

# **LIBRERIA SOCKET**

AUTHOR  
Version 1  
CREATEDATE



# Tabla de contenidos

Table of contents

## Índice de estructura de datos

### Jerarquía de la clase

Esta lista de herencias esta ordenada aproximadamente por orden alfabético:

Socket.....	3
tcp.....	6
TCPclient.....	6
TCPserver.....	9
Udp.....	11
UDPclient.....	12
UDPserver.....	14

## Índice de estructura de datos

### Estructura de datos

Lista de estructuras con una breve descripción:

Socket (Clase genérica de tipo socket ) .....	3
tcp (Clase genérica de tipo TCP ) .....	6
TCPclient (Clase de un cliente de tipo TCP ) .....	6
TCPserver (Clase de un servidor tipo TCP ) .....	9
Udp (Clase genérica de tipo UDP ) .....	11
UDPclient (Clase de un cliente de tipo UDP ) .....	12
UDPserver (Clase de un servidor tipo UDP ) .....	14

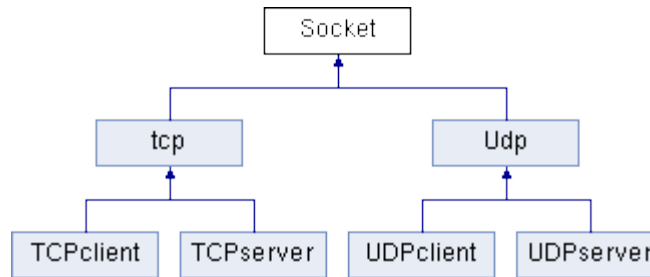
## Documentación de las estructuras de datos

### Referencia de la Clase Socket

Clase genérica de tipo socket.

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

Diagrama de herencias de Socket



## Métodos públicos

- 1 void **bufferChar2String** (char \*cbuf)  
*Conversión de tipo <char> a tipo <string>*
- 2 void **bufferString2Char** (string sbuf)  
*Conversión de tipo <string> a tipo <char>*
- 3 void **print** ()  
*Imprimimos por pantalla lo que esté almacenado en la variable ccbuffer.*
- 4 virtual void **checkConnection** (int argc)=0  
*Comprobamos la conexión.*
- 5 virtual void **socket** (char \*argv[])=0  
*Creación del socket.*
- 6 virtual void **enviar** (string buffer)=0  
*Función Enviar.*
- 7 virtual void **recibir** ()=0  
*Función Recibir.*
- 8 string **getSbuffer** ()  
*Función que devuelve el valor del atributo sbuffer.*

## Atributos protegidos

- 9 int **sockfd**  
*File descriptor de nuestro socket.*
- 10 int **n**  
*Aquí almacenamos los <return> de **enviar()** y **recibir()***
- 11 string **sbuffer**  
*Aquí almacenamos lo que recibimos en formato <string>*
- 12 char \* **cbuffer**  
*Aquí almacenamos los char que enviamos.*
- 13 char **ccbuffer** [1024]  
*Aquí almacenamos los char que recibimos.*

---

## Descripción detallada

Clase genérica de tipo socket.

Definición en la línea 22 del archivo socket.h.

---

## Documentación de las funciones miembro

### **void Socket::bufferChar2String (char \*cbuf)**

Conversión de tipo <char> a tipo <string>

< Borramos lo que hubiera guardado en sbuffer

< Copiamos lo que haya en cbuf a sbuffer

Definición en la línea 11 del archivo socket.cpp.

```
{  
    sbuffer.clear();  
    sbuffer = cbuf;  
}
```

### **virtual void Socket::checkConnection (intargc) [pure virtual]**

Comprobamos la conexión.

#### **Parámetros:**

<i>argc</i>	Contiene el número de argumentos del programa.
-------------	--

Si el número de argumentos es menos del deseado entonces se muestra un error por pantalla y se sale del programa

Implementado en **TCPclient** (p.7), **TCPserver** (p.10), **UDPclient** (p.13) y **UDPserver** (p.15).

### **virtual void Socket::enviar (stringbuffer) [pure virtual]**

Función Enviar.

#### **Parámetros:**

<i>buffer</i>	Mensaje a enviar
---------------	------------------

Implementado en **TCPclient** (p.8), **TCPserver** (p.10), **UDPclient** (p.13) y **UDPserver** (p.15).

### **virtual void Socket::socket (char \*argv[]) [pure virtual]**

Creación del socket.

#### **Parámetros:**

<i>argv[]</i>	Aquí se pasa por parametro la dirección IP del servidor y el número de puerto.
---------------	--

Creamos el socket y vamos comprobando que no hay errores durante el proceso.

