return -1;
    }

    int enable = 1;
    setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &enable, sizeof(int));

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    //servaddr.sin_addr.s_addr = 0;
    //servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(atoi(argv[1]));

    if(bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
        perror("bind failed");
        return -1;
    }
```

```c
struct sockaddr_in servaddr
```

```c
struct sockaddr_in {
    sa_family_t     sin_fmaily;
    uint16_t        sin_port;
    struct in_addr  sin_addr;
    char            sin_zero[8];
}

struct sockaddr {
    u_short     sa_fmaily;
    char        sa_data[14];
}
```

서버

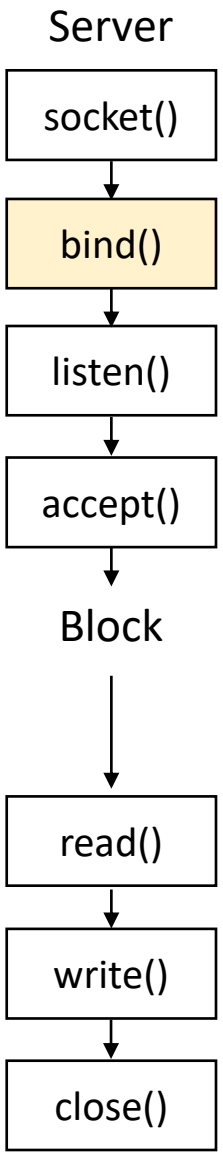



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>

int main(int argc, char** argv) {
    int sockfd, cSockfd;
    struct sockaddr_in servaddr, cliaddr;
    char buf[1024];
    socklen_t len;

    if(argc < 2) {
        printf("usage:./server localPort\n");
        return -1;
    }

    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket creation failed");
        return -1;
    }

    int enable = 1;
    setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &enable, sizeof(int));

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    //servaddr.sin_addr.s_addr = 0;
    //servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(atoi(argv[1]));

    if(bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
        perror("bind failed");
        return -1;
    }
```

`htonl()` : **Host to Network Long**
Network Byte order
(Big Endian)

INADDR_ANY : 0.0.0.0

서버




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

int main(int argc, char** argv) {
    int sockfd, cSockfd;
    struct sockaddr_in servaddr, cliaddr;
    char buf[1024];
    socklen_t len;

    if(argc < 2) {
        printf("usage:./server localPort\n");
        return -1;
    }

    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket creation failed");
        return -1;
    }

    int enable = 1;
    setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &enable, sizeof(int));

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    //servaddr.sin_addr.s_addr = 0;
    //servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(atoi(argv[1]));

    if(bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
        perror("bind failed");
        return -1;
    }
```

```c
bind(int fd, const struct sockaddr* addr, socklen_t len)
```

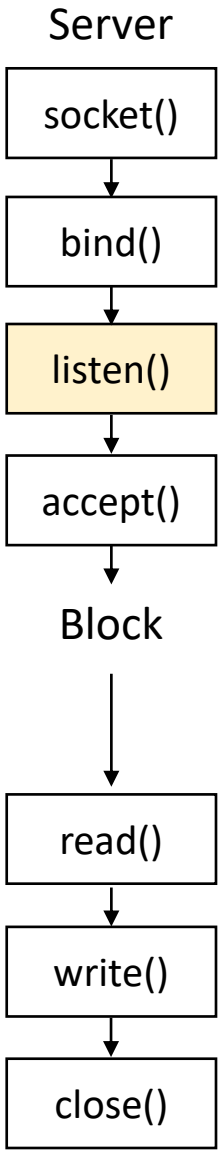
fd : **소켓 파일 디스크립터**

addr: **소켓 주소 구조체**

sockaddr_in → sockaddr

len: **구조체 길이**

서버

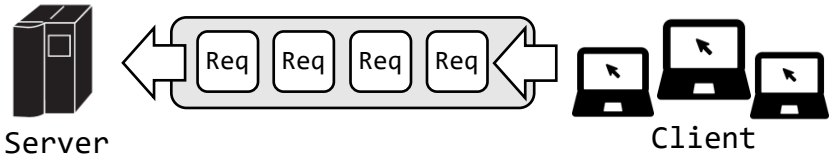



```c
    if(listen(sockfd, 5) < 0) {
        perror("socket failed");
        return -1;
    }

    if((cSockfd = accept(sockfd, (struct sockaddr *)&cliaddr, &len)) < 0) {
        perror("accept error");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    read(cSockfd, buf, sizeof(buf));
    printf("%s\n", buf);

    strcat(buf, " by server");
    write(cSockfd, buf, strlen(buf));

    close(cSockfd);
    close(sockfd);
    return 0;
}
```

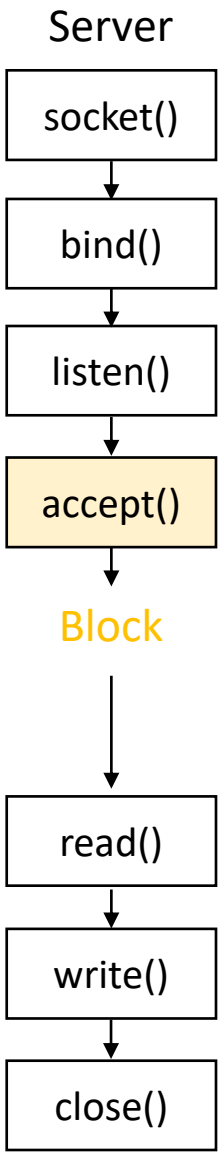
```
listen(int fd, int backlog)
```

fd : **소켓 파일 디스크립터**

backlog: **큐 크기**



서버




