12장 소켓을 이용한 통신 (1)

- 서론
- 예제 프로그램
- 함수
  - inet_addr
  - inet_ntoa
  - socket
  - bind
  - listen
  - accept
  - connect
  - recv
  - send

- 파이프를 사용하여 통신을 하기 위한 시스템 호출/표준 라이브러리 함수

함수	의미
inet_addr	문자열 형태의 인터넷 주소를 바이너리 형태로 변환한다.
inet_ntoa	바이너리 형태의 인터넷 주소를 문자열 형태로 변환한다.
socket	통신에 사용하기 위해 소켓을 생성한다.
bind	호스트의 로컬 주소를 소켓과 연결한다.
listen	소켓을 연결 요청 대기 상태로 만든다.
accept	연결 요청을 수락한다.
connect	연결을 요청한다.
recv	소켓을 통해 데이터를 수신한다.
send	소켓을 통해 데이터를 전송한다.

【예제12-1】 ex 12-01s.c , 서버(1/2)


```
01 #include <sys/types.h>
02 #include <sys/socket.h>
03 #include <netinet/in.h>
04
05 #define SIZE    sizeof(struct sockaddr_in)
06
07 main()
08 {
09    int sockfd_listen;
10     int sockfd_connect;
11
12    char c;
13    struct sockaddr_in server = {AF_INET, 5000, INADDR_ANY};
14
15     printf("socket()₩n");
16     sockfd_listen = socket(AF_INET, SOCK_STREAM, 0);
17
18     printf("bind()₩n");
19    bind(sockfd_listen, (struct sockaddr *)&server, SIZE);
```

09 연결 요청을 받기 위한 소켓 기술자

10 실제 통신을 위한 소켓 기술자

13 자신의 통신 유형에 관한 정보

16 소켓을 생성한다

19 통신 유형 정보와 소켓을 연결한다

표준입력 스트림

【예제12-1】 ex 12-01s.c , 서버(2/2)


```
20     printf("listen()₩n");
21    listen(sockfd_listen, 5);
22
23     printf("wating for client₩n");
24     sockfd_connect = accept(sockfd_listen, NULL, NULL);
25     printf("accepted₩n");
26
27     recv(sockfd_connect, &c, 1, 0);
28     printf("recv %c from client₩n", c);
29
30     c++;
31     printf("send %c to client₩n", c);
32     send(sockfd_connect, &c, 1, 0);
33
34     printf("close()₩n");
35     close(sockfd_connect);
36     close(sockfd_listen);
37 }
```

21 소켓을 클라이언트의 연결 요청을 기다리는 상태로 바꾼다.

24 대기 상태로 있다가 클라이언트가 연결을 요청하면 이를 수락하면서 새로운 소켓을 생성한다.

27 클라이언트가 보낸 메시지를 수신한다

32 클라이언트로 메시지를 전송한다

35 소켓을 닫는다

표준입력 스트림

【예제12-1】 ex 12-01c.c , 클라이언트(1/2)


```
01 #include <sys/types.h>
02 #include <sys/socket.h>
03 #include <netinet/in.h>
04
05 #define SIZE    sizeof(struct sockaddr_in)
06
07 main()
08 {
09     int sockfd;
10     char send_c, recv_c;
11     struct sockaddr_in server = {AF_INET, 5000};
12
13     server.sin_addr.s_addr = inet_addr("127.0.0.1");
14
15     printf("socket()₩n");
16   sockfd = socket(AF_INET, SOCK_STREAM, 0);
17
18     printf("connect()₩n");
19     connect(sockfd, (struct sockaddr *)&server, SIZE);
```

11 자신의 통신 유형에 관한 정보

13 연결 요청할 서버의 인터넷 주소

16 소켓을 생성한다

19 서버로 연결을 요청한다

표준입력 스트림

【예제12-1】 ex 12-01c.c , 클라이언트(1/2)

