

Servidor y cliente IRC

Generated by Doxygen 1.8.11

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	_ Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Field Documentation	5
3.1.2.1	sck	5
3.1.2.2	ssl	5
3.2	audioArgs Struct Reference	6
3.2.1	Detailed Description	6
3.2.2	Field Documentation	6
3.2.2.1	ip	6
3.2.2.2	port	6
3.3	fileReceiver_args Struct Reference	6
3.3.1	Detailed Description	6
3.3.2	Field Documentation	7
3.3.2.1	length	7
3.3.2.2	path	7
3.3.2.3	sckF	7
3.4	fileSender_args Struct Reference	7
3.4.1	Detailed Description	7
3.4.2	Field Documentation	7
3.4.2.1	data	7
3.4.2.2	length	7
3.4.2.3	sck	7

4	File Documentation	9
4.1	/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P1-command.h File Reference	9
4.2	/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P1-socket.h File Reference	9
4.2.1	Detailed Description	10
4.2.2	Function Documentation	10
4.2.2.1	acceptSocket(int sck)	10
4.2.2.2	bindSocket(int sck, uint16_t port, int max_clients, protocol p)	11
4.2.2.3	connectClientSocket(int sck, char *host_name, int port)	11
4.2.2.4	openSocket(protocol p)	11
4.2.2.5	receiveData(int sck, SSL *ssl, protocol p, char *dest_addrUDP, int portUDP, char *data, int len)	11
4.2.2.6	sendData(int sck, SSL *ssl, protocol p, char *dest_addrUDP, int portUDP, char *data, int len)	12
4.3	/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-audio.h File Reference	12
4.3.1	Detailed Description	13
4.3.2	Function Documentation	13
4.3.2.1	audioChat(char *sender, char *msg)	13
4.3.2.2	audioRecv(void *args)	13
4.3.2.3	audioSend(void *args)	14
4.4	/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-files.h File Reference	14
4.4.1	Detailed Description	14
4.4.2	Function Documentation	14
4.4.2.1	fileDialog(char *nick, char *msg)	14
4.4.2.2	fileReceiver(void *args)	15
4.4.2.3	fileSender(void *fs)	15
4.5	/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-messages.h File Reference	15
4.5.1	Detailed Description	17
4.5.2	Function Documentation	17
4.5.2.1	msgAway(char *m_in)	17
4.5.2.2	msgBack(char *m_in)	17
4.5.2.3	msgCreated(char *m_in)	17

4.5.2.4	msgDefault(char *m_in)	18
4.5.2.5	msgEndNames(char *m_in)	18
4.5.2.6	msgIsSupport(char *m_in)	18
4.5.2.7	msgJoin(char *m_in)	19
4.5.2.8	msgKick(char *m_in)	19
4.5.2.9	msgList(char *m_in)	19
4.5.2.10	msgMode(char *m_in)	20
4.5.2.11	msgModeChanged(char *m_in)	20
4.5.2.12	msgMOTD(char *m_in)	20
4.5.2.13	msgMOTDEnd(char *m_in)	20
4.5.2.14	msgMOTDStart(char *m_in)	21
4.5.2.15	msgMyInfo(char *m_in)	21
4.5.2.16	msgNames(char *m_in)	21
4.5.2.17	msgNickChanged(char *m_in)	22
4.5.2.18	msgNoNickOrChannel(char *m_in)	22
4.5.2.19	msgNotice(char *m_in)	22
4.5.2.20	msgNotOperator(char *m_in)	23
4.5.2.21	msgPart(char *m_in)	23
4.5.2.22	msgPing(char *m_in)	23
4.5.2.23	msgPrivmsg(char *m_in)	23
4.5.2.24	msgQuit(char *m_in)	24
4.5.2.25	msgRplAway(char *m_in)	24
4.5.2.26	msgTopicChanged(char *m_in)	24
4.5.2.27	msgWelcome(char *m_in)	25
4.5.2.28	msgWho(char *m_in)	25
4.5.2.29	msgWhoisChannels(char *m_in)	25
4.5.2.30	msgWhoisEnd(char *m_in)	26
4.5.2.31	msgWhoisIdle(char *m_in)	26
4.5.2.32	msgWhoisMode(char *m_in)	26
4.5.2.33	msgWhoisServer(char *m_in)	26

