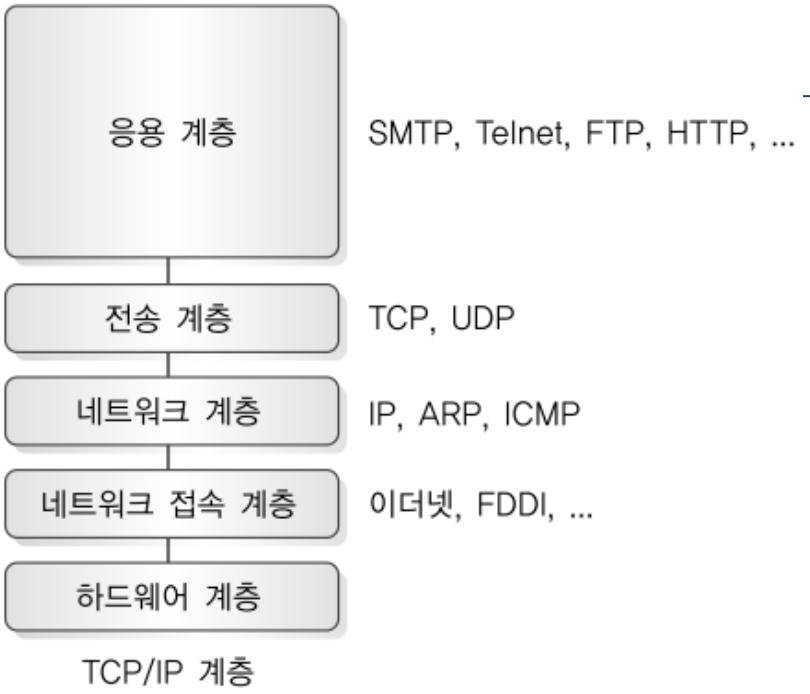
Socket Programming 1

Jo, Heeseung

TCP/IP 개요

TCP/IP

- 인터넷의 표준 프로토콜
- 5계층(4계층)으로 구성



TCP와 UDP의 차이

TCP	UDP
연결지향형(connection-oriented)	비연결형(connectionless)
신뢰성(reliability) 보장	신뢰성을 보장하지 않음
흐름 제어 기능(flow-control) 제공	흐름 제어 기능 없음
순서 보장(sequenced)	순서를 보장하지 않음(no sequence)

IP주소와 호스트명

IP주소와 호스트명

- IP주소 : 인터넷 주소로 점(.)으로 구분된 32비트 숫자
  - Ex. 10.198.123.33
- 호스트명 : 시스템에 부여된 이름
  - Ex. cslab.cbnu.ac.kr
- DNS : 호스트명(도메인명)과 IP주소를 관리하는 서비스

호스트명과 IP주소 변환

```
hosts: files dns
```

- /etc/hosts 파일 또는 DNS, NIS 등
- /etc/nsswitch.conf 파일에 주소변환을 누가 할 것인지 지정

IP주소와 호스트명

호스트명으로 정보 검색: gethostbyname(3)

```
#include <netdb.h>
struct hostent *gethostbyname(const char *name);
```

```
struct hostent{
    char *h_name;       //공식 도메인 이름
    char **h_aliases;   //공식 이외 도메인 이름들
    int h_addrtype;     //주소정보 체계(IPv4: AF_INET, IPv6: AF_INET6)
    int h_length;       //IP주소의 크기를 담는다. (IPv4는 4)
    char **h_addr_list; //도메인 이름에 대한 IP주소가 정수 형태로 반환될 때
                          이 멤버 변수를 이용
}
```

IP주소와 호스트명

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main()
{
    char *name = "www.google.com";
    struct hostent *host;

    host = gethostbyname(name);

    printf("%s\n", host->h_name);

    for(int i=0; host->h_aliases[i] != NULL; i++)
        printf("aliases : %s\n", host->h_aliases[i]);

    for(int i=0; host->h_addr_list[i] != NULL; i++)
        printf("addrList : %s\n",
            inet_ntoa(*(struct in_addr*)host->h_addr_list[i]));

    return 0;
}
```

```
# ex11_3.out
www.google.com
addrList : 142.251.8.94
```

IP주소와 호스트명

IP주소로 정보 검색: gethostbyaddr(3)

