

[3주차_2차시] 4장 TCP 서버-클라이언트 소켓 함수

서버 소켓 함수 프로토타입과 함수 구성 요소 (교재 page 56)

socket()

- 역할 : listen socket 생성
- 함수 반환값 : SOCKET

```
원도우 #include <winsock2.h>
        SOCKET socket(
            int af,          // 주소 체계 지정
            int type,        // 소켓 타입 지정
            int protocol     // 사용할 프로토콜 지정
        );
                                     성공: 새로운 소켓, 실패: INVALID_SOCKET
```

```
원도우 #include <winsock2.h>
        SOCKET socket(
            int af,          // 주소 체계 지정
            int type,        // 소켓 타입 지정
            int protocol     // 사용할 프로토콜 지정
        );
                                     성공: 새로운 소켓, 실패: INVALID_SOCKET
```

bind()

- 역할 : listen socket과 port (서버의 서비스) 연결
- 함수 반환값 : int

원도우

```
#include <winsock2.h>
int bind(
    ❶ SOCKET sock,
    ❷ const struct sockaddr *addr,
    ❸ int addrlen
);
```

성공: 0, 실패: SOCKET_ERROR

listen()

- 역할 : listen socket의 최대 허용 클라이언트 정의
- 함수 반환값 : int

원도우

```
#include <winsock2.h>
int listen(
    ❶ SOCKET sock,
    ❷ int backlog
);
```

성공: 0, 실패: SOCKET_ERROR

accept()

- 역할 : listen socket을 통해 접속한 클라이언트 소켓 생성
- 함수 반환값 : SOCKET

원도우

```
#include <winsock2.h>
SOCKET accept(
    ❶ SOCKET sock,
    ❷ struct sockaddr *addr,
    ❸ int *addrlen
);
```

성공: 새로운 소켓, 실패: INVALID_SOCKET

클라이언트 소켓 함수 프로토타입과 함수 구성 요소

connect()

- 역할 : 서버와 연결
- 함수 반환값 : int

원도우

```
#include <winsock2.h>
int connect(
    ❶ SOCKET sock,
    ❷ const struct sockaddr *addr,
    ❸ int addrlen
);
```

성공: 0, 실패: SOCKET_ERROR

데이터 전송 소켓 함수 프로토타입과 함수 구성 요소

send ()

- 역할 : 데이터 전송
- 함수 반환값 : int

원도우

```
#include <winsock2.h>
int send(
    ❶ SOCKET sock,
    ❷ const char *buf,
    ❸ int len,
    ❹ int flags
);
```

성공: 보낸 바이트 수, 실패: SOCKET_ERROR

recv ()

- 역할 : 데이터 수신
- 함수 반환값 : int

원도우

```
#include <winsock2.h>
int recv(
    ❶ SOCKET sock,
    ❷ char *buf,
    ❸ int len,
    ❹ int flags
);
```

성공: 받은 바이트 수 또는 0(연결 종료 시), 실패: SOCKET_ERROR

bind() 함수

- 0.0.0.0은 "모든 IP(와일드카드)"를 뜻함
- IPv4에서 0.0.0.0은 **INADDR_ANY**(값 0)로, "이 머신의 모든 인터페이스로 오는 연결/패킷을 받겠다"는 의미.
- 주소를 만드는 게 아니라, 바인드할 "대상"을 와일드카드로 지정

IPv4

```
struct sockaddr_in addr;  
memset(&addr, 0, sizeof(addr));  
addr.sin_family = AF_INET;  
addr.sin_port = htons(8080);           // 원하는 포트  
addr.sin_addr.s_addr = htonl(INADDR_ANY); // == 0, 모든 인터페이스  
bind(fd, (struct sockaddr*)&addr, sizeof(addr));
```

IPv6

```
struct sockaddr_in6 a6 = {0};  
a6.sin6_family = AF_INET6;  
a6.sin6_port = htons(8080);  
a6.sin6_addr = in6addr_any; // "::" (IPv6 와일드카드)  
bind(fd, (struct sockaddr*)&a6, sizeof(a6));
```