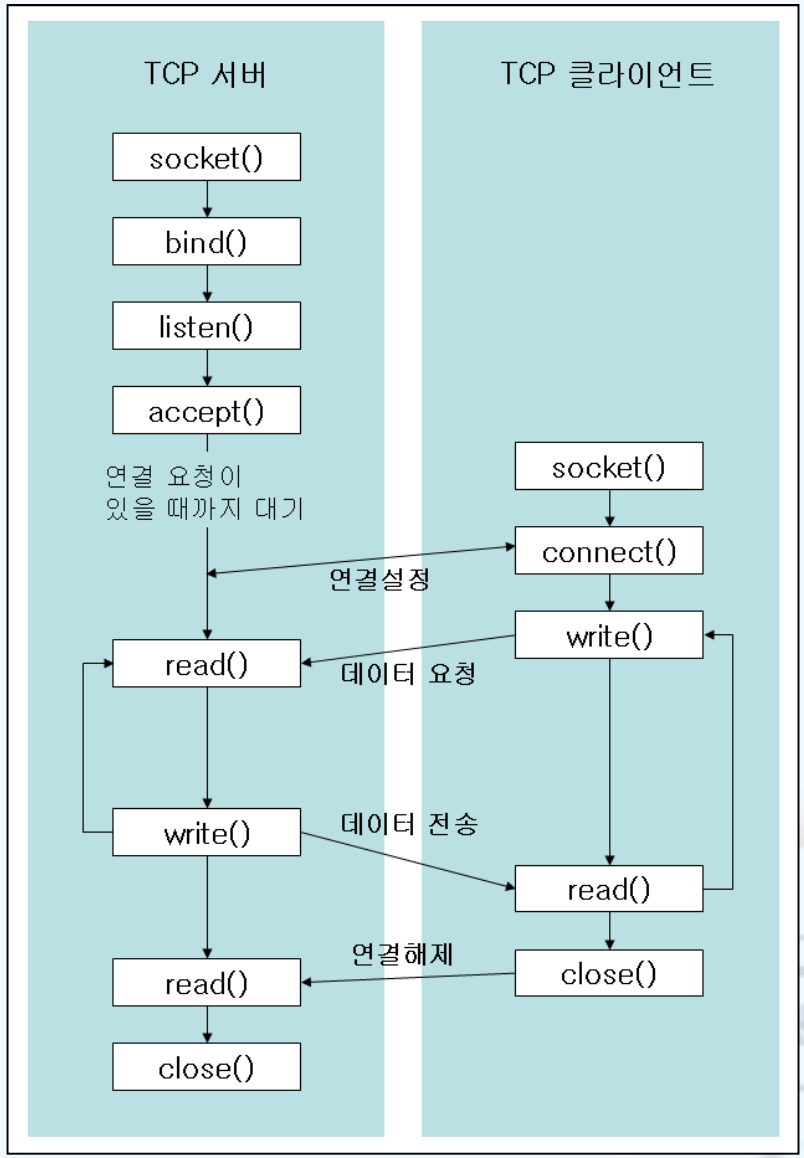

```
20     send_c = 'A';
21
22     printf("send %c to server₩n", send_c);
23     send(sockfd, &send_c, 1, 0);
24
25     recv(sockfd, &recv_c, 1, 0);
26     printf("recv %c from server₩n", recv_c);
27
28     printf("close()₩n");
29     close(sockfd);
30 }
```

23 서버로 메시지를 전송한다

26 서버가 보낸 메시지를 수신한다

29 소켓을 닫는다

표준입력 스트림


TCP 서버
TCP 클라이언트
socket()
bind()
listen()
accept()
연결 요청이
있을 때까지 대기
socket()
connect()
연결설정
write()
read()
데이터 요청
write()
데이터 전송
read()
연결해제
read()
close()
close()

실행 결과

서버 쪽	클라이언트 쪽
$ ex12-01s socket() bind() listen() waiting for client accepted recv A from client send B to client close() $	$ ex12-01c socket() connect() send A to server recv B from server close() $

인터넷 주소를 문자열에서 바이너리로 또는 바이너리에서 문자열로 변환한다

```
#include <arpa/inet.h>

in_addr_t inet_addr(const char *ip_addr);
char *inet_ntoa(struct in_addr n_addr);
```

ip_addr	문자열로 표현된 인터넷 호스트 주소이다.
n_addr	바이너리로 표현된 인터넷 호스트 주소이다
반환값	inet_addr은 호출이 성공하면 바이너리 형식으로 변환된 인터넷 주소가 반환되고, 실패하면 -1이 반환된다. inet_ntoa는 호출이 성공하면 문자열 형식으로 변환된 인터넷 주소의 포인터를 반환한다.

