7. Homework

12201922 이규민

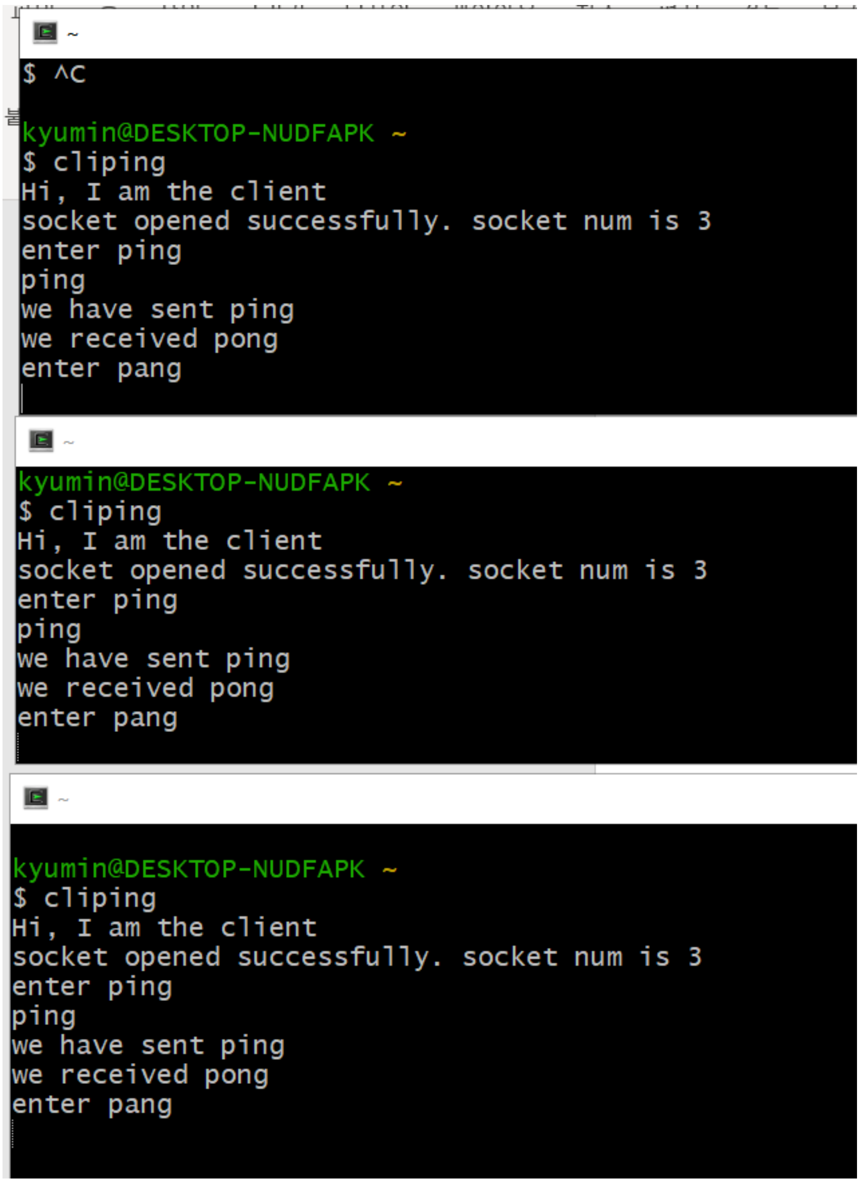
1) Download cliping.c and servping.c into your directory, modify IP and port number appropriately, and compile them. Run the server first and run client 3 times each in different window. Check if the server can handle multiple clients at the same time.

cliping1 -> servping: ping

cliping2 -> servping: ping

cliping3 -> servping: ping

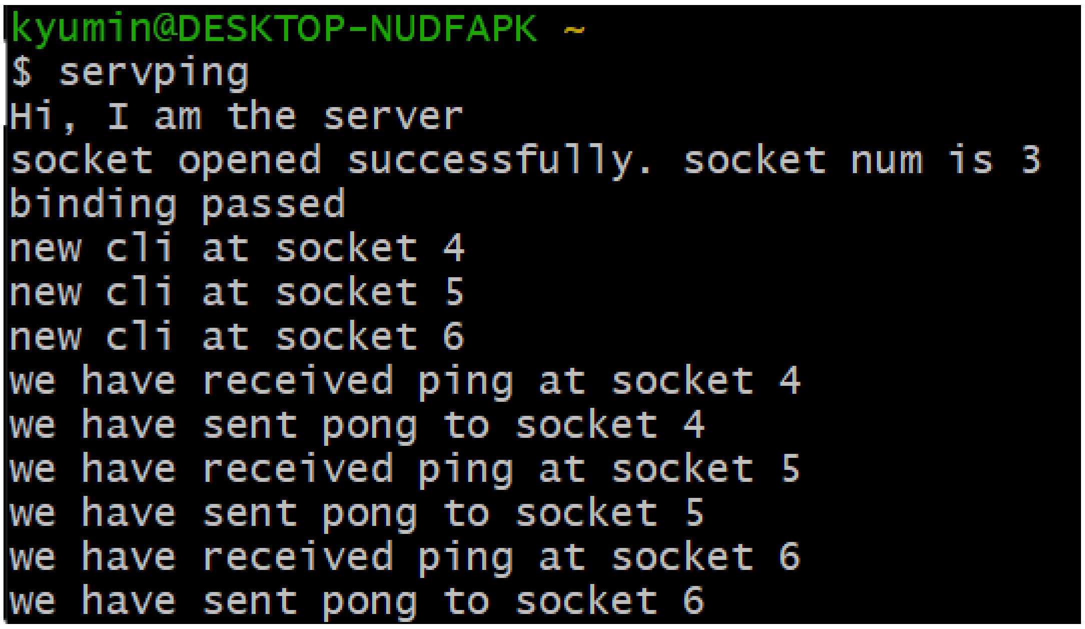
....



```
$ ^C

kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
ping
we have sent ping
we received pong
enter pang
```

```
kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
ping
we have sent ping
we received pong
enter pang
```

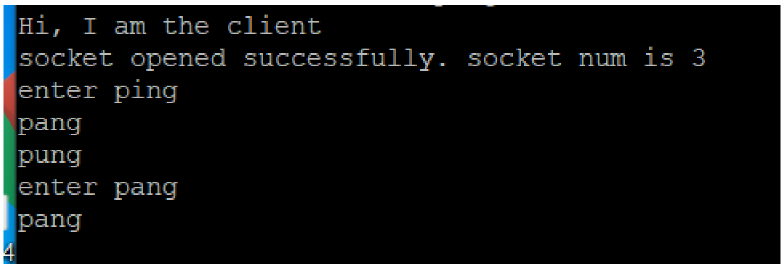
```
kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
ping
we have sent ping
we received pong
enter pang
```



```
kyumin@DESKTOP-NUDFAPK ~
$ servping
Hi, I am the server
socket opened successfully. socket num is 3
binding passed
new cli at socket 4
new cli at socket 5
new cli at socket 6
we have received ping at socket 4
we have sent pong to socket 4
we have received ping at socket 5
we have sent pong to socket 5
we have received ping at socket 6
we have sent pong to socket 6
```

먼저 서버를 실행한 후 여러 클라이언트를 동시에 실행했다. 그리고 각 클라이언트마다 “ping”이라는 메시지를 보냈고, 정상적으로 송신 및 수신이 되는 것을 볼 수 있다.

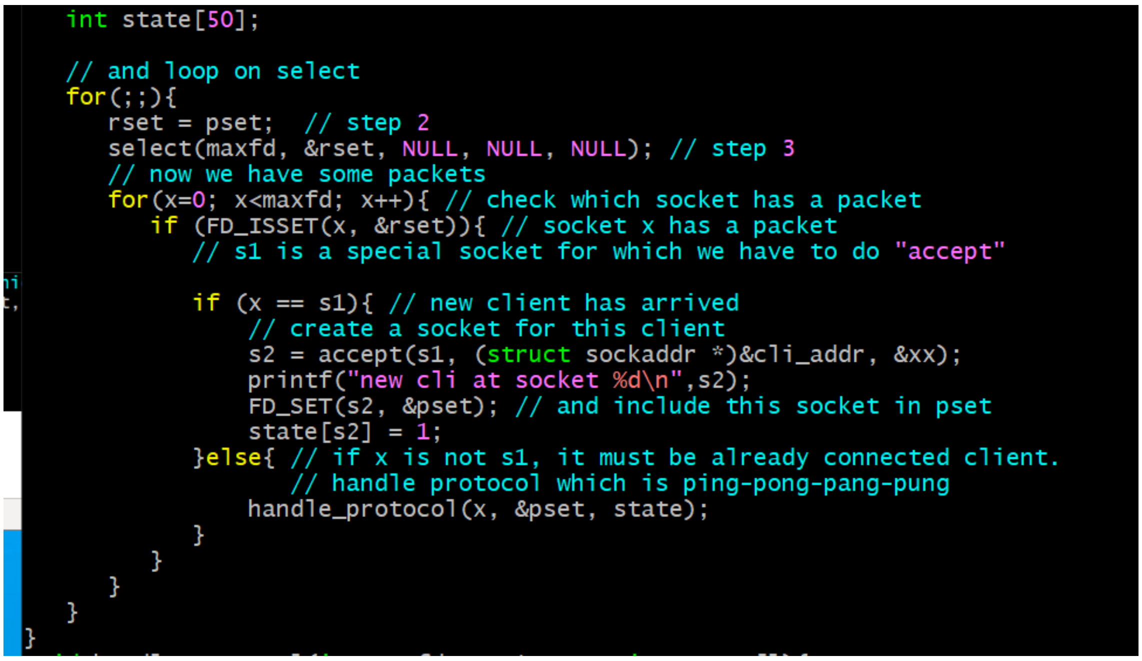
2) The server in Prob 1) cannot give error message to clients even when the client doesn't follow the protocol. Run server and run client and let the client send "pang" instead of "ping" as the first message. The server gives "pung" instead of error message as below.



```
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
pang
pung
enter pang
pang
```

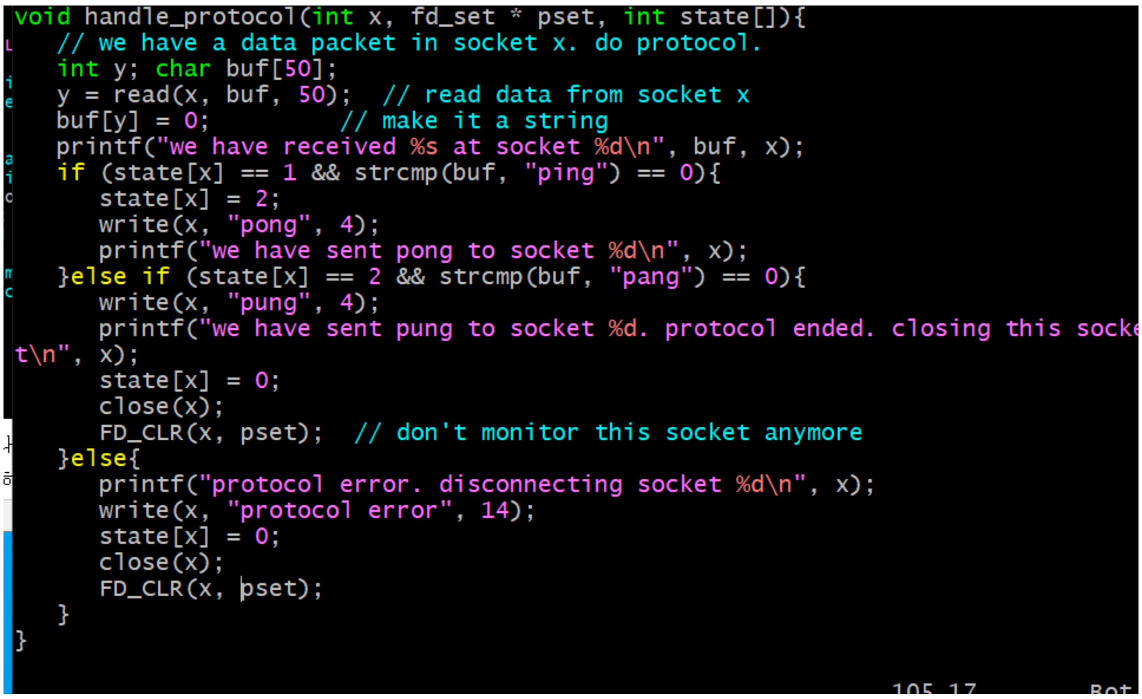
Modify servping.c so that it can send error message when the client sends something other than "ping" for the first message. You need to remember the state of each socket to do this. The beginning state of each socket is 1 which means this socket is waiting for the first message “ping”. If a socket at state 1 receives a message other than “ping”, the server should send protocol error message. Try to solve this problem by yourself and look at the code in Problem 2-1) if you need help.

[servping.c]



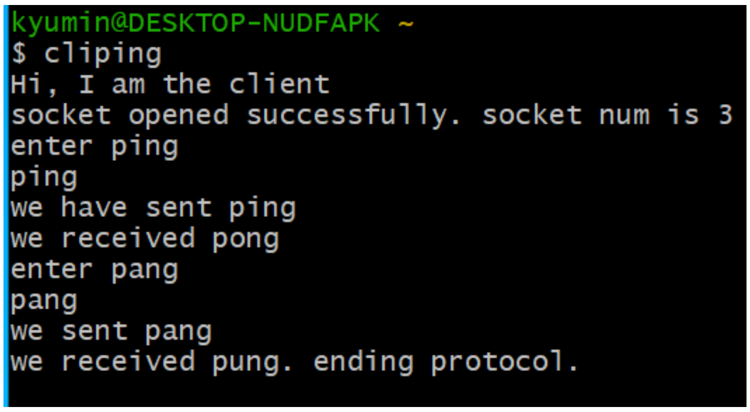
```
    int state[50];

    // and loop on select
    for(;;){
       rset = pset;  // step 2
       select(maxfd, &rset, NULL, NULL, NULL); // step 3
       // now we have some packets
       for(x=0; x<maxfd; x++){ // check which socket has a packet
          if (FD_ISSET(x, &rset)){ // socket x has a packet
             // s1 is a special socket for which we have to do "accept"

             if (x == s1){ // new client has arrived
                 // create a socket for this client
                 s2 = accept(s1, (struct sockaddr *)&cli_addr, &xx);
                 printf("new cli at socket %d\n",s2);
                 FD_SET(s2, &pset); // and include this socket in pset
                 state[s2] = 1;
             }else{ // if x is not s1, it must be already connected client.
                   // handle protocol which is ping-pong-pang-pung
                 handle_protocol(x, &pset, state);
             }
          }
       }
    }
}
```

함수 state를 배열 형태로 선언했다. 새로운 클라이언트와 연결이 될 때 s2 값이 몇 번째소켓인지 표시해주는 값인데 state[s2] = 1을 하면 특정 클라이언트의 상태를 1로 표시할 수 있다.

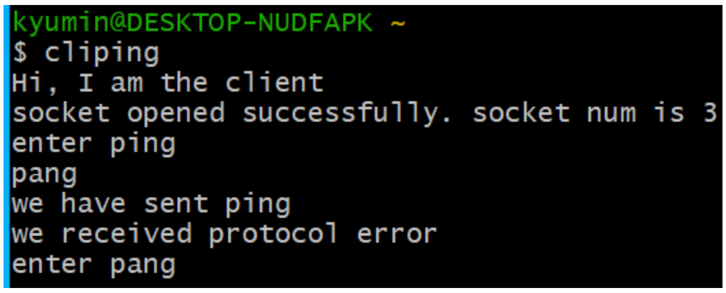


```
void handle_protocol(int x, fd_set * pset, int state[]){
   // we have a data packet in socket x. do protocol.
   int y; char buf[50];
   y = read(x, buf, 50);  // read data from socket x
   buf[y] = 0;           // make it a string
   printf("we have received %s at socket %d\n", buf, x);
   if (state[x] == 1 && strcmp(buf, "ping") == 0){
      state[x] = 2;
      write(x, "pong", 4);
      printf("we have sent pong to socket %d\n", x);
   }else if (state[x] == 2 && strcmp(buf, "pang") == 0){
      write(x, "pung", 4);
      printf("we have sent pung to socket %d. protocol ended. closing this socke
t\n", x);
      state[x] = 0;
      close(x);
      FD_CLR(x, pset);  // don't monitor this socket anymore
   }else{
      printf("protocol error. disconnecting socket %d\n", x);
      write(x, "protocol error", 14);
      state[x] = 0;
      close(x);
      FD_CLR(x, pset);
   }
}
                                                              105,17        Bot
```

연결되어있는 클라이언트로부터 메시지를 입력 받으면 프로토콜이 실행되는데 이 때 x 또한 몇 번째 소켓인지 표시해주는 값이다. 프로토콜이 실행되고, state값에 따라 몇 번째 입력된 메시지인지 확인할 수 있다. 이 때 state가 1이면서 ping이 입력되어야만 pong을 반환해주고, 아니라면 프로토콜 에러 메시지를 출력하며 연결을 끊는다.



```
kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
ping
we have sent ping
we received pong
enter pang
pang
we sent pang
we received pung. ending protocol.
```



```
kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
pang
we have sent ping
we received protocol error
enter pang
```

Ping 이 입력되었다면 정상 작동하지만 pang 이 입력된다면 에러 메시지가 출력된다.