INADDR_ANY

- IPv4에서 값 0(주소 0.0.0.0)을 뜻하는 매크로
 - **INADDR_ANY** = "모든 인터페이스로 바인드" (0.0.0.0).
 - **bind()** 에 넣으면 이 머신의 모든 인터페이스(모든 NIC)의 해당 포트에 들어오는 트래픽을 받겠다는 의미
- 사용

```

struct sockaddr_in addr = {0};
addr.sin_family = AF_INET;
addr.sin_port = htons(8080);
addr.sin_addr.s_addr = htonl(INADDR_ANY); // == 0.0.0.0
bind(fd, (struct sockaddr*)&addr, sizeof(addr));

```

- `INADDR_ANY` 는 0이라 `htonl()` 유무와 상관없지만, 적어두면 안전

• 효과

- **TCP**: 해당 포트로 오는 모든 로컬 IP에 대한 연결 수락.
- **UDP**: 해당 포트로, 모든 로컬 IP로 도착한 datagram을 수신.
- 방화벽이 열려 있으면 **외부에서도 접근 가능**
- 로컬에서만 받으려면 `127.0.0.1` (loopback)로 바인드.
- 특정 NIC로만 받으려면 그 **특정 IP**로 바인드.
- **IPv6 대응**: `in6addr_any` (주소 `::`)

```

struct sockaddr_in6 a6 = {0};
a6.sin6_family = AF_INET6;
a6.sin6_port = htons(8080);
a6.sin6_addr = in6addr_any; // ::

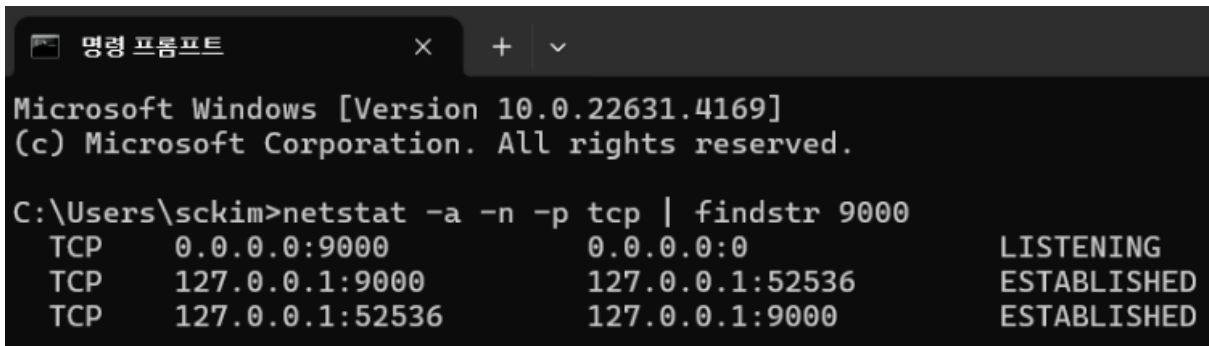
```

- `INADDR_ANY` : 0으로 정의
 - `memset()` 함수는 `serveraddr` 변수 메모리 영역을 0으로 초기화함
 - `serveraddr` 변수 초기화
 - `sin_family`에는 `AF_INET` 값
 - `s_addr`에는 `INADDR_ANY` (0)
 - `s_port`는 9000 으로

```

19 // bind()
20 struct sockaddr_in serveraddr;
21 memset(&serveraddr, 0, sizeof(serveraddr));
22 serveraddr.sin_family = AF_INET;
23 serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
24 serveraddr.sin_port = htons(SERVERPORT);
25 retval = bind(listen_sock, (struct sockaddr *)&serveraddr, sizeof(serveraddr));
26 if (retval == SOCKET_ERROR) err_quit("bind()");

