

INTRODUCCIÓN A LA PROGRAMACIÓN CON SOCKETS

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

- - -

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

ÍNDICE

1. Introducción.
2. API de sockets:
 - Llamadas al sistema
 - Estructuras de datos.
3. Esquemas cliente/servidor.

The logo for Cartagena99 features the text 'Cartagena99' in a stylized, green, serif font. The text is set against a background of a light blue and orange gradient that resembles a stylized sun or a wave. The '99' is slightly larger and more prominent than the 'Cartagena' part.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

- - -

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

BIBLIOGRAFÍA

- Básica:
 - M.J. Donahoo, K.L. Calvert (2009). *TCP/IP Sockets in C. Practical Guide for Programmers*. 2nd Edition. Morgan Kaufmann Publishers.
 - W. Richard Stevens; Bill Fenner & Andrew M. Rudoff (2003). *Unix Network Programming, Volume 1: The Sockets Networking API*. 3rd Edition. Addison-Wesley Professional.
 - Capítulo 21 de D. E. Comer (2014). *Internetworking with TCP/IP vol I*, 6 Ed Pearson.
 - Beej's Guide to Network Programming (http://beej.us/guide/bgnet/pdf/bgnet_A4_2.pdf).
- Complementaria:

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Introducción

- TCP/IP no define el interfaz con los programas de aplicación (API).
 - Depende mucho del sistema operativo.
- El interfaz *socket*, de BSD UNIX, se ha convertido en el estándar “de facto”.
 - A principios de los 80s, ARPA funda un grupo en la Universidad de California en Berkeley para integrar TCP/IP en el sistema operativo UNIX.
 - El sistema que desarrollaron se conoce como Berkeley UNIX o BSD UNIX.
 - Como parte del proyecto, se define un API C para que las aplicaciones puedan usar TCP/IP para comunicarse: interfaz *socket*.
 - Comunicación entre procesos (IPC).
 - Aparece a partir de la versión 4.1 de BSD
 - Se basa en las llamadas de E/S de UNIX.

The logo for Cartagena99 features the text 'Cartagena99' in a stylized, green, serif font. The '99' is significantly larger and more prominent than the 'Cartagena' part. The text is set against a light blue and white background with a subtle wave-like pattern.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Llamadas al sistema en UNIX

- UNIX: sistema operativo de tiempo compartido desarrollado a principios de los 70s.
- Orientado a procesos:
 - Cada proceso se ejecuta con un privilegio.
- Las aplicaciones interactúan con el sistema operativo mediante llamadas al sistema (“*system calls*”):
 - Cada vez que una aplicación necesita acceder a un servicio del sistema operativo, lo hace a través de una llamada al sistema.
 - Por ejemplo, las operaciones de E/S.
 - La llamada se ejecuta con privilegios del sistema operativo.
- Para el programador, una llamada al sistema funciona como una llamada a una función:
 - Se le pasan unos argumentos.

Cartagena99

CLÁSES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Entrada/Salida

- Llamadas al sistema para E/S:
 - `open()`
 - Prepara el dispositivo o el fichero para hacer la entrada o la salida.
 - Devuelve un entero (*descriptor de fichero*) que se usará en el resto de las llamadas para hacer referencia a ese dispositivo.
`int desc;`
`desc = open("filename", O_RDWR, 0);`
 - `read()`
 - Lee datos del dispositivo o fichero.
`read(desc,buffer,128)`
 - `write()`
 - Escribe datos en el dispositivo o fichero.
`write(desc,buffer,128)`
 - `close()`
 - Termina el uso de ese dispositivo.
`close(desc)`

Otras llamadas:

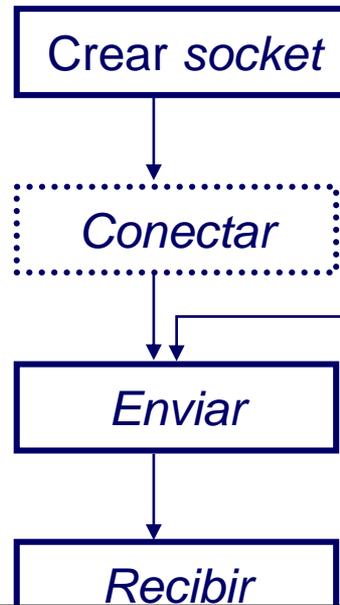
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Comunicación entre procesos (IPC)

- Sigue una estructura parecida a la Entrada/Salida:



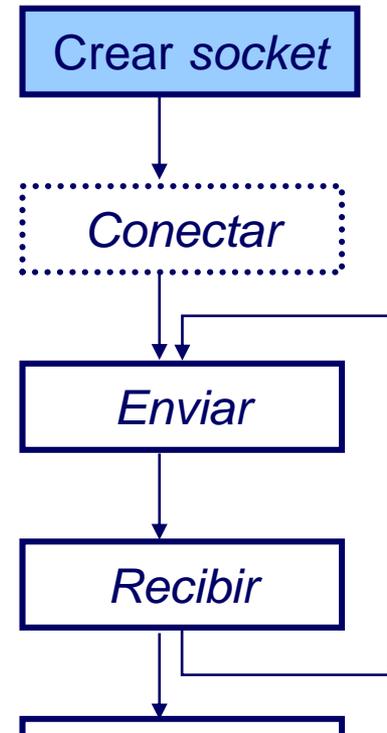
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Crear el socket

- Se hace con la llamada `socket()`.



Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Crear el socket: socket ()

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

int socket (int domain, int type, int protocol)
```

- **domain:** dominio de aplicación que representa a una familia de protocolos.
 - **PF_INET:** DARPA Internet (TCP/IP). Comunicaciones sobre IPv4.
 - **PF_INET6:** IPv6. Comunicaciones sobre IPv6.
 - **PF_UNIX:** Sistema de ficheros UNIX (pipe).
 - **PF_PACKET:** Packet sockets (Linux).
- **type:** tipo de comunicación.
 - **SOCK_STREAM:** fiable y ordenada.
 - **SOCK_DGRAM:** no fiable y desordenada, conservando los límites de los paquetes.
 - **SOCK_RAW:** acceso directo a los protocolos de bajo nivel.
- **protocol:** protocolo.
 - Para especificar un protocolo concreto cuando varios pueden dar ese servicio.
 - Si se da el valor 0, el sistema elige un protocolo adecuado según el tipo.
 - `getprotobyname ()` devuelve el número de protocolo a partir del nombre del protocolo

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Crear el *socket*: clasificación

Llamada		Tipo		Acceso a:
<code>socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)</code>	→	TCP socket	→	TCP
<code>socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)</code>	→	UDP socket	→	UDP
<code>socket(PF_INET, SOCK_RAW, IPPROTO_ICMP)</code>	→	Raw socket	→	IP

Cartagena99

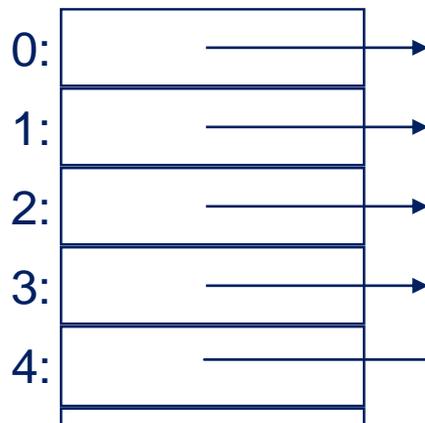
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Estructura de datos por *socket*

- Al realizar la llamada a `socket()`, el sistema operativo lo que crea es una estructura de datos para mantener la información asociada a esa comunicación.

Tabla de descriptores
(una por proceso)



Estructura de datos
de un *socket*



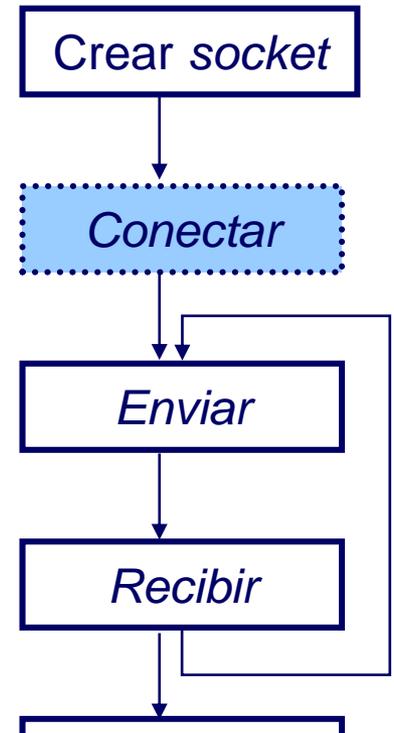
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

Conectar el socket

- Consiste en indicar las IP y puertos que se va a usar en la comunicación.
- Este paso difiere según sea un cliente o servidor y *stream* o *dgram*:
 - Servidor *stream*:
 - `bind()`
 - `listen()`
 - `accept()`
 - Cliente *stream*:
 - `bind()` (opcional)
 - `connect()`
 - Servidor *dgram*:
 - `bind()`



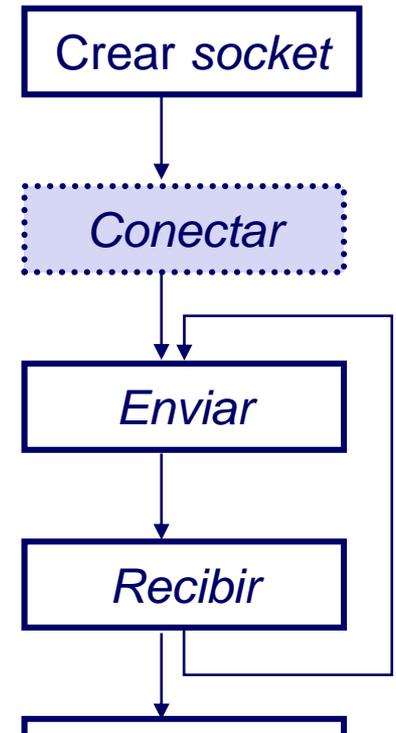
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Conectar el socket

- `bind()`
 - Se suele usar sólo en servidor (*stream* o *dgram*), aunque también se puede usar en el cliente.
 - Si no se hace se asignan valores por defecto.
 - Asigna al socket la dirección IP y puerto locales
- `listen()`
 - Se usa en servidor *stream*.
 - Indica que el socket se va a quedar a la escucha esperando que un cliente se conecte.
- `accept()`
 - Se usa en servidor *stream*.
 - Cuando un cliente se conecta, nos devuelve un nuevo socket por el que nos comunicaremos con él.
- `connect()`
 - Se usa en cliente *stream*.



Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

- No se establece conexión. Simplemente toma nota de la

Especificar dirección local: `bind()`

- Inicialmente el *socket* se crea sin dirección origen ni destino:
 - En TCP/IP esto significa que no han sido asignados: dirección IP local, puerto local, dirección IP destino, puerto destino.

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

int bind(int sockfd, struct sockaddr *my_addr, int addrlen);
```