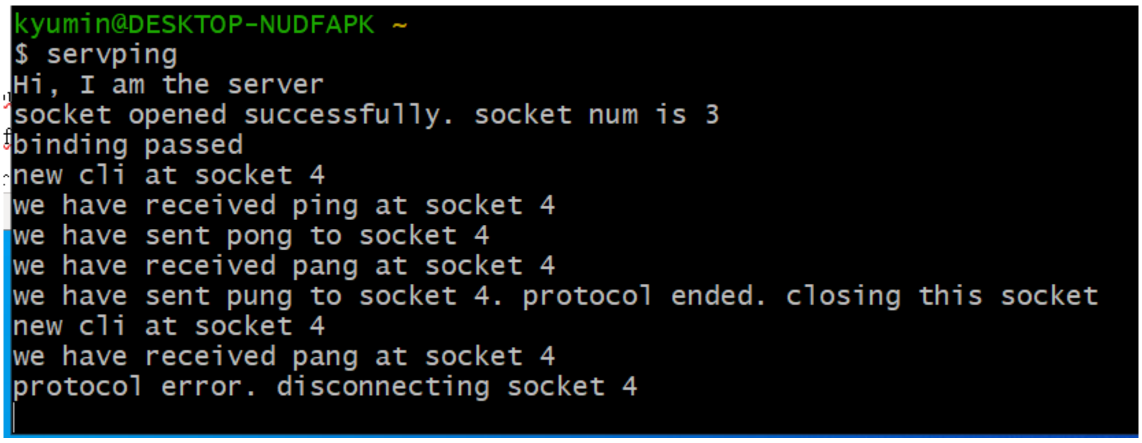
2-1) Modify the server such that it disconnects the connection if the client doesn't follow the protocol. You need to keep track of the state of each client to do this. (The client will act strange when the server disconnects it. You don't have to change the client code for this since we don't care about what happens to the client when it does not follow the protocol.)

```
    int state[50]; // state of each client (state of each client socket)
                   // 1: the server is waiting for "ping" from this client
                   // 2: the server is waiting for "pang" from this client
    ...........
    for(x=0;x<maxfd;x++){ // check all fd
        if (FD_ISSET(x, &rset)){ // if we have packet in socket x
            if (x==s1){ // if x is the connection accepting socket, we have a new client
                        // and we must have the connection request packet(SYN) at x
                s2=accept(s1, ........); // now s2 is this client's socket
                printf("new cli at socket %d. it's sate is 1₩n", s2);
                state[s2]=1;   // init the state of this client.
                               // the server is expecting "ping" from this client
                FD_SET(s2, &pset);
            }else{ // we must have the data packet at socket x
                handle_protocol(x, &pset, state);
            } // else
        }//if
    }//for
    ................
void handle_protocol(int x, fd_set * pset, int state[]){
// we have data packet in socket x. state[x] shows the state of socket x.
// handle the protocol.
    int y; char buf[50];
    y=read(x, buf, 50); // read the data
    buf[y]=0; // make it a string
   printf("we have received %s at socket %d₩n", buf, x);
    if (state[x]==1){ // the state of this socket is 1 meaning we are
                      // expecting "ping" from this socket
        handle_state_1(x, pset, buf, state);
    }else if (state[x]==2){ // expecting "pang"
```

```
         handle_state_2(x, pset, buf, state);
     }
}
void handle_state_1(int x, fd_set *pset, char* buf, int state[]){
// socket x is in state 1. Expecting "ping" in buf. if we have ping, send "pong" and
// just update state[x]=2; otherwise send error message and disconnect the connection
       printf("socket %d is in state 1 waiting for ping₩n", x);
       if (strcmp(buf, "ping")==0){ // yes we have "ping"
           printf("yes it is ping. send pong₩n");
           write(x, "pong", 4);   // send pong to this client
           printf("we sent pong to socket %d. now state of socket %d is 2₩n", x, x);
           state[x]=2; // now we are waiting for "pang" from this client
       }else{ // no we didn't receive "ping"
           printf("protocol error. disconnected skt %d₩n", x);
           write(x, "protocol error", 14); // send err message to the client
           close(x);     // end the connection
           FD_CLR(x, pset); // remove from the watch list.
                              // we don't monitor socket x any more
       }
}
void handle_state_2(int x, fd_set *pset, char* buf, int state[]){
// socket x is in state 2. we are expecting "pang" in buf. If we have "pang", send "pung"
// and close the connection. If we didn't receive "pang", send "protocol error" to the
// client and disconnect.
printf("socket %d is in state 2 waiting for pang₩n", x);
       if (strcmp(buf, "pang")==0){ // yes we have "pang"
           printf("yes it is pang. send pung₩n");
           write(x, "pung", 4);   // send pong to this client
           printf("we sent pung to socket %d. protocol done₩n", x);
           close(x);
           FD_CLR(x, pset);

       }else{ // no we didn't receive "pang"
           printf("protocol error. disconnected skt %d₩n", x);
           write(x, "protocol error", 14); // send err message to the client
           close(x);     // end the connection
```

```
            FD_CLR(x, pset); // remove from the watch list.
                             // we don't monitor socket x any more
        }
}
```

[servping 결과]



```
kyumin@DESKTOP-NUDFAPK ~
$ servping
Hi, I am the server
socket opened successfully. socket num is 3
binding passed
new cli at socket 4
we have received ping at socket 4
we have sent pong to socket 4
we have received pang at socket 4
we have sent pung to socket 4. protocol ended. closing this socket
new cli at socket 4
we have received pang at socket 4
protocol error. disconnecting socket 4
```

이전 문제에서 만든 코드 그대로 실행하면 된다. 클라이언트와 연결이 되고, ping 과 pang 이 순서대로 입력되면 정상적으로 진행 후 종료된다.

그러나 pang이 먼저 입력된 경우 에러가 발생한 것을 볼 수 있다.

3) Modify the protocol such that the server expects a final “ping” again from the client. Make sure the server give error message and disconnect the client if the client doesn't follow the protocol.

cli=>serv: ping

serv=>cli: pong

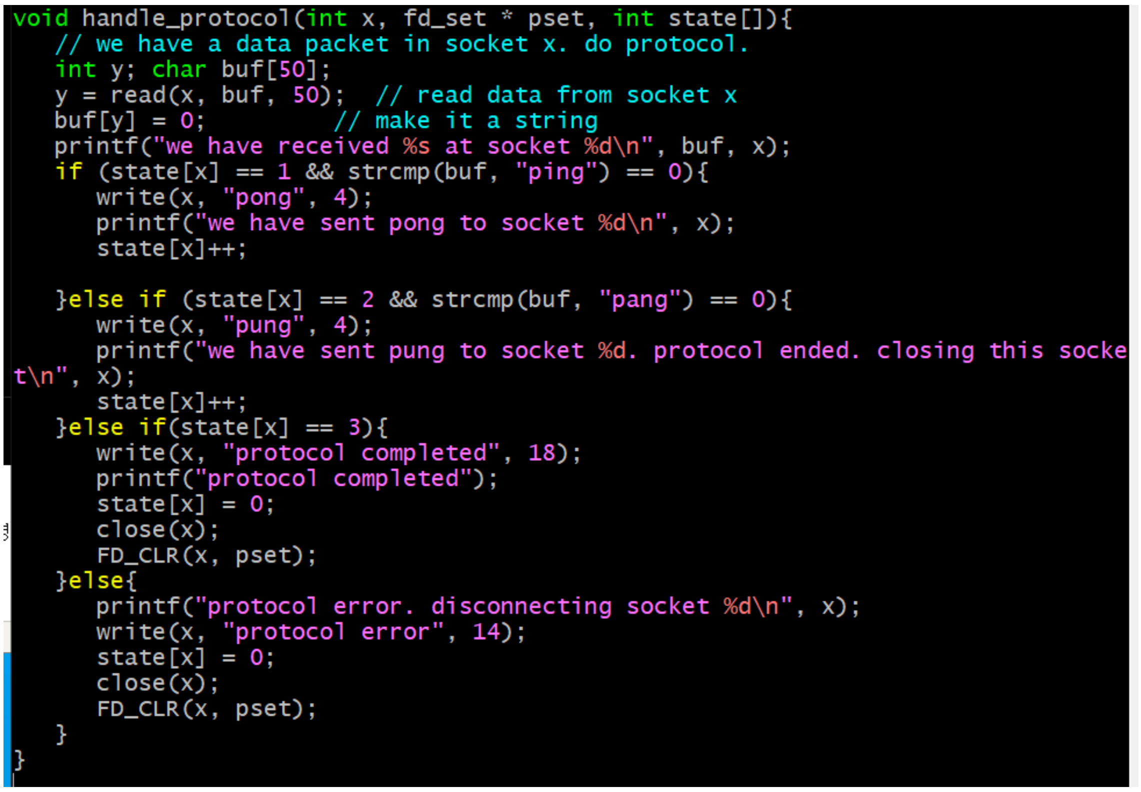
cli=>serv: pang

serv=>cli: pung

cli=>serv: ping (final ping)

serv=>cli: protocol completed

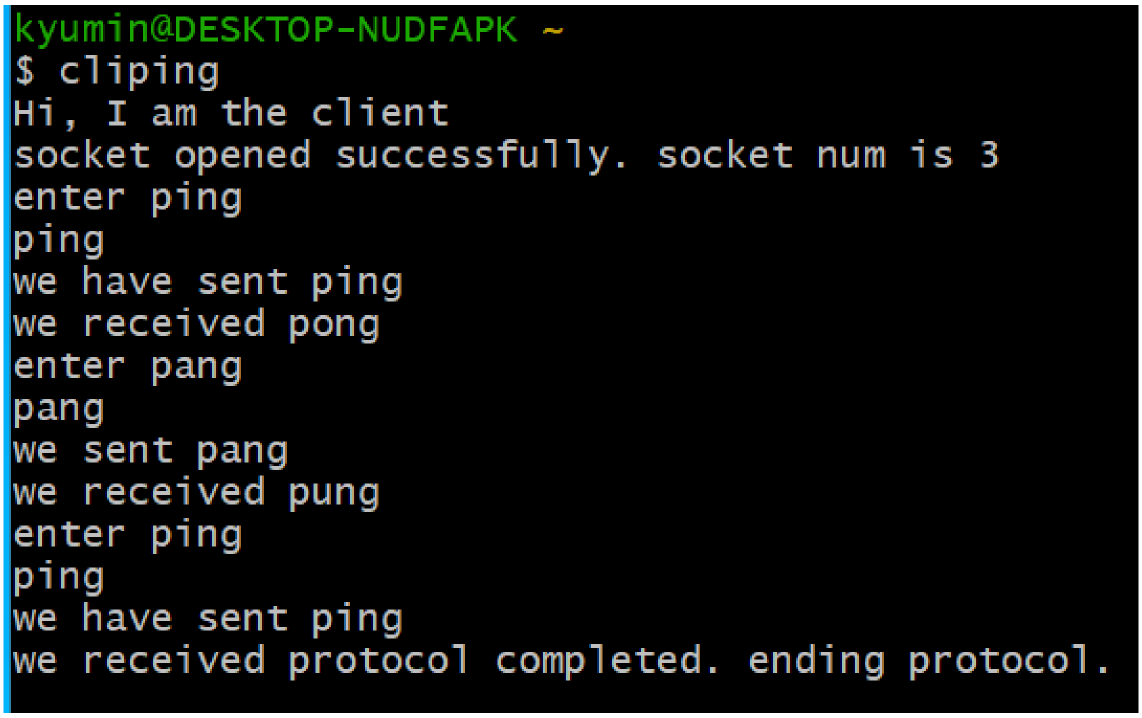
[servping.c]



```
void handle_protocol(int x, fd_set * pset, int state[]){
   // we have a data packet in socket x. do protocol.
   int y; char buf[50];
   y = read(x, buf, 50);  // read data from socket x
   buf[y] = 0;          // make it a string
   printf("we have received %s at socket %d\n", buf, x);
   if (state[x] == 1 && strcmp(buf, "ping") == 0){
      write(x, "pong", 4);
      printf("we have sent pong to socket %d\n", x);
      state[x]++;

   }else if (state[x] == 2 && strcmp(buf, "pang") == 0){
      write(x, "pung", 4);
      printf("we have sent pung to socket %d. protocol ended. closing this socke
t\n", x);
      state[x]++;
   }else if(state[x] == 3){
      write(x, "protocol completed", 18);
      printf("protocol completed");
      state[x] = 0;
      close(x);
      FD_CLR(x, pset);
   }else{
      printf("protocol error. disconnecting socket %d\n", x);
      write(x, "protocol error", 14);
      state[x] = 0;
      close(x);
      FD_CLR(x, pset);
   }
}
```

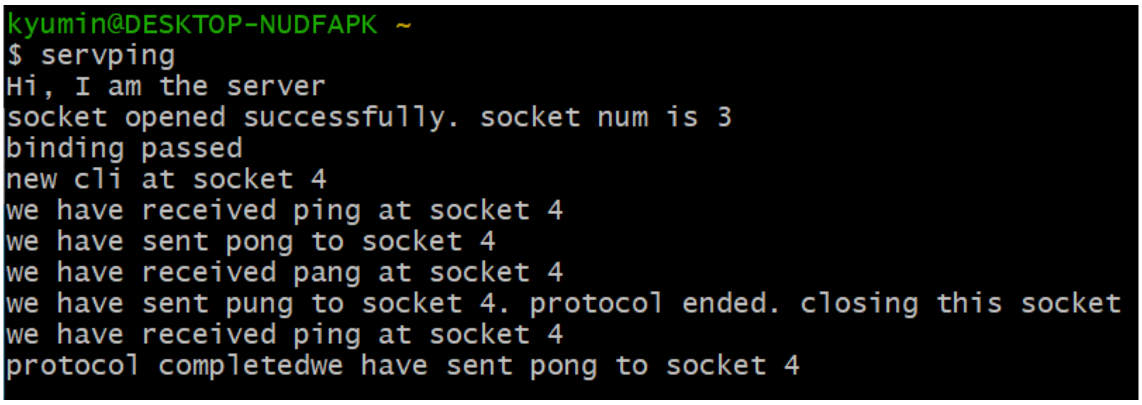
Ping 또는 pang 이 입력될 경우 state[x]++;을 해서 state를 바꿔준다.
상태가 3일 때 ping이 입력된다면 완료 메시지를 보내고, 통신을 끊는다.

