Programación con Sockets

Se tienen los siguientes tres códigos:

socket.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <fcntl.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <ctype.h>

#define NOBLOCK         0x0FFFB
extern int errno;

int socket_servidor(side)
int side;
{
   struct  sockaddr_in server;
   int   sock, x, flags;
   sock= socket(AF_INET, SOCK_STREAM, 0);
   if (sock < 0)
   {
      perror ("NO SE PUEDE CREAR SOCKET");
      exit (1);
   }
   bzero(&server, sizeof(server));
   server.sin_family = AF_INET;
   server.sin_addr.s_addr = INADDR_ANY;
   server.sin_port = htons(side);
   x = bind(sock, &server, sizeof(server));
   if (x<0)
   {
      close(sock);
      perror("NO SE PUEDE ENLAZAR");
      exit(1);
   }
   if (listen(sock, 5) < 0)
   {
      perror ("NO ESCUCHA");
      exit (1);
   }
   return sock;
}

int acepta_conexion(sock)
int sock;
{ struct  sockaddr_in server;
  int   adrl, x, flags;
  adrl = sizeof (struct sockaddr_in);
  x = accept (sock, &server, &adrl);
  return x;
}
/********** cliente **********************/

int socket_cliente(host, port)
char *host;
int port;
{
   int csock;
   struct  sockaddr_in client;
   struct  hostent *hp, *gethostbyname();
   bzero(&client, sizeof(client));
   client.sin_family = AF_INET;
   client.sin_port = htons(port);
   if (isdigit(host[0]))
      client.sin_addr.s_addr = inet_addr(host);
   else
   {
      hp = gethostbyname(host);
      if (hp == NULL)
      {  perror (" ¿A QUE HOST QUIERO CONECTARSE? ");
         exit (1);
      }
      bcopy(hp->h_addr, &client.sin_addr, hp->h_length);
   }
   csock = socket(AF_INET, SOCK_STREAM, 0);
   if (csock < 0)
   {  perror ("NO SE PUEDE CREAR EL SOCKET");
      exit (1);
   }
   connect(csock, &client, sizeof(client));
   return csock;
}
```

servidor.c

```
#include <stdio.h>
#include <string.h>
extern  acepta_conexion();
extern  call_on_sock();
struct PDU
{
   char cabeza;
   char mensaje[100];
};

main()
{
   int dirlisten, conexion;
   struct PDU peticion, respuesta;

   dirlisten=socket_servidor(55055);
   printf("--------- INICIO DEL SERVIDOR ----------\n");
   while(1)
   {
      while((conexion=acepta_conexion(dirlisten))<0);
      read(conexion, &peticion, sizeof(peticion));
      printf("\nDATO RECIBIDO:%s\n",peticion.mensaje);
      printf("Escuchando ....   ");
      strcpy(respuesta.mensaje,"Mensaje recibido");
      write(conexion, &respuesta, sizeof(respuesta));
      close(conexion);
    }
}
```

cliente.c

```
#include <stdio.h>
extern  socket_cliente();
struct PDU
{
   char cabeza;
   char mensaje[100];
};

main()
{
   char dato[25];
   int  conexion;
   struct PDU peticion, respuesta;

   printf("--------- INICIO DE CLIENTE --------\n");
   printf("Ingrese dato a enviar:");
   scanf("%s",dato);

   conexion=socket_cliente("loki",55055);
   strcpy(peticion.mensaje,dato);
   write(conexion,&peticion,sizeof(peticion));
   read(conexion, &respuesta, sizeof(respuesta));
   printf("RESPUESTA DE SERVIDOR: %s\n", respuesta.mensaje);
   close(conexion);
}
```

Ambiente

- Se sugiere dejar todos los programas en un directorio específico. Por ejemplo, crearse una carpeta llamada **sockets**. En este ejemplo se ja creado la carpeta **sockets** dentro de otra carpeta llamada **prgs**

```
ennio@loki:~/prgs/sockets$ ls -l
total 12
-rw-r--r-- 1 ennio ennio  551 nov  2 17:33 cliente.c
-rw-r--r-- 1 ennio ennio  663 nov  2 17:31 servidor.c
-rw-r--r-- 1 ennio ennio 1790 dic 16 11:16 socket.c
ennio@loki:~/prgs/sockets$
```

-

Compilación

- Para compilar el programa servidor se debe ejecutar el linux
  cc -w servidor.c socket.c -o <ejecutable>

  donde <ejecutable> es el archivo binario ejecutable, por ejemplo

```
ennio@loki:~ /prgs/sockets$ cc -w servidor.c socket.c -o srvr
ennio@loki:~ /prgs/sockets$
```

  Deberá aparecer un nuevo archivo llamado **srvr** en el directorio donde se realizó la compilación:

```
ennio@loki:~/prgs/sockets$ ls -l
total 28
-rw-r--r-- 1 ennio ennio  551 nov  2 17:33 cliente.c
-rw-r--r-- 1 ennio ennio  663 nov  2 17:31 servidor.c
-rw-r--r-- 1 ennio ennio 1790 dic 16 11:16 socket.c
-rwxr-xr-x 1 ennio ennio 16220 dic 16 11:49 srvr
ennio@loki:~/prgs/sockets$
```

- Para compilar el programa cliente se debe ejecutar el linux
  cc -w cliente.c socket.c -o *<ejecutable>*

  donde <ejecutable> es el archivo binario ejecutable, por ejemplo

```
ennio@loki:~/prgs/sockets $ cc -w cliente.c socket.c -o hcli
ennio@loki:~/prgs/sockets $
```

  Deberá aparecer un nuevo archivo llamado **hcli** en el directorio donde se realizó la compilación:

```
ennio@loki:~/prgs/sockets$ ls -l
total 44
-rw-r--r-- 1 ennio ennio  551 nov  2 17:33 cliente.c
-rwxr-xr-x 1 ennio ennio 16304 dic 16 11:50 hcli
-rw-r--r-- 1 ennio ennio  663 nov  2 17:31 servidor.c
-rw-r--r-- 1 ennio ennio 1790 dic 16 11:16 socket.c
-rwxr-xr-x 1 ennio ennio 16220 dic 16 11:49 srvr
ennio@loki:~/prgs/sockets$
```

Ejecución

Ejecución del servidor

- En un terminal de Linux ejecutar el programa servidor primero. Siguiendo el ejemplo anterior se debe ejecutar así

  `./srvr`

  Donde **`srvr`** es el nombre del ejecutable creado anteriormente.

  Ejemplo de ejecución del servidor:

```
ennio@loki:~/prgs/sockets$
ennio@loki:~/prgs/sockets$ ./srvr
--------- INICIO DEL SERVIDOR ----------
```

  El terminal quedará esperando la conexión de un cliente.

Ejecución del cliente

- En un segundo terminal de Linux ejecutar el programa cliente. Siguiendo el ejemplo anterior se debe ejecutar así

  `./hcli`

  Donde **`hcli`** es el nombre del ejecutable creado anteriormente.

  Ejemplo de ejecución del cliente:

```
ennio@loki:~/prgs/sockets$
ennio@loki:~/prgs/sockets$ ./hcli
--------- INICIO DE CLIENTE --------
Ingrese dato a enviar:
```

  El programa pedirá un "dato" a enviar. Puede ser una palabra o un número. En este caso se enviará el dato Hola

- Inmediatamente el programa termina la ejecución así:

```
ennio@loki:~/prgs/sockets$ ./hcli
--------- INICIO DE CLIENTE --------
Ingrese dato a enviar:Hola
RESPUESTA DE SERVIDOR: Mensaje recibido
ennio@loki:~/prgs/sockets$
```

- El cliente recibe el mensaje "Mensaje recibido" desde el servidor.
- Si se revisa la terminal del servidor se verá que recibió el mensaje "Hola" que imprime en pantalla

  Terminal del servidor:

```
ennio@loki:~/prgs/sockets$
ennio@loki:~/prgs/sockets$ ./srvr
--------- INICIO DEL SERVIDOR ----------

DATO RECIBIDO:Hola
```

- El servidor sigue en espera de nuevos mensajes desde el cliente, es decir, se puede volver a ejecutar el cliente con otro dato a enviar.

  NOTA: El programa solo envía una palabra, si se agregan más palabras enviar sólo considera la primera.

Terminar el servidor

- Se puede abortar la ejecución del servidor con un control+c

```
ennio@loki:~/prgs/sockets$ ./srvr
--------- INICIO DEL SERVIDOR ----------

DATO RECIBIDO:Hola
^C
ennio@loki:~/prgs/sockets$
```

Observación Importante:

Eventualmente al intentar ejecutar el servidor nuevamente obtengamos el siguiente mensaje de error.

```
ennio@loki:~/prgs/sockets$ ./srvr
NO SE PUEDE ENLAZAR: Address already in use
ennio@loki:~/prgs/sockets$
```

Esto es por que aún no sea liberado el puerto de comunicación que se usó.
Hay dos posibles soluciones

1. Esperar a que se libere el recurso (puede ser necesario esperar un par de minutos)
2. Modificar los programas fuente del servidor y cliente para cambiar el puerto de comunicación que actualmente está configurado con el 55055.