Implementado en **TCPclient** (p.8), **TCPserver** (p.11), **UDPclient** (p.14) y **UDPserver** (p.16).

La documentación para esta clase fue generada a partir de los siguientes ficheros:

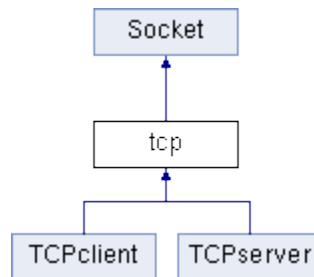
```
14 src/lib/socket.h
15 src/lib/socket.cpp
```

## Referencia de la Clase tcp

Clase genérica de tipo TCP.

```
#include <TCP.h>
```

Diagrama de herencias de tcp



### Atributos protegidos

```
16 struct sockaddr_in serv_addr
```

*Estructura donde se almacenan los datos del servidor.*

---

## Descripción detallada

Clase genérica de tipo TCP.

Es hija de **Socket**

Definición en la línea 12 del archivo TCP.h.

---

La documentación para esta clase fue generada a partir de los siguientes ficheros:

```
17 src/lib/TCP.h
18 src/lib/TCP.cpp
```

## Referencia de la Clase TCPclient

Clase de un cliente de tipo TCP.

```
#include <tcpclient.h>
```

Diagrama de herencias de TCPclient



## Métodos públicos

```

19 void socket (char *argv[])
   Creación del socket.
20 void checkConnection (int argc)
   Comprobamos la conexión.
21 void enviar (string buffer)
   Función Enviar.
22 void recibir ()
   Función Recibir.

```

## Atributos privados

```

23 struct hostent * server
   Estructura donde se almacena la dirección del servidor tcp.

```

---

## Descripción detallada

Clase de un cliente de tipo TCP.

Es hija de TCP

Definición en la línea 12 del archivo tcpclient.h.

---

## Documentación de las funciones miembro

**void TCPclient::checkConnection (intargc) [virtual]**

Comprobamos la conexión.

### Parámetros:

<i>argc</i>	Contiene el número de argumentos del programa.
-------------	--

Si el número de argumentos es menos del deseado entonces se muestra un error por pantalla y se sale del programa

Implementa **Socket** (p.5).

Definición en la línea 32 del archivo tcpclient.cpp.

```

{
    if (argc < 3)
    {

```

```

        cout<<"ERROR, falta la ip del servidor o el puerto"<<endl;
        exit(0);
    }
}

```

**void TCPClient::enviar (stringbuffer) [virtual]**

Función Enviar.

**Parámetros:**

<i>buffer</i>	Mensaje a enviar
---------------	------------------

Implementa **Socket** (p.5).

Definición en la línea 42 del archivo tcpclient.cpp.

```

{
    bufferString2Char(buffer);
    n = write(sockfd,cbuffer,strlen(cbuffer));
    if (n < 0) error("ERROR writing to socket");
}

```

**void TCPClient::socket (char \*argv[]) [virtual]**

Creación del socket.

**Parámetros:**

<i>argv[]</i>	Aquí se pasa por parametro la dirección IP del servidor y el número de puerto.
---------------	--

Creamos el socket y vamos comprobando que no hay errores durante el proceso.

Implementa **Socket** (p.5).

Definición en la línea 10 del archivo tcpclient.cpp.

```

{
    sockfd = ::socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    server = gethostbyname(argv[1]);
    if (server == NULL) {
        fprintf(stderr,"ERROR, no such host\n");
        exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr,
        (char *)&serv_addr.sin_addr.s_addr,
        server->h_length);
    serv_addr.sin_port = htons(atoi(argv[2]));
    if (connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr)) < 0)
        error("ERROR connecting");
}

```

---

**La documentación para esta clase fue generada a partir de los siguientes ficheros:**

- 24 src/lib/tcpclient.h
- 25 src/lib/tcpclient.cpp

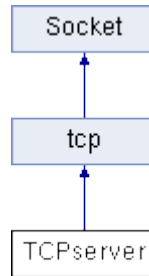


## Referencia de la Clase TCPserver

Clase de un servidor tipo TCP.

```
#include <tcpserver.h>
```

Diagrama de herencias de TCPserver



### Métodos públicos

26 void **socket** (char \*argv[])

*Creación del socket.*

27 void **checkConnection** (int argc)

*Comprobamos la conexión.*

28 void **conectar** ()

*Conectamos el servidor con el cliente.*

29 void **enviar** (string buffer)

*Función Enviar.*

30 void **recibir** ()

*Función Recibir.*

### Atributos privados

31 int **newsockfd**

*File descriptor del socket creado para la conexión con el cliente.*

32 int **clilen**

*Tamaño de la estructura donde almacenamos el cliente a enlazar.*

33 struct sockaddr\_in **cli\_addr**

*Estructura de nuestro cliente.*

---

### Descripción detallada

Clase de un servidor tipo TCP.