[cliping 결과]



```
kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
ping
we have sent ping
we received pong
enter pang
pang
we sent pang
we received pung
enter ping
ping
we have sent ping
we received protocol completed. ending protocol.
```

[servping결과]



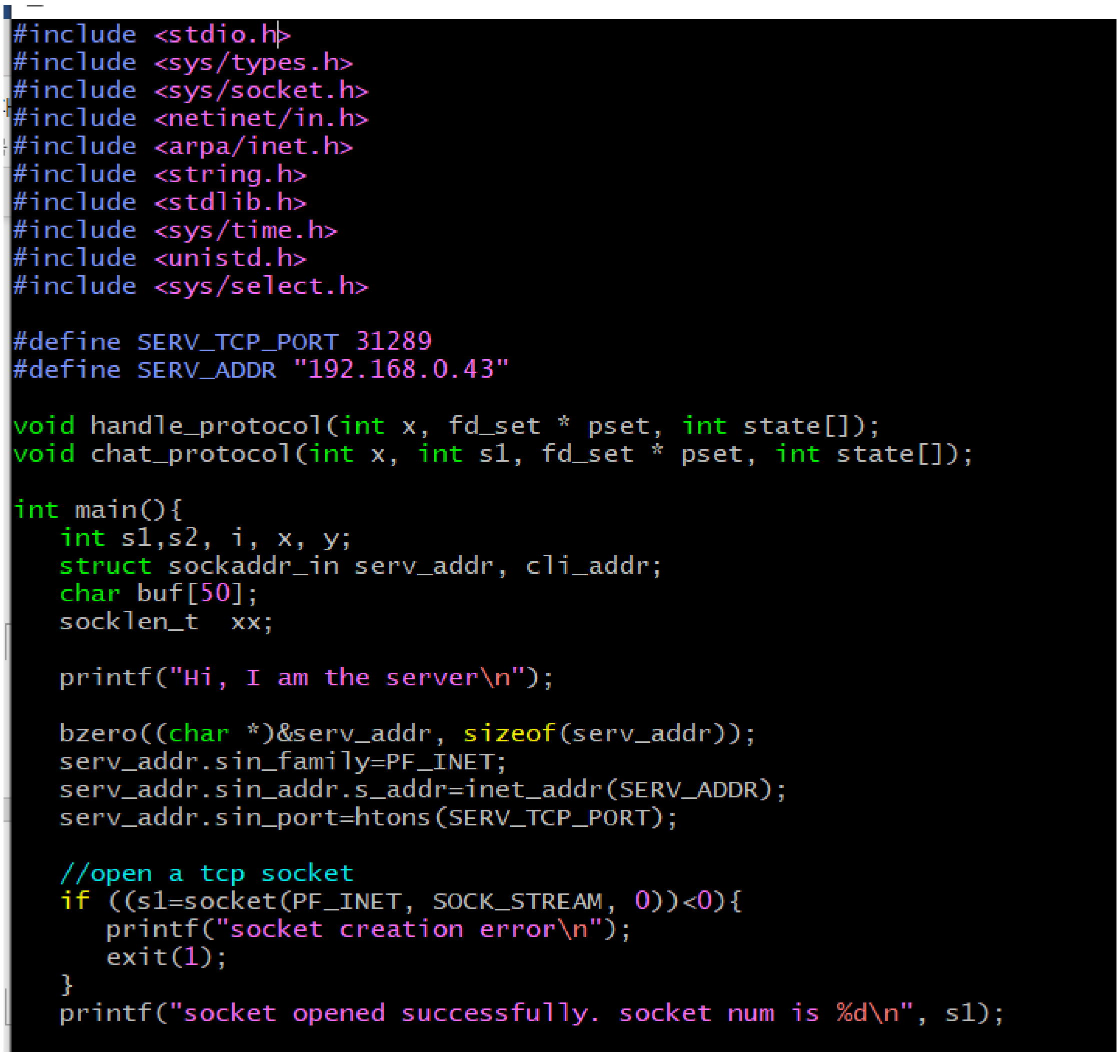
```
kyumin@DESKTOP-NUDFAPK ~
$ servping
Hi, I am the server
socket opened successfully. socket num is 3
binding passed
new cli at socket 4
we have received ping at socket 4
we have sent pong to socket 4
we have received pang at socket 4
we have sent pung to socket 4. protocol ended. closing this socket
we have received ping at socket 4
protocol completedwe have sent pong to socket 4
```

Ping/pang/ping 입력하니 정상적으로 종료된 것을 볼 수 있다.

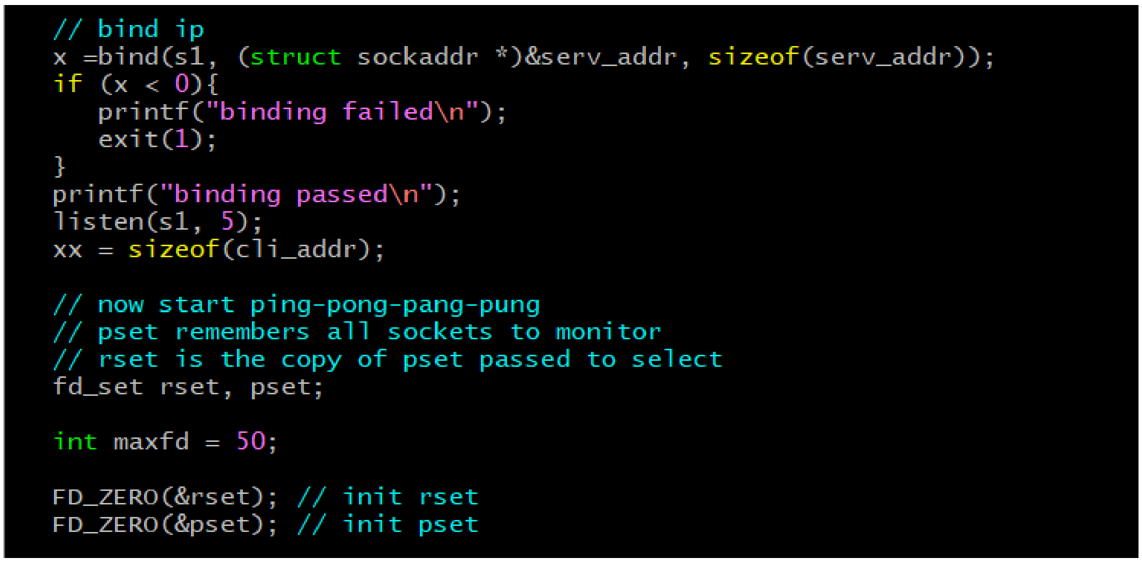
4) Modify the protocol such that the server relays a message from a client to all other clients after the “ping-pong-pang-pung” sequence is completed. The clients should fork itself after the “ping-pong-pang-pung” sequence so that the parent part keeps reading while the child part keeps writing. The server does not fork since it doesn't do the chatting by itself; it just relays a message from one client to all other clients. The server checks state[] array to see which socket is ready to receive message.

```
cli at socket 3 => serv: ping
serv => cli at socket 3 : pong
cli at socket 3 => serv: pang
serv => cli at socket 3 : pung. Protocol completed. Start chatting.

cli at socket 4 => serv: ping
serv => cli at socket 4 : pong
cli at socket 4 => serv: pang
serv => cli at socket 4 : pung. Protocol completed. Start chatting.

cli at socket 5 => serv: ping
serv => cli at socket 5 : pong
cli at socket 5 => serv: pang
serv => cli at socket 5 : pung. Protocol completed. Start chatting.

cli at socket 3 => serv: hello
serv => cli at socket 4, 5 : hello
cli at socket 4 => serv: hi
serv => cli at socket 3, 5: hi
.................
```

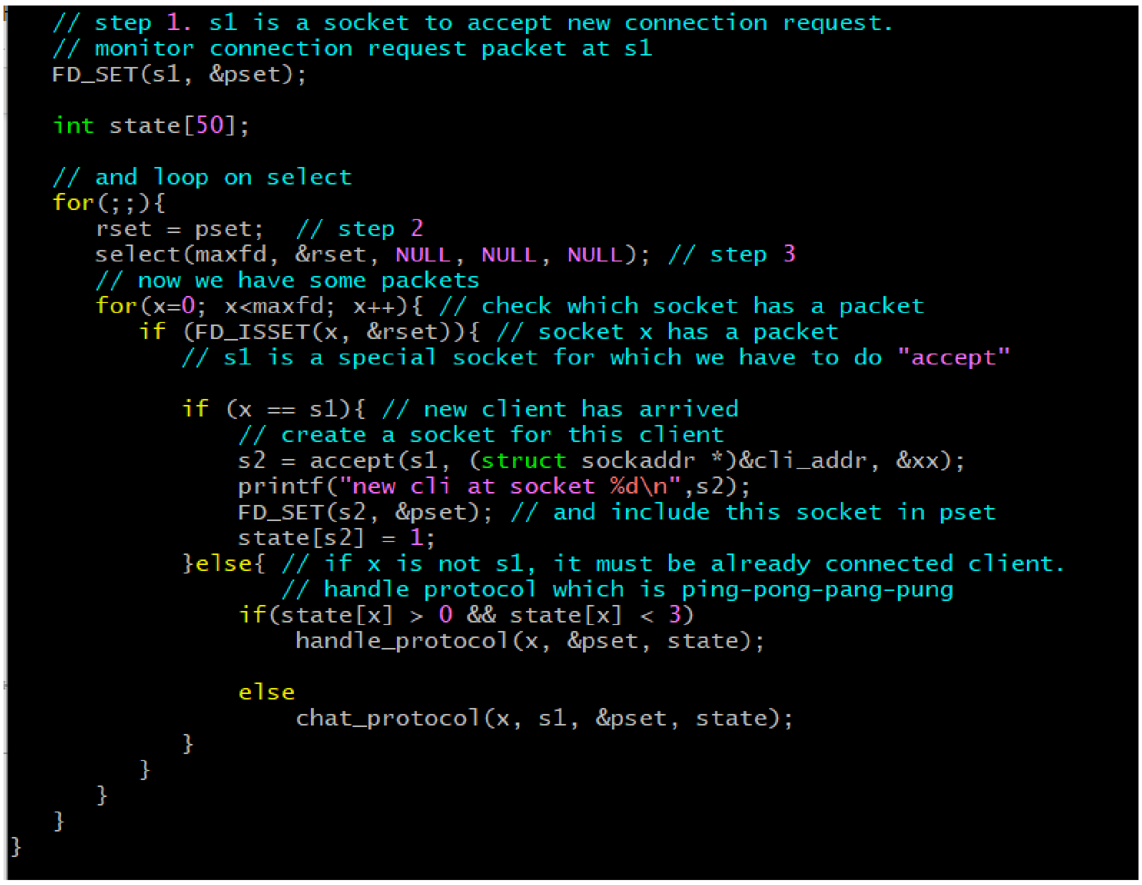
[servping.c]



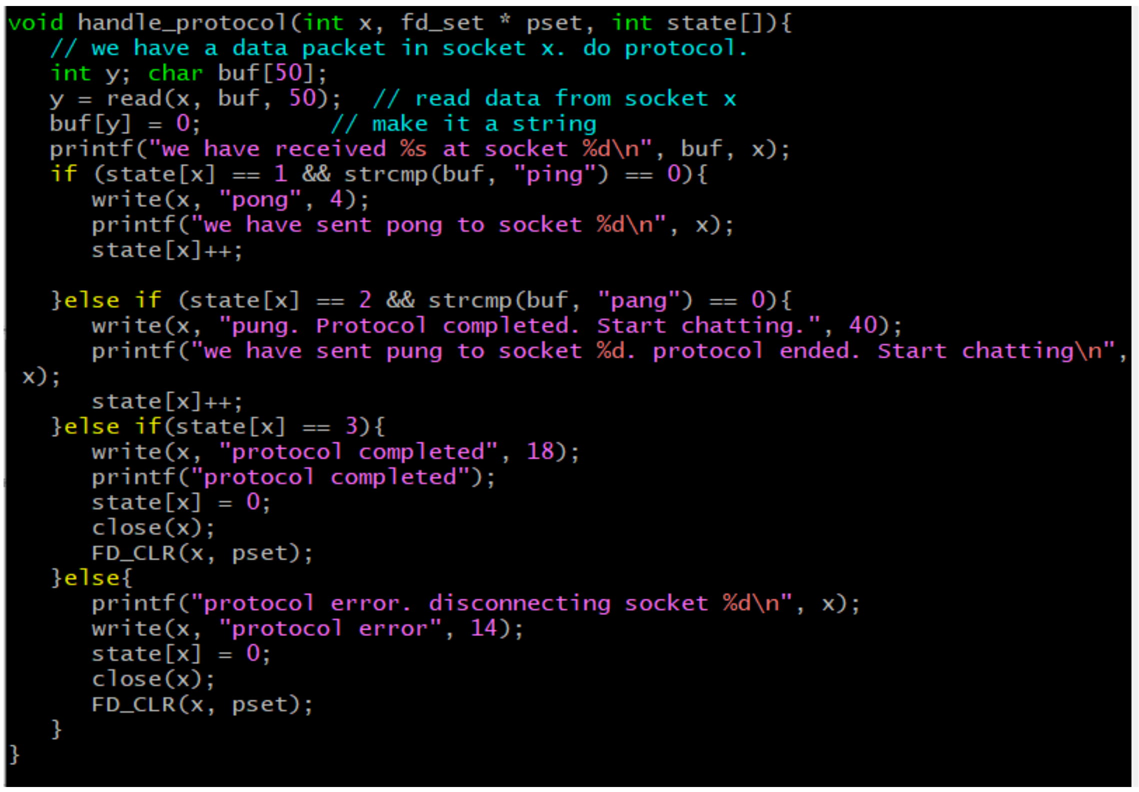
```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdlib.h>
#include <sys/time.h>
#include <unistd.h>
#include <sys/select.h>

#define SERV_TCP_PORT 31289
#define SERV_ADDR "192.168.0.43"

void handle_protocol(int x, fd_set * pset, int state[]);
void chat_protocol(int x, int s1, fd_set * pset, int state[]);

int main(){
   int s1,s2, i, x, y;
   struct sockaddr_in serv_addr, cli_addr;
   char buf[50];
   socklen_t  xx;

   printf("Hi, I am the server\n");

   bzero((char *)&serv_addr, sizeof(serv_addr));
   serv_addr.sin_family=PF_INET;
   serv_addr.sin_addr.s_addr=inet_addr(SERV_ADDR);
   serv_addr.sin_port=htons(SERV_TCP_PORT);

   //open a tcp socket
   if ((s1=socket(PF_INET, SOCK_STREAM, 0))<0){
      printf("socket creation error\n");
      exit(1);
   }
   printf("socket opened successfully. socket num is %d\n", s1);
```



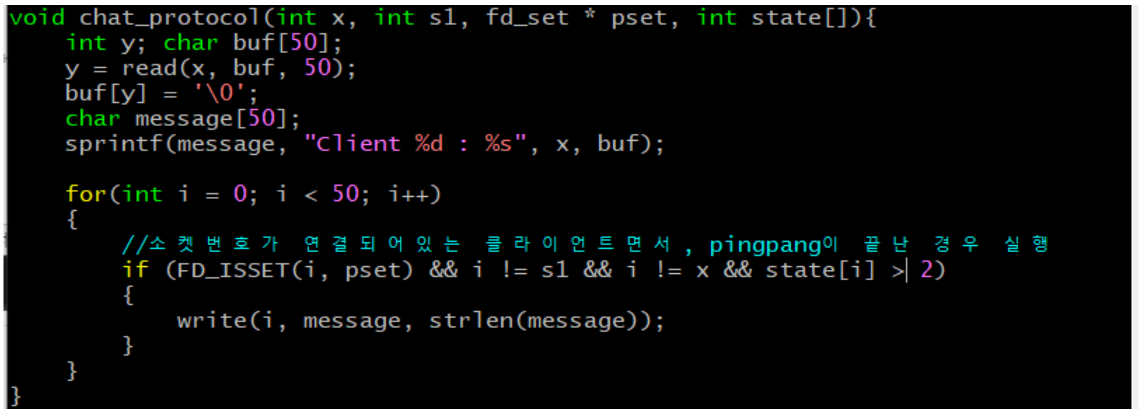
```c
   // bind ip
   x =bind(s1, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
   if (x < 0){
      printf("binding failed\n");
      exit(1);
   }
   printf("binding passed\n");
   listen(s1, 5);
   xx = sizeof(cli_addr);

   // now start ping-pong-pang-pung
   // pset remembers all sockets to monitor
   // rset is the copy of pset passed to select
   fd_set rset, pset;

   int maxfd = 50;

   FD_ZERO(&rset); // init rset
   FD_ZERO(&pset); // init pset
```



```c
   // step 1. s1 is a socket to accept new connection request.
   // monitor connection request packet at s1
   FD_SET(s1, &pset);

   int state[50];

   // and loop on select
   for(;;){
      rset = pset;  // step 2
      select(maxfd, &rset, NULL, NULL, NULL); // step 3
      // now we have some packets
      for(x=0; x<maxfd; x++){ // check which socket has a packet
         if (FD_ISSET(x, &rset)){ // socket x has a packet
            // s1 is a special socket for which we have to do "accept"

            if (x == s1){ // new client has arrived
                // create a socket for this client
                s2 = accept(s1, (struct sockaddr *)&cli_addr, &xx);
                printf("new cli at socket %d\n",s2);
                FD_SET(s2, &pset); // and include this socket in pset
                state[s2] = 1;
            }else{ // if x is not s1, it must be already connected client.
                   // handle protocol which is ping-pong-pang-pung
                if(state[x] > 0 && state[x] < 3)
                    handle_protocol(x, &pset, state);

                else
                    chat_protocol(x, s1, &pset, state);
            }
         }
      }
   }
}
```



```c
void handle_protocol(int x, fd_set * pset, int state[]){
   // we have a data packet in socket x. do protocol.
   int y; char buf[50];
   y = read(x, buf, 50);  // read data from socket x
   buf[y] = 0;           // make it a string
   printf("we have received %s at socket %d\n", buf, x);
   if (state[x] == 1 && strcmp(buf, "ping") == 0){
      write(x, "pong", 4);
      printf("we have sent pong to socket %d\n", x);
      state[x]++;

   }else if (state[x] == 2 && strcmp(buf, "pang") == 0){
      write(x, "pung. Protocol completed. Start chatting.", 40);
      printf("we have sent pung to socket %d. protocol ended. Start chatting\n",
 x);
      state[x]++;
   }else if(state[x] == 3){
      write(x, "protocol completed", 18);
      printf("protocol completed");
      state[x] = 0;
      close(x);
      FD_CLR(x, pset);
   }else{
      printf("protocol error. disconnecting socket %d\n", x);
      write(x, "protocol error", 14);
      state[x] = 0;
      close(x);
      FD_CLR(x, pset);
   }
}
```



```
void chat_protocol(int x, int s1, fd_set * pset, int state[]){
    int y; char buf[50];
    y = read(x, buf, 50);
    buf[y] = '\0';
    char message[50];
    sprintf(message, "Client %d : %s", x, buf);

    for(int i = 0; i < 50; i++)
    {
        //소켓번호가 연결되어있는 클라이언트면서, pingpang이 끝난 경우 실행
        if (FD_ISSET(i, pset) && i != s1 && i != x && state[i] > 2)
        {
            write(i, message, strlen(message));
        }
    }
}
```

```
if(state[x] > 0 && state[x] < 3)
                    handle_protocol(x, &pset, state);
else
                    chat_protocol(x, s1, &pset, state);
```

을 보면 state 에 따라 handle 프로토콜이 실행되거나 chat 프로토콜이 실행된다.

