12주차 3차시 소켓 프로그래밍

【학습목표】

1. 소켓 프로그래밍의 함수를 사용할 수 있다.
2. 소켓을 이용하여 통신 프로그램을 작성 할 수 있다.

학습내용1 : 소켓 프로그래밍 함수

1. 소켓 인터페이스 함수

① 함수 종류
- 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

② 소켓 생성하기: socket(2)
- domain에 지정한 소켓의 형식과 type에 지정한 통신방법을 지원하는 소켓 생성
- 성공 시 : 소켓 기술자 리턴
- 실패 시 : -1 리턴

```
#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, 보통 0 지정

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

③ 소켓에 이름 지정하기: bind(3)
- socket함수가 생성한 소켓 기술자 s에 sockaddr 구조체인 name지정한 정보 연결
- 성공 시 : 0
- 실패 시 : -1 리턴

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

- s : socket 함수가 생성한 소켓 기술자
- name : 소켓의 이름을 표현하는 구조체
- namelen : name의 크기

* 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");
memset(&(sin.sin_zero), 0, 8);
bind(sd, (struct sockaddr *)&sin, sizeof(struct sockaddr));
```

④ 클라이언트 연결 기다리기: listen(3)

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

- s : socket 함수가 생성한 소켓 기술자
- backlog : 최대 허용 클라이언트 수

* 예 : 클라이언트 연결 요청 받아들일 준비 마쳤고 최대 10개까지만 받음.

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

⑤ 연결 요청 수락하기: accept(3)

* 클라이언트 : 연결 요청 수락

- 서버 : 소켓 s를 통해 요청한 클라이언트와의 연결을 수락

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

- s : socket 함수가 생성한 소켓 기술자
- addr: 접속을 요청한 클라이언트의 IP 정보
- addrlen : addr 크기

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

⑥ 서버와 연결하기: connect(3)

- 클라이언트가 서버에 연결 요청시 사용
- 성공 시 : 0
- 실패 시 : -1

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

- s : socket 함수가 생성한 소켓 기술자
- name : 접속하려는 서버의 IP정보
- namelen : name의 크기

* 예

```
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));
```

⑦ 데이터 보내기: send(3)

- 소켓 s를 통해 크기가 len인 메시지 msg를 flags에 지정한 방법으로 전송
- 성공 시 : 실제로 전송한 데이터의 바이트 수 리턴
- 실패 시 : -1

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

ssize_t send(int s, const void *msg, size_t len, int flafs);
```

* flages

- MSG_OOB : 영역밖의 데이터 처리, SOCK_STREAM에서만 사용
- MSG_DONTROUTE : 데이터의 라우팅 설정을 해제.

* 예

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

⑧ 데이터 받기: recv(3)

* 소켓 s를 통해 전송받은 메시지를 크기가 len인 buf에 저장

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

ssize_t recv(int s, void *buf, size_t len, int flags);
```

* 예

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

⑨ UDP 데이터 보내기: sendto(3)

* UDP 프로토콜 활용 데이터 전송 함수
* 매번 목적지 주소 지정 필요

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

ssize_t sentto(int s, const void *msg, size_t len, int flags,
                const struct sockaddr *to, int tolen);
```

- s : socket 함수가 생성한 소켓 기술자
- msg : 전송할 메시지를 저장한 메모리 주소
- len : 메모리의 크기
- flags : 데이터를 주고받는 방법을 지정한 플래그
- to : 메시지를 받을 호스트의 주소
- tolen : to의 크기

* 예

```
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);
}
```

⑩ UDP 데이터 받기: recvfrom(3)

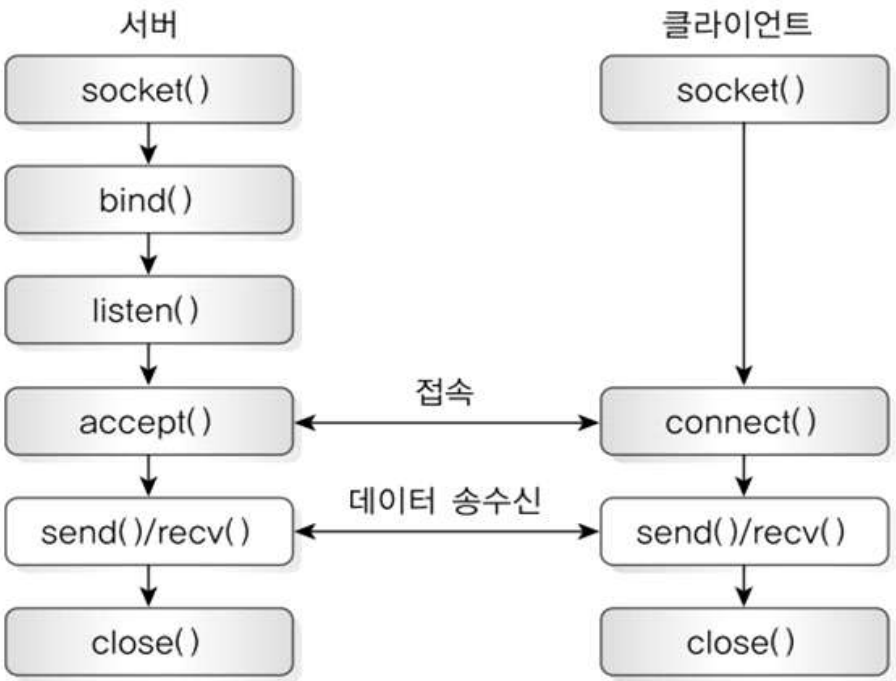
```
#include <sys/types.h>
#include <sys/socket.h>