4.5.2.34	msgWhoisUser(char *m_in)	27
4.5.2.35	msgYourHost(char *m_in)	27
4.6	/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-userCommands.h File Reference	27
4.6.1	Detailed Description	28
4.6.2	Function Documentation	28
4.6.2.1	userCommandAway(char *command, SSL *ssl)	28
4.6.2.2	userCommandBack(char *command, SSL *ssl)	29
4.6.2.3	userCommandDefault(char *command, SSL *ssl)	29
4.6.2.4	userCommandJoin(char *command, SSL *ssl)	29
4.6.2.5	userCommandKick(char *command, SSL *ssl)	30
4.6.2.6	userCommandList(char *command, SSL *ssl)	30
4.6.2.7	userCommandMode(char *command, SSL *ssl)	30
4.6.2.8	userCommandNames(char *command, SSL *ssl)	30
4.6.2.9	userCommandNick(char *command, SSL *ssl)	31
4.6.2.10	userCommandPart(char *command, SSL *ssl)	31
4.6.2.11	userCommandPrivmsg(char *command, SSL *ssl)	31
4.6.2.12	userCommandQuit(char *command, SSL *ssl)	32
4.6.2.13	userCommandTopic(char *command, SSL *ssl)	32
4.6.2.14	userCommandWhois(char *command, SSL *ssl)	32
4.7	/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P3-redes2.h File Reference	33
4.7.1	Detailed Description	33
4.8	/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P3-ssl.h File Reference	33
4.8.1	Detailed Description	34
4.8.2	Function Documentation	34
4.8.2.1	aceptar_canal_seguro_SSL(SSL_CTX *ctx, int sock)	34
4.8.2.2	cerrar_canal_SSL(SSL *ssl, SSL_CTX *ctx)	34
4.8.2.3	conectar_canal_seguro_SSL(SSL_CTX *ctx, int sock)	34
4.8.2.4	enviar_datos_SSL(SSL *ssl, char *buffer, int nbytes)	35
4.8.2.5	evaluar_post_conectar_SSL(const SSL *ssl)	35
4.8.2.6	fijar_contexto_SSL(char *ca_cert, char *clserv_cert)	35
4.8.2.7	inicializar_nivel_SSL(char *ca_cert, char *clserv_pem)	36
4.8.2.8	recibir_datos_SSL(SSL *ssl, char *buffer, int nbytes)	36

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

_	5
audioArgs	
Estructura con los parametros necesarios para las distintas funciones de audio	6
fileReceiver_args	
Estructura argumento de la funcion fileReceiver, con los parámetros necesarios para esta . . .	6
fileSender_args	
Estructura argumento de la funcion fileSender, con los parámetros necesarios para esta . . .	7

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P1-command.h	9
/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P1-socket.h	9
/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-audio.h	12
/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-files.h	14
/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-messages.h	15
/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-userCommands.h	27
/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P3-redes2.h	33
/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P3-ssl.h	33

Chapter 3

Data Structure Documentation

3.1 _ Struct Reference

```
#include <G-2301-04-P1-command.h>
```

Data Fields

- int [sck](#)
- SSL * [ssl](#)

3.1.1 Detailed Description

estructura para guardar los argumentos de attendClient

3.1.2 Field Documentation

3.1.2.1 sck

socket en caso del cliente. Parametro obligtorio

3.1.2.2 ssl

Estructura ssl necesaria si se ha iniciado una conexion segura

The documentation for this struct was generated from the following file:

- [/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P1-command.h](#)

3.2 audioArgs Struct Reference

Estructura con los parametros necesarios para las distintas funciones de audio.

```
#include <G-2301-04-P2-audio.h>
```

Data Fields

- char * [ip](#)
- int [port](#)

3.2.1 Detailed Description

Estructura con los parametros necesarios para las distintas funciones de audio.

3.2.2 Field Documentation

3.2.2.1 ip

Direccion IP del que envía o recibe el audio

3.2.2.2 port

Puerto en el que se establece la comunicación

The documentation for this struct was generated from the following file:

- [/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-audio.h](#)

3.3 fileReceiver_args Struct Reference

Estructura argumento de la funcion fileReceiver, con los parámetros necesarios para esta.

```
#include <G-2301-04-P2-files.h>
```

Data Fields

- int [sckF](#)
- char * [path](#)
- long unsigned int [length](#)

3.3.1 Detailed Description

Estructura argumento de la funcion fileReceiver, con los parámetros necesarios para esta.