```c
    if(listen(sockfd, 5) < 0) {
        perror("socket failed");
        return -1;
    }

    if((cSockfd = accept(sockfd, (struct sockaddr *)&cliaddr, &len)) < 0) {
        perror("accept error");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    read(cSockfd, buf, sizeof(buf));
    printf("%s\n", buf);

    strcat(buf, " by server");
    write(cSockfd, buf, strlen(buf));

    close(cSockfd);
    close(sockfd);
    return 0;
}
```

```c
accept(int fd, struct sockaddr* addr, socklen_t* len)
```

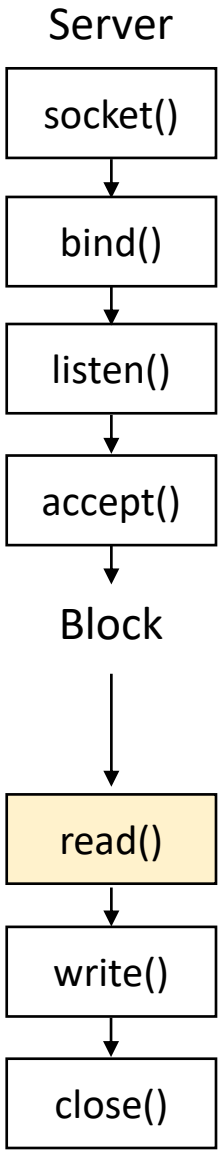
fd : **소켓 파일 디스크립터**

cliaddr: **소켓 주소 구조체**

sockaddr_in → sockaddr

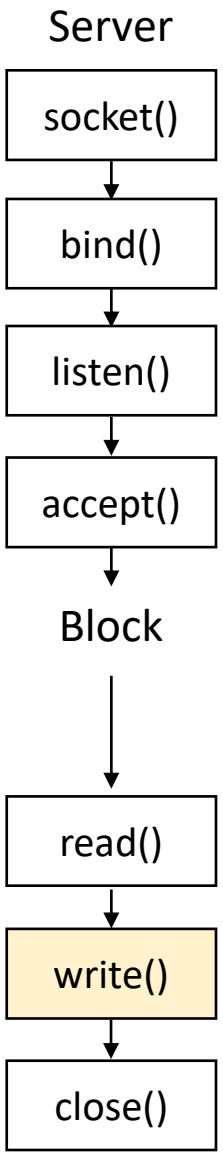
len: **구조체 길이를 받아올 포인터**

서버




```c
    if(listen(sockfd, 5) < 0) {
        perror("socket failed");
        return -1;
    }

    if((cSockfd = accept(sockfd, (struct sockaddr *)&cliaddr, &len)) < 0) {
        perror("accept error");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    read(cSockfd, buf, sizeof(buf));
    printf("%s\n", buf);

    strcat(buf, " by server");
    write(cSockfd, buf, strlen(buf));

    close(cSockfd);
    close(sockfd);
    return 0;
}
```

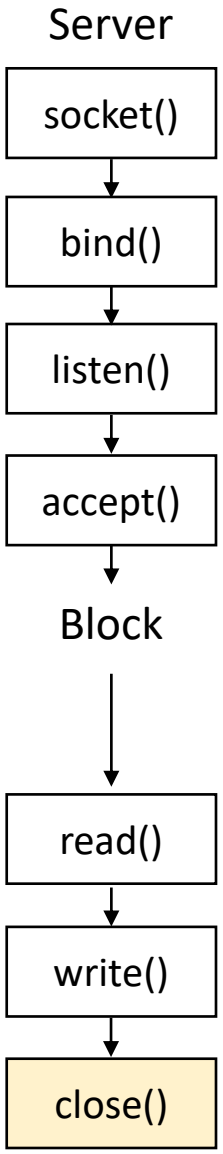
서버




```c
    if(listen(sockfd, 5) < 0) {
        perror("socket failed");
        return -1;
    }

    if((cSockfd = accept(sockfd, (struct sockaddr *)&cliaddr, &len)) < 0) {
        perror("accept error");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    read(cSockfd, buf, sizeof(buf));
    printf("%s\n", buf);

    strcat(buf, " by server");
    write(cSockfd, buf, strlen(buf));

    close(cSockfd);
    close(sockfd);
    return 0;
}
```

```
strcat()  : append string to buf
```

서버




```c
    if(listen(sockfd, 5) < 0) {
        perror("socket failed");
        return -1;
    }

    if((cSockfd = accept(sockfd, (struct sockaddr *)&cliaddr, &len)) < 0) {
        perror("accept error");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    read(cSockfd, buf, sizeof(buf));
    printf("%s\n", buf);

    strcat(buf, " by server");
    write(cSockfd, buf, strlen(buf));

    close(cSockfd);
    close(sockfd);
    return 0;
}
```

실행 결과

Server

```
smalldragon@SD-DESKTOP:~/Workspace/socket/socket1$ gcc server.c -o server
smalldragon@SD-DESKTOP:~/Workspace/socket/socket1$ ./server 9190
Hello from client
smalldragon@SD-DESKTOP:~/Workspace/socket/socket1$
```

Client