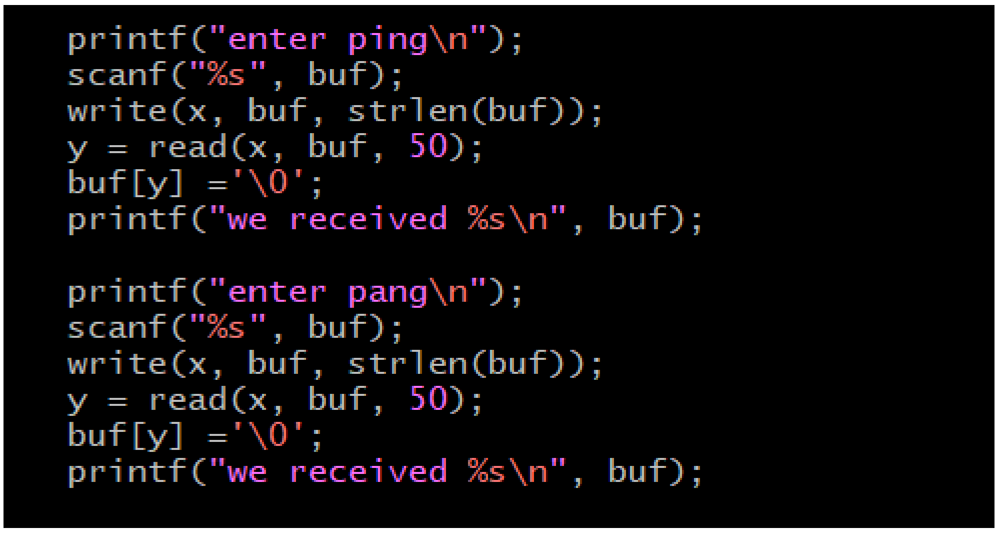
우선 handle 프로토콜에서는 ping pong 이 이뤄질 때 마다 state 값을 올려준다.

State 가 3 이상이 된다면 chat 프로토콜이 실행이 가능한데, 여기선 우선 클라이언트의 메시지를 읽는다. 그리고 그 메시지를 모든 사용자한테 보내준다.

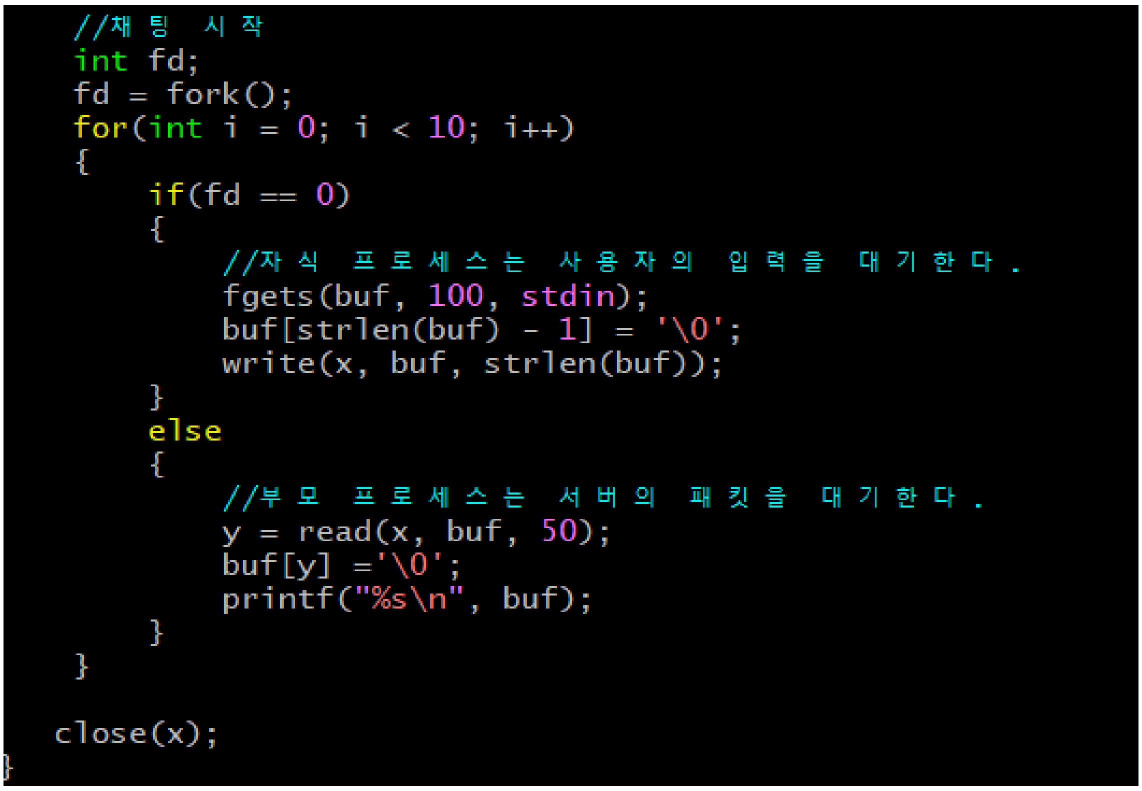
```
if (FD_ISSET(i, pset) && i != s1 && i != x && state[i] > 2)
```

사용자 확인을 위해서 조건은 위와 같이 넣었다. Pset 의 상태가 true 여야 하며, s1 과 x 가 아니고 state 는 채팅을 할 수 있는 값이여야 한다. 만약 모든 조건을 만족한다면 message 를 해당 클라이언트로 보낸다.

[clipping.c]



```
    printf("enter ping\n");
    scanf("%s", buf);
    write(x, buf, strlen(buf));
    y = read(x, buf, 50);
    buf[y] ='\0';
    printf("we received %s\n", buf);

    printf("enter pang\n");
    scanf("%s", buf);
    write(x, buf, strlen(buf));
    y = read(x, buf, 50);
    buf[y] ='\0';
    printf("we received %s\n", buf);
```

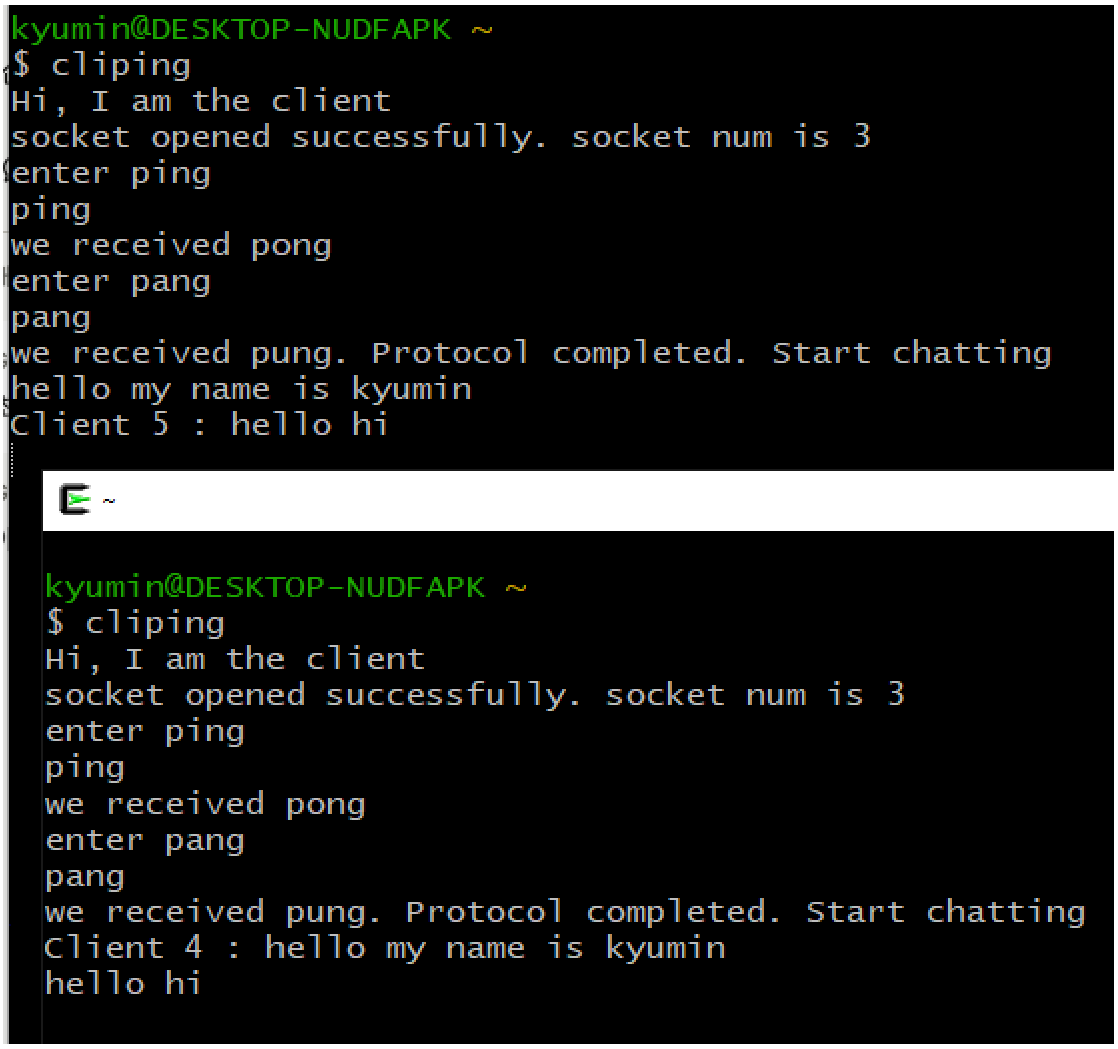


```
    //채팅 시작
    int fd;
    fd = fork();
    for(int i = 0; i < 10; i++)
    {
        if(fd == 0)
        {
            //자식 프로세스는 사용자의 입력을 대기한다.
            fgets(buf, 100, stdin);
            buf[strlen(buf) - 1] = '\0';
            write(x, buf, strlen(buf));
        }
        else
        {
            //부모 프로세스는 서버의 패킷을 대기한다.
            y = read(x, buf, 50);
            buf[y] ='\0';
            printf("%s\n", buf);
        }
    }

   close(x);
}
```

우선 ping pong 을 진행하고, 채팅을 시작한다.

Fgets 는 사용자의 입력이 있을 때까지 다음 코드로 넘어가지 않고, read 함수도 패킷을 받기 전까지 다음 코드로 넘어가지 않는다. 즉 읽기와 쓰기를 동시에 할 수 없다.

그래서 fork 를 사용해서 두 개의 프로세스로 나눠준다. 자식 프로세스는 쓰기를 하고, 부모 프로세스는 읽기를 한다.



```
kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
ping
we received pong
enter pang
pang
we received pung. Protocol completed. Start chatting
hello my name is kyumin
Client 5 : hello hi
```

```
kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
ping
we received pong
enter pang
pang
we received pung. Protocol completed. Start chatting
Client 4 : hello my name is kyumin
hello hi
```

두 클라이언트를 실행한 후 ping pong 을 마치고 메시지를 입력해보았다. 서로 입력한 메시지가 화면에 나타나는 것을 볼 수 있다. 즉 채팅 프로그램을 구현한 것이다.

5) Modify your code in Problem 4) such that the server attaches the client's name and age in the message. For this purpose, the server should ask name and age for each client and store them in cli[] array which is an array of client{} structure to store name and age of each client. cli[x] will remember the client information whose socket number is x.

```
struct client{
    char name[20];   // this client's name
    char age[5];         // this client's age as a string
};
......
struct client cli[50];   // max 50 clients
```

```
 cli   aaa=> serv: ping
 serv => cli aaa: pong
 cli   aaa=> serv: pang
 serv => cli aaa: pung. name?
 cli   aaa=> serv: aaa
 serv => cli aaa: age?
 cli aaa => serv: 19

 cli   bbb=> serv: ping
 serv => cli bbb: pong
 cli   bbb=> serv: pang
 serv => cli bbb: pung. name?
 cli   bbb=> serv: bbb
 serv => cli aaa: age?
 cli aaa => serv: 22

 cli   ccc=> serv: ping
 serv => cli ccc: pong
 cli   ccc=> serv: pang
 serv => cli ccc: pung. name?
 cli   ccc=> serv: ccc
 serv => cli aaa: age?
 cli aaa => serv: 21
```

serv => cli aaa: start chatting

serv => cli bbb: start chatting

serv => cli bbb: start chatting

cli aaa => serv: "hello there"

serv=> cli bbb: "aaa 19 to bbb 22: hello there"

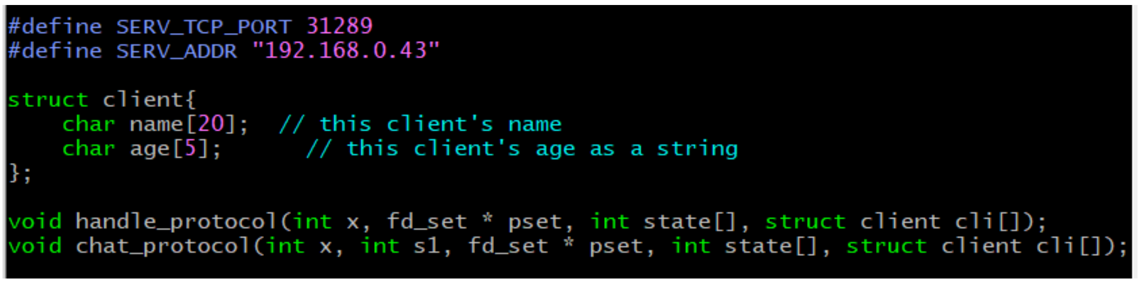
serv=> cli ccc: "aaa 19 to ccc 21: hello there"

cli bbb=> serv: "hi"

