[예제 11-7] (1) 인터넷 소켓(서버)-server1.c

이 포트가 사용중인지 알 수 있는 명령어
netstat -anp | grep 9002
kill -9 프로세스 넘버

전에 했던 서버를 에코 서버로 만드는 실습 파일을 인터넷 버전으로 바꾼 것이 server0.c, client0.c

```
...
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      }
```

포트번호

실제로 실습할 때는 포트넘버 9000, 9001, 9002 를 사용하면 안됨.
9000 + 학번 뒤에 두 자리를 사용.

에러가 나는 경우, 문의.

인터넷을 통해 통신하려면 #include <netinet/in.h>
를 사용해야 함.

소켓 생성

소켓 주소 구조체 생성

IP 는 같은 서버로 보내고 받는 프로그램을 사용.
우리 서버의 IP 주소: 203.250.148.46

소켓기술자와 소켓 주소 구조체 연결

[예제 11-7] (1) 인터넷 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

IP 주소를 문자열로 바꿈.

클라이언트로 데이터 보내기

[예제 11-7] (2) 인터넷 소켓(클라이언트)-client1.c

```
...
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

소켓 생성

소켓 주소 구조체 생성

h 에서 n 으로 바꿔줌.

[예제 11-7] (2) 인터넷 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 -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
```

TCP 기반 프로그래밍

두 가지 종류의 서버 프로그램

□ 반복서버

서버 프로세스

- 데몬 프로세스가 직접 모든 클라이언트의 요청을 차례로 처리
- 따라서 한번에 한 클라이언트의 요청만 처리할 수 있고, 여러 클라이언트가 서비스를 요청할 경우 순차적으로 처리

□ 동시동작서버

부모(서버 or 데몬) 프로세스가 리슨을 하고,

서버 프로세스와 서비스를 처리하는 프로세스(자식 프로세스)가 따로 있음.

- 데몬 프로세스가 직접 서비스를 제공하지 않고, 서비스를 대신 처리할 프로세스를 fork 함수로 생성해 클라이언트와 연결시켜준다.

[예제 12-1] (1) 반복서버(서버)-server2.c

```
...
10  #define PORTNUM 9001
11
12  int main(void) {
13      char buf[256];
14      struct sockaddr_in sin, cli;
15      int sd, ns, clientlen = sizeof(cli);
16
17      memset((char *)&sin, '\0', sizeof(sin));
18      sin.sin_family = AF_INET;
19      sin.sin_port = htons(PORTNUM);
20      sin.sin_addr.s_addr = inet_addr("192.168.162.133");
21
22      if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
23          perror("socket");
24          exit(1);
25      }
26
27      if (bind(sd, (struct sockaddr *)&sin, sizeof(sin))) {
28          perror("bind");
29          exit(1);
30      }
31
32      if (listen(sd, 5)) {
33          perror("listen");
34          exit(1);
35      }
```

소켓 주소구조체 생성

소켓 생성

클라이언트 접속 대기

[예제 12-1] (1) 반복서버(서버)

```
37      while (1) {
38          if ((ns = accept(sd, (struct sockaddr *)&cli, &clientlen)) == -1) {
39              perror("accept");
40              exit(1);
41          }
42          sprintf(buf, "%s", inet_ntoa(cli.sin_addr));
43          printf("*** Send a Message to Client(%s)\n", buf);
44
45          strcpy(buf, "Welcome to Network Server!!!");
46          if (send(ns, buf, strlen(buf) + 1, 0) == -1) {
47              perror("send");
48              exit(1);
49          }
50
51          if (recv(ns, buf, strlen(buf), 0) == -1) {
52              perror("recv");
53              exit(1);
54          }
55          printf("** From Client : %s\n", buf);
56          close(ns);
57      }
58      close(sd);
59
60      return 0;
61  }
```

클라이언트 접속

클라이언트에 정보전송

클라이언트의 데이터 수신

[예제 12-1] (2) 반복서버(클라이언트)-client2.c

```
...
11  #define PORTNUM 9001
12
13  int main(void) {
14      int sd;
15      char buf[256];
16      struct sockaddr_in sin;
17
18      memset((char *)&sin, '\0', sizeof(sin));
19      sin.sin_family = AF_INET;
20      sin.sin_port = htons(PORTNUM);
21      sin.sin_addr.s_addr = inet_addr("192.168.162.133");
22
23      if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
24          perror("socket");
25          exit(1);
26      }
27
28      if (connect(sd, (struct sockaddr *)&sin, sizeof(sin))) {
29          perror("connect");
30          exit(1);
31      }
```

