

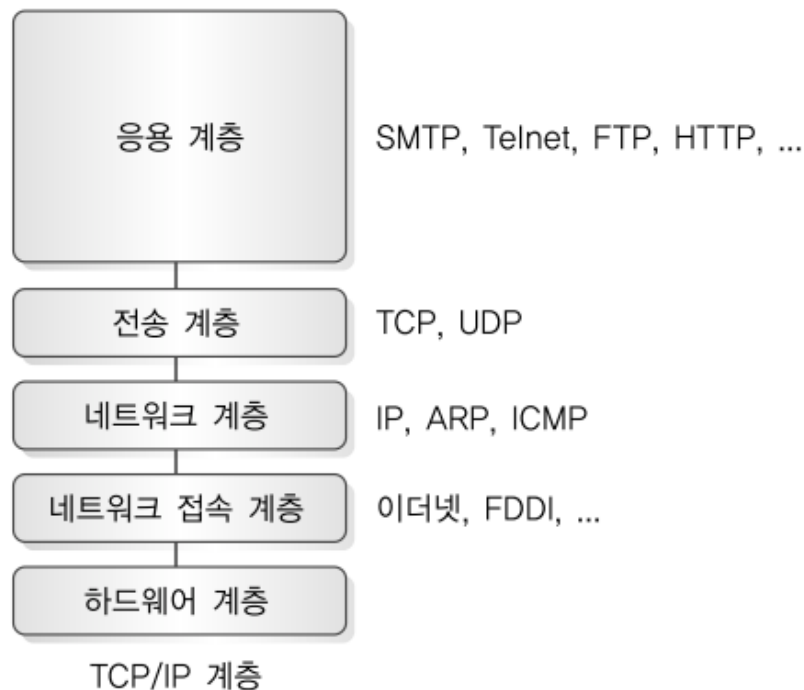
실습(1)



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[1]

□ IP주소와 호스트명

- IP주소 : 인터넷을 이용할 때 사용하는 주소로 점(.)으로 구분된 32비트 숫자
- 호스트명 : . 시스템에 부여된 이름
- 호스트명(도메인명)과 IP주소를 관리하는 서비스 -> DNS

□ 호스트명과 IP주소 변환

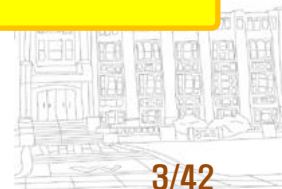
- /etc/hosts 파일 또는 DNS, NIS 등
- /etc/nsswitch.conf 파일에 주소변환을 누가 할 것인지 지정

```
hosts: files dns
```

□ 호스트명과 주소 읽어오기: gethostent(3), sethostent(3), endhostent(3)

```
#include <netdb.h>
struct hostent *gethostent(void);
int sethostent(int stayopen);
int endhostent(void);
```

P441 설명 참조



IP주소와 호스트명[1]

- **gethostent** : 호스트명과 IP주소를 읽어와서 **hostent** 구조체에 저장
- **sethostent** : 데이터베이스의 읽기 위치를 시작위치로 재설정
- **endhostent** : 데이터베이스를 닫는다

□ **hostent** 구조체

```
struct hostent {  
    char *h_name;  
    char **h_aliases;  
    int h_addrtype;  
    int h_length;  
    char **h_addr_list;  
}
```



[예제 11-1] 호스트명 읽어오기 (test1.c)

```
01  #include <netdb.h>
02  #include <stdio.h>
03
04  int main(void) {
05      struct hostent *hent;
06
07      sethostent(0);
08
09      while ((hent = gethostent()) != NULL)
10          printf("Name=%s\n", hent->h_name);
11
12      endhostent();
13
14      return 0;
15  }
```

처음 위치로 이동

DB의 내용을 차례로 읽어오기

DB 닫기

```
# gcc ex11_1.c
```

정의되지 않음

기호

endhostent

gethostent

sethostent

ld: 치명적: 기호 참조 오류. a.out에 출력이 기록되지 않음

collect2: ld returned 1 exit status

첫번째 참조된

파일:

/var/tmp//ccwQu9hN.o

/var/tmp//ccwQu9hN.o

/var/tmp//ccwQu9hN.o

왜 실행파일이 생성되지 않을까?

