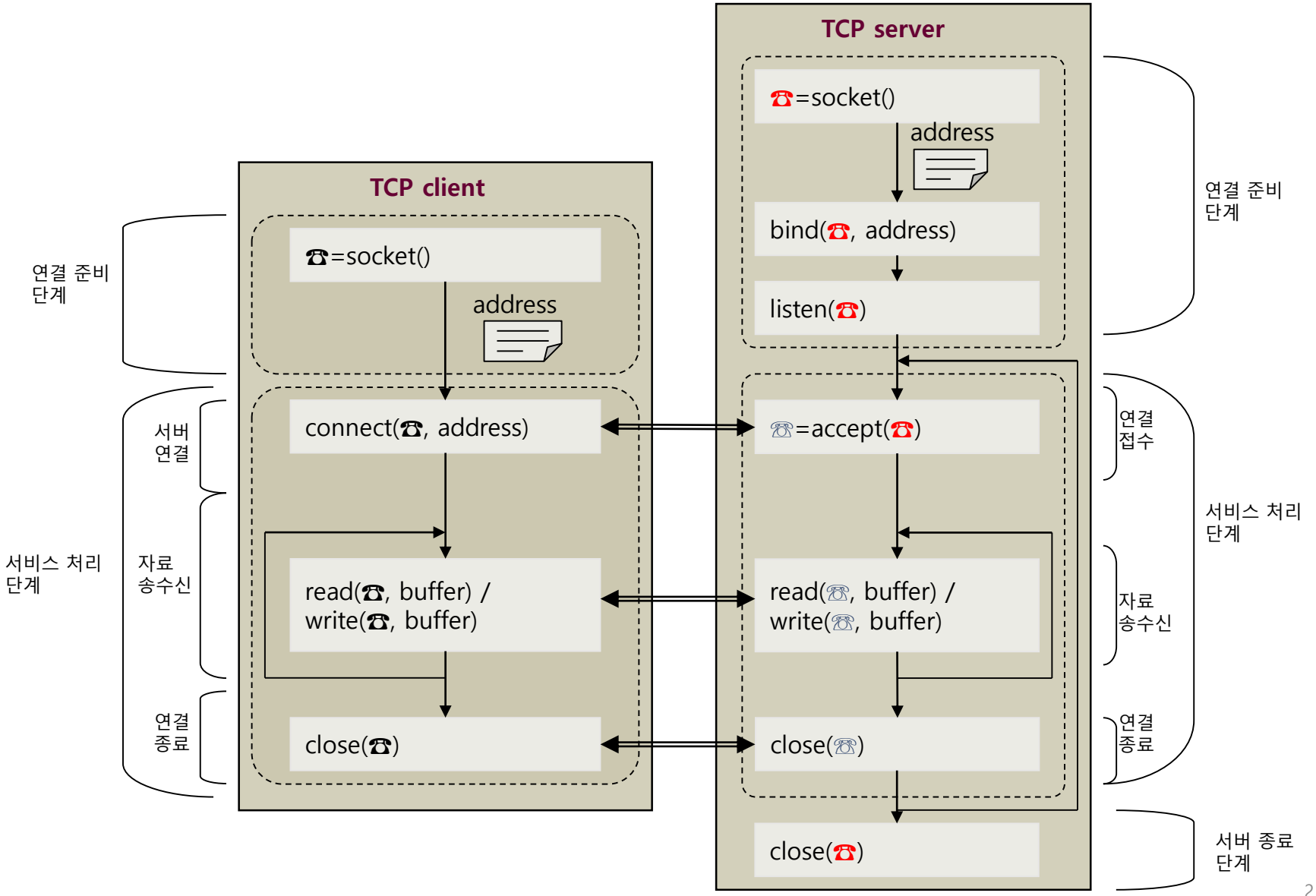
C언어로 구현하는 TCP/IP 소켓 프로그래밍

TCP 소켓 프로그래밍


TCP server
☎=socket()
address
bind(☎, address)
listen(☎)
TCP client
☎=socket()
address
connect(☎, address)
☎=accept(☎)
read(☎, buffer) /
write(☎, buffer)
read(☎, buffer) /
write(☎, buffer)
close(☎)
close(☎)
close(☎)
연결 준비
단계
서버
연결
서비스 처리
단계
자료
송수신
연결
종료
연결 준비
단계
연결
접수
서비스 처리
단계
자료
송수신
연결
종료
서버 종료
단계