3.3.2 Field Documentation

3.3.2.1 length

Longitud del fichero

3.3.2.2 path

Ruta en la que se va a guardar el fichero

3.3.2.3 sckF

Socket en el que previamente se ha abierto la conexion con el otro extremo

The documentation for this struct was generated from the following file:

- [/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-files.h](#)

3.4 fileSender_args Struct Reference

Estructura argumento de la funcion fileSender, con los parámetros necesarios para esta.

```
#include <G-2301-04-P2-files.h>
```

Data Fields

- char * [data](#)
- long unsigned int [length](#)
- int [sck](#)

3.4.1 Detailed Description

Estructura argumento de la funcion fileSender, con los parámetros necesarios para esta.

3.4.2 Field Documentation

3.4.2.1 data

Datos del fichero

3.4.2.2 length

Longitud del fichero

3.4.2.3 sck

Socket en el que se espera al receptor del fichero

The documentation for this struct was generated from the following file:

- [/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-files.h](#)

Chapter 4

File Documentation

4.1 /home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P1-command.h File Reference

```
#include <stdio.h>
#include <string.h>
#include <redes2/irc.h>
```

Include dependency graph for G-2301-04-P1-command.h:

4.2 /home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P1-socket.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <errno.h>
#include <netdb.h>
#include <redes2/irc.h>
#include <pthread.h>
#include <arpa/inet.h>
#include <signal.h>
#include <sys/stat.h>
#include <syslog.h>
#include <fcntl.h>
#include "G-2301-04-P3-ssl.h"
```

Include dependency graph for G-2301-04-P1-socket.h: This graph shows which files directly or indirectly include this file:

Macros

- `#define MAXLEN 512`
Longitud maxima de un comando IRC.

Enumerations

- enum `protocol` { `TCP`, `UDP` }

Enumeracion con los protocolos de nivel de transporte que es capaz de soportar nuestra aplicación.

Functions

- void `daemonizar` ()
Daemoniza el proceso.
- int `openSocket` (`protocol` p)
Abre un socket para comunicarse con el servidor.
- int `bindSocket` (int `sck`, uint16_t port, int max_clients, `protocol` p)
Dado un socket notifica al SO de la apertura del mismo.
- int `acceptSocket` (int `sck`)
Acepta conexiones de clientes.
- int `connectClientSocket` (int `sck`, char *host_name, int port)
Dado un socket, conecta con una IP y un puerto específico. Específica para clientes.
- int `sendData` (int `sck`, SSL *ssl, `protocol` p, char *dest_addrUDP, int portUDP, char *data, int len)
Dado un socket o una estructura ssl envia datos al otro extremo de la conexion.
- int `receiveData` (int `sck`, SSL *ssl, `protocol` p, char *dest_addrUDP, int portUDP, char *data, int len)
Dado un socket o una estructura ssl recibe datos al otro extremo de la conexion.

4.2.1 Detailed Description

Modulo de sockets. Contiene funciones de alto nivel para el manejo de sockets así como la función que manejan los hilos.

Author

Antonio Amor Mourelle <antonio.amor@estudiante.uam.es>
 Esther Lopez Ramos <esther.lopezramos@estudiante.uam.es>
 Mario Santiago Yepes <mario.santiagoy@estudiante.uam.es>

4.2.2 Function Documentation

4.2.2.1 int acceptSocket (int `sck`)

Acepta conexiones de clientes.

Parameters

<code>sck</code>	identificador de fichero del socket
------------------	-------------------------------------

Returns

-1 en caso de error, identificador de fichero del socket del cliente

4.2.2.2 int bindSocket (int *sck*, uint16_t *port*, int *max_clients*, protocol *p*)

Dado un socket notifica al SO de la apertura del mismo.

Parameters

<i>sck</i>	descriptor del socket
<i>port</i>	puerto con el protocolo de nivel de aplicacion
<i>max_clients</i>	numero maximo de clientes esperando a ser atendidos
<i>p</i>	protocolo de nivel de aplicaion TCP o UDP

Returns

0 si se realiza todo correctamente, -1 en caso de error

4.2.2.3 int connectClientSocket (int *sck*, char * *host_name*, int *port*)

Dado un socket, conecta con una IP y un puerto especifico. Especifica para clientes.

Parameters

<i>sck</i>	identificador de fichero con el socket
<i>host_name</i>	url o direccion IP del servidor
<i>port</i>	numero de puerto

Returns

0 si todo ha ido bien, -1 en caso de error

4.2.2.4 int openSocket (protocol *p*)

Abre un socket para comunicarse con el servidor.

Parameters

<i>p</i>	UDP en caso de querer conexion no fiable TCP conexion segura (valor por defecto)
----------	----------------------------------------------------------------------------------

Returns

devuelve el descriptor de fichero del socket, -1 en caso de error

4.2.2.5 int receiveData (int *sck*, SSL * *ssl*, protocol *p*, char * *dest_addrUDP*, int *portUDP*, char * *data*, int *len*)

Dado un socket o una estructura ssl recibe datos al otro extremo de la conexion.

Parameters

<i>sck</i>	identificador de fichero con el socket
<i>ssl</i>	puntero a una estructura ssl. Si vale NULL se asume canal no seguro
<i>p</i>	protocolo TCP o UDP (se asume TCP por defecto)
<i>dest_addrUDP</i>	nombre del host al que vamos a enviar en UDP
<i>portUDP</i>	puerto al que enviamos los datos en UDP, debe coincidir con el del socket
<i>data</i>	datos que se envian
<i>len</i>	de los datos que se envian

Returns

numero de bytes leidos si todo correcto
 -1 en caso de error

4.2.2.6 `int sendData (int sck, SSL * ssl, protocol p, char * dest_addrUDP, int portUDP, char * data, int len)`

Dado un socket o una estructura ssl envia datos al otro extremo de la conexion.

Parameters

<i>sck</i>	identificador de fichero con el socket
<i>ssl</i>	puntero a una estructura ssl. Si vale NULL se asume canal no seguro
<i>p</i>	protocolo TCP o UDP (se asume TCP por defecto)
<i>dest_addrUDP</i>	nombre del host al que vamos a enviar en UDP
<i>portUDP</i>	puerto al que enviamos los datos en UDP, debe coincidir con el del socket
<i>data</i>	datos que se envian
<i>len</i>	de los datos que se envian

Returns

numero de bytes leidos si todo correcto
 -1 en caso de error

4.3 /home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-audio.h File Reference

