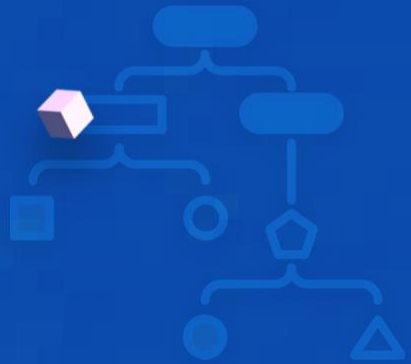
강원혁신플랫폼

리눅스프로그래밍

TCP 서버와 클라이언트 프로그래밍



Linux 네트워크 프로그래밍에서 두 엔드포인트 간에
안정적인 연결 지향 통신 프로토콜은 무엇인가요?

TCP
(Transmission Control Protocol)

학습목표

학습 내용

1. TCP 네트워크 프로그래밍
2. TCP 클라이언트·서버 함수

학습 목표

- TCP 네트워크 프로그래밍에 대해 설명할 수 있다.
- TCP 클라이언트·서버 함수에 대해 설명할 수 있다.

TCP 네트워크 프로그래밍

TCP 네트워크 프로그래밍

- Linux 운영 체제에서 TCP(Transmission Control Protocol) 연결을 통해 통신하는 애플리케이션 개발

TCP

[두 엔드포인트 간에 안정적인 연결 지향 통신]