hello, world 출력 소켓 프로그램 / 클라이언트

```
1.  #include <unistd.h>
2.  #include <stdio.h>
3.  #include <string.h>
4.  #include <arpa/inet.h>
5.  #include <sys/socket.h>

6.  #define PORT 9001
7.  #define IPADDR "127.0.0.1"

8.  int main(void){

9.      int c_socket;
10.     struct sockaddr_in c_addr;
11.     int len;
12.     int n;

13.     char rcvBuffer[BUFSIZ];

14.     c_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);   ①
15.
16.     memset(&c_addr, 0, sizeof(c_addr));
17.     c_addr.sin_addr.s_addr = inet_addr(IPADDR);
18.     c_addr.sin_family = AF_INET;
19.     c_addr.sin_port = htons(PORT);                          ②

20.     if(connect(c_socket, (struct sockaddr*)&c_addr, sizeof(c_addr)) == -1){   ③
21.         printf("Can not connect ₩n");
22.         perror("Error:");
23.         close(c_socket);
24.         return -1;
25.     }

26.     if((n = read(c_socket, rcvBuffer, sizeof(rcvBuffer))) < 0){   ④
27.         return -1;
28.     }

29.     rcvBuffer[n] = '₩0';                                    ⑤
30.     printf("received Data : %s₩n", rcvBuffer);

31.     close(c_socket);                                        ⑥

32.     return 0;
33. }
```

① **소켓을 생성**

② **연결할 서버의 주소 설정**

③ **소켓을 서버에 연결**

④, ⑤ **서비스 요청과 처리**

⑥ **소켓 연결을 종료**

socket()을 이용한 소켓 생성

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

- 인터넷과 연결하기 위한 접점인 소켓(endpoint socket) 생성
  - domain : 소켓의 사용 영역을 정의
  - type : 소켓 유형을 정의
  - protocol : 소켓이 사용할 프로토콜을 정의

socket()을 이용한 소켓 생성

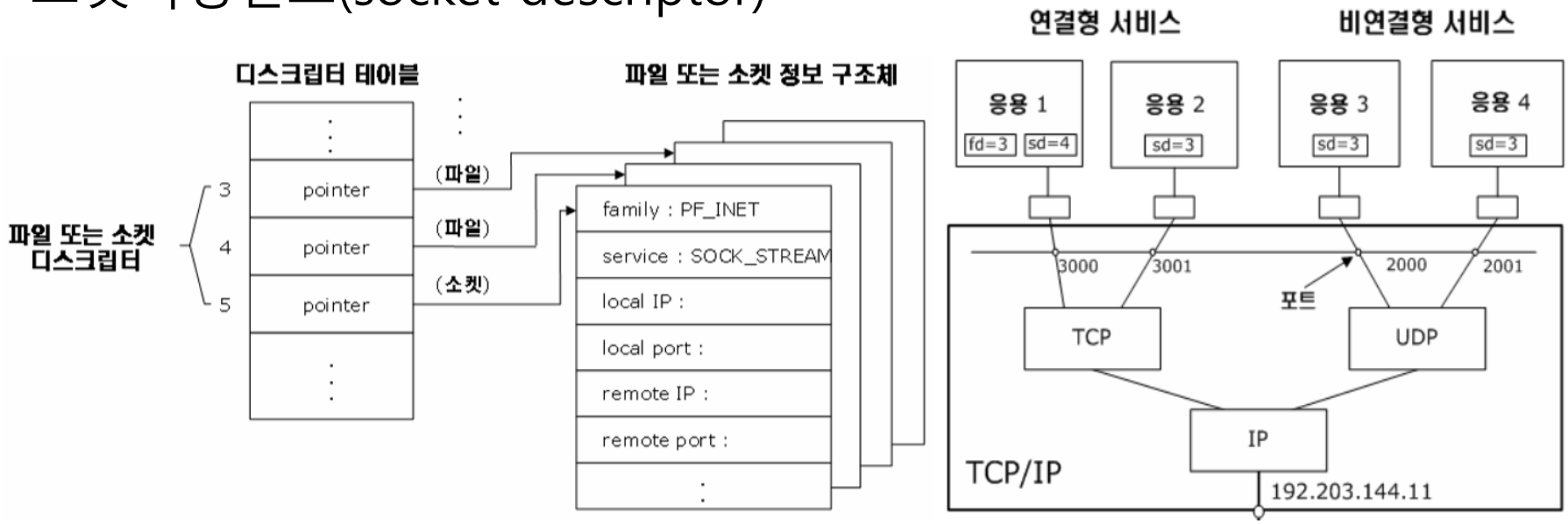
- socket 함수를 이용한 소켓 생성의 예
  1. TCP 소켓 :
     - **socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)**
  2. UDP 소켓
     - socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)