```
#include "G-2301-04-P2-messages.h"
```

Include dependency graph for G-2301-04-P2-audio.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [audioArgs](#)

Estructura con los parametros necesarios para las distintas funciones de audio.

Functions

- int [audioChat](#) (char *sender, char *msg)
Gestiona el envio y recibo de archivos de audio.
- void * [audioSend](#) (void *args)
Gestiona el envio de ficheros de audio.
- void * [audioRecv](#) (void *args)
Gestiona el recibo de ficheros de audio.

4.3.1 Detailed Description

Modulo de audio. Contiene funciones de alto nivel para el envio de audio desde la interfaz de usuario.

Author

Antonio Amor Mourelle <antonio.amor@estudiante.uam.es>
Esther Lopez Ramos <esther.lopezramos@estudiante.uam.es>
Mario Santiago Yepes <mario.santiagoy@estudiante.uam.es>

4.3.2 Function Documentation

4.3.2.1 int audioChat (char * sender, char * msg)

Gestiona el envio y recibo de archivos de audio.

Parameters

<i>sender</i>	nick del cliente que envia audio
<i>msg</i>	comando que contiene la informacion necesaria para gestionar el envio

Returns

-1 en caso de error, identificador de fichero del socket del cliente

4.3.2.2 void* audioRecv (void * args)

Gestiona el recibo de ficheros de audio.

Parameters

<i>args</i>	estructura que contiene informacion sobre el recibo
-------------	-----------------------------------------------------

Returns

-1 en caso de error, identificador de fichero del socket del cliente

4.3.2.3 void* audioSend (void * args)

Gestiona el envio de ficheros de audio.

Parameters

<i>args</i>	estructura que contiene informacion sobre el envio
-------------	----------------------------------------------------

Returns

-1 en caso de error, identificador de fichero del socket del cliente

4.4 /home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-files.h File Reference

This graph shows which files directly or indirectly include this file:

Data Structures

- struct [fileReceiver_args](#)
Estructura argumento de la funcion fileReceiver, con los parámetros necesarios para esta.
- struct [fileSender_args](#)
Estructura argumento de la funcion fileSender, con los parámetros necesarios para esta.

Functions

- void * [fileReceiver](#) (void *args)
Gestiona el proceso de recepcion de ficheros.
- int [fileDialog](#) (char *nick, char *msg)
Acepta la recepcion del envio de ficheros y abre un socket para ello.
- void * [fileSender](#) (void *fs)
Gestiona el proceso de envio.

4.4.1 Detailed Description

Modulo de ficheros. Contiene funciones de alto nivel para el envio de ficheros desde la interfaz de usuario.

Author

Antonio Amor Mourelle <antonio.amor@estudiante.uam.es>
 Esther Lopez Ramos <esther.lopezramos@estudiante.uam.es>
 Mario Santiago Yepes <mario.santiagoy@estudiante.uam.es>

4.4.2 Function Documentation

4.4.2.1 int fileDialog (char * nick, char * msg)

Acepta la recepcion del envio de ficheros y abre un socket para ello.

Parameters

<i>nick</i>	nick del emisor
<i>msg</i>	comando que contiene la informacion necesaria para gestionar la recepcion

Returns

-1 en caso de error, identificador de fichero del socket del cliente

4.4.2.2 void* fileReceiver (void * args)

Gestiona el proceso de recepcion de ficheros.

Parameters

<i>args</i>	estructura que contiene informacion sobre la recepcion
-------------	--------------------------------------------------------

Returns

-1 en caso de error, identificador de fichero del socket del cliente

4.4.2.3 void* fileSender (void * fs)

Gestiona el proceso de envio.

Parameters

<i>fs</i>	estructura que contiene informacion sobre el envio
-----------	----------------------------------------------------

Returns

-1 en caso de error, identificador de fichero del socket del cliente

4.5 /home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-messages.h File Reference