```
#include <netdb.h>
struct hostent *gethostbyaddr(const char *addr, int len, int type);
```

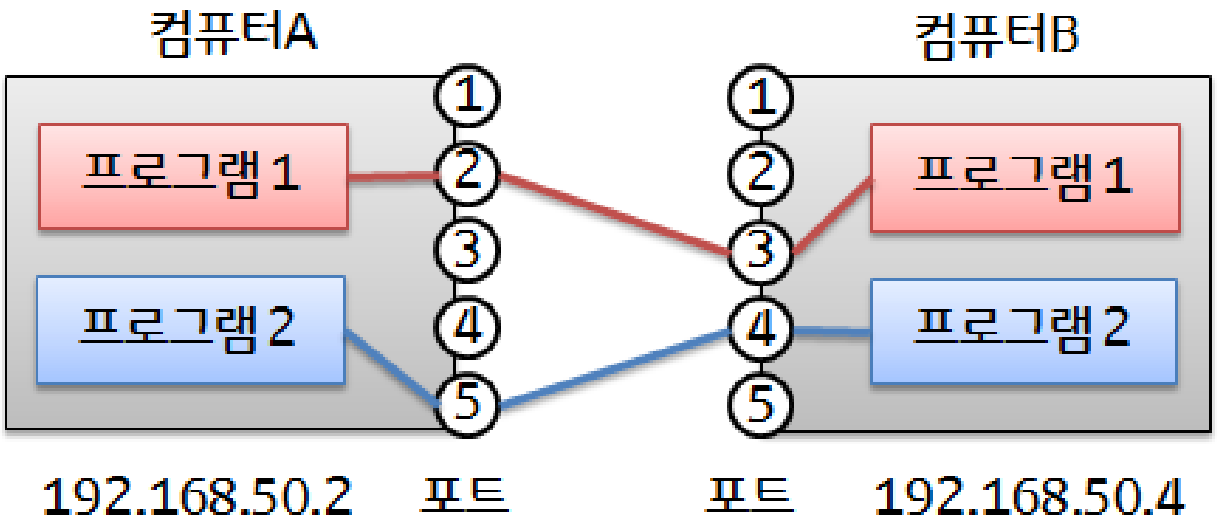
- type에 지정할 수 있는 값

```
AF_UNSPEC       0       /* 미지정 */
AF_UNIX         1       /* 호스트 내부 통신 */
AF_INET         2       /* 인터네트워크 통신: UDP, TCP 등 */
AF_IMPLINK      3       /* Arpanet의 IMP 주소 */
AF_PUP          4       /* PUP 프로토콜 : BSP 등 */
AF_CHAOS        5       /* MIT의 CHAOS 프로토콜 */
AF_NS           6       /* XEROX의 NS 프로토콜 */
AF_NBS          7       /* NBS 프로토콜 */
…
```

포트번호

포트번호

- 호스트에서 동작하고 있는 서비스를 구분하는 번호
- 2바이트 정수로 0~65535까지 사용가능
- Well known port : 이미 정해져 있고 자주 사용하는 포트
  - ssh(22), HTTP(80), HTTPS(443), mysql(3306), ...
- 관련 파일 : /etc/services



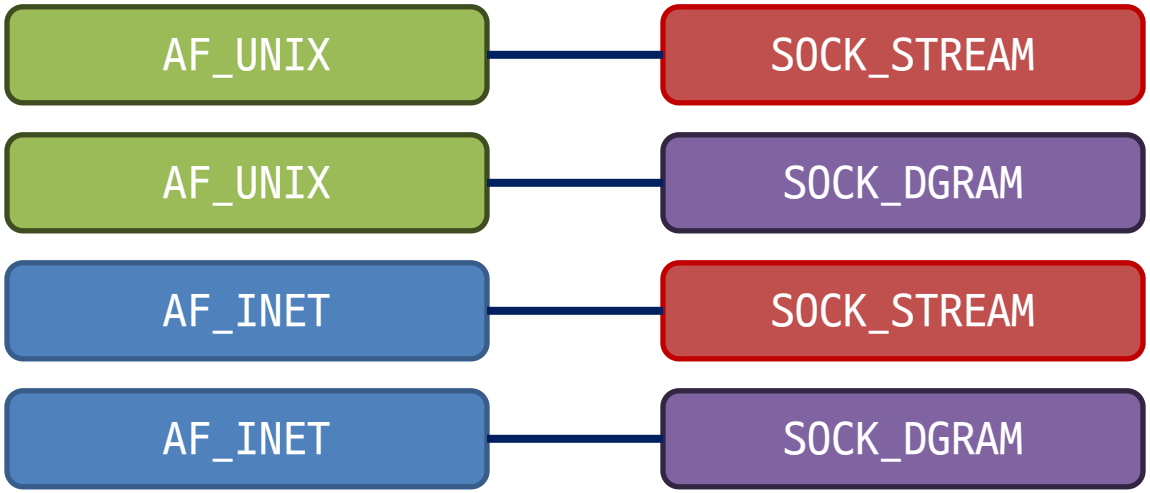
소켓 프로그래밍 기초[1]

소켓의 종류

- AF_UNIX : 유닉스 도메인 소켓 (시스템 내부 프로세스간 통신)
- AF_INET : 인터넷 소켓 (네트워크를 이용한 통신)

소켓의 통신 방식

- SOCK_STREAM : TCP 사용
- SOCK_DGRAM : UDP 사용



소켓 프로그래밍 기초[2]

소켓 주소 구조체

- 유닉스 도메인 소켓의 주소 구조체