- **sockfd**: descriptor del socket.
- **my_addr**: es un puntero a una estructura **sockaddr**.
- **addrlen**: longitud de la estructura anterior.
- Resultado:

The logo for Cartagena99, featuring the text 'Cartagena99' in a stylized font with a blue and orange gradient background.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Estructura sockaddr

```
struct sockaddr {
    sa_family_t    short sa_family; // unsigned integer
    char          sa_data[14];
};
```

- Mantiene información sobre direcciones de sockets.
- Campos:
 - **sa_family**: familia de direcciones (admite varios valores, como por ejemplo **AF_INET**, **AF_INET6**).
 - **sa_data**: dirección y número de puerto para el socket.

The logo for Cartagena99 features the text 'Cartagena99' in a stylized, green, serif font. The '99' is significantly larger and more prominent than the rest of the text. The logo is set against a light blue and orange gradient background.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

AF_ versus PF_

- En la llamada *sockets*, al hablar de familias de protocolos hemos visto unas constantes que empiezan por `PF_`.
 - `PF_INET`, `PF_INET6`, `PF_UNIX`, `PF_PACKET`...
- Al hablar de familias de direcciones aparecen constantes que empiezan por `AF_`.
 - `AF_INET`, `AF_INET6`...
- Ambas constantes están definidas en `<sys/socket.h>` y en realidad apuntan al mismo valor.

```
/* Supported address families. */
...
#define AF_INET 2
...
#define AF_INET6 10
...
/* Protocol families, same as address families. */
...
#define PF_INET AF_INET
...
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Estructura sockaddr_in

```

struct sockaddr_in {
    sa_family_t    sin_family;           // Internet protocol (AF_INET)
    in_port_t      sin_port;            // Puerto(uint16_t, 16 bits)
    struct in_addr sin_addr;            // Dirección IPv4 (32 bits)
    char           sin_zero[8];         // Sin usar
};

struct in_addr {
    uint32_t      s_addr;                // Dirección IPv4 (32 bits)
};

```

- Particularización de sockaddr para contener información de sockets IPv4 (direcciones IPv4 y puerto de transporte).
- Campos:
 - **sin_family**: admite varios valores, como **AF_INET**.
 - **sin_port**: número de puerto para el socket.
 - **sin_addr**: dirección IP para el socket.
 - **sin_zero**: relleno para mantener el mismo tamaño que **sockaddr** (debe rellenarse a

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
 LLAMA O ENVIA WHATSAPP: 689 45 44 70

 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
 CALL OR WHATSAPP:689 45 44 70

Estructura sockaddr_in6

- Equivalentes a las anteriores para direcciones IPv6

```
struct sockaddr_in6 {
    sa_family_t    sin6_family;    /* AF_INET6 */
    in_port_t      sin6_port;      /* port number */
    uint32_t       sin6_flowinfo; /* IPv6 flow information */
    struct in6_addr sin6_addr;      /* IPv6 address */
    uint32_t       sin6_scope_id; /* Scope ID (new in 2.4) */
};