[예제 11-1] 실행결과

□ 표준 C 라이브러리에 없는 함수들이기 때문에

- endhostent, gethostent, sethostent
- libnsl.so 라이브러리를 링크해야 -> /usr/lib 디렉토리에 위치

```
# gcc -o ex11_1.out ex11_1.c -lnsl
```

- /etc/host파일의 내용이 다음과 같을 때

```
# cat /etc/hosts
#
# Internet host table
#
127.0.0.1      localhost
218.237.65.4   www.hanb.co.kr
192.168.162.133 hanbit
```

- 실행결과

```
# ex11_1.out
Name=localhost
Name=www.hanb.co.kr
Name=hanbit
```



IP주소와 호스트명[2]

□ 호스트명으로 정보 검색: gethostbyname(3)

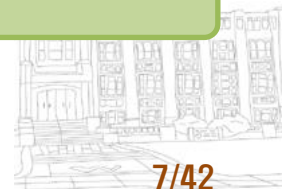
```
#include <netdb.h>
struct hostent *gethostbyname(const char *name);
```

□ 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	/* 인터넷워크 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[1]

□ 포트번호

- 호스트에서 동작하고 있는 서비스를 구분하는 번호
- 2바이트 정수로 0~65535까지 사용가능
- 잘 알려진 포트 : 이미 정해져 있고 자주 사용하는 포트
 - 텔넷(23), HTTP(80), FTP(21)
- 관련 파일 : /etc/services

□ 포트 정보 읽어오기: getservent(3), setservent(3), endservent(3)

```
#include <netdb.h>
struct servent *getservent(void);
int setservent(int stayopen);
int endservent(void);
```

- **getservent** : 포트 정보를 읽어 servent 구조체로 리턴
- **setservent** : 읽기 위치를 시작으로 재설정
- **endservent** : 데이터베이스 닫기

```
struct servent {
    char *s_name;
    char **s_aliases;
    int s_port;
    char **s_proto;
}
```


[예제 11-2] getservent 함수로 포트 정보 읽어오기(test2.c)

```
01 #include <netdb.h>
02 #include <stdio.h>
03
04 int main(void) {
05     struct servent *port;
06     int n;
07
08     setservent(0);
09
10     for (n = 0; n < 5; n++) {
11         port = getservent();
12         printf("Name=%s, Port=%d\n", port->s_name, port->s_port);
13     }
14
15     endservent();
16
17     return 0;
18 }
```

처음 위치로 이동

DB의 내용을 차례로 5개만 읽어오기

DB닫기

socket 라이브러리를 지정해서 컴파일해야 한다.
gcc -o ex11_2.out ex11_2.c -lsocket

```
# ex11_2.out
Name=tcpmux, Port=256
Name=echo, Port=1792
Name=echo, Port=1792
Name=discard, Port=2304
Name=discard, Port=2304
```

□ 서비스명으로 정보 검색: getservbyname(3)

```
#include <netdb.h>
struct servent *getservbyname(const char *name, const char *proto);
```

- name : 검색할 포트명
- proto : tcp 또는 udp 또는 NULL

□ 포트 번호로 정보 검색: getservbyport(3)

```
#include <netdb.h>
struct servent *getservbyport(int port, const char *proto);
```

- proto : tcp 또는 udp 또는 NULL



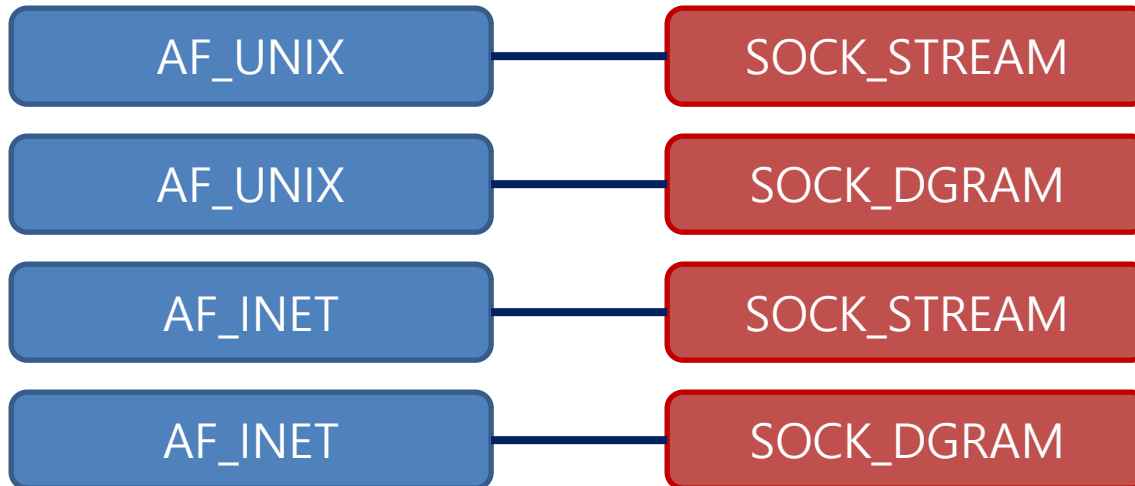
소켓 프로그래밍 기초[1]