- 소켓 생성
  - socket() 시스템 호출을 사용하여 TCP 소켓을 생성

서버측 프로그래밍

바인딩

서버 응용 프로그램의 경우 bind() 시스템 호출을 사용하여 소켓을 특정 로컬 IP 주소 및 포트에 바인딩

듣기

소켓이 들어오는 연결을 수락할 수 있도록 하려면 listen() 시스템 호출을 사용

연결 수락

accept() 시스템 호출을 사용하여 들어오는 클라이언트 연결을 수락

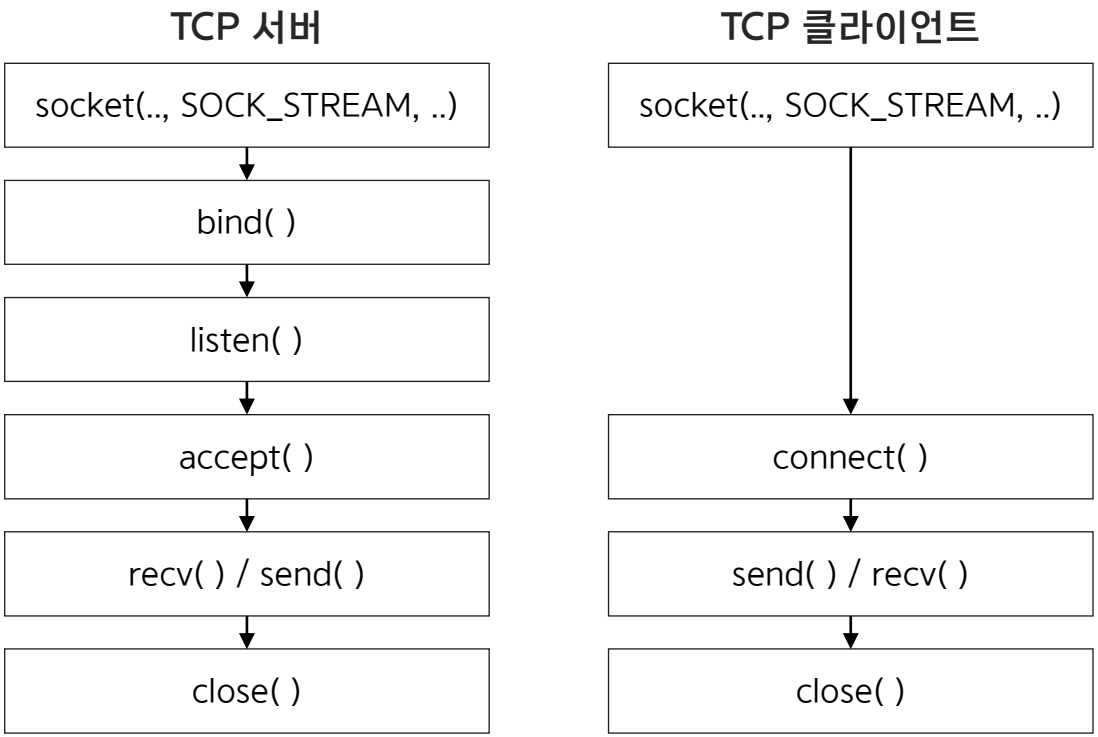
TCP 네트워크 프로그래밍

클라이언트측 프로그래밍

클라이언트 애플리케이션의 경우 connect() 시스템 호출을 사용하여 원격 서버에 연결

close() 시스템 호출을 사용

TCP의 서버와 클라이언트의 통신 설정



출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

TCP 클라이언트·서버 함수

TCP 클라이언트 함수

connect() : 클라이언트 응용 프로그램에서 원격 서버와
TCP 연결을 설정하는 데 사용

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

int connect_result = connect(socket_fd, (struct sockaddr*)&server_addr,
sizeof(server_addr));
```

TCP 클라이언트 함수

recv() : TCP 연결에서 데이터를 수신하는 데 사용

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

char recv_buffer[1024];
int bytes_received = recv(socket_fd, recv_buffer, sizeof(recv_buffer), 0);
```

TCP 클라이언트 함수

send() : TCP 연결을 통해 데이터를 보내는 데 사용

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

char send_buffer[] = "Hello, server!";
int bytes_sent = send(socket_fd, send_buffer, sizeof(send_buffer), 0);
```

TCP 서버 함수

listen() : 서버 응용 프로그램에서 TCP 소켓이 들어오는 연결을 수락

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

int listen(int sock_fd, int backlog)
```

TCP 서버 함수

accept() : 서버 응용 프로그램에서 수신 소켓에서 들어오는 클라이언트 연결 수락

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

struct sockaddr_in client_addr;
socklen_t addr_len = sizeof(client_addr);
int client_socket_fd = accept(socket_fd, (struct sockaddr*)&client_addr, &addr_len);
```

send( )나 recv( ) 함수의 flags 인자

디렉터리	상수	내용	비고
공용	MSG_OOB	긴급(Out-Of-Band) 데이터로 전송	
	MSG_DONTWAIT	넌블록킹(non blocking) 입출력 설정	리눅스 2.2
recv( )	MSG_PEEK	버퍼에 들어온 데이터가 있는지 확인	
	MSG_TRUNC	Raw 소켓이나 UDP 소켓에서 실제 패킷의 길이를 반환	리눅스 2.2
	MSG_ERRQUEUE	소켓 오류 큐에 저장된 오류들을 획득	
	MSG_WAITALL	요청된 조건이 모두 만족할 때까지 대기	
	MSG_CMSG_CLOEXEC	close-on-exec 플래그를 설정해서 exec( )이 수행되면 파일 디스크립터를 닫도록 설정	리눅스 2.6.23

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

send( )나 recv( ) 함수의 flags 인자

디렉터리	상수	내용	비고
send( )	MSG_DONTROUTE	데이터 전송 시 라우팅 테이블을 사용하지 않고 로컬 네트워크 내로 전송	
	MSG_MORE	보낼 데이터가 더 있다는 것을 명시	리눅스 2.4.4
	MSG_NOSIGNAL	소켓이 끊어져도 SIGPIPE 등의 시그널을 보내지 않도록 설정	리눅스 2.2

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

따라하기 tcp_server.c


```
... 상부 생략
#define TCP_PORT 5100                    /* 서버의 포트 번호 */

int main(int argc, char **argv) {
    int ssock;                           /* 소켓 디스크립트 정의 */
    socklen_t clen;
    int n;
    struct sockaddr_in servaddr, cliaddr;       /* 주소 구조체 정의 */
    char mesg[BUFSIZ];
```

따라하기 tcp_server.c


```
/* 서버 소켓 생성 */
if((ssock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    perror("socket()");
    return -1;
}

/* 주소 구조체에 주소 지정 */
memset(&servaddr, 0, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(TCP_PORT);        /* 사용할 포트 지정 */
```

따라하기 tcp_server.c