Es hija de TCP

Definición en la línea 12 del archivo tcpserver.h.

---

## Documentación de las funciones miembro

### **void TCPserver::checkConnection (intargc) [virtual]**

Comprobamos la conexión.

#### **Parámetros:**

<i>argc</i>	Contiene el número de argumentos del programa.
-------------	--

Si el número de argumentos es menos del deseado entonces se muestra un error por pantalla y se sale del programa

Implementa **Socket** (p.5).

Definición en la línea 26 del archivo tcpserver.cpp.

```
{
    if (argc < 2)
    {
        cout<<"ERROR, falta el puerto"<<endl;
        exit(1);
    }
}
```

### **void TCPserver::conectar ()**

Conectamos el servidor con el cliente.

En esta función el servidor se queda a la espera de conexiones entrantes y se conecta en cuanto recibe un mensaje.

Definición en la línea 35 del archivo tcpserver.cpp.

```
{
    newsockfd = accept(sockfd,
        (struct sockaddr *) &cli_addr, (socklen_t *) &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
}
```

### **void TCPserver::enviar (stringbuffer) [virtual]**

Función Enviar.

#### **Parámetros:**

<i>buffer</i>	Mensaje a enviar
---------------	------------------

Implementa **Socket** (p.5).

Definición en la línea 43 del archivo tcpserver.cpp.

```
{
    bufferString2Char(buffer);
    n = write(newsockfd, cbuffer, strlen(cbuffer));
    if (n < 0) error("ERROR writing to socket");
}
```

**void TCPserver::socket (char \*argv[]) [virtual]**

Creación del socket.

**Parámetros:**

<i>argv[]</i>	Aquí se pasa por parametro la dirección IP del servidor y el número de puerto.
---------------	--

Creamos el socket y vamos comprobando que no hay errores durante el proceso.

Implementa **Socket** (p.5).

Definición en la línea 10 del archivo tcpserver.cpp.

```
{
    sockfd = ::socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(atoi(argv[1]));
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
        sizeof(serv_addr)) < 0)
        error("ERROR on binding");
    listen(sockfd,5);
    cliilen = sizeof(cli_addr);
}
```

---

## Documentación de los campos

**int TCPserver::newsockfd [private]**

File descriptor del socket creado para la conexión con el cliente.

Definición en la línea 15 del archivo tcpserver.h.

---

**La documentación para esta clase fue generada a partir de los siguientes ficheros:**

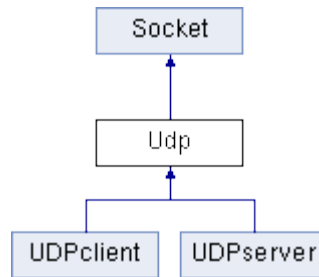
```
34 src/lib/tcpserver.h
35 src/lib/tcpserver.cpp
```

## Referencia de la Clase Udp

Clase genérica de tipo UDP.

```
#include <udp.h>
```

Diagrama de herencias de Udp



### Atributos protegidos

```

36 unsigned int length
37 socklen_t fromlen
38 struct sockaddr_in server from
39 struct hostent * hp
40 char buf [1024]
  
```

### Descripción detallada

Clase genérica de tipo UDP.

Es hija de **Socket**

Definición en la línea 12 del archivo udp.h.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

```

41 src/lib/udp.h
42 src/lib/udp.cpp
  
```

### Referencia de la Clase UDPclient

Clase de un cliente de tipo UDP.

```
#include <udpclient.h>
```

Diagrama de herencias de UDPclient



### Métodos públicos

```

43 void socket (char *argv[])
   Creación del socket.
44 void checkConnection (int argc)
   Comprobamos la conexión.
45 void enviar (string buffer)
  
```

*Función Enviar:*

46 void **recibir** ()

*Función Recibir:*

---

## Descripción detallada

Clase de un cliente de tipo UDP.

Es hija de UDP

Definición en la línea 12 del archivo udpclient.h.

---

## Documentación de las funciones miembro

**void UDPclient::checkConnection (intargc) [virtual]**

Comprobamos la conexión.

### Parámetros:

<i>argc</i>	Contiene el número de argumentos del programa.
-------------	--

Si el número de argumentos es menos del deseado entonces se muestra un error por pantalla y se sale del programa

Implementa **Socket** (p.5).