- socket 함수 반환 값
  - 성공적으로 소켓을 만들면 0보다 큰 int 값을 반환
  - 소켓지정번호, socket descriptor 라고 부름
  - 소켓을 지시하며, 이를 이용해서 소켓을 제어

```
s_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
```

socket()을 이용한 소켓 생성

- 소켓지정번호(socket descriptor)

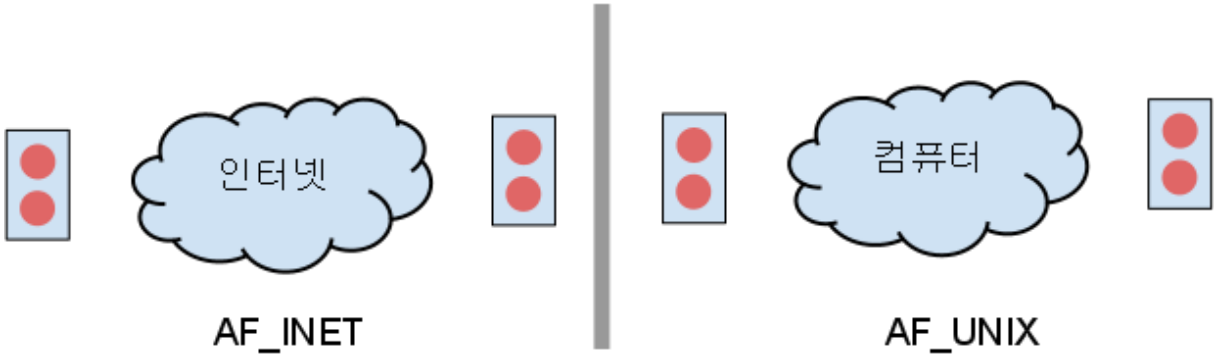


- descriptor
  - 유닉스에서 파일을 새로 열면 int형 타입의 파일 디스크립터를 리턴
  - 프로그램에서 이 파일을 액세스할 때 해당 파일 디스크립터를 사용
  - 유닉스에서는 각종 하드웨어 장치 파이프 소켓 등을 파일로 취급
- 소켓지정번호는 응용 프로그램 내에서 순서대로 배정됨
  - 프로그램 내에서만 유일하게 구분

socket()을 이용한 소켓 생성

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

- domain : 소켓이 사용되는 네트워크의 영역을 정의



TYPE	설명
AF_UNIX	프로세스간 통신(IPC)용
AF_INET	**일반 TCP/IP 인터넷 통신**
AF_IPX	노벨의 IPX
AF_X25	X.25 프로토콜

socket()을 이용한 소켓 생성

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

- Type : 통신에 사용할 패킷의 타입을 지정
- Protocol : 통신에 사용할 프로토콜 지정
- Type에 따라서 Protocol이 정해짐

Type	Protocol
SOCK_STREAM	**IPPROTO_TCP**
SOCK_DGRAM	IPPROTO_UDP
SOCK_RAW	

- SOCK_STREAM & IPPROTO_TCP : TCP 기반의 통신에 사용
- SOCK_DGRAM & IPPROTO_UDP : UDP 기반의 통신에 사용
- SOCK_RAW & (원하는 프로토콜) : RAW Socket으로 저수준에서 프로토콜을 직접 다룰 때 사용

connect()을 이용하여 연결하기

```
int connect(int sockfd, struct sockaddr *serv_addr, socklen_t addrlen);
```

- 연결하고자 하는 상대 Node의 IP 주소와 Port 번호를 이용하여 소켓 연결
  - sockfd : 소켓파일 지정 번호 → 클라이언트
  - serv_addr : 연결할 서버의 IP 주소와 Port 번호를 포함한 구조체
  - addrlen : 구조체 *serv_addr의 크기

connect()을 이용하여 연결하기

- connect 함수를 이용한 연결 요청의 예

```
#define PORT 9001
#define IPADDR "127.0.0.1"

struct sockaddr_in c_addr;

c_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
c_addr.sin_addr.s_addr = inet_addr(IPADDR);  // 127.0.0.1로 연결 요청
c_addr.sin_family = AF_INET;                 // internet TCP/IP 통신
c_addr.sin_port = htons(PORT);               // 9001포트에 연결된 프로그램을 요청

connect(c_socket, (struct sockaddr*)&c_addr, sizeof(c_addr));
```