```
/* bind 함수를 사용하여 서버 소켓의 주소 설정 */
if(bind(ssock, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
    perror("bind()");       return -1;
}
/* 동시에 접속하는 클라이언트의 처리를 위한 대기 큐를 설정 */
if(listen(ssock, 8) < 0) {      perror("listen()");      return -1;    }
clen = sizeof(cliaddr);
do {
    /* 클라이언트가 접속하면 접속을 허용하고 클라이언트 소켓 생성 */
    int n, csock = accept(ssock, (struct sockaddr *)&cliaddr, &clen);
   /* 네트워크 주소를 문자열로 변경 */
    inet_ntop(AF_INET, &cliaddr.sin_addr, mesg, BUFSIZ);
```

```
        printf("Client is connected : %s\n", mesg);
        if((n = read(csock, mesg, BUFSIZ)) <= 0)
            perror("read()");        printf("Received data : %s", mesg);

        /* 클라이언트로 buf에 있는 문자열 전송 */
        if(write(csock, mesg, n) <= 0)
            perror("write()");        close(csock);                /* 클라이언트 소켓을 닫음 */
    } while(strncmp(mesg, "q", 1));

    close(ssock);                    /* 서버 소켓을 닫음 */
    return 0;
}
```

따라하기 tcp_client.c


```
... 상부 헤더 생략
#define TCP_PORT 5100

int main(int argc, char **argv) {
    int ssock;    struct sockaddr_in servaddr;    char mesg[BUFSIZ];

    if(argc < 2) {
        printf("Usage : %s IP_ADRESS\n", argv[0]);        return -1;
    }

    /* 소켓을 생성 */
    if((ssock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
```

따라하기 tcp_client.c


```
    perror("socket()");        return -1;
}

/* 소켓이 접속할 주소 지정 */
memset(&servaddr, 0, sizeof(servaddr));    servaddr.sin_family = AF_INET;

/* 문자열을 네트워크 주소로 변경 */
inet_pton(AF_INET, argv[1], &(servaddr.sin_addr.s_addr));
servaddr.sin_port = htons(TCP_PORT);
```

따라하기 tcp_client.c


```
/* 지정한 주소로 접속 */
    if(connect(ssock, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
        perror("connect()");        return -1;
    }

    fgets(mesg, BUFSIZ, stdin);
    if(send(ssock, mesg, BUFSIZ, MSG_DONTWAIT) <= 0) {  /* 데이터를 소켓에 쓴다. */
        perror("send()");        return -1;
    }

    memset(mesg, 0, BUFSIZ);
```

따라하기 tcp_client.c


```
    if(recv(ssock, mesg, BUFSIZ, 0) <= 0) {          /* 데이터를 소켓으로부터 읽는다. */
        perror("recv()");
        return -1;
    }

    printf("Received data : %s", mesg);              /* 받아온 문자열을 화면에 출력 */

    close(ssock);                                    /* 소켓을 닫는다. */

    return 0;
}
```

따라하기 tcp 실행결과

- 실행결과

```
$ gcc -o tcp_server tcp_server.c
$ gcc -o tcp_client tcp_client.c
$ ./tcp_server
Client is connected : 127.0.0.1
Received data : Hello World
새로운 터미털 창에서
$ ./tcp_client
Usage : ./tcp_client IP_ADRESS
$ ./tcp_client 127.0.0.1
Hello World
Received data : Hello World
$
```

shutdown( ) 함수의 how 인자

- Linux에서 shutdown() 시스템 호출은 소켓을 정상적으로 종료하는 데 사용
- TCP 연결의 제어된 종료 또는 UDP 소켓의 양방향 종료를 허용

shutdown( ) 함수의 how 인자


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

int shutdown(int sockfd, int how);
```

shutdown( ) 함수의 how 인자

상수 값	모드	내용
0	SHUT_RD	입력 스트림을 종료함
1	SHUT_WR	출력 스트림을 종료함
2	SHUT_RDWR	입/출력 스트림을 종료함

출처 서영진, 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이, 제이펍, 2020

요약정리

01 • TCP 네트워크 프로그래밍

02 • TCP 클라이언트·서버 함수