```



The screenshot shows a Windows Command Prompt window titled "명령 프롬프트". The output of the command `netstat -a -n -p tcp | findstr 9000` is displayed, showing three TCP connections in a listening or established state on port 9000.

```

Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sckim>netstat -a -n -p tcp | findstr 9000
TCP        0.0.0.0:9000          0.0.0.0:0             LISTENING
TCP        127.0.0.1:9000        127.0.0.1:52536       ESTABLISHED
TCP        127.0.0.1:52536      127.0.0.1:9000        ESTABLISHED

```

TCPServer

```

#include "..\Common.h"

#define SERVERPORT 9000
#define BUFSIZE 512

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

    // 윈속 초기화
    WSADATA wsa;
    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
        return 1;

    // 소켓 생성

```

```

SOCKET listen_sock = socket(AF_INET, SOCK_STREAM, 0);
if (listen_sock == INVALID_SOCKET) err_quit("socket()");

// bind()
struct sockaddr_in serveraddr;
memset(&serveraddr, 0, sizeof(serveraddr));
serveraddr.sin_family = AF_INET;
serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
serveraddr.sin_port = htons(SERVERPORT);
retval = bind(listen_sock, (struct sockaddr*)&serveraddr, sizeof(serverad
dr));
if (retval == SOCKET_ERROR) err_quit("bind()");

// listen()
retval = listen(listen_sock, SOMAXCONN);
if (retval == SOCKET_ERROR) err_quit("listen()");

// 데이터 통신에 사용할 변수
SOCKET client_sock;
struct sockaddr_in clientaddr;
int addrlen;
char buf[BUFSIZE + 1];

while (1) {
    // accept()
    addrlen = sizeof(clientaddr);
    client_sock = accept(listen_sock, (struct sockaddr*)&clientaddr, &addr
len);
    if (client_sock == INVALID_SOCKET) {
        err_display("accept()");
        break;
    }

    // 접속한 클라이언트 정보 출력
    char addr[INET_ADDRSTRLEN];
    inet_ntop(AF_INET, &clientaddr.sin_addr, addr, sizeof(addr));
    printf("\n[TCP 서버] 클라이언트 접속: IP 주소=%s, 포트 번호=%d\n",
        addr, ntohs(clientaddr.sin_port));
}

```

```

// 클라이언트와 데이터 통신
while (1) {
    // 데이터 받기
    retval = recv(client_sock, buf, BUFSIZE, 0);
    if (retval == SOCKET_ERROR) {
        err_display("recv()");
        break;
    }
    else if (retval == 0)
        break;

    // 받은 데이터 출력
    buf[retval] = '\0';
    printf("[TCP/%s:%d] %s\n", addr, ntohs(clientaddr.sin_port), buf);

    // 데이터 보내기
    retval = send(client_sock, buf, retval, 0);
    if (retval == SOCKET_ERROR) {
        err_display("send()");
        break;
    }
}

// 소켓 닫기
closesocket(client_sock);
printf("[TCP 서버] 클라이언트 종료: IP 주소=%s, 포트 번호=%d\n",
    addr, ntohs(clientaddr.sin_port));
}

// 소켓 닫기
closesocket(listen_sock);

// 윈속 종료
WSACleanup();
return 0;
}