데이터 통신하기

- 데이터 쓰기

```
int write(int sockfd, const void *buf, size_t count);
```

- 데이터 읽기

```
int read(int sockfd, void *buf, size_t count);
```

- sockfd : 소켓 지정 번호
- buf : 통신에 사용할 데이터를 가리키는 포인터
- count : 통신에 사용할 데이터의 크기

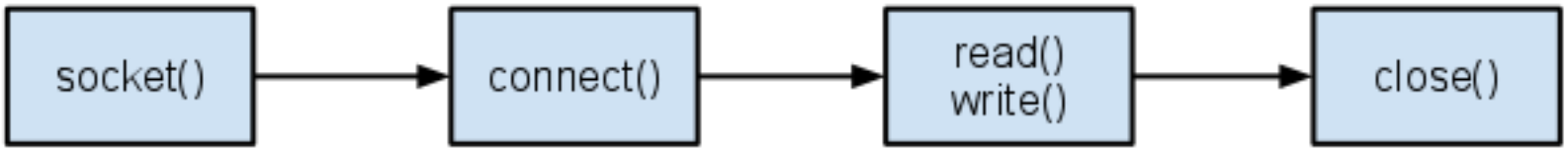
연결 종료

```
int close(int sockfd);
```

- 소켓을 닫고 연결 종료
  - 데이터 통신이 끝났다면, close 함수를 이용해서 소켓을 닫음
  - 소켓을 닫지 않을 경우 자원 누수 발생

클라이언트 네트워크 프로그램의 흐름

- 클라이언트 프로그램의 흐름
  - socket() : 소켓 생성
  - connect() : 연결 요청
  - read()/write() : 데이터 통신
  - close() : 소켓 닫기



hello, world 출력 소켓 프로그램 / 서버

```
1.  #include <unistd.h>
2.  #include <stdio.h>
3.  #include <string.h>
4.  #include <arpa/inet.h>
5.  #include <sys/socket.h>

6.  #define PORT 9001

7.  char buffer[BUFSIZ] = "Hello World₩n";

8.  int main(void){

9.   int s_socket, c_socket;
10.  struct sockaddr_in s_addr, c_addr;
11.  int len;
12.  int n;

13. s_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);   ①
14.
15. memset(&s_addr, 0, sizeof(s_addr));                    ②
16. s_addr.sin_addr.s_addr = htonl(INADDR_ANY);
17. s_addr.sin_family = AF_INET;
18. s_addr.sin_port = htons(PORT);
```

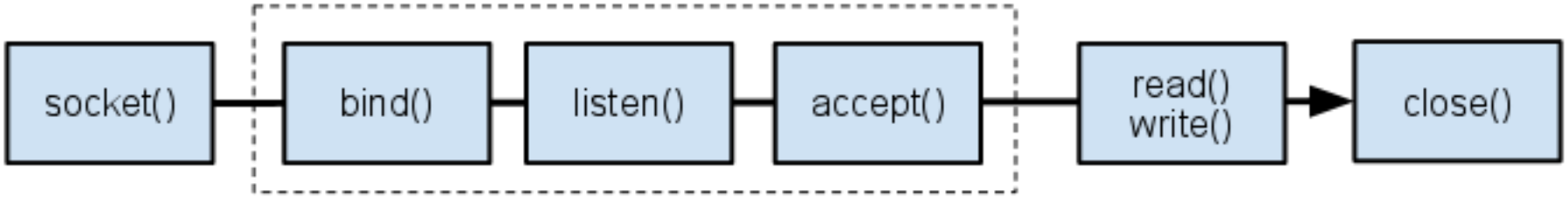
```
19. if(bind(s_socket, (struct sockaddr *)&s_addr, sizeof(s_addr)) == -1){   ③
20.         printf("Can not Bind₩n");
21.         return -1;
22. }

23. if(listen(s_socket, 5) == -1){                          ④
24.         printf("listen Fail₩n");
25.         return -1;
26. }

27. while(1){
28.         printf("Server waiting...₩n");
29.         len = sizeof(c_addr);                           ⑤
30.         c_socket = accept(s_socket, (struct sockaddr*)&c_addr, &len);
31.         printf("Connected IP : %s₩n", inet_ntoa(c_addr.sin_addr));
32.
33.         n = strlen(buffer);                             ⑥
34.         write(c_socket, buffer, n);
35.
36.         close(c_socket);                                ⑦
37. }
38.
39. close(s_socket);                                        ⑧
40. return 0;
41. }
```