struct in6_addr {
    unsigned char  s6_addr[16]; /* IPv6 address */
};
```

The logo for Cartagena99, featuring the text 'Cartagena99' in a stylized font with a blue and orange gradient background.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Tamaño de los enteros en C

- En C se definen varios tipos que pueden contener valores enteros: `char`, `short`, `int`, `long`.
- Pero no especifica el tamaño exacto (en bits) con el que se debe almacenar en memoria cada uno de ellos.
 - Sólo el valor mínimo que deben ser capaces de almacenar (`char` 8 bits, `short` e `int` 16 bits, `long` 32 bits), y que `char` ≤ `short` ≤ `int` ≤ `long`.
 - Depende del compilador, el sistema operativo, la arquitectura del procesador.
- Por esta razón, se debe **evitar estos tipos de datos** cuando queramos representar información que tenemos que enviar por la red.
 - En una máquina un `int` puede tener 16 bits, en otra 32, en otra 64...
 - Problemas al migrar el código de una máquina a otra.
- Las RFC que definen los protocolos nos indican claramente el número de bits con el que ocupa un determinado campo de una cabecera o de una información a enviar.
- Para estar seguro de que enviamos/recibimos el número de bits correcto, se usan otros tipos de enteros definidos en el lenguaje C en la versión C99.

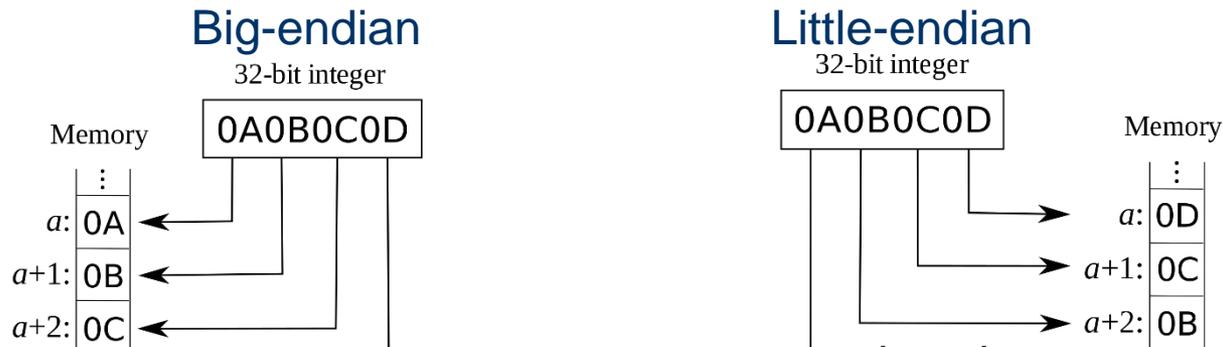
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Ordenación de octetos en memoria

- Cualquier tipo de datos de más de un octeto hay dos posibles formas de almacenarlo en memoria:
 - Primer octeto el más significativo.
 - Primer octeto el menos significativo.
- Por ejemplo, el entero de 32 bits 168496141 en hexadecimal es 0x0A0B0C0D. Se puede almacenar en memoria de dos maneras.



Cartagena99

Little-endian.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Ordenación de octetos de la red

- Sin embargo en Internet la “ordenación de octetos de la red” es (“*Big-Endian*”).
 - En el ejemplo anterior, 0x0A0B0C0D se enviaría por la red en el siguiente orden: 0A – 0B – 0C – 0D.
- Todo lo que se envíe por la red debe seguir la “ordenación de la red”:
 - Datos que se envían por el socket.
 - Algunos campos en determinadas estructuras (como por ejemplo los números de puerto).
- Por lo tanto en algunos casos es preciso realizar conversiones.
 - Depende de la ordenación que use el procesador de nuestra máquina.
- Para no tener que preocuparnos, la biblioteca `<netinet/in.h>` proporciona unas funciones que se encargan de hacer la conversión entre el orden de red y el de la máquina, caso de ser necesario.
- `#include <netinet/in.h>`:
 - `uint16_t htons(uint16_t)`: “*Host to Network Short*” (16 bit de máquina a red).
 - `uint32_t htonl(uint32_t)`: “*Host to Network Long*” (32 bit de máquina a red).

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Especificar dirección local: `bind()`

```
my_addr.sin_port = 0; /* selecciona, aleatoriamente, un puerto sin usar */  
my_addr.sin_addr.s_addr = htonl(INADDR_ANY); /* usa mi dirección IP */
```

- Consideraciones sobre la elección de puerto:
 - Todos los puertos por debajo del 1024 están RESERVADOS, a menos que seas súper-usuario.
 - Puedes usar cualquier número de puerto por encima de ese, hasta el 65535, siempre y cuando no lo esté usando ya otro programa.
- La llamada a `bind()` se emplea en programas servidores, en programas clientes no es necesario realizar esta llamada (se llama a `connect`).
- En raw sockets, ya que desaparece el concepto de puerto, no sería necesario usar `bind`. Sin embargo, sí es necesaria para asociar una dirección IP local (ó un interfaz, pei: eth1) con el raw socket (útil si hay

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Manipulación de direcciones IP

- ¿Cómo se guarda en una estructura `struct sockaddr_in` la dirección IP “163.117.139.233” ?:

```
ina.sin_addr.s_addr = inet_addr("163.117.139.233"); /* Ordenación de red*/
```

- Otras llamadas útiles para el tratamiento de direcciones IP:

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int inet_aton(const char *cp, struct in_addr *inp);

char *inet_ntoa(struct in_addr inp);
```