- **문자열 형식의 인터넷 주소**
  - **"202.31.200.123"과 같이 문자열로 구성된 인터넷 주소**
  - **실제 통신을 위해서는 바이너리로 변환해야 한다.**

【예제12-2】 ex 12-02.c


```
01 #include <arpa/inet.h>
02 #include <unistd.h>
03
04 main()
05 {
06     char *valid = "197.0.0.1";
07     char *invalid = "300.0.0.1";
08
09     in_addr_t ipaddr1;
10     struct in_addr ipaddr2;
11
12     if((ipaddr1 = inet_addr(valid)) == -1)
13         printf("invalid: %s₩n", valid);
14     else
15         printf("valid: %d.%d.%d.%d₩n",
16             (ipaddr1 >>  0) & 0xFF, (ipaddr1 >>  8) & 0xFF,
17             (ipaddr1 >> 16) & 0xFF, (ipaddr1 >> 24) & 0xFF );
18
19     ipaddr2.s_addr = ipaddr1;
20
21     if((ipaddr1 = inet_addr(invalid)) == -1)
22         printf("invalid: %s₩n", invalid);
23     else
24         printf("vaild: %x₩n", ipaddr1);
25
26     printf("%s₩n", inet_ntoa(ipaddr2));
27 }
```

표준입력 스트림

프로세스 간 통신을 위한 끝단(end-point, socket)을 생성한다

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

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

domain	통신 도메인(통신에 사용되는 프로토콜 집단)을 지정한다.
type	통신 방법을 선택한다.
protocol	선택된 프로토콜 군에서 특정 프로토콜을 선택한다. 대부분의 프로토콜 군은 하나의 프로토콜을 갖고 있다.
반환값	호출이 성공하면 소켓 기술자를 반환하고, 실패하면 -1을 반환한다.

- **파일을 다루기 위해 파일 기술자가 필요한 것처럼 통신은 소켓이 필요하다.**
- **소켓을 생성할 때 다음을 결정한다.**
  - **프로토콜 집단(유형)**
  - **연결형 또는 비연결형**

int domain; 프로토콜 집단 (protocol family)

이름 (매크로 상수)	의미
AF_UNIX, AF_LOCAL	로컬 통신 (동일한 시스템에 있는 프로세스 간의 통신이다.)
AF_INET	IP version 4의 인터넷 통신 프로토콜
AF_INET6	IP Version 6의 인터넷 통신 프로토콜
AF_IPX	노벨 네트워크의 통신 프로토콜

int type; 통신 유형

이름 (매크로 상수)	의미
SOCK_STREAM	연결형 통신을 사용한다.
SOCK_DGRAM	비연결형 통신을 사용한다.

int protocol; 통신용 프로토콜

보통 0을 설정 (선택한 프로토콜 집단과 통신 유형에 맞는 프로토콜은 하나뿐이다. 연결형이면 TCP, 비연결형이면 UDP가 선택된다.)

【예제12-3】 ex 12-03.c


```
01 #include <arpa/inet.h>
02 #include <sys/socket.h>
03 #include <unistd.h>
04
05 main()
06 {
07     int sockfd;
08
09     if( (sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
10     {
11         printf("fail to call socket()₩n");
12         exit(1);
13     }
14
15     printf("socket descriptor is %d₩n", sockfd);
16
17     /* 소켓을 통한 통신 기능을 수행한다. */
18
19     close(sockfd);
20 }
```

```
$ ex12-03
socket descriptor is 3
$
```

호스트의 로컬 주소를 소켓과 연결시킨다

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

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

sockfd	socket으로 미리 생성된 소켓 기술자이다.
my_addr	호스트의 로컬 주소이다.
addrlen	my_addr의 바이트 길이이다.
반환값	호출이 성공하면 0을 반환하고, 실패하면 -1을 반환한다.

