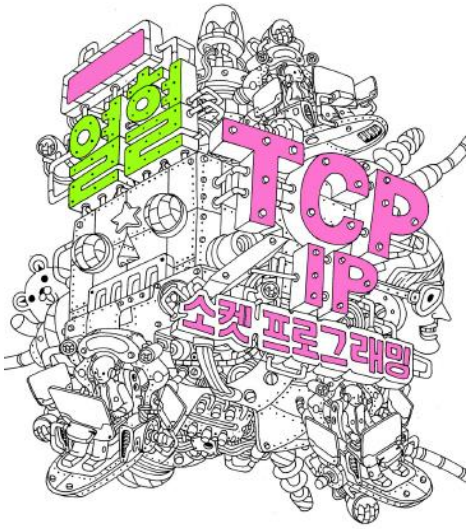

# 윤성우의 열혈 TCP/IP 소켓 프로그래밍

## 윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

## Chapter 02. 소켓의 타입과 프로토콜의 설정




# Chapter 02-1. 소켓의 프로토콜과 그에 따른 데이터 전송 특성

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

# 프로토콜의 이해와 소켓의 생성


- 프로토콜이란?
  - 개념적으로 약속의 의미를 담고 있다.
  - 컴퓨터 상호간의 데이터 송수신에 필요한 통신규약.
  - 소켓을 생성할 때 기본적인 프로토콜을 지정해야 한다.

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

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

➜ 성공 시 파일 디스크립터, 실패 시 -1 반환

- domain 소켓이 사용할 프로토콜 체계(Protocol Family) 정보 전달.
- type 소켓의 데이터 전송방식에 대한 정보 전달.
- protocol 두 컴퓨터간 통신에 사용되는 프로토콜 정보 전달.

매개변수 domain, type 그리고 protocol이 모두 프로토콜 정보와 관련이 있다.

# 프로토콜 체계(Protocol Family)

- 프로토콜 체계
  - 프로토콜도 그 종류에 따라서 부류가 나뉘는데, 그 부류를 가리켜 프로토콜 체계라 한다.
  - 프로토콜의 체계 PF_INET은 IPv4 인터넷 프로토콜 체계를 의미한다. 우리는 이를 기반으로 소켓 프로그래밍을 학습한다.

| 이름 | 프로토콜 체계(Protocol Family) |
|---|---|
| PF_INET | IPv4 인터넷 프로토콜 체계 |
| PF_INET6 | IPv6 인터넷 프로토콜 체계 |
| PF_LOCAL | 로컬 통신을 위한 UNIX 프로토콜 체계 |
| PF_PACKET | Low Level 소켓을 위한 프로토콜 체계 |
| PF_IPX | IPX 노벨 프로토콜 체계 |

대표적인 프로토콜 체계 정보

# 소켓의 타입(Type)

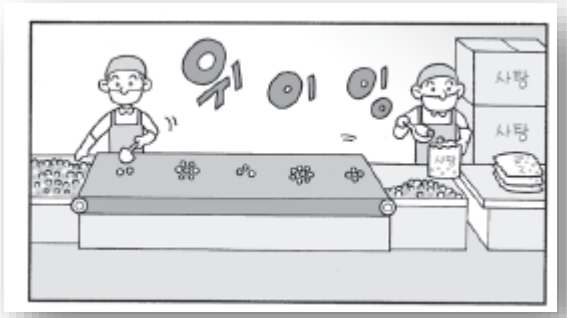

- 소켓의 타입
  - 데이터 전송방식을 의미함.
  - 소켓이 생성될 때 소켓의 타입도 결정되어야 한다.

- 프로토콜 체계 PF_INET의 대표적인 소켓 타입 둘
  - 연결 지향형 소켓 타입
  - 비 연결 지향형 소켓 타입.

# 두 타입의 소켓

TCP 소켓

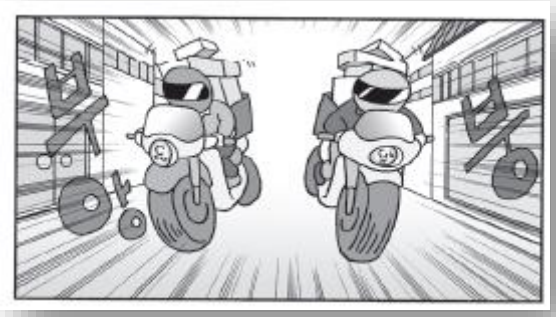
- 연결지향형 소켓(SOCK_STREAM)의 데이터 전송특성
  - 중간에 데이터 소멸되지 않는다.
  - 전송 순서대로 데이터가 수신된다.
  - 데이터의 경계가 존재하지 않는다.
  - 소켓 대 소켓의 연결은 반드시 1대 1의 구조.