The logo for Cartagena99, featuring the text 'Cartagena99' in a stylized, green, serif font with a blue and orange gradient background behind the letters.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Ejemplo de bind ()

```
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define MYPORT 3490

void main() {
    int sockfd;
    struct sockaddr_in my_addr;

    /* Chequear si devuelve -1! */
    sockfd = socket(PF_INET, SOCK_STREAM, 0);
    my_addr.sin_family = AF_INET; /* Ordenación de la máquina */
    my_addr.sin_port = htons(MYPORT); /* Ordenación de la red */
    my_addr.sin_addr.s_addr = inet_addr("132.241.5.10");
    /* Pone a cero el resto de la estructura */
    memset(&my_addr.sin_zero, \0, 8);
}
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Poner el *socket* a la escucha: `listen()`

- Se realiza en los servidores:
 - Se denominan “sockets pasivos”
 - Quedan a la espera de recibir peticiones de conexión.

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

int listen(int sockfd, int backlog);
```

- `sockfd`: descriptor del socket.
- `backlog`: número de conexiones permitidas en la cola de entrada.
- Resultado:
 - “-1” si se ha producido un error.

The logo for Cartagena99, featuring the text 'Cartagena99' in a stylized font with a blue and orange gradient background.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Poner el *socket* a la escucha: `accept()`

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

int accept(int sockfd, void *addr, int *addrlen);
```

- `sockfd`: descriptor del socket.
- `addr`: puntero a una estructura `struct sockaddr_in` local, donde se guardará la información de la conexión entrante y con ella se puede averiguar qué máquina se está conectando, y desde qué puerto.
- `addrlen`: longitud de la estructura anterior.
- Resultado:
 - Devuelve un descriptor de un socket conectado al cliente.
 - Será por este nuevo socket (no por el original) por el que haremos toda la comunicación con el cliente que se acaba de conectar.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Ejemplo de accept ()

```
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define MYPORT 3490 /* Puerto al que se conectarán los clientes */
#define BACKLOG 10 /* Número de conexiones que vamos a mantener en cola */

void main() {
    int sockfd, new_fd; /* Se escucha sobre sock_fd, nuevas conexiones sobre new_fd */
    struct sockaddr_in my_addr; /* Información sobre mi dirección */
    struct sockaddr_in their_addr; /* Información sobre la dirección remota */
    int sin_size;

    sockfd = socket(PF_INET, SOCK_STREAM, 0);

    my_addr.sin_family = AF_INET; /* Ordenación de la máquina */
    my_addr.sin_port = htons(MYPORT); /* Ordenación de la red */
    my_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    memset(&(my_addr.sin_zero), '\0', 8);
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Conectar el `socket`: `connect ()`

- Se realiza en los clientes:
 - Obligatorio en los *stream*, opcional en los *dgram*.
 - Se denominan “sockets activos”

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

int connect(int sockfd, struct sockaddr *serv_addr, int addrlen);
```

- `sockfd`: descriptor del socket.
- `serv_addr`: es un puntero a una estructura `sockaddr` que contiene el puerto y la dirección IP del destino.
- `addrlen`: longitud de la estructura anterior.
- Resultado:

The logo for Cartagena99, featuring the text 'Cartagena99' in a stylized font with a blue and orange gradient background.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Conectar el *socket*: `connect ()`

- Conectar un *socket* consiste en asociarlo con una dirección IP y un puerto destino:
 - Obligatorio en los *sockets stream* (`SOCK_STREAM`).
 - Hace un establecimiento de conexión (TCP).
 - Da error si no puede establecer la conexión.
 - Opcional en los *sockets datagram* (`SOCK_DGRAM`).
 - No establece conexión (UDP).
 - Sólo almacena la dirección destino localmente:
 - Aunque no dé error, puede no existir o ser incorrecta.
 - Simplemente permite transmitir sin tener que especificar la dirección destino cada vez que se envía un mensaje.

The logo for Cartagena99, featuring the text 'Cartagena99' in a stylized font with a blue and orange gradient background.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Ejemplo de connect ()

```
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define DEST_IP "10.12.110.57"
#define DEST_PORT 23

void main() {
    int sockfd;
    struct sockaddr_in dest_addr; /* Guardará la dirección de destino */

    sockfd = socket(PF_INET, SOCK_STREAM, 0); /* Chequear si devuelve -1! */
    dest_addr.sin_family = AF_INET; /* Ordenación de la máquina */
    dest_addr.sin_port = htons(DEST_PORT); /* Ordenación de la red */
    dest_addr.sin_addr.s_addr = inet_addr(DEST_IP);
    /* Pone a cero el resto de la estructura */
    memset(&dest_addr.sin_zero, 0, sizeof(dest_addr.sin_zero));
}
```

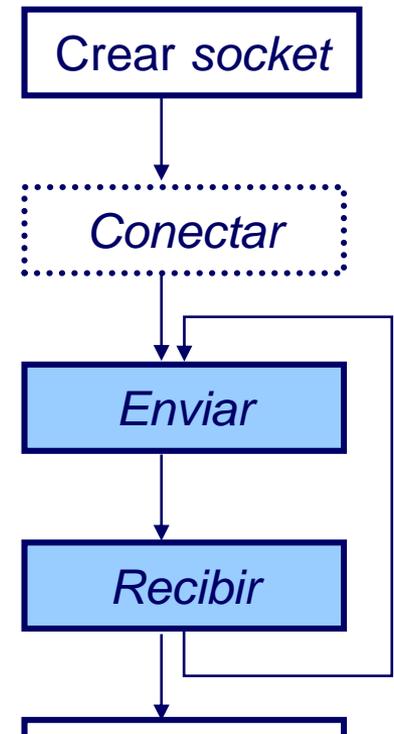
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Enviar y recibir datos

- Para sockets *stream* (o clientes *dgram* que hayan hecho `connect()`):
 - `send()`
 - `recv()`
- Para sockets *dgram*:
 - Hay que indicar en cada mensaje la IP y puerto del otro extremo.
 - `sendto()`
 - `recvfrom()`



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

Enviar datos *socket* conectado: `send()`

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

ssize_t send(int sockfd, const void *msg, size_t len, int flags);
```