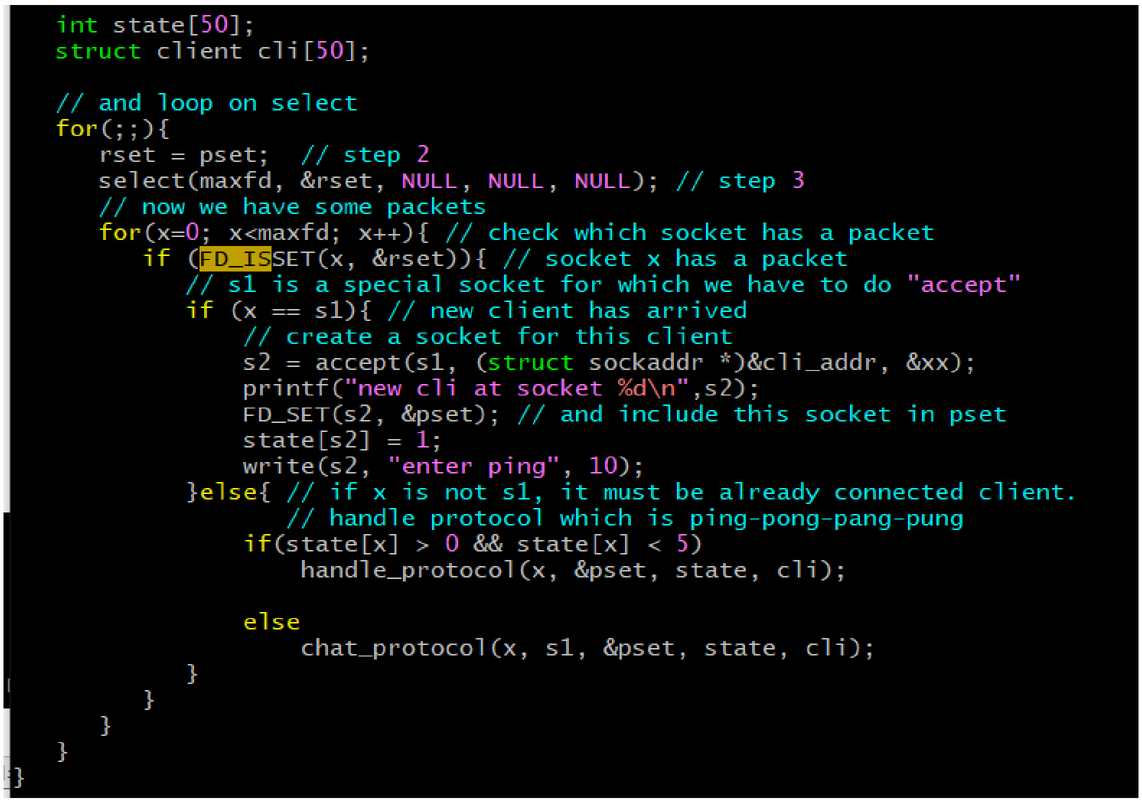
serv => cli aaa: "bbb 22 to aaa 19: hi"

serv => cli ccc: "bbb 22 to ccc 21: hi"

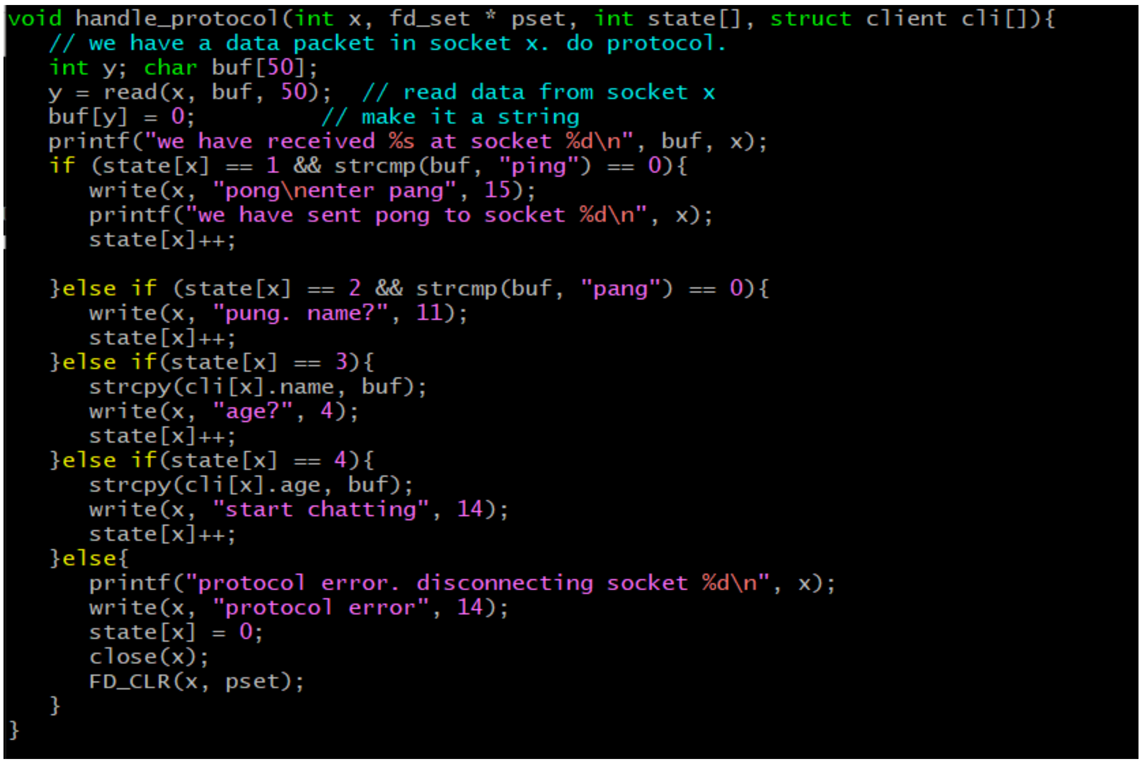
[servping.c]



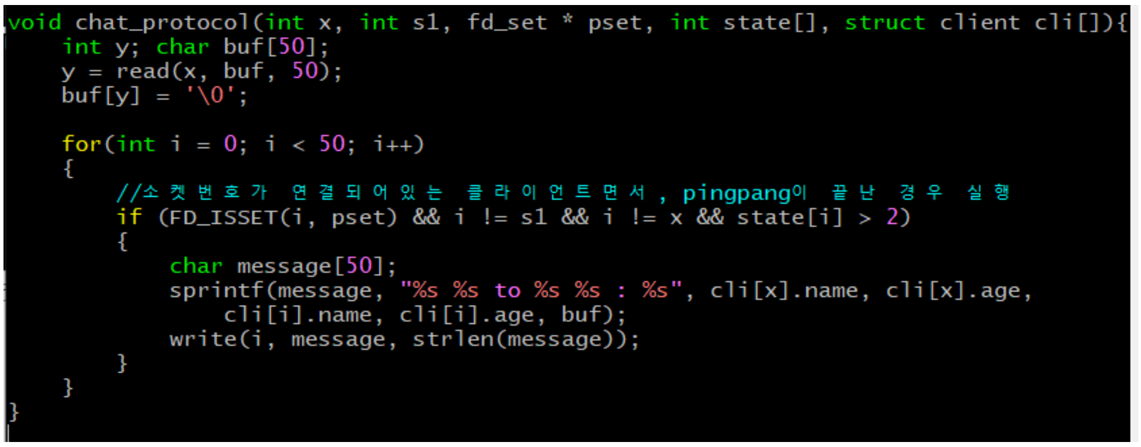
```c
#define SERV_TCP_PORT 31289
#define SERV_ADDR "192.168.0.43"

struct client{
    char name[20];  // this client's name
    char age[5];      // this client's age as a string
};

void handle_protocol(int x, fd_set * pset, int state[], struct client cli[]);
void chat_protocol(int x, int s1, fd_set * pset, int state[], struct client cli[]);
```



```c
   int state[50];
   struct client cli[50];

   // and loop on select
   for(;;){
      rset = pset;  // step 2
      select(maxfd, &rset, NULL, NULL, NULL); // step 3
      // now we have some packets
      for(x=0; x<maxfd; x++){ // check which socket has a packet
         if (FD_ISSET(x, &rset)){ // socket x has a packet
            // s1 is a special socket for which we have to do "accept"
            if (x == s1){ // new client has arrived
                // create a socket for this client
                s2 = accept(s1, (struct sockaddr *)&cli_addr, &xx);
                printf("new cli at socket %d\n",s2);
                FD_SET(s2, &pset); // and include this socket in pset
                state[s2] = 1;
                write(s2, "enter ping", 10);
            }else{ // if x is not s1, it must be already connected client.
                   // handle protocol which is ping-pong-pang-pung
                if(state[x] > 0 && state[x] < 5)
                    handle_protocol(x, &pset, state, cli);

                else
                    chat_protocol(x, s1, &pset, state, cli);
            }
         }
      }
   }
}
```



```c
void handle_protocol(int x, fd_set * pset, int state[], struct client cli[]){
   // we have a data packet in socket x. do protocol.
   int y; char buf[50];
   y = read(x, buf, 50);  // read data from socket x
   buf[y] = 0;           // make it a string
   printf("we have received %s at socket %d\n", buf, x);
   if (state[x] == 1 && strcmp(buf, "ping") == 0){
      write(x, "pong\nenter pang", 15);
      printf("we have sent pong to socket %d\n", x);
      state[x]++;

   }else if (state[x] == 2 && strcmp(buf, "pang") == 0){
      write(x, "pung. name?", 11);
      state[x]++;
   }else if(state[x] == 3){
      strcpy(cli[x].name, buf);
      write(x, "age?", 4);
      state[x]++;
   }else if(state[x] == 4){
      strcpy(cli[x].age, buf);
      write(x, "start chatting", 14);
      state[x]++;
   }else{
      printf("protocol error. disconnecting socket %d\n", x);
      write(x, "protocol error", 14);
      state[x] = 0;
      close(x);
      FD_CLR(x, pset);
   }
}
```

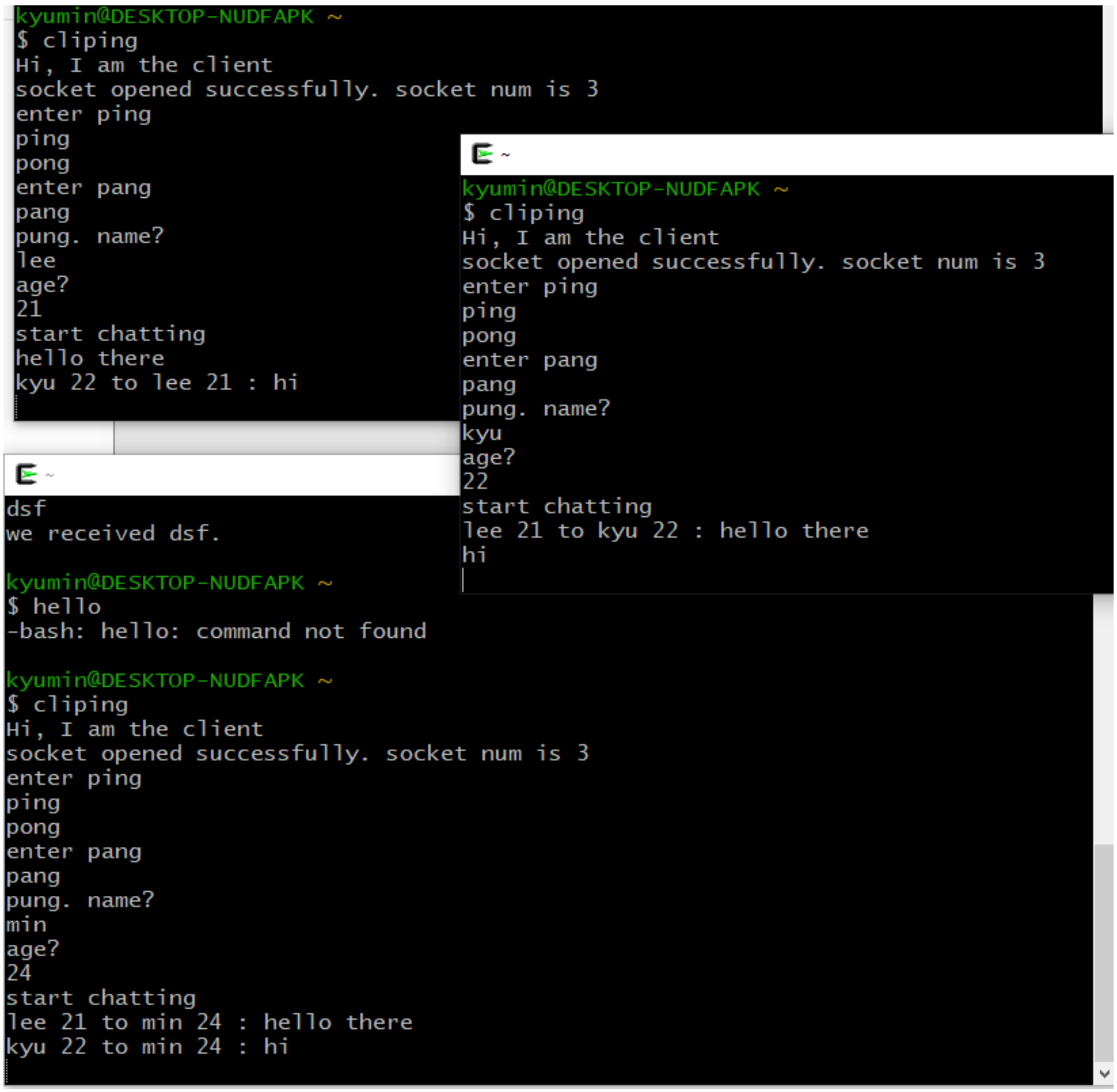


```c
void chat_protocol(int x, int s1, fd_set * pset, int state[], struct client cli[]){
    int y; char buf[50];
    y = read(x, buf, 50);
    buf[y] = '\0';

    for(int i = 0; i < 50; i++)
    {
        //소켓번호가 연결되어있는 클라이언트면서, pingpang이 끝난 경우 실행
        if (FD_ISSET(i, pset) && i != s1 && i != x && state[i] > 2)
        {
            char message[50];
            sprintf(message, "%s %s to %s %s : %s", cli[x].name, cli[x].age,
                cli[i].name, cli[i].age, buf);
            write(i, message, strlen(message));
        }
    }
}
```

```c
struct client cli[50];
```

구조체를 정의하고 선언할 때 client 의 수만큼 필요하므로 위와 같이 선언했다.

그리고 함수 내부에서는 이름을 저장하거나 출력할 때 사용된다.



```
kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
ping
pong
enter pang
pang
pung. name?
lee
age?
21
start chatting
hello there
kyu 22 to lee 21 : hi
```

```
kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
ping
pong
enter pang
pang
pung. name?
kyu
age?
22
start chatting
lee 21 to kyu 22 : hello there
hi
```

```
dsf
we received dsf.

kyumin@DESKTOP-NUDFAPK ~
$ hello
-bash: hello: command not found

kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
ping
pong
enter pang
pang
pung. name?
min
age?
24
start chatting
lee 21 to min 24 : hello there
kyu 22 to min 24 : hi
```

3개의 프로그램을 실행했다. 세 클라이언트는 각자 이름과 나이를 입력했다. 그리고 메시지를 입력하면 보낸 사람과 받는 사람의 이름/나이가 출력되고, 메시지가 출력된다.