Definición en la línea 24 del archivo udpclient.cpp.

```
{
    if (argc != 3)
    {
        cout<<"ERROR, falta la ip del servidor o el puerto"<<endl;
        exit(1);
    }
}
```

**void UDPclient::enviar (stringbuffer) [virtual]**

Función Enviar.

### Parámetros:

<i>buffer</i>	Mensaje a enviar
---------------	------------------

Implementa **Socket** (p.5).

Definición en la línea 34 del archivo udpclient.cpp.

```
{
    bufferString2Char(buffer);
    n=sendto(sockfd,cbuffer,strlen(cbuffer),0,(const struct sockaddr
*)&server,length);
    if (n < 0) error("Sendto");
}
```

**void UDPclient::socket (char \*argv[]) [virtual]**

Creación del socket.

**Parámetros:**

<code>argv[]</code>	Aquí se pasa por parametro la dirección IP del servidor y el número de puerto.
---------------------	--

Creamos el socket y vamos comprobando que no hay errores durante el proceso.

Implementa **Socket** (p.5).

Definición en la línea 9 del archivo udpclient.cpp.

```
{
    sockfd = ::socket(AF_INET, SOCK_DGRAM, 0);
    if(sockfd<0) error("socket");
    server.sin_family = AF_INET;
    hp = gethostbyname(argv[1]);
    if (hp==0) error("Unknown host");

    bcopy((char *)hp->h_addr,
          (char *)&server.sin_addr,
          hp->h_length);
    server.sin_port = htons(atoi(argv[2]));
    length=sizeof(struct sockaddr_in);
}
```

---

**La documentación para esta clase fue generada a partir de los siguientes ficheros:**

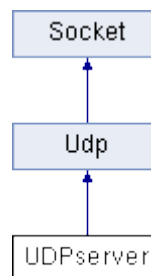
```
47 src/lib/udpclient.h
48 src/lib/udpclient.cpp
```

## Referencia de la Clase UDPserver

Clase de un servidor tipo UDP.

```
#include <udpserver.h>
```

Diagrama de herencias de UDPserver



### Métodos públicos

```
49 void socket (char *argv[])
    Creación del socket.
50 void checkConnection (int argc)
    Comprobamos la conexión.
51 void enviar (string buffer)
```

*Función Enviar:*

52 void **recibir** ()

*Función Recibir:*

---

## Descripción detallada

Clase de un servidor tipo UDP.

Es hija de UDP

Definición en la línea 12 del archivo udpserver.h.

---

## Documentación de las funciones miembro

**void UDPserver::checkConnection (int`argc`) [virtual]**

Comprobamos la conexión.

### Parámetros:

<code>argc</code>	Contiene el número de argumentos del programa.
-------------------	--

Si el número de argumentos es menos del deseado entonces se muestra un error por pantalla y se sale del programa

Implementa **Socket** (p.5).

Definición en la línea 25 del archivo udpserver.cpp.

```
{
    if (argc != 2)
    {
        cout<<"ERROR, falta el puerto"<<endl;
        exit(0);
    }
}
```

**void UDPserver::enviar (string`buffer`) [virtual]**

Función Enviar.

### Parámetros:

<code>buffer</code>	Mensaje a enviar
---------------------	------------------

Implementa **Socket** (p.5).

Definición en la línea 35 del archivo udpserver.cpp.

```
{
    bufferString2Char(buffer);
    n = sendto(sockfd,cbuffer,strlen(cbuffer),0,(struct sockaddr *)&from,fromlen);
    if (n < 0) error("sendto");
}
```

**void UDPserver::socket (char \*argv[]) [virtual]**

Creación del socket.

**Parámetros:**

<i>argv[]</i>	Aquí se pasa por parametro la dirección IP del servidor y el número de puerto.
---------------	--

Creamos el socket y vamos comprobando que no hay errores durante el proceso.

Implementa **Socket** (p.5).

Definición en la línea 10 del archivo udpserver.cpp.

```
{
    sockfd = ::socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) error("Opening socket");
    length = sizeof(server);
    bzero(&server,length);
    server.sin_family=AF_INET;
    server.sin_addr.s_addr=INADDR_ANY;
    server.sin_port=htons(atoi(argv[1]));
    // Enlazamos el socket a la dirección IP y puerto del servidor
    if (bind(sockfd, (struct sockaddr *)&server,length)<0)
        error("binding");
    fromlen = sizeof(struct sockaddr_in);
}
```

---

**La documentación para esta clase fue generada a partir de los siguientes ficheros:**

53 src/lib/udpserver.h  
54 src/lib/udpserver.cpp

## Índice

INDEX