```
struct sockaddr_un {
    sa_family_t  sun_family;
    char         sun_path[108]
};
```

- sun_family : AF_UNIX
- sun_path : 통신에 사용할 파일의 경로명

- 인터넷 소켓의 주소 구조체

```
struct sockaddr_in {
    sa_family_t    sin_family;
    in_port_t      sin_port;
    struct in_addr sin_addr;
    char sin_zero[8]; // 전체 크기를 16B로 맞추기 위한 dummy
};

struct in_addr {
    in_addr_t s_addr;      // unsigned long (4B)
};
```

IP주소 변환 함수

IP주소의 형태

- 192.168.10.1과 같이 점(.)으로 구분된 형태
- 시스템 내부 저장 방법 : 이진값 (FF.FF.FF.FF, 4B) 으로 바꿔서 저장
- 사용자 사용 형태 : 문자열로 사용

문자열 형태의 IP주소를 숫자형태로 변환 : inet_addr(3)

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
in_addr_t inet_addr(const char *cp);
```

"192.168.11.11"

구조체 형태의 IP주소를 문자열형태로 변환: inet_ntoa(3)

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
char *inet_ntoa(const struct in_addr in);
```

IP 주소 변환하기

```
...
09  int main(void) {
10      in_addr_t addr;
11      struct hostent *hp;
12      struct in_addr in;
13
14      if ((addr = inet_addr("210.115.161.30")) == (in_addr_t)-1) {
15          printf("Error : inet_addr()\n");
16          exit(1);
17      }
18
19      hp = gethostbyaddr((char *)&addr, 4, AF_INET);
20      if (hp == NULL) {
21          printf("Host information not found\n");
22          exit(2);
23      }
24
25      printf("Name=%s\n", hp->h_name);
26
27      memcpy(&in.s_addr, *hp->h_addr_list, sizeof (in.s_addr));
28      printf("IP=%s\n", inet_ntoa(in));
29
30      return 0;
31  }
```

문자열 형태를 이진형태로 변환

주소로 호스트명 검색

구조체 형태에서 문자열로 변환하여 출력

```
struct sockaddr_in {
    sa_family_t sin_family;
    in_port_t sin_port;
    struct in_addr sin_addr;
    char sin_zero[8];
};