① 소켓을 생성
② 연결 요청을 수신할 주소 설정
③ 소켓을 주소와 포트에 연결
④ 수신대기열 생성
⑤ 클라이언트 연결 요청 수신
⑥ 클라리언트 요청 서비스 제공
⑦ 클라이언트와 연결 종료
⑧ 서버 종료

- s_socket : 클라이언트의 연결 요청을 처리하는 듣기소켓
- c_socket : 연결된 클라이언트의 소켓과 직접 통신하는 연결소켓

서버 네트워크 프로그램의 흐름

- 서버 프로그램의 흐름
  - socket() : 소켓 생성
  - bind() : 소켓을 인터넷 주소와 포트 번호에 묶음
  - listen() : 수신 대기열 생성 (listen queue)
  - accept() : 연결 대기
  - read()/write() : 데이터 통신
  - close() : 소켓 닫기



서버 프로그램 만들기-bind()

- bind 함수
  - 소켓을 인터넷 주소(IP 주소 & Port 번호)에 묶어줌

```
int bind(int sockfd, struct sockaddr *my_addr, socklen_t addrlen);
```

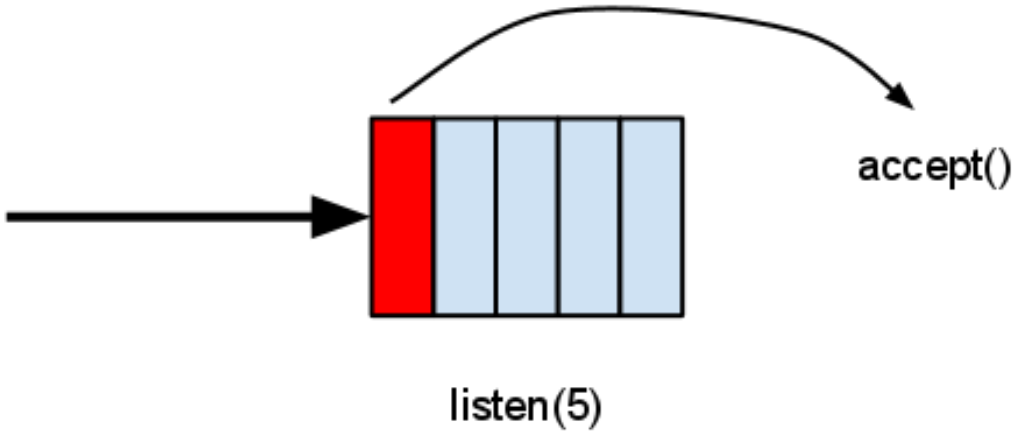
```
#define PORT 9001

struct sockaddr_in s_addr;

s_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
s_addr.sin_addr.s_addr = htonl(INADDR_ANY);  // 자신에게 오는 것은 모두..
s_addr.sin_family = AF_INET;                 // internet TCP/IP 통신
s_addr.sin_port = htons(PORT);               // 9001포트에 연결된 프로그램을 요청

bind(s_socket, (struct sockaddr *)&s_addr, sizeof(s_addr));
```

서버 프로그램 만들기-listen()

- listen 함수 : 수신 대기열 생성
  - 클라이언트의 요청은 먼저 수신 대기열에 들어감

```
int listen(int sockfd, int backlog);
```

  - sockfd: 소켓 지정 번호
  - backlog: 연결 대기열의 크기



서버 프로그램 만들기-accept()