```


TCPClient

```
#include "..\\Common.h"

char* SERVERIP = (char*)"127.0.0.1";
#define SERVERPORT 9000
#define BUFSIZE 512

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

    // 명령행 인수가 있으면 IP 주소로 사용
    if (argc > 1) SERVERIP = argv[1];

    // 윈속 초기화
    WSADATA wsa;
    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
        return 1;

    // 소켓 생성
    SOCKET sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == INVALID_SOCKET) err_quit("socket()");

    // connect()
    struct sockaddr_in serveraddr;
    memset(&serveraddr, 0, sizeof(serveraddr));
    serveraddr.sin_family = AF_INET;
    inet_pton(AF_INET, SERVERIP, &serveraddr.sin_addr);
    serveraddr.sin_port = htons(SERVERPORT);
    retval = connect(sock, (struct sockaddr*)&serveraddr, sizeof(serveraddr));
    if (retval == SOCKET_ERROR) err_quit("connect()");

    // 데이터 통신에 사용할 변수
    char buf[BUFSIZE + 1];
    int len;
```

```

// 서버와 데이터 통신
while (1) {
    // 데이터 입력
    printf("\n[보낼 데이터] ");
    if (fgets(buf, BUFSIZE + 1, stdin) == NULL)
        break;

    // '\n' 문자 제거
    len = (int)strlen(buf);
    if (buf[len - 1] == '\n')
        buf[len - 1] = '\0';
    if (strlen(buf) == 0)
        break;

    // 데이터 보내기
    retval = send(sock, buf, (int)strlen(buf), 0);
    if (retval == SOCKET_ERROR) {
        err_display("send()");
        break;
    }
    printf("[TCP 클라이언트] %d바이트를 보냈습니다.\n", retval);

    // 데이터 받기
    retval = recv(sock, buf, retval, MSG_WAITALL);
    if (retval == SOCKET_ERROR) {
        err_display("recv()");
        break;
    }
    else if (retval == 0)
        break;

    // 받은 데이터 출력
    buf[retval] = '\0';
    printf("[TCP 클라이언트] %d바이트를 받았습니다.\n", retval);
    printf("[받은 데이터] %s\n", buf);
}

// 소켓 닫기

```

```

    closesocket(sock);

    // 윈속 종료
    WSACleanup();
    return 0;
}

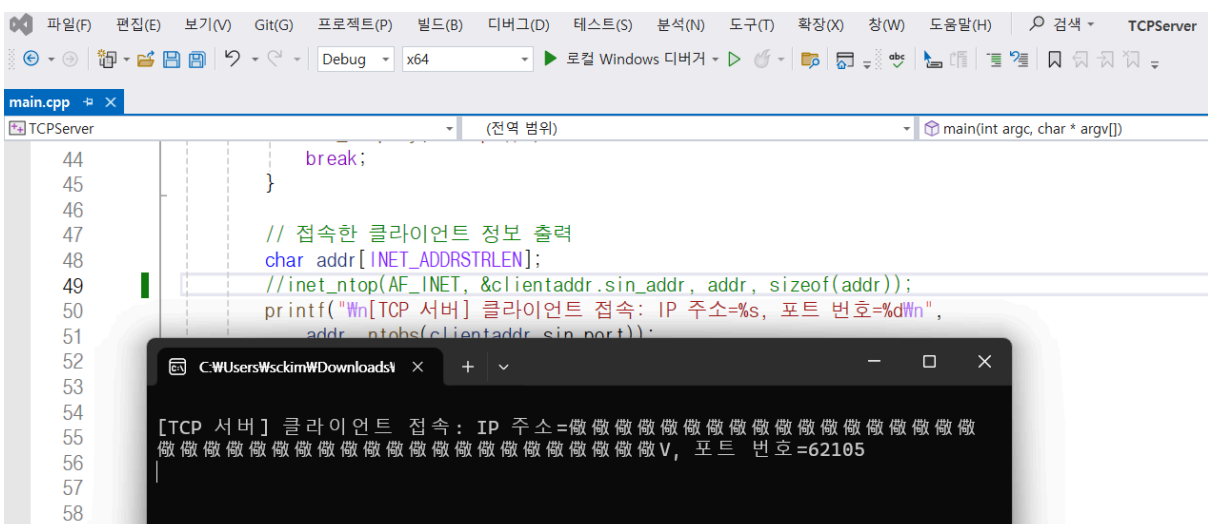
```