struct in_addr {
    in_addr_t s_addr;
};
```

```
# gcc -o ex11_5.out ex11_5.c
# ex11_5.out
Name=mobile.chungbuk.ac.kr
IP=210.115.161.30
```

소켓 인터페이스 함수[1]

소켓 인터페이스 함수

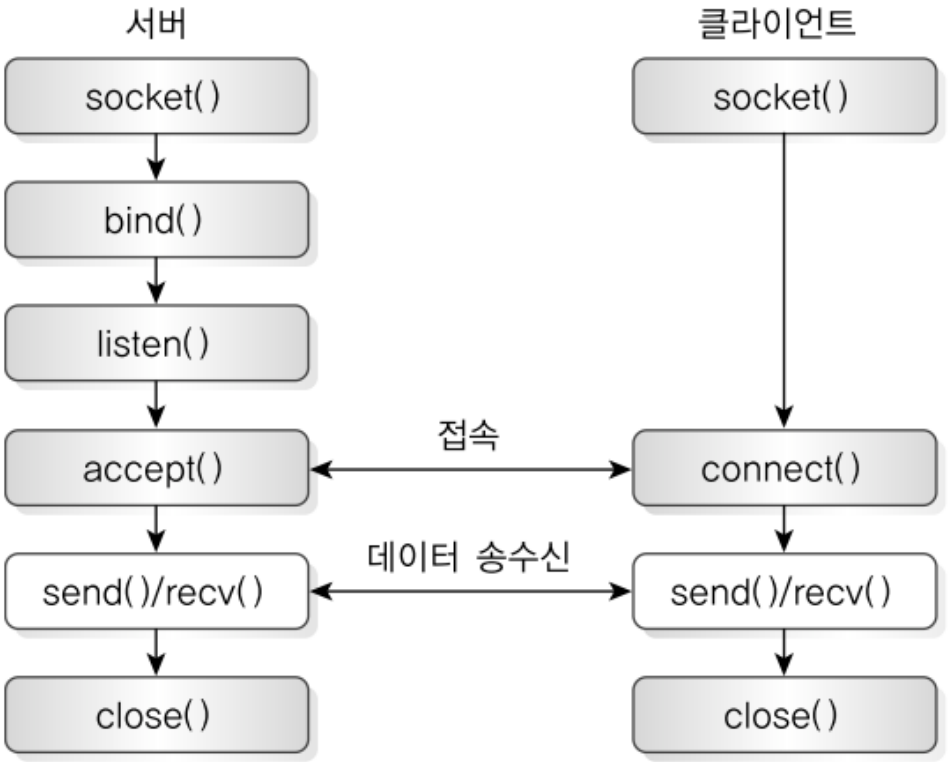
- socket : 소켓 파일기술자 생성
- bind : 소켓 파일기술자를 지정된 IP 주소/포트번호와 결합(bind)
- listen : 클라이언트의 접속 요청 대기
- accept : 클라이언트의 접속 허용

- connect : 클라이언트가 서버에 접속 요청

- recv : 데이터 수신(SOCK_STREAM) / recvfrom : 데이터 수신(SOCK_DGRAM)
- send : 데이터 송신(SOCK_STREAM) / sendto : 데이터 송신(SOCK_DGRAM)

- close : 소켓 파일기술자 종료

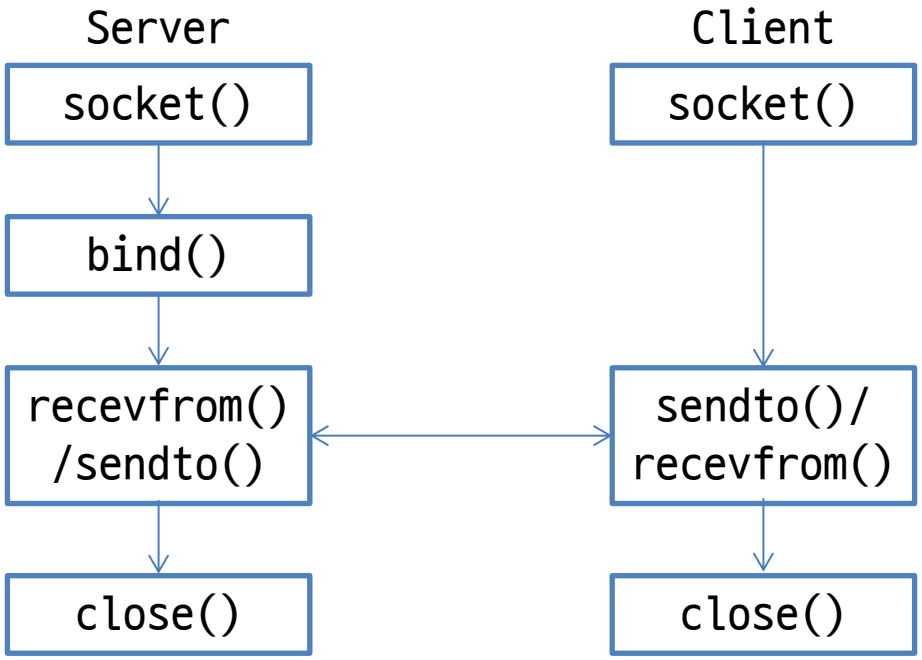
TCP 소켓 함수의 호출 순서



Office Telephone Analogy for Server

- Socket : 핸드폰 구매
- Bind : 통신사 가입
- Listen : 전원켜고 기지국과 연결 (수신 대기 상태)
- Accept : 전화가 울릴 때 받음

UDP 소켓 함수의 호출 순서



소켓 인터페이스 함수[2]

소켓 생성하기: socket(2)

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int domain, int type, int protocol);
```