- **SOCK_STREAM으로 설정되어 있고 서버 쪽에서 연결 요청을 받는 소켓일 경우 bind를 사용하여 로컬 주소를 소켓에 연결해야 한다.**

  → my_addr로 지정한 정보에 해당하는 통신 시도는 sockfd로 지정한 소켓으로 전달한다.

struct sockaddr_in 또는 struct sockaddr

- struct sockaddr_in
  - 서로 다른 시스템에 있는 프로세스 간의 통신
- struct sockaddr
  - 동일한 시스템에 있는 프로세스 간의 통신

- 두 번째 인자로 적용할 수 있는 데이터 형식은 struct sockaddr형이다.
  - struct sockaddr_in 형은 형 변환(type-casting)이 필요하다.

struct sockaddr과 struct sockaddr_in

```
struct sockaddr {
    sa_family_t     sa_family;      /* 주소 계열, AF_xxx        */
    char            sa_data[14];    /* 프로토콜 주소            */
};
```

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


struct sockaddr_in {
    sa_family_t         sin_family;     /* 주소 계열                */
    unsigned short int  sin_port;       /* 포트 번호                */
    struct in_addr      sin_addr;       /* 인터넷 주소              */


    /* 'struct sockaddr'과 크기를 맞추기 위한 패드 */
    unsigned char       __pad[__SOCK_SIZE__ - sizeof(short int) -
                        sizeof(unsigned short int) - sizeof(struct in_addr)];
};
```

【예제12-4】 ex 12-04.c


```
01 #include <sys/socket.h>
02 #include <unistd.h>
03 #include <netinet/in.h>
04
05 #define SIZE    sizeof(struct sockaddr_in)
06
07 main()
08 {
09     int sockfd;
10     struct sockaddr_in addr;
11
12     addr.sin_family = AF_INET;
13     addr.sin_port = 1004;
14     addr.sin_addr.s_addr = INADDR_ANY;
15
16     /* socket()으로 소켓을 생성하는 코드 */
17
18     if(bind(sockfd, (struct sockaddr *)&addr, SIZE) == -1) {
19             printf("fail to call bind()₩n");
20             exit(1);
21     }
22
23    /* 통신을 위한 나머지 코드들.. */
24
25     close(sockfd);
26 }
```

표준입력 스트림

소켓을 통해 들어오는 연결 요청을 기다린다

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

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

sockfd	socket으로 미리 생성된 소켓 기술자이다.
backlog	큐의 사이즈이다. 큐는 동시에 들어오는 여러 연결 요청을 넣어둔다.
반환값	호출이 성공하면 0을 반환하고, 실패하면 -1을 반환한다.

- **listen은 지정한 소켓을 연결 요청을 받을 수 있는 상태로 만든다.**
  - **SOCK_STREAM으로 설정한 것과 같이 연결형 통신 방법으로 설정된 서버 쪽 소켓에 지정한다.**
  - **이렇게 지정된 소켓은 메시지를 주고 받는 실제 통신에 사용되는 것이 아니라 클라이언트의 연결 요청을 받아 들이는 용도로 사용된다.**

    클라이언트는 connect를 호출하여 연결을 요청한다.

int backlog

- 동시에 받아들일 수 있는 연결 요청의 수를 의미한다.
- 동시에 여러 개의 연결 요청이 있을 경우
  - 먼저 들어온 요청을 처리하고 나머지는 backlog에서 지정한 크기의 큐에서 대기하게 된다.
- backlog는 0이 될 수 없다.

【예제12-5】 ex 12-05.c