```
smalldragon@SD-DESKTOP:~/Workspace/socket/socket1$ gcc client.c -o client
smalldragon@SD-DESKTOP:~/Workspace/socket/socket1$ ./client 127.0.0.1 9190
Message from server: Hello from client by server
smalldragon@SD-DESKTOP:~/Workspace/socket/socket1$
```

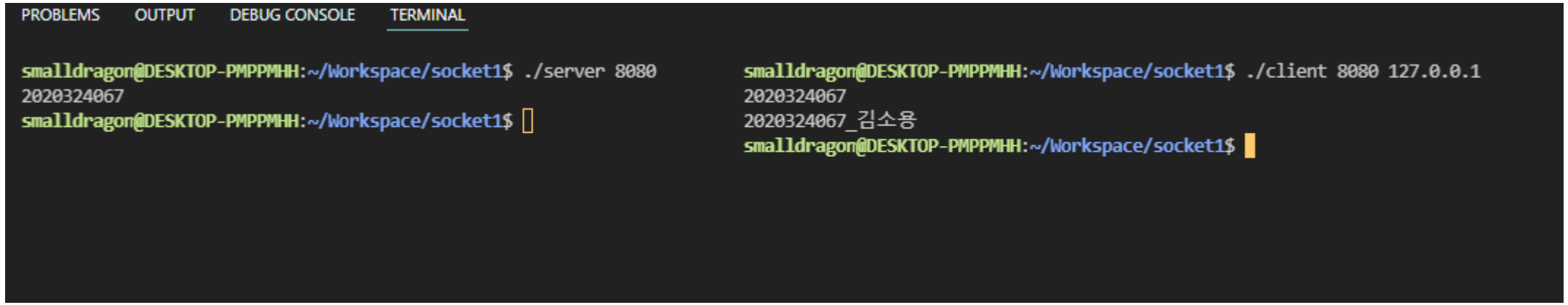
Network Programming Assignment #1

- **server.c**

1. 서버 실행 시 main 함수의 매개변수로 포트번호를 받아서 실행
   Ex) ./server 8080
2. INADDR_ANY와 매개변수로 받은 포트번호로 소켓 바인드
3. 클라이언트의 요청을 대기하다가 학번 정보를 받게 되면 이를 출력
4. 뒤에 자신의 이름을 붙여 클라이언트에게 전송. 형태는 아래와 같음
   Ex) 2020324067_김소용
5. 응답 이후 소켓을 닫고 프로그램 종료

- **client.c**

1. 클라이언트 실행 시 main 함수의 매개변수로 포트번호와 서버의 IP주소 순으로 받아서 실행
   Ex) ./client 8080 127.0.0.1 (순서 확인)
2. 매개변수를 활용하여 서버에게 연결 요청
3. 서버에게 데이터를 전송하기 전에 자신의 학번을 출력
4. 서버에게 자신의 학번을 아래와 같은 형태로 전송
   Ex) 2020324067
5. 서버에게서 받은 데이터를 그대로 출력
6. 소켓을 닫고 프로그램 종료



Network Programming Assignment #1

- 참고사항

1. 서버가 의도한 것 이외의 값을 받는 케이스를 예외 처리할 필요 없음
2. 예시 이외에 다른 쓰레기 값이 출력되면 안됨
3. 과제에서 의도한 대로 데이터를 주고받고 이를 출력하는 방식이 아닌, 겉으로 출력 결과만 똑같이 보인다면 점수 없음

- 제출관련

1. 서버 프로그램은 server.c, 클라이언트 프로그램은 client.c로 명명
2. 빌드 시(gcc) Warning이 발생해서는 안됨
3. 제출 시 두 파일을 "자신의 학번.tar" 파일로 제출

   Ex) 2020324067.tar

   ~/Workspace/socket1/(server.c, client.c)

```
smalldragon@DESKTOP-PMPPMHH:~/Workspace$ tar cvf 2020324067.tar -C socket1 server.c client.c
server.c
client.c
```

압축파일명 / 폴더명 / 파일명 / 파일명

4. 과제는 10점 만점
5. 제출 기한: 2023.03.24(금) PM 11:59
6. 지각 제출 허용: 2023.03.28(화) PM 11:59 / 하루 늦을 때 마다 최종 점수에서 2점 씩 감점

   지각 제출 시 보낼 이메일: minji001011@naver.com
7. 기한 안에 제출을 하지 않으면 점수 없음