- domain : 소켓 종류(AF_UNIX, AF_INET)
- type : 통신방식(TCP, UDP)
- protocol : 소켓에 이용할 프로토콜

```
int sd;
sd = socket(AF_INET, SOCK_STREAM, 0);
```

소켓 인터페이스 함수[3]

소켓에 이름 지정하기: bind(3)

```
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
int bind(int s, const struct sockaddr_in *name, int namelen);
```

- name : 소켓의 이름을 표현하는 구조체

```
struct sockaddr_in {
    sa_family_t sin_family;
    in_port_t sin_port;
    struct in_addr sin_addr;
    char sin_zero[8];
};

struct in_addr {
    in_addr_t s_addr;
};
```

```
int sd;
sd = socket(AF_INET, SOCK_STREAM, 0);

struct sockaddr_in sin;
memset((char *)&sin, '\0', sizeof(sin));
sin.sin_family = AF_INET;
sin.sin_port = htons(9000);
sin.sin_addr.s_addr = inet_addr("192.168.100.1");
memset(&(sin.sin_zero), 0, 8);

bind(sd, (struct sockaddr *)&sin, sizeof(struct sockaddr));
```

소켓 인터페이스 함수[4]

클라이언트 연결 기다리기: listen(3)

```
#include <sys/types.h>
#include <sys/socket.h>
int listen(int s, int backlog);
```

- backlog : 최대 허용 클라이언트 수 (대기 큐의 수)

```
listen(sd, 10);
```

연결 요청 수락하기: accept(3)

```
#include <sys/types.h>
#include <sys/socket.h>
int accept(int s, struct sockaddr_in *addr, socklen_t *addrlen);
```

- addr: 접속을 요청한 클라이언트의 IP 정보

```
int sd, new_sd;
struct sockaddr_in sin, clisin;
new_sd = accept(sd, &clisin, &sizeof(struct sockaddr_in));
```

- connect가 들어올 때까지 block

소켓 인터페이스 함수[5]

서버와 연결하기: connect(3) - client 에서 수행

```
#include <sys/types.h>
#include <sys/socket.h>
int connect(int s, const struct sockaddr_in *name, int namelen);
```

- name : 접속하려는 서버의 IP정보

```
int sd;

struct sockaddr_in sin;
memset((char *)&sin, '\0', sizeof(sin));
sin.sin_family = AF_INET;
sin.sin_port = htons(9000);
sin.sin_addr.s_addr = inet_addr("192.168.100.1");
memset(&(sin.sin_zero), 0, 8);

connect(sd, (struct sockaddr *)&sin, sizeof(struct sockaddr));
```

```
struct sockaddr_in {
    sa_family_t sin_family;
    in_port_t sin_port;
    struct in_addr sin_addr;
};

struct in_addr {
    in_addr_t s_addr;
};
```

소켓 인터페이스 함수[6]

TCP 데이터 보내기: send(3)

```
#include <sys/types.h>
#include <sys/socket.h>
ssize_t send(int s, const void *msg, size_t len, int flags);
```

- flags : 전송 데이터에 대한 option을 여러가지로 설정 가능

```
char *msg = "Send Test\n";
int len = strlen(msg) + 1;
if (send(sd, msg, len, 0) == -1) {
    perror("send");
    exit(1);
}
```

소켓 인터페이스 함수[6]

TCP 데이터 받기: recv(3)

```
#include <sys/types.h>
#include <sys/socket.h>
ssize_t recv(int s, void *buf, size_t len, int flags);
```

```
char buf[80];
int len, rlen;
if ((rlen = recv(sd, buf, len, 0)) == -1) {
    perror("recv");
    exit(1);
}
```

소켓 인터페이스 함수[7]