```
01 #include <...h>
02
03 #define SIZE    sizeof(struct sockaddr_in)
04
05 main()
06 {
07     int sockfd_listen;
08     char c;
09     struct sockaddr_in server = {AF_INET, 5000, INADDR_ANY};
10
11     /* socket()을 호출하는 부분 */
12
13     /* bind()를 호출하는 부분 */
14
15     if(listen(sockfd_listen, 5) == -1) {
16         printf("fail to call listen()₩n");
17         exit(1);
18     }
19
20     /* 통신을 수행하는 부분 */
21 }
```

표준입력 스트림

연결형 소켓으로 통신을 할 때 상대의 연결 요청을 수락한다

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

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

sockfd	socket으로 미리 생성된 소켓 기술자이다.
addr	연결 요청한 상대에 대한 정보가 저장된다.
addrlen	addr의 바이트 길이이다.
반환값	호출이 성공하면 음수가 아닌 정수형의 새로운 소켓 기술자가 반환되고, 실패하면 -1이 반환된다.

- 연결형 통신 모델에서 서버 쪽 프로세스가 사용한다.
- accept를 호출하면
  - 연결 요청 대기 큐의 첫 번째 연결 요청을 가져온다.
  - 실제 통신을 위한 새로운 소켓 기술자를 할당한다.

listen의 소켓과 accept가 만드는 새로운 소켓

- listen의 소켓
  - ▸클라이언트의 연결 요청을 받기 위한 용도의 소켓이다.
  - ▸실제 메시지를 주고 받는 용도로 사용되지 않는다.
- accept가 만드는 새로운 소켓
  - ▸listen의 소켓으로 들어온 클라이언트의 연결 요청에 응한 결과 클라이언트와 실제 메시지를 주고 받기 위해 새롭게 만들어진다.

struct sockaddr *addr

- 연결을 요청한 클라이언트 쪽 프로세스에 대한 정보로 채워진다.

【예제12-6】 ex 12-06.c(1/2)


```
01 #include <….h>
02
03 #define SIZE    sizeof(struct sockaddr_in)
04
05 int sockfd_connect;
06
07 main()
08 {
09     int sockfd_listen;
10     char c;
11     struct sockaddr_in server = {AF_INET, 5000, INADDR_ANY};
12
13     if((sockfd_listen = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
14         printf("fail to call socket()₩n");
15         exit(1);
16     }
17
18     if(bind(sockfd_listen, (struct sockaddr *)&server, SIZE) == -1) {
19         printf("fail to call bind()₩n");
20         exit(1);
21     }
```

【예제12-6】 ex 12-06.c(2/2)


```
22      if(listen(sockfd_listen, 5) == -1) {
23          printf("fail to call listen()₩n");
24          exit(1);
25      }
26
27      while(1) {
28          if((sockfd_connect = accept(sockfd_listen, NULL, NULL)) == -1) {
29              printf("fail to call accept()₩n");
30              continue;
31          }
32
33          /* sockfd_connect를 사용하여 통신을 수행 */
34
35      }
36  }
```

listen 상태로 대기 중인 상대 프로세스에게 연결을 요청한다

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

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

sockfd	소켓 기술자이다.
serv_addr	통신을 연결할 상대의 주소 등에 관한 정보가 저장되어 있다.
addrlen	serv_addr의 크기이다.
반환값	호출이 성공하면 0을 반환하고, 실패하면 -1을 반환한다.

- **클라이언트 쪽 프로세스가 호출한다.**
  listen 상태로 대기 중인 서버 쪽 프로세스에게 연결을 요청한다.

serv_addr

- **연결을 요청할 대상인 서버 쪽에 대한 정보가 담겨 있다.**

【예제12-7】 ex 12-07.c