- `sockfd`: descriptor del socket.
- `msg`: puntero a los datos que se quieren enviar.
- `len`: longitud de los datos en octetos.
- `flags`: normalmente a 0.
- Resultado:
 - Número de octetos enviados. (¡¡¡ Puede ser menor que `len`!!!)
 - “-1” si se ha producido un error.

Cartagena99

CLÁSES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Enviar datos *socket* no conectado: `sendto ()`

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

ssize_t sendto(int sockfd, const void *msg, size_t len, int flags,
               const struct sockaddr *to, socklen_t tolen);
```

- `sockfd`: descriptor del socket.
- `msg`: puntero a los datos que se quieren enviar.
- `len`: longitud de los datos en octetos.
- `flags`: normalmente a 0.
- `to`: contiene información sobre la IP y puerto del destino.
- `tolen`: longitud de la estructura anterior.
- Resultado:
 - Número de octetos enviados. (¡¡¡ Puede ser menor que `len`!!!)

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Recibir datos *socket* conectado: `recv()`

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

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

- `sockfd`: descriptor del socket.
- `buf`: el buffer donde se va a depositar la información leída .
- `len`: longitud máxima del buffer.
- `flags`: normalmente a 0.
- Resultado:
 - Número de octetos recibidos.
 - "0" la máquina remota ha cerrado la conexión.
 - "-1" si se ha producido un error.

The logo for Cartagena99, featuring the text 'Cartagena99' in a stylized, green, serif font with a blue and orange gradient background behind the letters.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Recibir datos *socket* no conectado: `recvfrom()`

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

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

- **sockfd**: descriptor del socket.
- **buf**: el buffer donde se va a depositar la información leída .
- **len**: longitud máxima del buffer.
- **flags**: normalmente a 0.
- **from**: contiene información sobre la IP y puerto de la máquina de origen.
- **fromlen**: longitud de la estructura anterior.
- Resultado:
 - Número de octetos recibidos.
 - "0" la máquina remota ha cerrado la conexión.

The logo for Cartagena99, featuring the text "Cartagena99" in a stylized, green, serif font with a blue and orange gradient background behind the letters.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Recepción de datos no bloqueante

- Las operaciones de lectura sobre un *socket* son normalmente bloqueantes.
 - Cuando se realiza una lectura de sobre un *socket*, el proceso queda dormido esperando que complete la operación (llegan datos o se cierra la conexión).
- En ocasiones es recomendable que sea no bloqueante y poder realizar otras tareas en vez de esperar a que los datos estén disponibles.
- Dos formas de hacerlo:
 - `fcntl()`
 - `select()`

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Recepción de datos no bloqueante: `fcntl()`

- `fcntl()` es una función de control que nos permite realizar diferentes operaciones sobre descriptores (de ficheros, de sockets,...) en general.

```
#include <fcntl.h>

int fcntl(int fd, int cmd, /* int arg*/);
```

- Cada descriptor tiene asociado un conjunto de flags que nos permiten conocer ciertas características del descriptor.
 - Para obtener el valor de estos flags, se realiza una llamada a `fcntl()` con el parámetro `cmd` al valor `F_GETFL`. Para modificar el valor de los flags, se utiliza el valor `F_SETFL`.
- Para indicar que las operaciones de entrada y salida sobre un socket no sean bloqueantes, es necesario activar el flag

○ `NONBLOCK` en el descriptor del socket. El código necesario para

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

```
if ( fcntl(sd, F_SETFL, O_NONBLOCK) < 0 )
```

```
return fcntl(sd, F_SETFL, O_NONBLOCK | O_NONBLOCK);
```

Recepción de datos no bloqueante: `select()`