UDP 데이터 보내기: sendto(3)

```
#include <sys/types.h>
#include <sys/socket.h>
ssize_t sendto(int s,  const  void  *msg,  size_t  len,  int flags,
                const struct sockaddr_in *to, int  tolen);
```

- to : 메시지를 받을 호스트의 주소

```
char *msg = "Send Test\n";
int len = strlen(msg) + 1;

struct sockaddr_in sin;
int size = sizeof(struct sockaddr_in);
memset((char *)&sin, '\0', sizeof(sin));
sin.sin_family = AF_INET;
sin.sin_port = htons(9000);
sin.sin_addr.s_addr = inet_addr("192.168.10.1");
memset(&(sin.sin_zero), 0, 8);

if (sendto(sd, msg, len, 0, (struct sockaddr *)&sin, size) == -1) {
    perror("sendto");
    exit(1);
}
```

소켓 인터페이스 함수[3]

UDP 데이터 받기: recvfrom(3)

```
#include <sys/types.h>
#include <sys/socket.h>
ssize_t recvfrom(int s, void *buf, size_t  len,  int  flags,
                 struct sockaddr_in *from, int *fromlen);
```

- from : 메시지를 보내는 호스트의 정보 (채워져서 리턴됨)

```
char buf[80];
int len, size;
struct sockaddr_in sin;
if (recvfrom(sd, buf, len, 0, (struct sockaddr *)&sin, &size) == -1) {
    perror("recvfrom");
    exit(1);
}
```

소켓 인터페이스 함수[4]

Port 재사용을 위한 setsockopt()

- 프로그램에서 port를 사용 후 종료시 해당 port는 일정시간 동안 다시 사용되지 안도록 되어 있음
- 반복적인 테스트 단계에서 이를 해제하기 위하여 아래와 같은 방법을 이용

```
....
....
if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        perror("socket");
        exit(1);
}
int optvalue=1;
setsockopt(sd, SOL_SOCKET, SO_REUSEADDR, &optvalue, sizeof(optvalue));
....
....
```

(1) 인터넷 TCP 소켓(서버)

```
...
09  #define PORTNUM 9000
10
11  int main(void) {
12      char buf[256];
13      struct sockaddr_in sin, cli;
14      int sd, ns, clientlen = sizeof(cli);
15
16      if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
17          perror("socket");
18          exit(1);
19      }
20
21      memset((char *)&sin, '\0', sizeof(sin));
22      sin.sin_family = AF_INET;
23      sin.sin_port = htons(PORTNUM);
24      sin.sin_addr.s_addr = inet_addr("0.0.0.0");
25
26      if (bind(sd, (struct sockaddr *)&sin, sizeof(sin))) {
27          perror("bind");
28          exit(1);
29      }
```

포트번호

iter에서는 아래 두 파일을 include
#include <arpa/inet.h>
#include <sys/un.h>

소켓 생성

0.0.0.0 인 경우 모든 IP으로부터의 요청을 받음

소켓 주소 구조체 생성

소켓기술자와 소켓 주소 구조체 연결

(1) 인터넷 TCP 소켓(서버)

```
31      if (listen(sd, 5)) {
32          perror("listen");
33          exit(1);
34      }
35
36      if ((ns = accept(sd, (struct sockaddr *)&cli, &clientlen))==-1){
37          perror("accept");
38          exit(1);
39      }
40
41      sprintf(buf, "Your IP address is %s", inet_ntoa(cli.sin_addr));
42      if (send(ns, buf, strlen(buf) + 1, 0) == -1) {
43          perror("send");
44          exit(1);
45      }
46      close(ns);
47      close(sd);
48
49      return 0;
50  }
```

클라이언트 접속요청 대기

클라이언트와 연결

클라이언트로 데이터 보내기

(2) 인터넷 TCP 소켓(클라이언트)

```
...
09  #define PORTNUM 9000
10
11  int main(void) {
12      int sd;
13      char buf[256];
14      struct sockaddr_in sin;
15
16      if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
17          perror("socket");
18          exit(1);
19      }
20
21      memset((char *)&sin, '\0', sizeof(sin));
22      sin.sin_family = AF_INET;
23      sin.sin_port = htons(PORTNUM);
24      sin.sin_addr.s_addr = inet_addr("192.168.162.133");
25
```

포트번호
(각자 자기만의 포트번호 사용)