inet_ntop() 함수

TCPServer에서 클라이언트에 대해 inet_ntop() 함수를 사용하지 않을 때

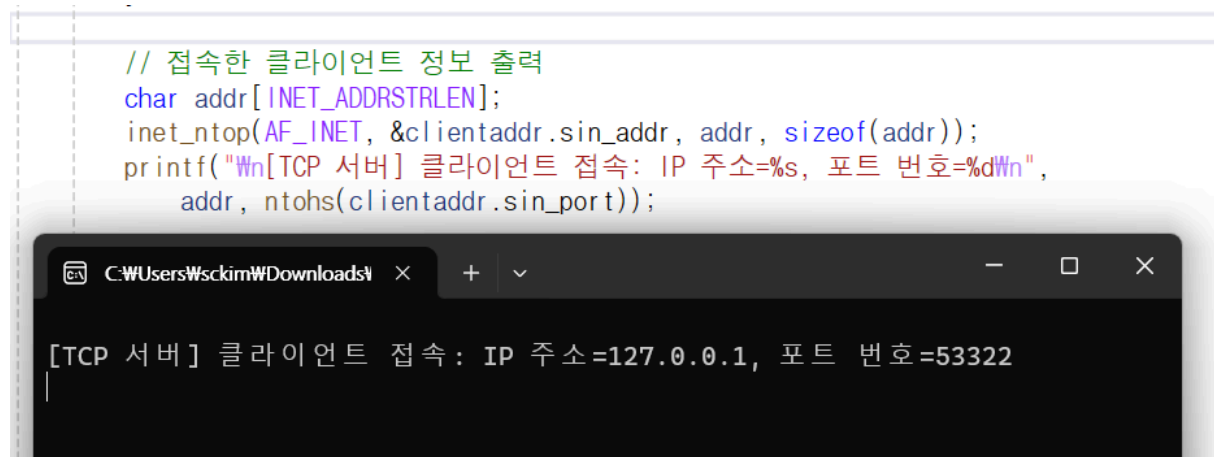
- [illegible]

```
// 접속한 클라이언트 정보 출력
char addr[INET_ADDRSTRLEN];
//inet_ntop(AF_INET, &clientaddr.sin_addr, addr, sizeof(addr));
printf("\n[TCP 서버] 클라이언트 접속: IP 주소=%s, 포트 번호=%d\n",
        addr, ntohs(clientaddr.sin_port));
```



TCPServer에서 클라이언트에 대해 inet_ntop() 함수를 사용할 때

- 네트워크 바이트 정렬된 IP 주소가 서버에서 호스트 바이트로 해석되어 제대로 된 IP 주소가 나타남



ntoh()함수

```
// 접속한 클라이언트 정보 출력
char addr[INET_ADDRSTRLEN];
inet_ntop(AF_INET, &clientaddr.sin_addr, addr, sizeof(addr));
printf("\n[TCP 서버] 클라이언트 접속: IP 주소=%s, 포트 번호=%d\n",
      addr, ntohs(clientaddr.sin_port));
```

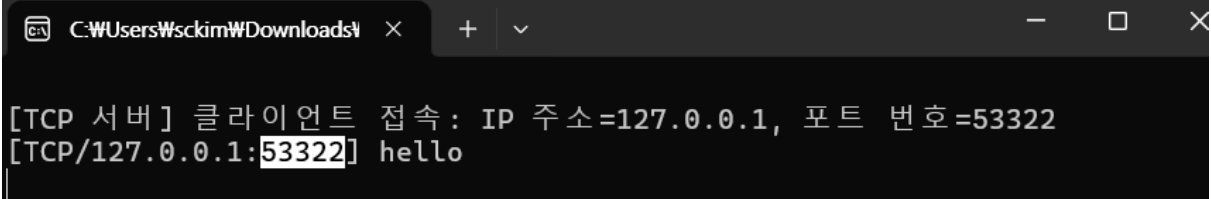
- `char addr[INET_ADDRSTRLEN];`
 - IPv4 문자열을 담을 버퍼 생성
 - `INET_ADDRSTRLEN` 은 **16**(예: `"255.255.255.255\0"`)이라 넉넉함
- `inet_ntop(AF_INET, &clientaddr.sin_addr, addr, sizeof(addr));`
 - `accept()` 가 채워준 `clientaddr.sin_addr` (이진 IPv4 주소)를 가독성 있는 점표기 문자열로 바꿔 `addr` 에 저장.
 - 첫 인자 `AF_INET` → IPv4임을 명시
 - 성공 시 `addr` 이 채워지고, 실패 시 `NULL` 을 반환(에러 체크 가능).
- `printf(..., addr, ntohs(clientaddr.sin_port));`