```
01 #include <….h>
02
03 #define SIZE    sizeof(struct sockaddr_in)
04
05 main()
06 {
07     int sockfd;
08     char send_c, recv_c;
09     struct sockaddr_in server = {AF_INET, 5000};
10
11     server.sin_addr.s_addr = inet_addr("127.0.0.1");
12
13     if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
14         printf("fail to call socket()₩n");
15         exit(1);
16     }
17
18     if(connect(sockfd, (struct sockaddr *)&server, SIZE) == -1) {
19         printf("fail to call connect()₩n");
20         exit(1);
21     }
22
23     /* 메시지를 주고받는 부분 */
24 }
```

표준입력 스트림

지정한 소켓을 사용하여 메시지를 전송하거나 수신한다.

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

int send(int sockfd, const void *msg, size_t msg_len, int flags);
int recv(int sockfd, void *msg, size_t msg_len, int flags);
```

sockfd	메시지를 전송하거나 수신할 소켓 기술자이다.
msg	전송하거나 수신한 메시지를 저장하고 있다.
msg_len	msg의 길이이다.
flags	메시지를 전송하거나 수신하는 방법을 선택한다.
반환값	send는 호출이 성공할 경우 전송한 문자 수를 반환하고, 실패할 경우 -1을 반환한다. recv는 호출이 성공할 경우 수신한 바이트 수를 반환하고, 실패할 경우 -1을 반환한다.

- **int sockfd**
  - 실제 메시지를 주고 받을 소켓의 기술자
    - ▸서버 쪽은 accept 호출로 만들어진 소켓이고 클라이언트 쪽은 connect를 호출할 때 지정한 소켓이다.
- **const void msg**
  - 상대 프로세스에게 전송할 또는 상대 프로세스로부터 수신한 메시지를 담고 있는 버퍼이다.
- **size_t msg_len**
  - 송수신할 메시지의 길이이다.
  - 실제 메시지의 길이와 상관없이 msg_len으로 지정한 크기만큼 송수신한다.

size_t msg_len

- send와 recv의 송수신 방법을 결정한다.
- 보통 0을 설정한다.
  - write, read와 동일하게 동작한다.

함수	심볼형 상수	의미
send	MSG_OOB	대역을 벗어난 데이터를 보낸다. 프로토콜이 대역을 벗어난 데이터를 지원해야 한다.
	MSG_DONTROUTE	직접 연결된 네트워크에 존재하는 호스트로 전송할 때 게이트웨이를 사용하지 않고 전송한다.
	MSG_NOSIGNAL	연결형 소켓에서 상대 프로세스가 연결을 끊었을 때 발생하는 SIGPIPE 시그널을 받지 않는다.
recv	MSG_OOB	일단 데이터가 아닌 대역을 벗어난 데이터를 수신한다.
	MSG_PEEK	도착한 메시지 큐에서 첫 번째 메시지를 제거하지 않고 읽어온다. 일반적인 호출이라면 큐에서 메시지를 가져오면서 제거한다.
	MSG_NOSIGNAL	연결형 소켓에서 상태 프로세스가 연결을 끊었을 때 발생하는 SIGPIPE 시그널을 받지 않는다.

【예제12-8】 ex 12-08.c


```
01 #define SIZE    sizeof(struct sockaddr_in)
02
03 int sockfd_connect;
04
05 main()
06 {
07     /* sock_listen 소켓 생성 */
08     /* sock_listen으로 bind 호출 */
09     /* sock_listen으로 listen 호출 */
10
11     while(1) {
12
13         /* accept를 호출하여 sockfd_connet를 생성 */
14
15         while(recv(sockfd_connect, &c, 1, 0) > 0)
16             send(sockfd_connect, &c, 1, 0);
17
18         close(sockfd_connect);
19     }
20 }
```

표준입력 스트림

소켓 또는 소켓 기술자

- 통신을 위해 생성하는 것이다.
  - 파일을 다루기 위한 파일 기술자와 같은 셈이다.
- 통신을 완료하여 더 이상 필요 없어진 소켓은 닫는다.

- 문제점 : 파일 기술자를 닫을 때와 다른 점
  - 어느 한쪽의 프로세스가 비정상적으로 종료되는 경우

    → 반대쪽의 프로세스는 상대 프로세스의 비정상적인 종료를 알 수가 없다.

    그래서, 존재하지 않는 상대에게 메시지를 전송하거나 메시지를 수신하려고 한다.

    → 서버 쪽 프로세스는 클라이언트의 비정성적인 종료에 대한 대비가 있어야 한다.