소켓 주소구조체 생성

소켓 생성

서버에 연결 요청

[예제 12-1] (2) 반복서버(클라이언트)

```
33      if (recv(sd, buf, sizeof(buf), 0) == -1) {
34            perror("recv");
35            exit(1);
36      }
37
38      printf("** From Server : %s\n", buf);
39
40      strcpy(buf, "I want a HTTP Service.");
41      if (send(sd, buf, sizeof(buf) + 1, 0) == -1) {
42            perror("send");
43            exit(1);
44      }
45
46      close(sd);
47
48      return 0;
49  }
```

서버의 데이터 수신

서버에 데이터 송신

서버

```
# ex12_1s.out
```

클라이언트

```
# ex12_1c.out
** From Server : Welcome to Network Server!!!
```

서버

```
# ex12_1s.out
*** Send a Message to Client(192.168.162.133)
** From Client : I want a HTTP Service.
```

클라이언트

```
# ex12_1s.out
*** Send a Message to Client(192.168.162.131)
** From Client : I want a FTP Service.
```

[예제 12-2] (1) 동시 동작 서버(서버)-server3.c

클라이언트 부분은 크게 달라질 게 없음.

```
...
10  #define PORTNUM 9002
11
12  int main(void) {
13      char buf[256];
14      struct sockaddr_in sin, cli;
15      int sd, ns, clientlen = sizeof(cli);
16
17      if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
18          perror("socket");
19          exit(1);
20      }
21
22      memset((char *)&sin, '\0', sizeof(sin));
23      sin.sin_family = AF_INET;
24       sin.sin_port = htons(PORTNUM);
25      sin.sin_addr.s_addr = inet_addr("192.168.162.133");
26
27      if (bind(sd, (struct sockaddr *)&sin, sizeof(sin))) {
28          perror("bind");
29          exit(1);
30      }
31
32      if (listen(sd, 5)) {
33          perror("listen");
34          exit(1);
35      }
```

[예제 12-2] (1) 동시 동작 서버(서버)

```
37      while (1) {
38          if ((ns = accept(sd, (struct sockaddr *)&cli, &clientlen)) == -1) {
39              perror("accept");
40              exit(1);
41          } 자식이 수행하도록 시킴.
42          switch (fork()) {
43              case 0:
44                  close(sd); 반드시 close()   자식 프로세스는 listen 하지 않으므로,
45                  strcpy(buf, "Welcome to Server");
46                  if (send(ns, buf, strlen(buf) + 1, 0) == -1) {
47                      perror("send");
48                      exit(1);
49                  }
50
51                  if (recv(ns, buf, strlen(buf), 0) == -1) {
52                      perror("recv");
53                      exit(1);
54                  }
55                  printf("** From Client: %s\n", buf);
56                  sleep(5);
57                  exit(0);
58          }
59          close(ns);   부모 프로세스는 데이터 통신을 하지 않으므로,
60      }
61
62      return 0;
63  }
```

fork로 자식 프로세스 생성

자식 프로세스가 클라이언트로 메시지 보내고 데이터 수신

[예제 12-3] (1) 동시동작서버 – server4.c

```
...
40      while (1) {
41          if ((ns = accept(sd, (struct sockaddr *)&cli,
                              &clientlen)) == -1) {
42              perror("accept");
43              exit(1);
44          }
45          printf("** Accept Client\n");
46
47          switch (fork()) {
48              case 0:
49                  printf("** Fork Client\n");
50                  close(sd);
51                  dup2(ns, STDIN_FILENO);
52                  dup2(ns, STDOUT_FILENO);
53                  close(ns);
54                  execl("./han", "han", (char *)0);
55          }
56          close(ns);
57      }
58
59      return 0;
60  }
```

표준 입력,

read 를 하게 되면, client 가 send 한 것을 받게 됨.

클라이언트의 요청 처리를 위한 별도의 프로그램(han) 실행

표준 출력으로 카피

화면에 출력되는 대신에 send 가 됨.

[예제 12-3] (2) 동시동작서버 – han 프로그램

sample1.c

sample1.c

```
01  #include <unistd.h>
02  #include <stdio.h>
03
04  int main(void) {
05      printf("Welcome to Server, from Han!");
06      sleep(5);
07
08      return 0;
09  }
```

출력이 client 쪽으로 send 가 됨.