TCP 데이터 전송특성

UDP 소켓

- 비 연결지향형 소켓(SOCK_DGRAM)의 데이터 전송특성
  - 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

# 프로토콜의 최종선택!

IPv4 인터넷 프로토콜 체계에서 동작하는 연결지향형 데이터 전송 소켓

```
int tcp_socket=socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
```

TCP 소켓

IPv4 인터넷 프로토콜 체계에서 동작하는 비 연결지향형 데이터 전송 소켓

```
int udp_socket=socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP);
```

UDP 소켓

첫 번째, 두 번째 인자로 전달된 정보를 통해서 소켓의 프로토콜이 사실상 결정되기 때문에 세 번째 인자로 0을 전달해도 된다!

# 연결지향형 소켓! TCP 소켓의 예


전송되는 데이터의 경계(boundary)가 존재하지 않음을 확인하자!

```
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);
```

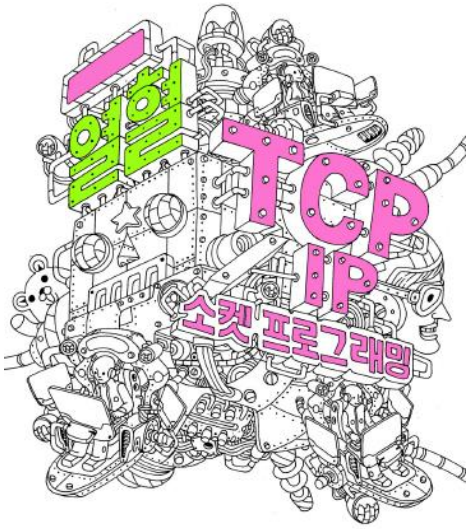
tcp_server.c의 데이터 전송 :
한 번에 전송

```
if(connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
    error_handling("connect() error!");
while(read_len=read(sock, &message[idx++], 1))
{
    if(read_len==-1)
    {
        error_handling("read() error!");
        break;
    }
    str_len+=read_len;
}
printf("Message from server: %s \n", message);
printf("Function read call count: %d \n", str_len);
```

실행결과

```
root@my_linux:/tcpip# gcc tcp_client.c -o hclient
root@my_linux:/tcpip# ./hclient 127.0.0.1 9190
Message from server: Hello World!
Function read call count: 13
```

tcp_client.c의 데이터 수신 :
1바이트 단위로 반복 수신




# Chapter 02-2. 윈도우 기반에서 이해 및 확인하기

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

# 윈도우 운영체제의 socket 함수

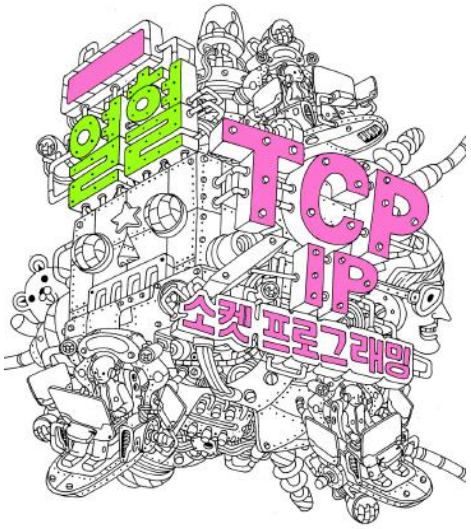
```
#include <winsock2.h>

SOCKET socket(int af, int type, int protocol);
```

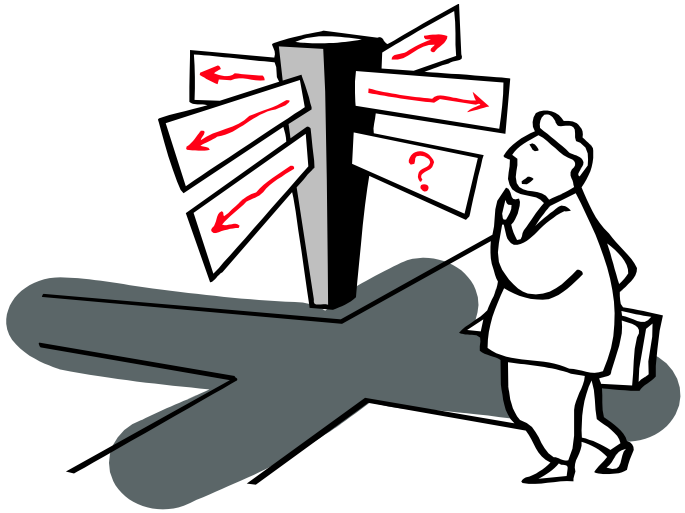
➜ 성공 시 소켓 핸들, 실패 시 INVALID_SOCKET 반환

```
SOCKET soc=socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
if(soc==INVALID_SOCKET)
    ErrorHandling(". . . ");
```

윈도우 소켓 생성의 예

프로토콜은 표준이다! 따라서 소켓의 타입에 따른 데이터의 전송특성은 운영체제와 상관없이 동일하다. 예제 tcp_server_win.c와 tcp_client_win.c의 실행을 통해서 이를 확인할 수 있다.


Chapter 02가 끝났습니다. 질문 있으신지요?