6) Modify your code in Problem 5) such that the client can now specify which client it wants to chat with. Add "partner" to client{} strucure to remember the socket number of the chatting partner. The server should ask which partner the clients wants to talk with and remember the partner's socket number in the client{} structure. Assume if cli A points to cli B as a partner, cli B also points to cli A as a partner.

```
struct client{
    char name[20];   // this client's name
    char age[5];         // this client's age as a string
    int   partner;        // the socket number of the chatting partner of this client
};
```

```
 cli   aaa=> serv: ping
 serv => cli aaa: pong
 cli   aaa=> serv: pang
 serv => cli aaa: name?
 cli   aaa=> serv: aaa

 cli   bbb=> serv: ping
 serv => cli bbb: pong
 cli   bbb=> serv: pang
 serv => cli bbb: name?
 cli   bbb=> serv: bbb

 cli   ccc=> serv: ping
 serv => cli ccc: pong
 cli   ccc=> serv: pang
 serv => cli ccc: name?
 cli   ccc=> serv: ccc

 cli   ddd=> serv: ping
 serv => cli ddd: pong
 cli   ddd=> serv: pang
 serv => cli ddd: name?
 cli   ddd=> serv: ddd

 serv=>cli aaa: chat partner?
```

cli aaa=>serv: bbb

serv=>cli bbb: chat partner?

cli bbb=>serv: aaa

serv=>cli ccc: chat partner?

cli ccc=>serv: ddd

serv=>cli ddd: chat partner?

cli ddd=>serv: ccc

serv => cli aaa: start chatting

serv => cli bbb: start chatting

serv => cli ccc: start chatting

serv => cli ddd: start chatting

cli aaa => serv: hello there

serv=> cli bbb: aaa to bbb: hello there

cli bbb=> serv: hi

serv => cli aaa: bbb to aaa: hi

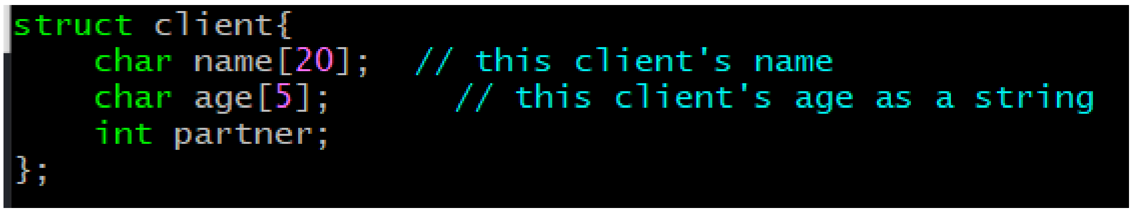
cli ccc => serv: hear me

serv=> cli ddd: ccc to ddd: hear me

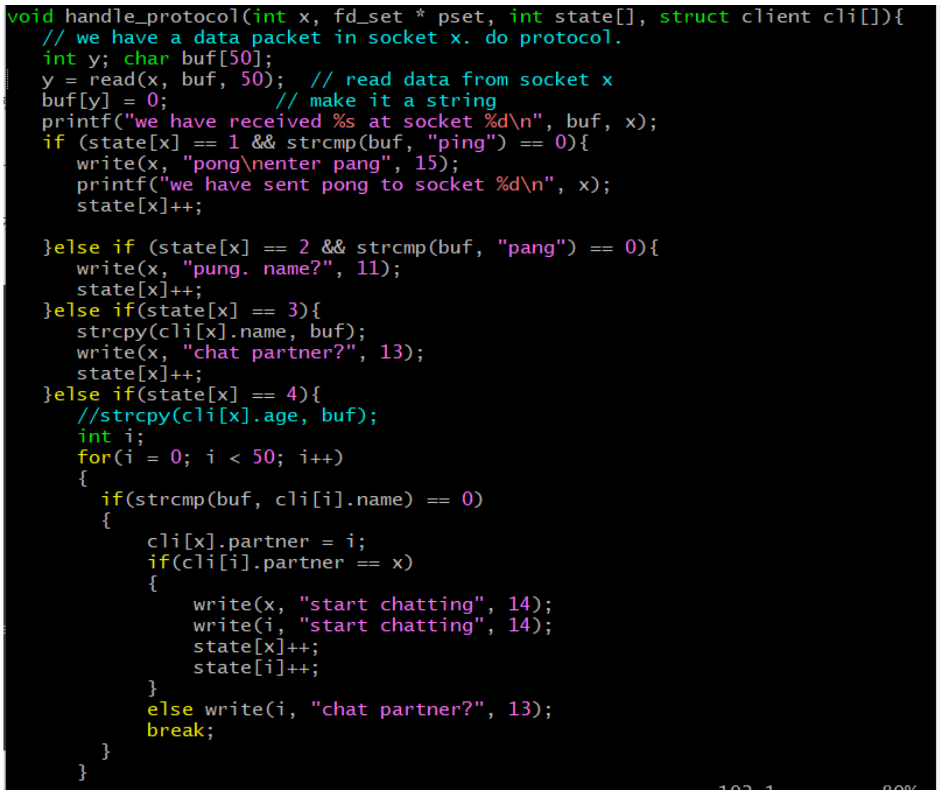
cli ddd=> serv: hi there

serv => cli ccc: ddd to ccc: hi there

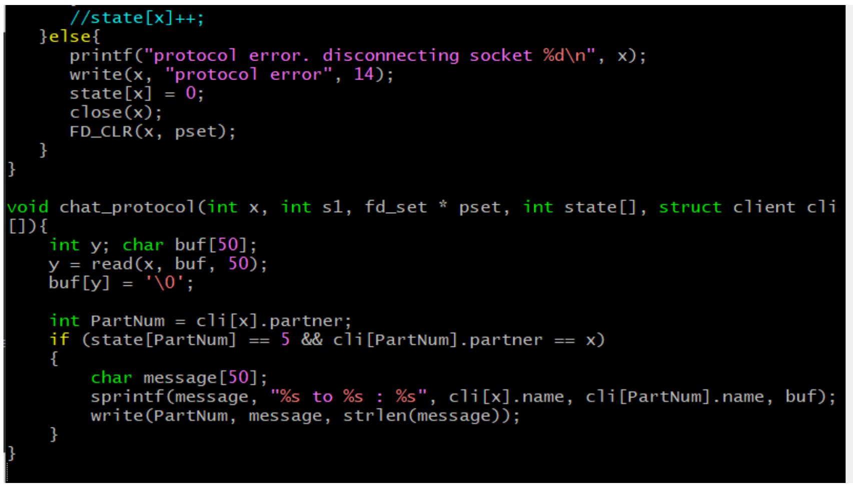
[servping.c]



```
struct client{
    char name[20];  // this client's name
    char age[5];      // this client's age as a string
    int partner;
};
```



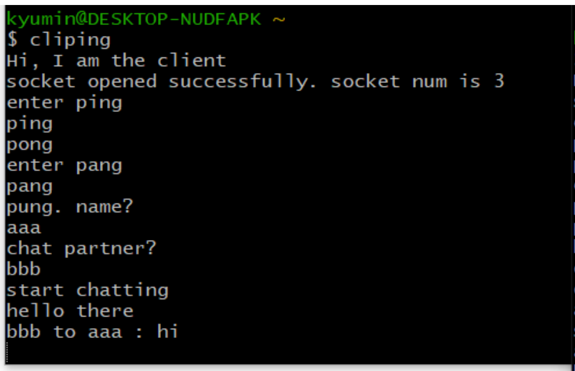
```
void handle_protocol(int x, fd_set * pset, int state[], struct client cli[]){
   // we have a data packet in socket x. do protocol.
   int y; char buf[50];
   y = read(x, buf, 50);  // read data from socket x
   buf[y] = 0;          // make it a string
   printf("we have received %s at socket %d\n", buf, x);
   if (state[x] == 1 && strcmp(buf, "ping") == 0){
      write(x, "pong\nenter pang", 15);
      printf("we have sent pong to socket %d\n", x);
      state[x]++;

   }else if (state[x] == 2 && strcmp(buf, "pang") == 0){
      write(x, "pung. name?", 11);
      state[x]++;
   }else if(state[x] == 3){
      strcpy(cli[x].name, buf);
      write(x, "chat partner?", 13);
      state[x]++;
   }else if(state[x] == 4){
      //strcpy(cli[x].age, buf);
      int i;
      for(i = 0; i < 50; i++)
      {
        if(strcmp(buf, cli[i].name) == 0)
        {
            cli[x].partner = i;
            if(cli[i].partner == x)
            {
                write(x, "start chatting", 14);
                write(i, "start chatting", 14);
                state[x]++;
                state[i]++;
            }
            else write(i, "chat partner?", 13);
            break;
        }
      }
```



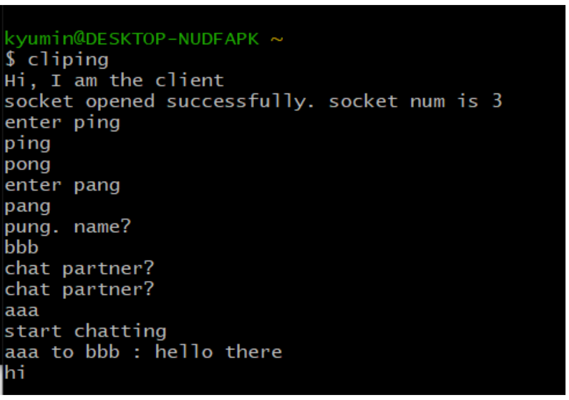
```
      //state[x]++;
   }else{
      printf("protocol error. disconnecting socket %d\n", x);
      write(x, "protocol error", 14);
      state[x] = 0;
      close(x);
      FD_CLR(x, pset);
   }
}

void chat_protocol(int x, int s1, fd_set * pset, int state[], struct client cli
[]){
    int y; char buf[50];
    y = read(x, buf, 50);
    buf[y] = '\0';

    int PartNum = cli[x].partner;
    if (state[PartNum] == 5 && cli[PartNum].partner == x)
    {
        char message[50];
        sprintf(message, "%s to %s : %s", cli[x].name, cli[PartNum].name, buf);
        write(PartNum, message, strlen(message));
    }
}
```

5번 문제의 코드에서 약간의 변화를 주었다. Ping pong을 진행하다가 state가 4가 되었을 경우 파트너의 이름을 입력할 수 있게 된다. 이 때 for문으로 구조체의 name을 하나씩 확인한다. 만약 파트너의 이름을 찾게된다면 우선 상대방의 구조체의 파트너가 자기 자신인지 확인한다. 만약 일치하지 않는다면 상대방에게 파트너를 물어본다. 만약 일치한다면 두 client의 state를 올려서 chat_protocol이 실행되도록 만든다. 그러면 이제부터 서로에게만 메시지를 보낼 수 있게된다.

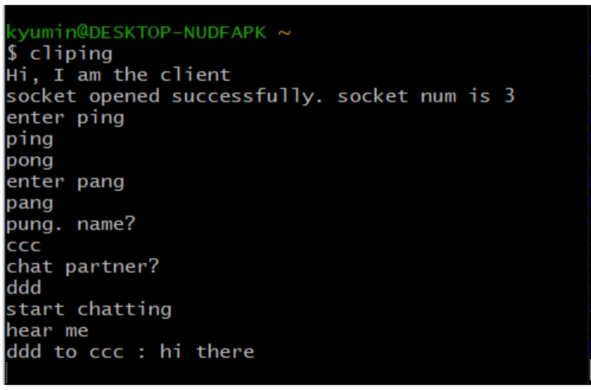
State가 4일 때와 5일 때를 나눈 이유는 만약 aaa가 bbb와 파트너를 원하지만 상대방이 aaa가 아닌 다른 client를 파트너로 선택할 경우 aaa는 다시 파트너를 선택할 수 있어야하므로 이렇게 만들었다.



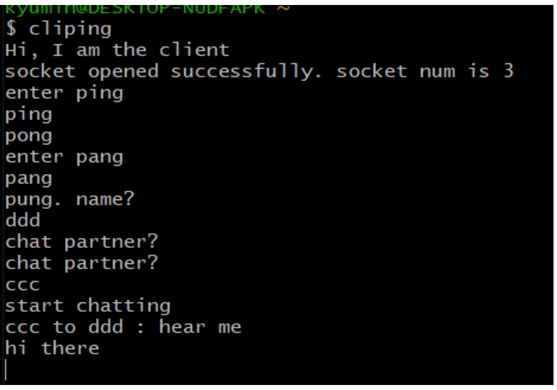
```
kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
ping
pong
enter pang
pang
pung. name?
aaa
chat partner?
bbb
start chatting
hello there
bbb to aaa : hi
```



```
kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
ping
pong
enter pang
pang
pung. name?
bbb
chat partner?
chat partner?
aaa
start chatting
aaa to bbb : hello there
hi
```



```
kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
ping
pong
enter pang
pang
pung. name?
ccc
chat partner?
ddd
start chatting
hear me
ddd to ccc : hi there
```



```
kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
enter ping
ping
pong
enter pang
pang
pung. name?
ddd
chat partner?
chat partner?
ccc
start chatting
ccc to ddd : hear me
hi there
```

Aaa와 bbb가 서로 파트너가 되었고 ccc와 ddd가 서로 파트너가 되었다. Aaa와 bbb가 대화를 할 경우 서로에게 메시지가 전달되지만 다른 client에게는 전달이 되지 않는다.

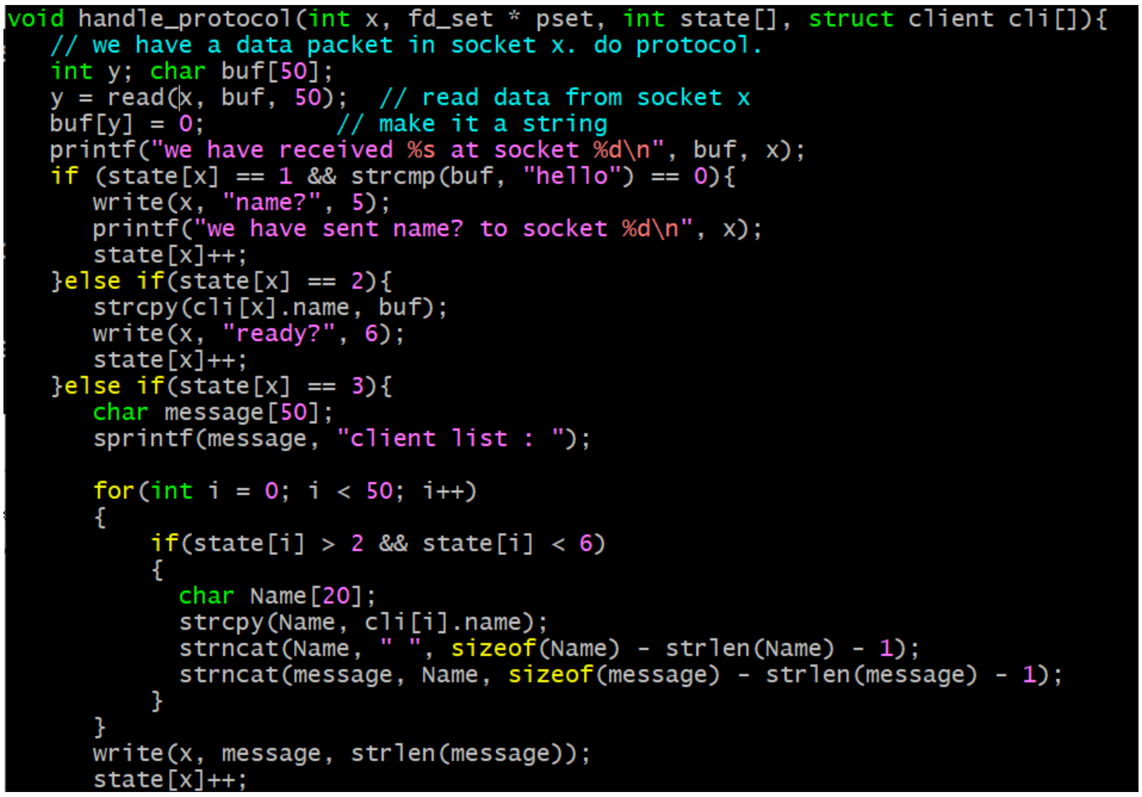
7) Implement a chatting server. The state of the client during the protocol is as follows. At any moment multiple pair of clients should be able to talk at the same time.

state 1 : The server is expecting "hello" for this client. When "hello" arrives, the server sends "name?"
state 2 : The server is expecting client ID from this client. The server rembers this client's ID in cli[x].name, where x is the socket number of this client. The server asks "ready?".
state 3 : The server is expecting "yes" from this client. The server sends all client name whose state is greater than or equal to 3.
state 4 : The server is expecting the chatting partner's ID from this client. The server remembers partner socket number in cli[x].partner. Send "go" to the client.
state 5 : The client is now in chatting session. The server is expecting some chat message from this client. The server sends this message to cli[x].partner.