- accept 함수
  - 수신 대기열의 맨 앞에 있는 클라이언트 요청을 읽음
  - 클라이언트와의 통신을 담당할 소켓 지정 번호를 반환

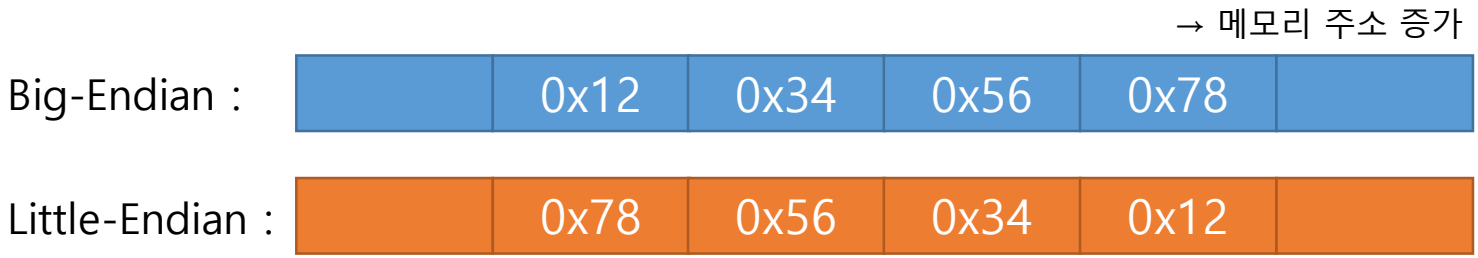
```
int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
```

  - sockfd: 소켓 지정 번호
  - *addr: 연결 요청을 한 클라이언트의 소켓 주소 구조체
  - *addrlen: *addr 구조체 크기의 포인터

기타 함수

```
s_addr.sin_addr.s_addr = htonl(INADDR_ANY);  // 자신에게 오는 것은 모두..
s_addr.sin_family = AF_INET;                 // internet TCP/IP 통신
s_addr.sin_port = htons(PORT);               // 9001포트에 연결된 프로그램을 요청
```

- 컴퓨터마다 메모리에 데이터를 저장하는 방식이 다름
  - 빅엔디안(Big-Endian), 리틀엔디안(Little-Endian)
  - 예: 0x12345678 저장 시



- 호스트 바이트 순서
  - 컴퓨터가 내부 메모리에 숫자를 저장하는 순서
  - CPU종류에 따라 빅엔디안 또는 리틀엔디안 방식 사용

기타 함수

```
s_addr.sin_addr.s_addr = htonl(INADDR_ANY);  // 자신에게 오는 것은 모두..
s_addr.sin_family = AF_INET;                 // internet TCP/IP 통신
s_addr.sin_port = htons(PORT);               // 9001포트에 연결된 프로그램을 요청
```

- 네트워크 바이트 순서
  - 포트번호나 주소와 같은 정보를 바이트 단위로 네트워크로 전송하는 순서
  - 빅엔디안 사용

- 바이트 순서가 바뀌는 문제의 해결방법
  - 네트워크로 전송하기 전에 htons()함수를 사용하여 네트워크 바이트 순서로 바꿈
  - 네트워크로부터 수신한 숫자는 ntohs()함수를 사용하여 자신의 호스트 바이트 순서로 바꿈

cf) htonl() / ntohl()

기타 함수

```
#define PORT 9001
#define IPADDR "127.0.0.1"

struct sockaddr_in c_addr;

c_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
c_addr.sin_addr.s_addr = inet_addr(IPADDR);  // 127.0.0.1로 연결 요청
c_addr.sin_family = AF_INET;              // internet TCP/IP 통신
c_addr.sin_port = htons(PORT);         // 9001포트에 연결된 프로그램을 요청
```

- IP 주소변환

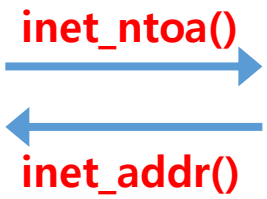
  - 4바이트 IP주소를 dotted decimal 방식으로 상호 변환 가능

01110010	11001011	10111101	10100000

IP주소(binary)

114.203.189.160

dotted dicimal



  - inet_ntoa(): 바이너리 IP주소를 dotted decimal 형태의 IP주소로
  - inet_addr(): dotted decimal 형태의 IP주소를 바이너리 IP주소로

참고문헌

- 정석용의 TCP/IP 소켓 프로그래밍, 정석용 저, 프리렉
- 뇌를 자극하는 TCP/IP 소켓 프로그래밍, 윤상배 저, 한빛미디어
- 열혈강의 TCP/IP 소켓 프로그래밍, 윤성우 저, 오렌지미디어
- 리눅스 네트워크 & 시스템 프로그래밍, 한국소프트웨어 진흥원