```
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>

int select(int numfds, fd_set *readfds, fd_set *writefds,
           fd_set *exceptfds, struct timeval *timeout);
```

Permite manejar en nuestra aplicación varios sockets y comprobar si nos ha llegado datos por alguno de ellos. Le pasamos un conjunto de descriptors de sockets a comprobar:

- **numfds**: valor del mayor descriptor de fichero más uno.
- **readfds**: conjunto de descriptors de lectura.
- **writefds**: conjunto de descriptors de escritura.
- **exceptfds**: conjunto de descriptors que producen excepciones.
- **timeout**: temporizador que hará regresar a `select()` incluso aunque ningún descriptor de fichero esté listo.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

- "-1" si se ha producido un error.

Otras llamadas para enviar/recibir

- Existen otras llamadas equivalentes:
 - `int sendmsg(int sockfd, const struct msghdr *msg, int flags)`
 - `int recvmsg(int s, struct msghdr *msg, int flags)`
- Se puede utilizar también los métodos de escritura y lectura de E/S:
 - `int write(socket, buffer, length)`
 - `int read(socket, buffer, length)`

The logo for Cartagena99, featuring the text 'Cartagena99' in a stylized font with a blue and orange gradient background.

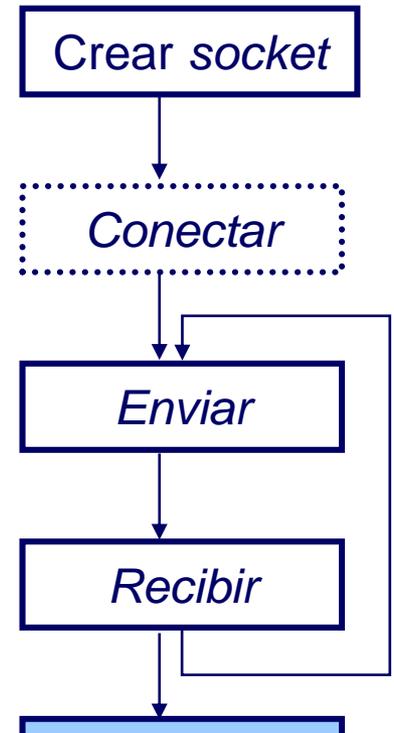
CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

- - -

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cerrar el socket

- Se puede hacer con dos llamadas:
 - `close()`
 - `shutdown()`



Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cerrar el *socket*: `close ()`, `shutdown ()`

```
int close(sockfd) ;
```

- `sockfd`: descriptor del socket.
- Si se cierra un socket con `close` el otro extremo cuando intente leer o escribir de ese socket recibirá un error. Para tener más control y realizar por ejemplo un “half-close” es necesario emplear la llamada:

```
int shutdown(int sockfd, int how) ;
```

- `sockfd`: descriptor del socket.
- `how`: toma uno de los siguientes valores:
 - “0” no se permite recibir más datos.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Llamadas auxiliares: gethostbyname ()

```
#include <netdb.h>

struct hostent *gethostbyname(const char *name);
```

```
struct hostent {
    char    *h_name;
    char    **h_aliases;
    int     h_addrtype;
    int     h_length;
    char    **h_addr_list;
};
#define h_addr h_addr_list[0]
```

- **h_name** : nombre oficial de la máquina.
- **h_aliases** : vector [array] terminado en **NULL** de nombres alternativos de máquina.
- **h_addrtype** : tipo de la dirección que se devuelve; usualmente **AF_INET**.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Ejemplo de gethostbyname ()

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

int main(int argc, char *argv[]){
    struct hostent *h;

    if (argc != 2) { // Comprobación de errores en la línea de comandos
        fprintf(stderr,"usage: getip address\n");
        exit(1);
    }

    if ((h=gethostbyname(argv[1])) == NULL) { // Obtener información del host
        perror("gethostbyname");
        exit(1);
    }
}
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Llamadas auxiliares: gethostbyaddr ()

```
#include <netdb.h>
```

```
struct hostent *gethostbyaddr(const void *addr, int len, int type);
```

```
struct sockaddr_in clientaddr;  
int clientlen = sizeof(clientaddr);  
int new_fd = accept(sockfd, (struct sockaddr *)&clientaddr, &clientlen);  
if (new_fd == -1) {  
    perror("ERROR on accept");  
    exit(0);  
}  
  
/* gethostbyaddr: determine who sent the message */  
struct hostent *host = gethostbyaddr((const char *)&clientaddr.sin_addr.s_addr,  
                                     sizeof(clientaddr.sin_addr.s_addr), AF_INET);  
  
if (host == NULL) {  
    perror("ERROR on gethostbyaddr");  
    exit(0);  
}  
char *hostaddr = inet_ntoa(clientaddr.sin_addr);
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Otras llamadas auxiliares

```
#include <unistd.h>

int gethostname(char *hostname, size_t size);
```

- **hostname**: puntero a una cadena de caracteres donde se almacenará el nombre de la máquina cuando la función retorne.
- **size**: longitud en octetos de la cadena de caracteres anterior.

```
#include <unistd.h>

int sethostname(char *hostname, size_t size);
```

- **hostname**: puntero a una cadena de caracteres donde se indica el nombre que se quiere poner a la máquina.

The logo for Cartagena99, featuring the text 'Cartagena99' in a stylized font with a blue and orange gradient background.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Otras llamadas auxiliares

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

int getsockopt(int sockfd, int level, int optname, void *optval,
               socklen_t *optlen);
```

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

int setsockopt(int sockfd, int level, int optname, const void *optval,
               socklen_t *optlen);
```

- Permiten obtener y establecer, respectivamente, opciones del socket o del protocolo (por ejemplo, temporizadores, si permite difusión,...).
- Lo usaremos habitualmente en servidores para poder reutilizar inmediatamente un número de puerto, sin esperar 2MSL.