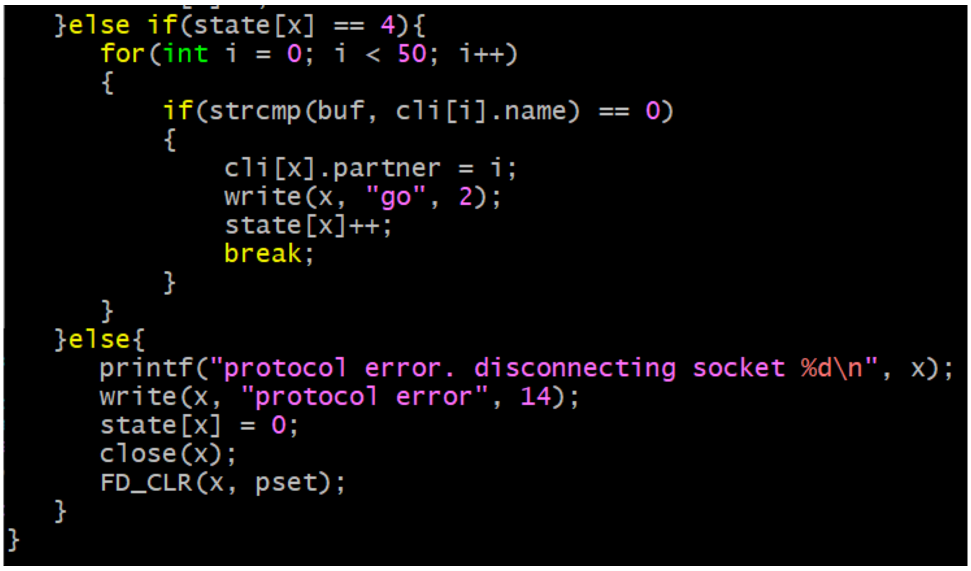
All client's initial state is 1.

```
cli eee => serv: hello
serv => cli eee: name?
cli eee=> serv: eee
serv => cli eee: ready?
cli eee => serv : yes
serv => cli eee: client list (aaa bbb ccc .....)
cli eee=> serv : bbb
serv => cli eee: go
................
cli bbb => serv : yes
serv => cli bbb : client list (aaa bbb ccc eee ...)
cli bbb => serv : eee
serv => cli bbb : go

cli eee => serv : hi how are you
serv => cli bbb : hi how are you
cli bbb => serv : hi there
serv => cli eee : hi there
..............
```

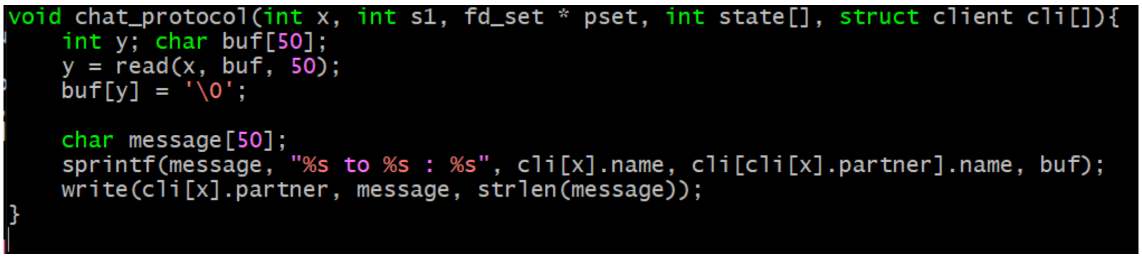
[servping.c]



```c
void handle_protocol(int x, fd_set * pset, int state[], struct client cli[]){
   // we have a data packet in socket x. do protocol.
   int y; char buf[50];
   y = read(x, buf, 50);   // read data from socket x
   buf[y] = 0;          // make it a string
   printf("we have received %s at socket %d\n", buf, x);
   if (state[x] == 1 && strcmp(buf, "hello") == 0){
      write(x, "name?", 5);
      printf("we have sent name? to socket %d\n", x);
      state[x]++;
   }else if(state[x] == 2){
      strcpy(cli[x].name, buf);
      write(x, "ready?", 6);
      state[x]++;
   }else if(state[x] == 3){
      char message[50];
      sprintf(message, "client list : ");

      for(int i = 0; i < 50; i++)
      {
          if(state[i] > 2 && state[i] < 6)
          {
            char Name[20];
            strcpy(Name, cli[i].name);
            strncat(Name, " ", sizeof(Name) - strlen(Name) - 1);
            strncat(message, Name, sizeof(message) - strlen(message) - 1);
          }
      }
      write(x, message, strlen(message));
      state[x]++;
```

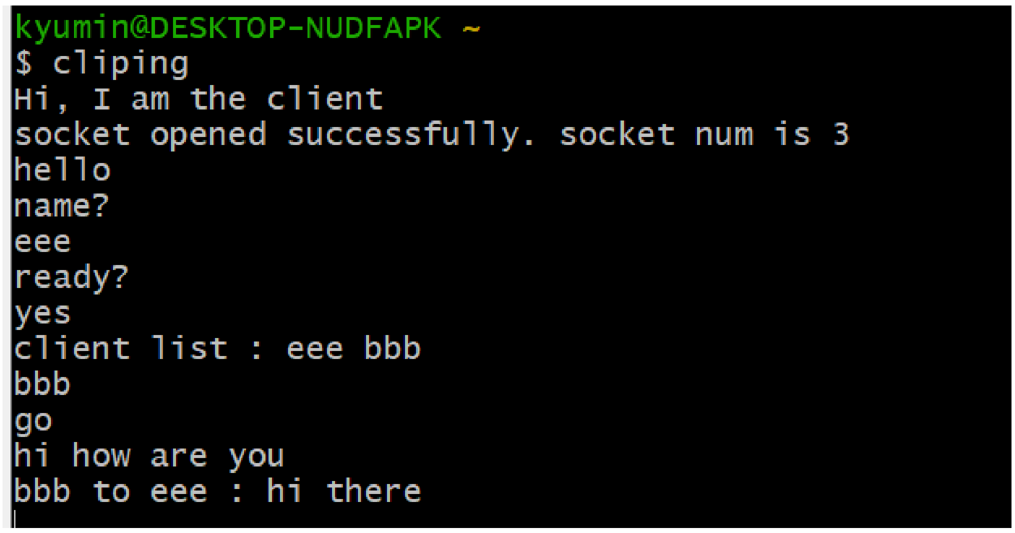


```c
   }else if(state[x] == 4){
      for(int i = 0; i < 50; i++)
      {
          if(strcmp(buf, cli[i].name) == 0)
          {
              cli[x].partner = i;
              write(x, "go", 2);
              state[x]++;
              break;
          }
      }
   }else{
      printf("protocol error. disconnecting socket %d\n", x);
      write(x, "protocol error", 14);
      state[x] = 0;
      close(x);
      FD_CLR(x, pset);
   }
}
```

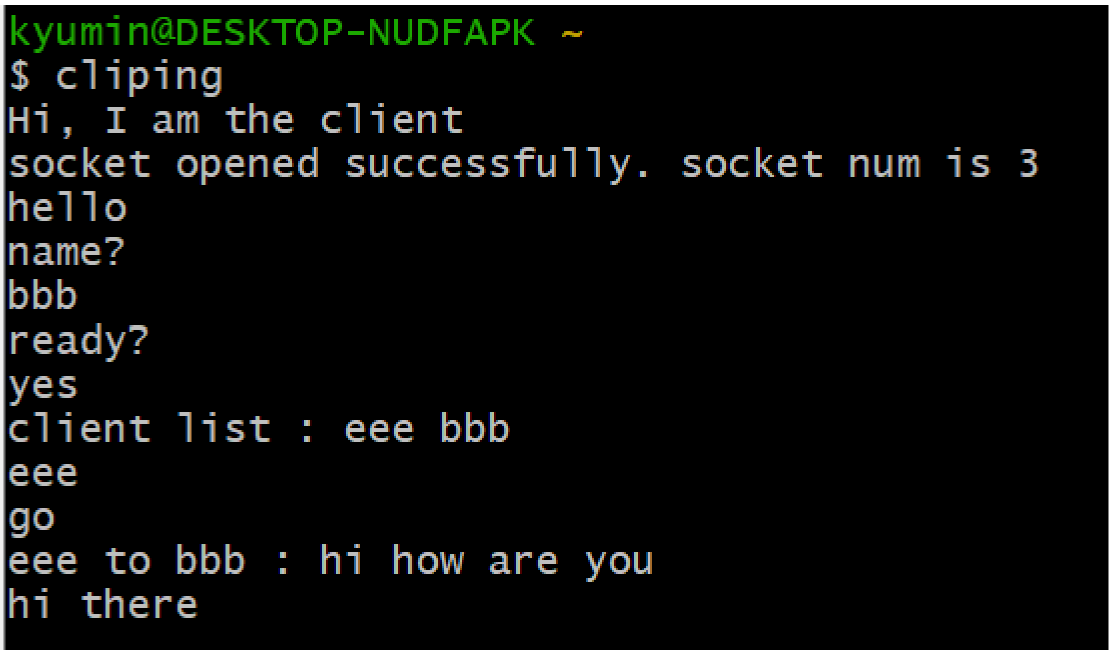


```c
void chat_protocol(int x, int s1, fd_set * pset, int state[], struct client cli[]){
    int y; char buf[50];
    y = read(x, buf, 50);
    buf[y] = '\0';

    char message[50];
    sprintf(message, "%s to %s : %s", cli[x].name, cli[cli[x].partner].name, buf);
    write(cli[x].partner, message, strlen(message));
}
```

이전 문제의 코드에서 프로토콜을 약간 수정하였다. 우선 클라이언트로부터 hello 메시지를 받으면 이름을 물어보는 메시지를 보낸다. 그리고 이름을 입력 받으면 cli[x].name에 저장한다. 이후 클라이언트로부터 메시지를 받으면 client 리스트를 보내야한다. 이 때 strncat을 사용하여 message라는 문자열에 이름을 하나씩 추가하고, write를 사용해서 message 문자열을 보냈다. 이름을 입력하면 모든 클라이언트의 name을 찾아보고, cli[x].partner에는 이름의 번호를 저장한다. 이후부터는 파트너에게 메시지를 보낼 수 있다.



```
kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
hello
name?
eee
ready?
yes
client list : eee bbb
bbb
go
hi how are you
bbb to eee : hi there
```

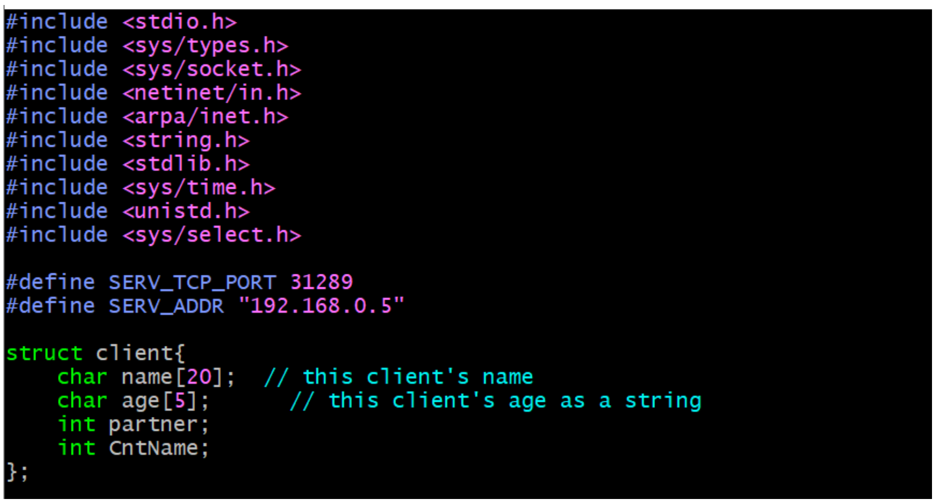


```
kyumin@DESKTOP-NUDFAPK ~
$ cliping
Hi, I am the client
socket opened successfully. socket num is 3
hello
name?
bbb
ready?
yes
client list : eee bbb
eee
go
eee to bbb : hi how are you
hi there
```

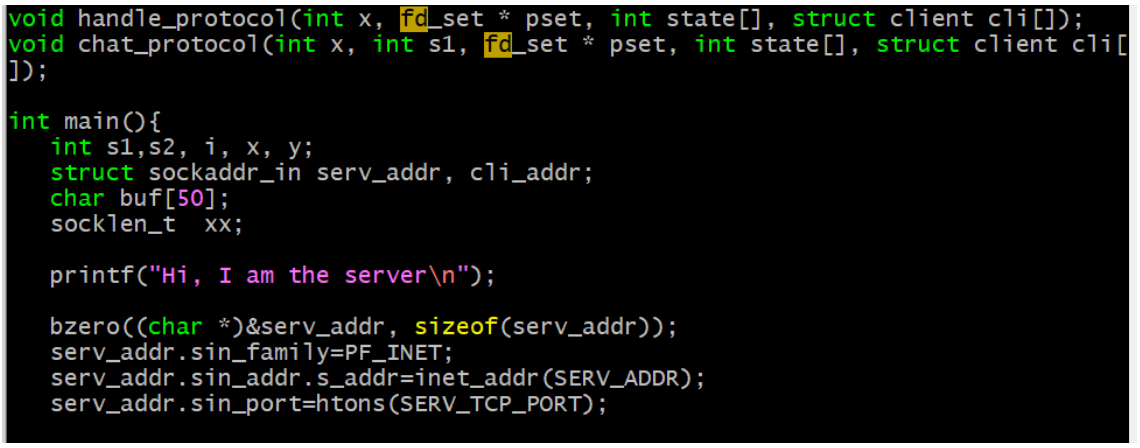
절차대로 진행할 경우 클라이언트 리스트 확인이 가능하며 원하는 상대방을 선택 후 채팅을 보내면 상대방에게 보이는 것을 알 수 있다.

8) Modify your chatting server in Problem 7 such that the server checks whether the client's name is in "login.txt" file. If yes, proceed as before; if not, ask the client's name again. If the client fails to give registered name for 5 times, the server should disconnect this client.

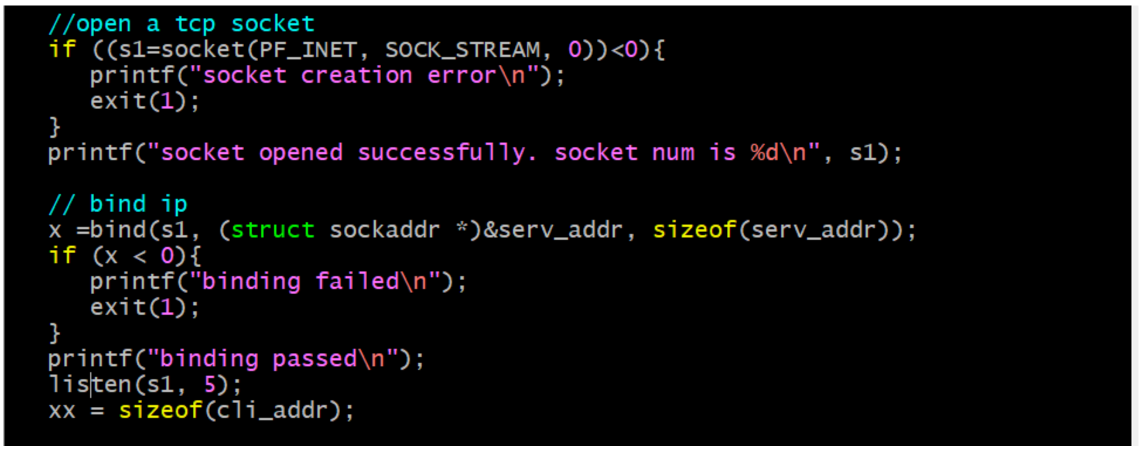
[servping.c]



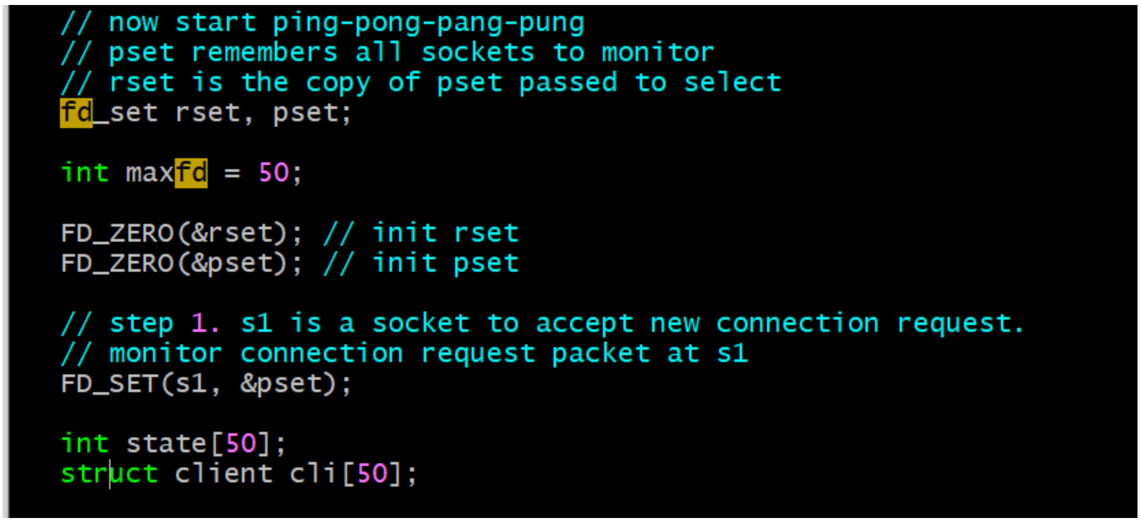
```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdlib.h>
#include <sys/time.h>
#include <unistd.h>
#include <sys/select.h>

#define SERV_TCP_PORT 31289
#define SERV_ADDR "192.168.0.5"

struct client{
    char name[20];  // this client's name
    char age[5];      // this client's age as a string
    int partner;
    int CntName;
};
```