SIGPIPE 시그널

- 비정상적으로 끊어진 상대방의 소켓에 대해서 메시지 송신이나 수신을 시도하면 SIGPIPE 시그널이 발생한다.
- 유연한 대처를 위해 SIGPIPE 시그널을 처리하기 위한 대비가 있어야 한다.

ex12-09.c

```
...

while(1) {
    sockfd_connect = accept(sockfd_listen, NULL, NULL);

    /* send, recv를 호출하는 부분 */

    close(sockfd_connect);
}

...
```

[예제 프로그램 완성] 연결형 모델, 서버쪽(1/3)


```
01 #include <sys/types.h>
02 #include <sys/socket.h>
03 #include <netinet/in.h>
04 #include <signal.h>
05
06 #define SIZE    sizeof(struct sockaddr_in)
07
08 void closesock(int sig);
09
10 int sockfd_connect;
11
12 main()
13 {
14     int sockfd_listen;
15     char c;
16     struct sockaddr_in server = {AF_INET, 5000, INADDR_ANY};
17     struct sigaction act;
18
19     act.sa_handler = closesock;
20     sigfillset(&(act.sa_mask));
21    sigaction(SIGPIPE, &act, NULL);
```

표준입력 스트림

【예제 프로그램 완성】 연결형 모델, 서버쪽(2/3)


```
22    if((sockfd_listen = socket(AF_INET, SOCK_STREAM, 0)) == -1)
23    {
24        printf("fail to call socket()₩n");
25        exit(1);
26    }
27
28    if(bind(sockfd_listen, (struct sockaddr *)&server, SIZE) == -1)
29    {
30        printf("fail to call bind()₩n");
31        exit(1);
32    }
33
34    if(listen(sockfd_listen, 5) == -1)
35    {
36        printf("fail to call listen()₩n");
37        exit(1);
38    }
```

표준입력 스트림

[예제 프로그램 완성] 연결형 모델, 서버쪽(3/3)


```
39      while(1)
40      {
41          if((sockfd_connect = accept(sockfd_listen, NULL, NULL)) == -1)
42          {
43              printf("fail to call accept()₩n");
44              continue;
45          }
46         printf("accepted₩n");
47
48          while(recv(sockfd_connect, &c, 1, 0) > 0)
49              send(sockfd_connect, &c, 1, 0);
50
51          printf("close(sockfd_connect)₩n");
52          close(sockfd_connect);
53      }
54 }
55
56 void closesock(int sig)
57 {
58      close(sockfd_connect);
59      printf("connection is lost₩n");
60      exit(0);
61 }
```

표준입력 스트림

[예제 프로그램 완성] 클라이언트쪽(1/2)


```
01 #include <sys/types.h>
02 #include <sys/socket.h>
03 #include <netinet/in.h>
04
05 #define SIZE    sizeof(struct sockaddr_in)
06
07 main()
08 {
09     int sockfd;
10     char send_c, recv_c;
11     struct sockaddr_in server = {AF_INET, 5000};
12
13     server.sin_addr.s_addr = inet_addr("127.0.0.1");
14
15     if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
16         printf("fail to call socket()₩n");
17         exit(1);
18     }
19
20     if(connect(sockfd, (struct sockaddr *)&server, SIZE) == -1) {
21         printf("fail to call connect()₩n");
22         exit(1);
23     }
```

표준입력 스트림

[예제 프로그램 완성] 클라이언트쪽(2/2)


```
24      recv_c = '₩n';
25      while(1)
26      {
27          if(recv_c == '₩n')
28             printf("Input a message₩n");
29          send_c = getchar();
30
31          send(sockfd, &send_c, 1, 0);
32
33          if(recv(sockfd, &recv_c, 1, 0) > 0)
34              printf("%c", recv_c);
35          else
36          {
37              printf("server has no reply₩n");
38              close(sockfd);
39              exit(1);
40          }
41      }
42  }
```

표준입력 스트림

실행 결과

ex12-10s.c	ex12-10c.c
$ ex12-10s accepted close(sockfd_connect) ^C $	$ ex12-10c Input a message hello world! hello world! Input a message 안녕!! 안녕!! Input a message ^C $