□ 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;  
};  
  
struct in_addr {  
    in_addr_t s_addr;  
};
```



□ 바이트 순서 함수

- 정수를 저장하는 방식 : 빅엔디안, 리틀엔디안
- 빅엔디안 : 메모리의 낮은 주소에 정수의 첫 바이트를 위치 -> 모토로라, 썬
- 리틀엔디안 : 메모리의 높은 주소에 정수의 첫 바이트를 위치 -> 인텔
- TCP/IP 네트워크에서 바이트 순서 표준 : 빅엔디안
- 호스트 바이트 순서(HBO) : 시스템에서 사용하는 바이트 순서
- 네트워크 바이트 순서(NBO) : 네트워크에서 사용하는 바이트 순서

```
#include <sys/types.h>
#include <netinet/in.h>
#include <inttypes.h>
uint32_t htonl(uint32_t hostlong);
uint16_t htons(uint16_t hostshort);
uint32_t ntohl(uint32_t netlong);
uint16_t ntohs(uint16_t netshort);
```

- **htonl** : 32비트 HBO를 32비트 NBO로 변환
- **htons** : 16비트 HBO를 16비트 NBO로 변환
- **ntohl** : 32비트 NBO를 32비트 HBO로 변환
- **ntohs** : 16비트 NBO를 16비트 HBO로 변환



[예제 11-3] NBO를 HBO로 변환하기(test3.c)

```
01  #include <netdb.h>
02  #include <stdio.h>
03
04  int main(void) {
05      struct servent *port;
06      int n;
07
08      setservent(0);
09
10      for (n = 0; n < 5; n++) {
11          port = getservent();
12          printf("Name=%s, Port=%d\n", port->s_name,
13                  ntohs(port->s_port));
14      }
15      endservent();
16
17      return 0;
18  }
```

NBO를 HBO로 변환하기 위한 함수 호출

```
# ex11_3.out
Name=tcpmux, Port=1
Name=echo, Port=7
Name=echo, Port=7
Name=discard, Port=9
Name=discard, Port=9
```

[예제 11-4] HBO를 NBO로 변환하기(test4.c)

```
01 #include <netdb.h>
02 #include <stdio.h>
03
04 int main(void) {
05     struct servent *port;
06
07     port = getservbyname("telnet", "tcp");
08     printf("Name=%s, Port=%d\n", port->s_name, ntohs(port->s_port));
09
10     port = getservbyport(htons(21), "tcp");
11     printf("Name=%s, Port=%d\n", port->s_name, ntohs(port->s_port));
12
13     return 0;
14 }
```

이름으로 서비스 포트번호 검색

HBO를 NBO로 변환하여 포트번호 검색

```
# ex11_4.out
Name=telnet, Port=23
Name=ftp, Port=21
```



IP주소 변환 함수

□ IP주소의 형태

- 192.168.10.1과 같이 점(.)으로 구분된 형태
- 시스템 내부 저장 방법 : 이진값으로 바뀌서 저장
- 외부적 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);
```

□ 구조체 형태의 IP주소를 문자열 형태로 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);
```


[예제 11-5] IP 주소 변환하기(test5.c)

```
...
09 int main(void) {
10     in_addr_t addr;
11     struct hostent *hp;
12     struct in_addr in;
13
14     if ((addr = inet_addr("218.237.65.4")) == (in_addr_t)-1) {
15         printf("Error : inet_addr(218.237.65.4\n");
16         exit(1);
17     }
18
19     hp = gethostbyaddr((char *)&addr, 4, AF_INET);
20     if (hp == NULL) {
21         (void) printf("Host information not found\n");
22         exit(2);
23     }
24
25     printf("Name=%s\n", hp->h_name);
26
27     (void) memcpy(&in.s_addr, *hp->h_addr_list, sizeof (in.s_addr)); 0;
28     printf("IP=%s\n", inet_ntoa(in));
29
30     return 0;
31 }
```

문자열 행태를 이진형태로 변환

주소로 호스트명 검색

구조체 형태에서 문자열로 변환하여 출력