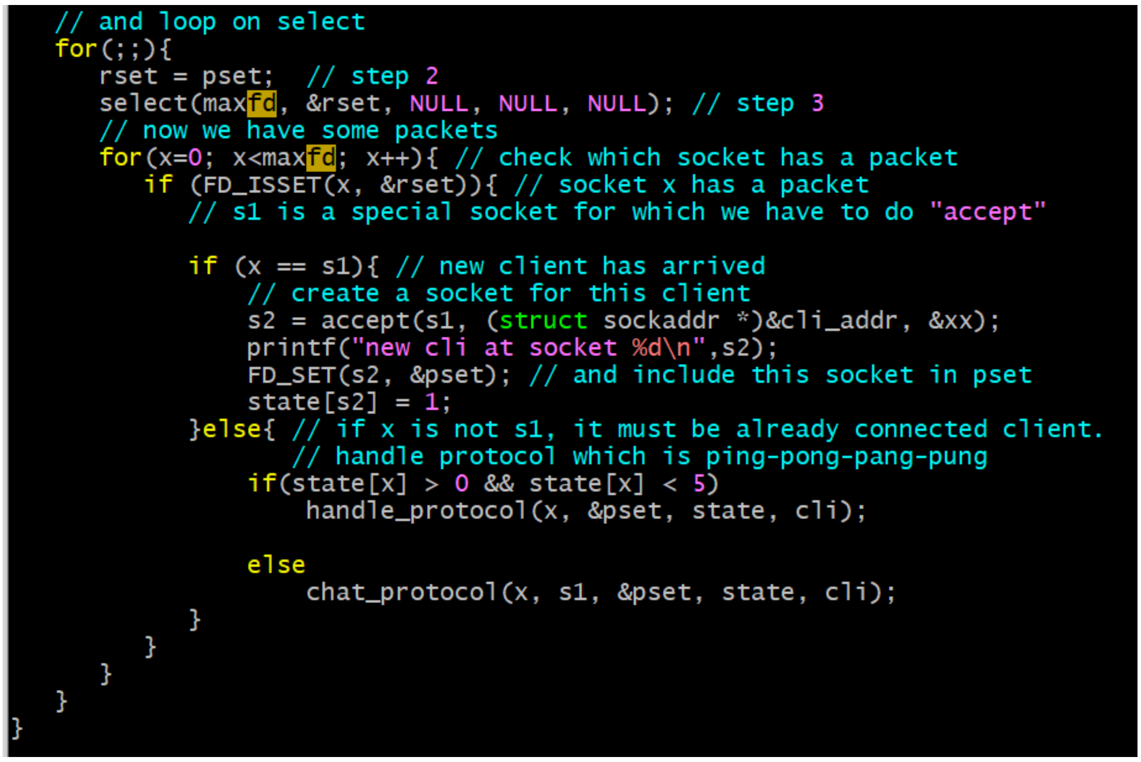
```
void handle_protocol(int x, fd_set * pset, int state[], struct client cli[]);
void chat_protocol(int x, int s1, fd_set * pset, int state[], struct client cli[
]);

int main(){
   int s1,s2, i, x, y;
   struct sockaddr_in serv_addr, cli_addr;
   char buf[50];
   socklen_t  xx;

   printf("Hi, I am the server\n");

   bzero((char *)&serv_addr, sizeof(serv_addr));
   serv_addr.sin_family=PF_INET;
   serv_addr.sin_addr.s_addr=inet_addr(SERV_ADDR);
   serv_addr.sin_port=htons(SERV_TCP_PORT);
```



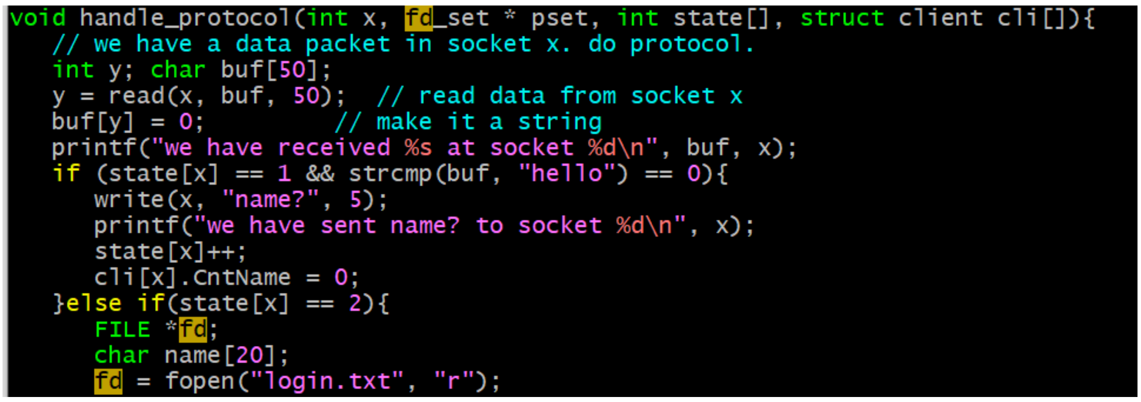
```
   //open a tcp socket
   if ((s1=socket(PF_INET, SOCK_STREAM, 0))<0){
      printf("socket creation error\n");
      exit(1);
   }
   printf("socket opened successfully. socket num is %d\n", s1);

   // bind ip
   x =bind(s1, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
   if (x < 0){
      printf("binding failed\n");
      exit(1);
   }
   printf("binding passed\n");
   listen(s1, 5);
   xx = sizeof(cli_addr);
```



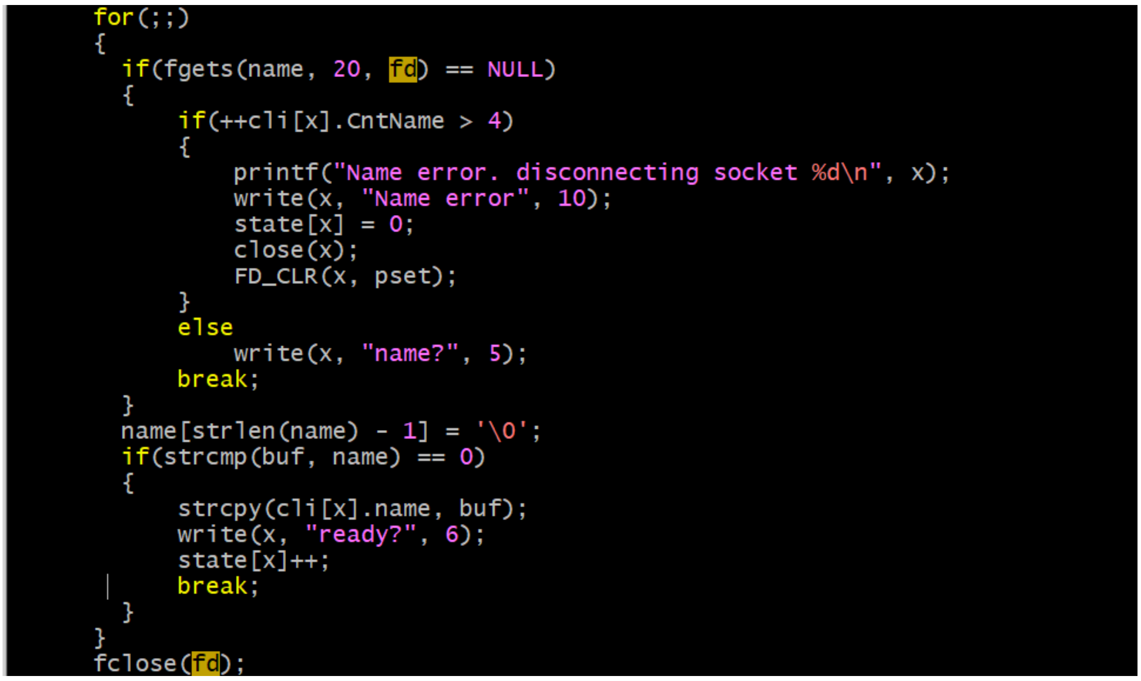
```
   // now start ping-pong-pang-pung
   // pset remembers all sockets to monitor
   // rset is the copy of pset passed to select
   fd_set rset, pset;

   int maxfd = 50;

   FD_ZERO(&rset); // init rset
   FD_ZERO(&pset); // init pset

   // step 1. s1 is a socket to accept new connection request.
   // monitor connection request packet at s1
   FD_SET(s1, &pset);

   int state[50];
   struct client cli[50];
```



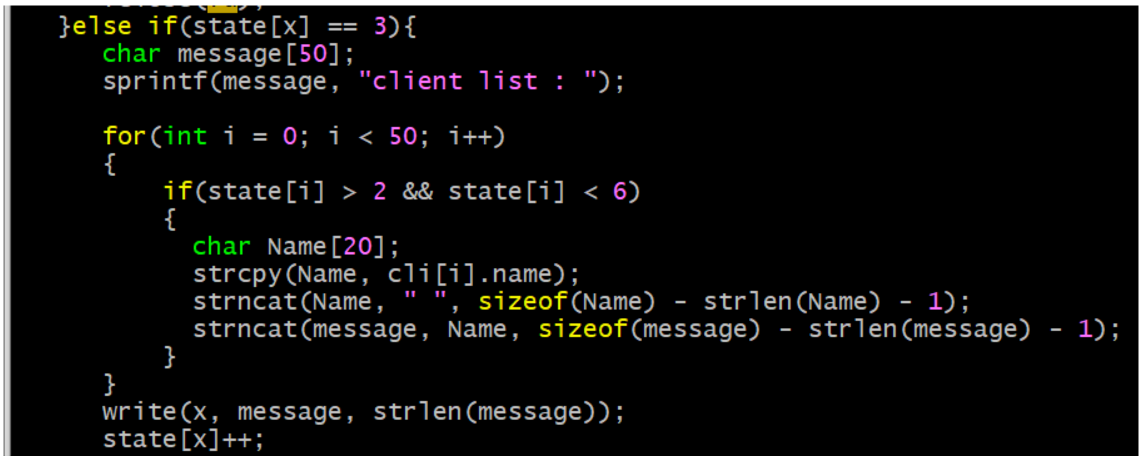
```
   // and loop on select
   for(;;){
      rset = pset;  // step 2
      select(maxfd, &rset, NULL, NULL, NULL); // step 3
      // now we have some packets
      for(x=0; x<maxfd; x++){ // check which socket has a packet
         if (FD_ISSET(x, &rset)){ // socket x has a packet
            // s1 is a special socket for which we have to do "accept"

            if (x == s1){ // new client has arrived
                // create a socket for this client
                s2 = accept(s1, (struct sockaddr *)&cli_addr, &xx);
                printf("new cli at socket %d\n",s2);
                FD_SET(s2, &pset); // and include this socket in pset
                state[s2] = 1;
            }else{ // if x is not s1, it must be already connected client.
                   // handle protocol which is ping-pong-pang-pung
                if(state[x] > 0 && state[x] < 5)
                    handle_protocol(x, &pset, state, cli);

                else
                    chat_protocol(x, s1, &pset, state, cli);
            }
         }
      }
   }
}
```



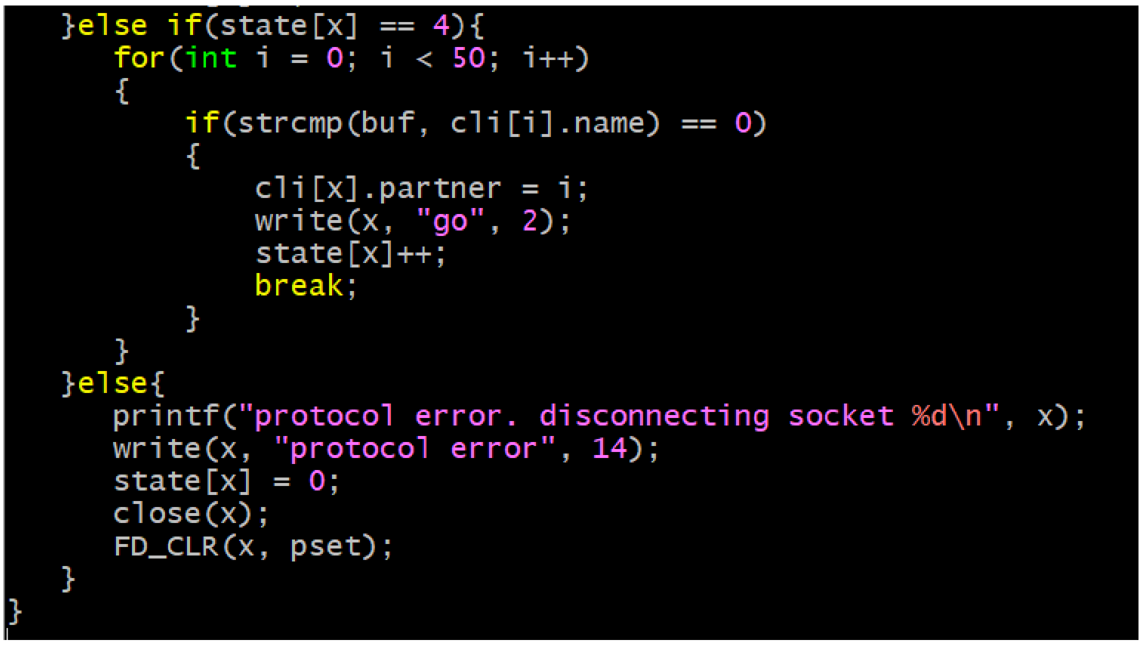
```
void handle_protocol(int x, fd_set * pset, int state[], struct client cli[]){
   // we have a data packet in socket x. do protocol.
   int y; char buf[50];
   y = read(x, buf, 50);  // read data from socket x
   buf[y] = 0;            // make it a string
   printf("we have received %s at socket %d\n", buf, x);
   if (state[x] == 1 && strcmp(buf, "hello") == 0){
      write(x, "name?", 5);
      printf("we have sent name? to socket %d\n", x);
      state[x]++;
      cli[x].CntName = 0;
   }else if(state[x] == 2){
      FILE *fd;
      char name[20];
      fd = fopen("login.txt", "r");
```



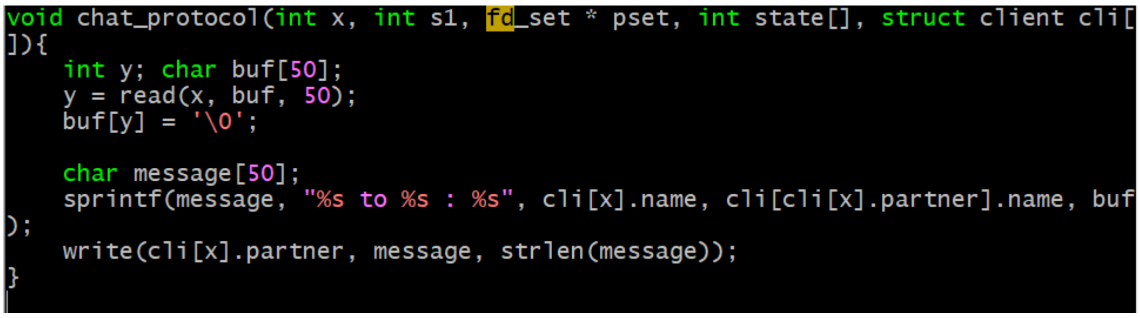
```
  for(;;)
  {
    if(fgets(name, 20, fd) == NULL)
    {
        if(++cli[x].CntName > 4)
        {
            printf("Name error. disconnecting socket %d\n", x);
            write(x, "Name error", 10);
            state[x] = 0;
            close(x);
            FD_CLR(x, pset);
        }
        else
            write(x, "name?", 5);
        break;
    }
    name[strlen(name) - 1] = '\0';
    if(strcmp(buf, name) == 0)
    {
        strcpy(cli[x].name, buf);
        write(x, "ready?", 6);
        state[x]++;
        break;
    }
  }
  fclose(fd);
```



```
}else if(state[x] == 3){
   char message[50];
   sprintf(message, "client list : ");

   for(int i = 0; i < 50; i++)
   {
       if(state[i] > 2 && state[i] < 6)
       {
         char Name[20];
         strcpy(Name, cli[i].name);
         strncat(Name, " ", sizeof(Name) - strlen(Name) - 1);
         strncat(message, Name, sizeof(message) - strlen(message) - 1);
       }
   }
   write(x, message, strlen(message));
   state[x]++;
```



```
    }else if(state[x] == 4){
        for(int i = 0; i < 50; i++)
        {
            if(strcmp(buf, cli[i].name) == 0)
            {
                cli[x].partner = i;
                write(x, "go", 2);
                state[x]++;
                break;
            }
        }
    }else{
        printf("protocol error. disconnecting socket %d\n", x);
        write(x, "protocol error", 14);
        state[x] = 0;
        close(x);
        FD_CLR(x, pset);
    }
}
```



```
void chat_protocol(int x, int s1, fd_set * pset, int state[], struct client cli[
]){
    int y; char buf[50];
    y = read(x, buf, 50);
    buf[y] = '\0';

    char message[50];
    sprintf(message, "%s to %s : %s", cli[x].name, cli[cli[x].partner].name, buf
);
    write(cli[x].partner, message, strlen(message));
}
```