간단한 환영메시지 출력

[예제 12-3] (3) 동시동작서버 – client4.c

```
...
28      printf("==> Create Socket\n");
29      if (connect(sd, (struct sockaddr *)&sin, sizeof(sin))) {
30          perror("connect");
31          exit(1);
32      }
33
34      printf("==> Connect Server\n");
35      if ((len = recv(sd, buf, sizeof(buf), 0)) == -1) {
36            perror("recv");
37            exit(1);
38      }
39      buf[len] = '\0';
40
41      printf("==> From Server : %s\n", buf);
42
43      close(sd);
44
45      return 0;
46  }
```

연결요청

메시지 수신

클라이언트

```
# ex12_3c.out
==> Create Socket
==> Connect Server
==> From Server : Welcome to Server, from Han!
```

```
# ps
PID TTY          TIME CMD
676 pts/2        0:00 ksh
760 pts/2        0:00 ex12_3s.out
763 pts/2        0:00 han
```

han 실행

[예제 12-4] (1) 명령행인자로 소켓 기술자 전달하기(서버)

server5.c

```
...
40      while (1) {
41          if ((ns = accept(sd, (struct sockaddr *)&cli,
                                &clientlen)) == -1) {
42              perror("accept");
43              exit(1);
44          }
45          printf("** Accept Client\n");
46
47          switch (fork()) {
48              case 0:
49                  printf("** Fork Client\n");
50                  close(sd);
51                  sprintf(buf, "%d", ns);
52                  execlp("./bit", "bit", buf, (char *)0);
53                  close(ns);
54          }
55          close(ns);
56      }
57
58      return 0;
59  }
```

클라이언트 접속 수용

bit프로그램 실행
명령행 인자로 소켓 전달

bit 실행파일의 명령행 인자로 전달.
자식이 사용하는 exec 파일로 전달.

[예제 12-4] (2) 명령행인자로 소켓 기술자 전달하기(bit)

sample2.c

sample2.c

```
...
08  int main(int argc, char *argv[]) {
09      char buf[256];
10      int len, ns;
11
12      ns = atoi(argv[1]);
13
14      strcpy(buf, "Welcome to Server, from Bit");
15      if ((send(ns, buf, strlen(buf) + 1, 0)) == -1) {
16          perror("send");
17          exit(1);
18      }
19
20      if ((len=recv(ns, buf, strlen(buf), 0)) == -1) {
21          perror("recv");
22          exit(1);
23      }
24      printf("@@ [Bit] From Client: %s\n", buf);
25      close(ns);
26
27      return 0;
28  }
```

명령행 인자로 받은 소켓을 숫자로 변환

클라이언트에 메시지 전달

클라이언트의 응답 받기

[예제 12-4] (3) 명령행인자로 소켓 기술자 전달(클라이언트)

client5.c

```
...
34      printf("==> Connect Server\n");
35 if ((len = recv(sd, buf, sizeof(buf), 0)) == -1) {
36          perror("recv");
37          exit(1);
38      }
39      buf[len] = '\0';
40
41      printf("==> From Server : %s\n", buf);
42
43      strcpy(buf, "I want a TELNET Service.");
44      if (send(sd, buf, sizeof(buf) + 1, 0) == -1) {
45            perror("send");
46            exit(1);
47      }
48
49      close(sd);
50
51      return 0;
52  }
```

서버의 메시지 수신

서버에 메시지 전송

[예제 12-4] 실행결과

서버

```
# ex12_4s.out
** Create Socket
** Bind Socket
** Listen Socket
** Accept Client
** Fork Client
@@ [Bit] From Client: I want a TELNET Service.
```

클라이언트

```
# ex12_4c.out
==> Create Socket
==> Connect Server
==> From Server : Welcome to Server, from Bit
```

[예제 12-5] (1) UDP 프로그래밍(서버) – server6.c

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

포트번호

UDP 프로토콜

소켓 생성(데이터그램)

소켓 주소 구조체 생성

소켓기술자와 소켓 주소 구조체 연결

[예제 12-5] (1) UDP 프로그래밍(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  }
```

서버에 클라이언트는 여러 개가 들어올 수 있으므로, IP 주소를 특정해야 하고.

클라이언트의 메시지 수신

클라이언트로 데이터 보내기

[예제 12-5] (2) UDP 프로그래밍(클라이언트) – client6.c

```
...
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

소켓 주소 구조체 생성

서버에 메시지 전송

[예제 12-5] (2) UDP 프로그래밍(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  }
```

NULL 을 쓰면 구분하지 않고 다 받음.
sendto 하면 서버로부터 받을 것이 뻔하므로,

서버가 보낸 데이터 읽기

서버

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

클라이언트

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