– Se puede hacer siempre y cuando no coincidan los cuatro valores IP origen, IP destino

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

– Los clientes usan `send()` igual que si fuera unicast, pero asignando una

dirección de difusión en la estructura `sockaddr_in` de `send()`.

Ejemplo de setsockopt ()

```
// Nota: faltan todos los #includes.
// Ejemplo de reutilización de puerto usado recientemente

main(int argc, char *argv[]) {
    int sockfd;
    int port = 5355;
    struct sockaddr_in serveraddr; // server's addr
    int optval = 1; // flag value for setsockopt

    sockfd = socket(PF_INET, SOCK_STREAM, 0);

    /* setsockopt: lets s rerun the server immediately after we kill it.
     * Eliminates "ERROR on binding: Address already in use" error. */
    setsockopt(new_fd, SOL_SOCKET, SO_REUSEADDR, (const void *)&optval ,
               sizeof(int));

    bzero((char *) &serveraddr, sizeof(serveraddr));
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Ejemplo de setsockopt ()

```
// Nota: faltan todos los #includes.
// Ejemplo de servidor que escucha por una dirección de difusión
#define PORT 5355
#define GROUP "239.255.255.253"
main(int argc, char *argv[]) {
    struct sockaddr_in addr;
    struct ip_mreq mreq;
    int fd;

    fd=socket(AF_INET,SOCK_DGRAM,0); // Hay que comprobar si devuelve -1
    memset(&addr,0,sizeof(addr));
    addr.sin_family=AF_INET;
    addr.sin_addr.s_addr=htonl(INADDR_ANY);
    addr.sin_port=htons(PORT);
    // Hay que comprobar si devuelve -1
    bind(fd,(struct sockaddr *)&addr,sizeof(addr));

    mreq.imr_multiaddr.s_addr=inet_addr(GROUP); // Se une al grupo de difusión
```

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

...

Cliente UDP

1. Averiguar la dirección IP y puerto del servidor con el que desea comunicarse.
2. Crear un socket: llamada a `socket ()`.
3. Especificar que UDP use cualquier puerto local libre.
4. Especificar a qué servidor deben enviarse los mensajes.
 - Con un `connect ()` se hacen los pasos 3 y 4.
 - Si no, habrá que especificar la dirección en cada mensaje que se envíe (en `sendto`).
5. Comunicarse con el servidor.
 - Enviar y recibir (llamadas a `sendto/recvfrom`, `send/recv`, ...).

The logo for Cartagena99, featuring the text 'Cartagena99' in a stylized font with a blue and orange gradient background.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Servidor UDP

1. Crear un socket: llamada a `socket()`.
2. Asociarlo al puerto local “bien conocido” que ofrece ese servicio: llamada a `bind()`.
3. Bucle:
 - Recibir y enviar datos al cliente (llamadas a `sendto/recvfrom`).
4. Cerrar socket: llamada a `close()`.

The logo for Cartagena99, featuring the text 'Cartagena99' in a stylized font with a blue and orange gradient background.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cliente TCP

1. Averiguar la dirección IP y puerto del servidor con el que desea comunicarse.
2. Crear un socket: llamada a `socket ()`
3. Especificar que TCP use cualquier puerto local libre.
 - Lo hace automáticamente el `connect ()` si no se hace previamente un `bind ()`.
4. Conectar con el servidor.
 - Esto abre la conexión TCP: llamada a `connect ()`.
5. Comunicarse con el servidor
 - Enviar y recibir por el socket (llamadas a `send/recv`).
6. Cerrar el socket: llamada a `close ()`.

The logo for Cartagena99, featuring the text 'Cartagena99' in a stylized font with a blue and orange gradient background.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Servidor TCP

1. Crear un socket: llamada a `socket()`.
2. Asociarlo al puerto local “bien conocido” que ofrece ese servicio: llamada a `bind()`.
3. Poner el socket en modo pasivo, dispuesto a aceptar peticiones de conexión: llamada a `listen()`.
4. Bucle:
 - Recibir petición de cliente.
 - Atender petición (por nuevo socket): llamada a `accept()`
 - Enviar/recibir por socket (llamadas a `send/recv`).
 - Cerrar nuevo socket (llamada a `close()`) y volver a punto 3.
5. Cerrar socket: llamada a `close()`.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Servidor TCP concurrente

1. Crear un socket.
2. Asociarlo al puerto local “bien conocido” que ofrece ese servicio.
3. Poner el socket en modo pasivo, dispuesto a aceptar peticiones de conexión.
4. Cuando recibe petición de cliente:
 - Crea un nuevo proceso hijo:
 - Le pasa el nuevo socket para atender esa petición.
 - Vuelve al punto 3.
5. Cerrar socket.
6. En paralelo, el proceso esclavo interactúa con el cliente.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

- - -

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70