ssize_t recvfrom(int s, void *buf, size_t len, int flags,
                 struct sockaddr *from, int *fromlen);
```

- s : socket 함수가 생성한 소켓 기술자
- msg : 전송받을 메시지를 저장한 메모리 주소
- len : 메모리의 크기
- flags : 데이터를 주고받는 방법을 지정한 플래그
- from : 메시지를 보내는 호스트의 주소
- tolen : from의 크기

* 예

```
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);
}
```

2. 소켓 함수의 호출 순서



학습내용2 : 소켓 프로그래밍 활용

1. 유닉스 도메인 소켓

* 유닉스 도메인 소켓 - 서버

```
...
#define SOCKET_NAME     "hbsocket"

int main(void) {
    char buf[256];
    struct sockaddr_un ser, cli;
    int sd, nsd, len, clen;

    if ((sd = socket(AF_UNIX, SOCK_STREAM, 0)) == -1) {
        perror("socket");
        exit(1);
    }

    memset((char *)&ser, 0, sizeof(struct sockaddr_un));
    ser.sun_family = AF_UNIX;
    strcpy(ser.sun_path, SOCKET_NAME);
    len = sizeof(ser.sun_family) + strlen(ser.sun_path);

    if (bind(sd, (struct sockaddr *)&ser, len)) {
        perror("bind");
        exit(1);
    }

    if (listen(sd, 5) < 0) {
        perror("listen");
        exit(1);
    }

    printf("Waiting ...\n");
    if ((nsd = accept(sd, (struct sockaddr *)&cli, &clen)) == -1) {
        perror("accept");
        exit(1);
    }

    if (recv(nsd, buf, sizeof(buf), 0) == -1) {
        perror("recv");
        exit(1);
}
    printf("Received Message: %s\n", buf);
    close(nsd);
    close(sd);

    return 0;
}
```

소켓 이름

유닉스 도메인 소켓 생성

소켓구조체에 값 지정

소켓기술자와 소켓 주소 구조체 연결

클라이언트 접속 대기

클라이언트 접속 수용

클라이언트가 보낸 메시지 읽기

* 유닉스 도메인 소켓 - 클라이언트

```
...
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      }
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  }
```

소켓 생성

소켓 주소 구조체에 값 지정

서버에 연결 요청

서버에 데이터 전송

서버

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

클라이언트

```
# ex11_6c.out
#
```

* 인터넷 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("192.168.162.133");
25
26      if (bind(sd, (struct sockaddr *)&sin, sizeof(sin))) {
27          perror("bind");
28          exit(1);
29      }
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  }
```

포트번호

소켓 생성

소켓 주소 구조체 생성

소켓기술자와 소켓 주소 구조체 연결

클라이언트 접속요청 대기

클라이언트와 연결

클라이언트로 데이터 보내기

* 인터넷 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
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  }
```

포트번호

소켓 생성

소켓 주소 구조체 생성

서버에 접속 요청

서버가 보낸 데이터 읽기

서버

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

# ex11_7s.out
```

클라이언트

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

【학습정리】

1. 소켓 프로그래밍 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

2. 소켓 프로그래밍 활용
- TCP/UDP 기반 프로그래밍의 차이점

항목	TCP기반 프로그래밍	UDP 기반 프로그래밍
통신 방식	SOCK_STREAM으로 지정	SOCK_DGRAM으로 지정
서버	bind, listen, accept함수를 순서대로 사용	bind함수를 수행한후 listen함수를 사용하지 않는다.
클라이언트	connect 함수를 사용해 명시적으로 서버 지정 서버와 연결을 확보한 후 recv() 함수나 send() 함수를 통해 데이터를 주고받음.	connect()함수를 사용하지 않음 클라이언트의 연결 요청이 별도로 없으므로 서버가 클라이언트로 데이터를 보낼 때 클라이언트의 주소를 구조체로 지정해야 함.
사용목적	신뢰성이 중요한 응용프로그램에 사용	신뢰성보다는 빠르게 전달하고 응답 받는 응용 프로그램에서 사용