```
# gcc -o ex11_5.out ex11_5.c -lsocket -lnsl
# ex11_5.out
Name=www.hanb.co.kr
IP=218.237.65.4
```

실습(2)



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
- **connect** : 클라이언트가 서버에 접속 요청
- **accept** : 클라이언트의 접속 허용
- **recv** : 데이터 수신(SOCK_STREAM)
- **send** : 데이터 송신(SOCK_STREAM)
- **recvfrom** : 데이터 수신(SOCK_DGRAM)
- **sendto** : 데이터 송신(SOCK_DGRAM)
- **close** : 소켓 파일기술자 종료



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>
int bind(int s, const struct sockaddr *name, int namelen);
```

- **name** : 소켓의 이름을 표현하는 구조체

```
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");
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

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

□ 연결 요청 수락하기: accept(3)

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



소켓 인터페이스 함수[5]

□ 서버와 연결하기: connect(3)

```
#include <sys/types.h>
#include <sys/socket.h>
int connect(int s, const struct sockaddr *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");
connect(sd, (struct sockaddr *)&sin, sizeof(struct sockaddr));
```



소켓 인터페이스 함수[6]

□ 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);
```

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

□ 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);
}
```

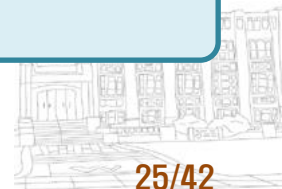

소켓 인터페이스 함수[7]

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



소켓 인터페이스 함수[3]

□ UDP 데이터 받기: recvfrom(3)

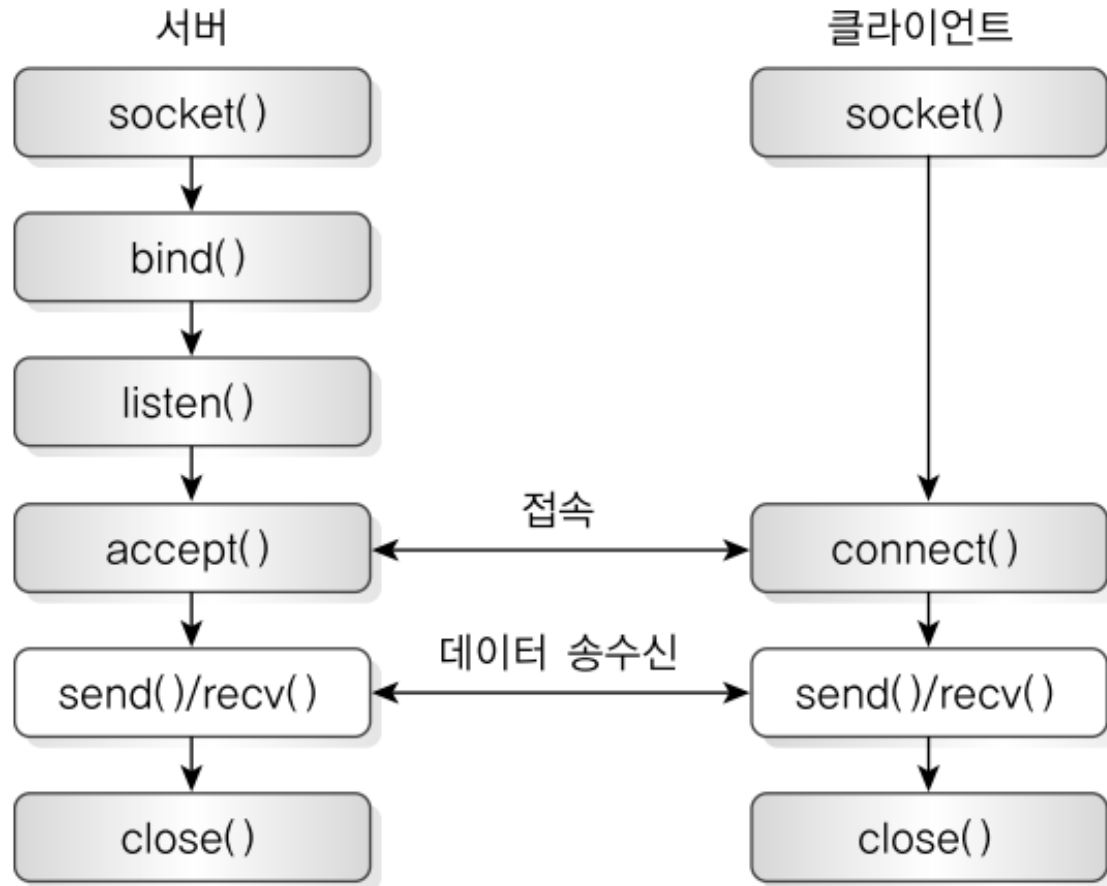
```
#include <sys/types.h>
#include <sys/socket.h>
ssize_t recvfrom(int s, void *buf, size_t len, int flags,
                 struct sockaddr *from, int *fromlen);
```

- **from** : 메시지를 보내는 호스트의 주소