소켓 생성

접속할 IP (서버 IP)
ifconfig 명령으로 확인 가능

소켓 주소 구조체 생성

(2) 인터넷 TCP 소켓(클라이언트)

```
26      if (connect(sd, (struct sockaddr *)&sin, sizeof(sin))) {
27          perror("connect");
28          exit(1);
29      }
30
31      if (recv(sd, buf, sizeof(buf), 0) == -1) {
32          perror("recv");
33          exit(1);
34      }
35      close(sd);
36      printf("From Server : %s\n", buf);
37
38      return 0;
39  }
```

서버에 접속 요청

서버가 보낸 데이터 읽기

서버

```
# gcc -o ex11_7s ex11_7-inet-s.c
# gcc -o ex11_7c ex11_7-inet-c.c

# ex11_7s.out
```

클라이언트

```
# ex11_7c.out
From Server : Your IP address is 192.168.162.131
```

(1) 인터넷 UDP 소켓(서버)

```
...
09  #define PORTNUM 9005
10
11  int main(void) {
12      char buf[256];
13      struct sockaddr_in sin, cli;
14      int sd, clientlen = sizeof(cli);
15
16      if ((sd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
17          perror("socket");
18          exit(1);
19      }
20
21      memset((char *)&sin, '\0', sizeof(sin));
22      sin.sin_family = AF_INET;
23      sin.sin_port = htons(PORTNUM);
24      sin.sin_addr.s_addr = inet_addr("0.0.0.0");
25
26      if (bind(sd, (struct sockaddr *)&sin, sizeof(sin))) {
27          perror("bind");
28          exit(1);
29      }
```

포트번호
(각자 자기만의 포트번호 사용)

소켓 생성(데이터그램)

0.0.0.0 인 경우 모든 IP으로부터의 요청을 받음

소켓 주소 구조체 생성

소켓기술자와 소켓 주소 구조체 연결

(1) 인터넷 UDP 소켓(서버)

```
31      while (1) {
32          if ((recvfrom(sd, buf, 255, 0,
33                  (struct sockaddr *)&cli, &clientlen)) == -1) {
34              perror("recvfrom");
35              exit(1);
36          }
37          printf("** From Client : %s\n", buf);
38          strcpy(buf, "Hello Client");
39          if ((sendto(sd, buf, strlen(buf)+1, 0,
40                  (struct sockaddr *)&cli, sizeof(cli))) == -1) {
41              perror("sendto");
42              exit(1);
43          }
44      }
45
46      return 0;
47  }
```

클라이언트의 메시지 수신

클라이언트로 데이터 보내기

(2) 인터넷 UDP 소켓(클라이언트)

```
...
09  #define PORTNUM 9005
10
11  int main(void) {
12      int sd, n;
13      char buf[256];
14      struct sockaddr_in sin;
15
16      if ((sd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
17          perror("socket");
18          exit(1);
19      }
20
21      memset((char *)&sin, '\0', sizeof(sin));
22      sin.sin_family = AF_INET;
23      sin.sin_port = htons(PORTNUM);
24      sin.sin_addr.s_addr = inet_addr("192.168.162.133");
25
26      strcpy(buf, "I am a client.");
27      if (sendto(sd, buf, strlen(buf)+1, 0,
28                 (struct sockaddr *)&sin, sizeof(sin)) == -1) {
29          perror("sendto");
30          exit(1);
31      }
```

포트번호

소켓 생성

접속할 IP (서버 IP)
ifconfig 명령으로 확인 가능

소켓 주소 구조체 생성

서버에 메시지 전송

(2) 인터넷 UDP 소켓(클라이언트)

```
33      n = recvfrom(sd, buf, 255, 0, NULL, NULL);
34      buf[n] = '\0';
35      printf("** From Server : %s\n", buf);
36
37      return 0;
38  }
```

서버가 보낸 데이터 읽기

서버

```
# ex12_5s.out
** From Client : I am a client.
```

클라이언트

```
# ex12_5c.out
** From Server : Hello Client
```

Exercise

파란색 태그가 있는 모든 프로그램 작성 후 테스트 해 볼 것

- server는 swist1
- client는 swist2 에서 작성

- swist1 ip : 10.198.138.212 (내부)
- swist2 ip : 10.198.138.213 (내부)
- port는 각자 적절히 생성 (10000 이상의 번호를 사용)