- 방금 만든 IP 문자열과, 네트워크 바이트 오더(big-endian)인 포트를 **호스트 오더**로 바꾼 값(`ntohs`)을 출력.

서버측 화면

- `ntoh()` 사용할 때,
- 포트 번호 = **53322 (`ntoh()` 사용함)**와 접속한 클라이언트 정보 TCP/127.0.0.1: **53322 (`ntoh()` 사용함)** 포트 번호가 같음

```
// 접속한 클라이언트 정보 출력
char addr[INET_ADDRSTRLEN];
inet_ntop(AF_INET, &clientaddr.sin_addr, addr, sizeof(addr));
printf("[TCP 서버] 클라이언트 접속: IP 주소=%s, 포트 번호=%d\n",
       addr, ntohs(clientaddr.sin_port));
```



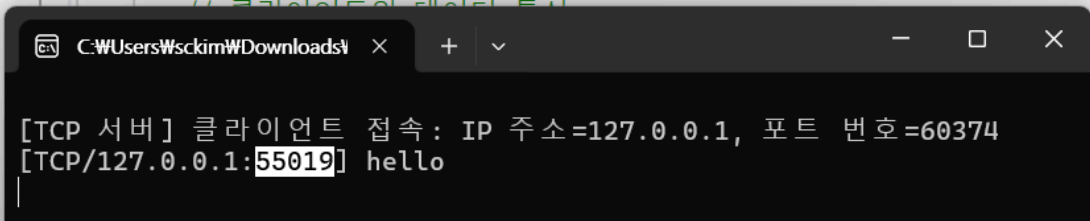
```
C:\Users\Wskim\Downloads\ >
[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=53322
[TCP/127.0.0.1:53322] hello
```

```
// 받은 데이터 출력
buf[retval] = '\0';
printf("[TCP/%s:%d] %s\n", addr, ntohs(clientaddr.sin_port), buf);
```

서버측 화면

- `ntoh()` 사용하지 않을 때,
- 포트 번호 = **60374 (`ntoh()` 사용하지 않음)**와 접속한 클라이언트 정보 TCP/127.0.0.1: **55019 (`ntoh()` 사용함)** 포트 번호가 다름

```
// 접속한 클라이언트 정보 출력
char addr[INET_ADDRSTRLEN];
inet_ntop(AF_INET, &clientaddr.sin_addr, addr, sizeof(addr));
printf("\n[TCP 서버] 클라이언트 접속: IP 주소=%s, 포트 번호=%d\n",
        addr, (clientaddr.sin_port));
```



```
[TCP 서버] 클라이언트 접속: IP 주소=127.0.0.1, 포트 번호=60374
[TCP/127.0.0.1:55019] hello
```

```
// 받은 데이터 출력
buf[retval] = '\0';
printf("[TCP/%s:%d] %s\n", addr, ntohs(clientaddr.sin_port), buf);
```

검토

- ntohs() 함수 적용 전
 - 포트 번호 십진수 : **60374**
 - 포트 번호 16진수 : **EBD6**
- 십진수 **60374** → 16진수 변환 : **EBD6**
 - 16진수 **EBD6** 는 ntohs() 함수 적용전임으로 네트워크 바이트 정렬된 16진수 값임
 - **EBD6** 를 호스트 바이트 정렬하면 **D6EB** 임
 - 16진수 **D6EB** → 10진수 **55,019** 됨
 - 바이트 정렬된 포트 번호는 **55,019** 임