```
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 함수의 호출 순서



[그림 11-3] 소켓 함수의 호출 순서



[예제 11-6] (1) 유닉스 도메인 소켓(서버)-server.c

```
...
08  #define SOCKET_NAME      "hbsocket"
09
10  int main(void) {
11      char buf[256];
12      struct sockaddr_un ser, cli;
13      int sd, nsd, len, clen;
14
15      if ((sd = socket(AF_UNIX, SOCK_STREAM, 0)) == -1) {
16          perror("socket");
17          exit(1);
18      }
19
20      memset((char *)&ser, 0, sizeof(struct sockaddr_un));
21      ser.sun_family = AF_UNIX;
22      strcpy(ser.sun_path, SOCKET_NAME);
23      len = sizeof(ser.sun_family) + strlen(ser.sun_path);
24
```

소켓 이름

유닉스 도메인 소켓 생성

소켓구조체에 값 지정



[예제 11-6] (1) 유닉스 도메인 소켓(서버)

```
25     if (bind(sd, (struct sockaddr *)&ser, len)) {
26         perror("bind");
27         exit(1);
28     }
29
30     if (listen(sd, 5) < 0) {
31         perror("listen");
32         exit(1);
33     }
34
35     printf("Waiting ...\n");
36     if ((nsd = accept(sd, (struct sockaddr *)&cli, &cliclen)) == -1) {
37         perror("accept");
38         exit(1);
39     }
40
41     if (recv(nsd, buf, sizeof(buf), 0) == -1) {
42         perror("recv");
43         exit(1);
44     }
45
46     printf("Received Message: %s\n", buf);
47     close(nsd);
48     close(sd);
49
50     return 0;
51 }
```

소켓기술자와 소켓 주소 구조체 연결

클라이언트 접속 대기

클라이언트 접속 수용

클라이언트가 보낸 메시지 읽기

[예제 11-6] (2) 유닉스 도메인 소켓(클라이언트)-client.c

```
...
08  #define SOCKET_NAME      "hbsocket"
09
10  int main(void) {
11      int sd, len;
12      char buf[256];
13      struct sockaddr_un ser;
14
15      if ((sd = socket(AF_UNIX, SOCK_STREAM, 0)) == -1) {
16          perror("socket");
17          exit(1);
18      }
19
20      memset((char *)&ser, '\0', sizeof(ser));
21      ser.sun_family = AF_UNIX;
22      strcpy(ser.sun_path, SOCKET_NAME);
23      len = sizeof(ser.sun_family) + strlen(ser.sun_path);
24
25      if (connect(sd, (struct sockaddr *)&ser, len) < 0) {
26          perror("bind");
27          exit(1);
28      }
```

소켓 생성

소켓 주소 구조체에 값 지정

서버에 연결 요청

[예제 11-6] (2) 유닉스 도메인 소켓(클라이언트)-client.c

```
30     strcpy(buf, "Unix Domain Socket Test Message");
31     if (send(sd, buf, sizeof(buf), 0) == -1) {
32         perror("send");
33         exit(1);
34     }
35     close(sd);
36
37     return 0;
38 }
```

서버에 데이터 전송

```
# ex11_6s.out
Waiting ...
Received Message: Unix Domain Socket Test Message
```

서버

```
# ex11_6c.out
#
```

클라이언트



실습(3)