```
#include <redes2/irc.h>
#include <redes2/ircxchat.h>
#include "G-2301-04-P1-socket.h"
```

Include dependency graph for G-2301-04-P2-messages.h: This graph shows which files directly or indirectly include this file:

Macros

- `#define MAX_TCP 65535`
Longitud máxima en bytes de un paquete TCP.

Typedefs

- typedef int(* [pMsg](#)) (char *m_in)
Definimos el tipo funcion mensaje.

Functions

- int [msgDefault](#) (char *m_in)
- int [msgNickChanged](#) (char *m_in)
- int [msgJoin](#) (char *m_in)
- int [msgMode](#) (char *m_in)
- int [msgNames](#) (char *m_in)
- int [msgWho](#) (char *m_in)
- int [msgNotice](#) (char *m_in)
- int [msgWelcome](#) (char *m_in)
- int [msgYourHost](#) (char *m_in)
- int [msgCreated](#) (char *m_in)
- int [msgMyInfo](#) (char *m_in)
- int [msgIsSupport](#) (char *m_in)
- int [msgMOTDStart](#) (char *m_in)
- int [msgMOTD](#) (char *m_in)
- int [msgMOTDEnd](#) (char *m_in)
- int [msgList](#) (char *m_in)
- int [msgEndNames](#) (char *m_in)
- int [msgNotOperator](#) (char *m_in)
- int [msgModeChanged](#) (char *m_in)
- int [msgWhoisUser](#) (char *m_in)
- int [msgWhoisServer](#) (char *m_in)
- int [msgWhoisChannels](#) (char *m_in)
- int [msgWhoisMode](#) (char *m_in)
- int [msgWhoisIdle](#) (char *m_in)
- int [msgWhoisEnd](#) (char *m_in)
- int [msgPrivmsg](#) (char *m_in)
- int [msgPing](#) (char *m_in)
- int [msgTopicChanged](#) (char *m_in)
- int [msgNoNickOrChannel](#) (char *m_in)
- int [msgPart](#) (char *m_in)
- int [msgQuit](#) (char *m_in)
- int [msgKick](#) (char *m_in)
- int [msgAway](#) (char *m_in)
- int [msgBack](#) (char *m_in)
- int [msgRplAway](#) (char *m_in)

Variables

- SSL * [ssl_channel](#)
Variable con la estructura ssl del cliente IRC.
- int [sck](#)
Variable con el descriptor de fichero del socket del cliente IRC.
- char [hostName](#) [[MAXLEN](#)]
Nombre de host que nos asigna el servidor al entrar en el. Usado para el envío de ficheros.
- int [stopAudio](#)
Variable que le indica al hilo emisor o receptor de audio que debe parar de enviar.
- [pMsg](#) [Messages](#) [[2048](#)]
Array de punteros a las funciones de mensajes del servidor.

4.5.1 Detailed Description

Modulo de mensajes. Contiene funciones de alto nivel para la gestion de mensajes que llegan al servidor.

Author

Antonio Amor Mourelle <antonio.amor@estudiante.uam.es>
Esther Lopez Ramos <esther.lopezramos@estudiante.uam.es>
Mario Santiago Yepes <mario.santiagoy@estudiante.uam.es>

4.5.2 Function Documentation

4.5.2.1 int msgAway (char * m_in)

Parsea el mensaje de Away

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.2 int msgBack (char * m_in)

Parsea el mensaje de Back (UNAWAY)

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.3 int msgCreated (char * m_in)

Parsea el mensaje de CREATED

Parameters

m_{\leftarrow} _in	mensaje que se va a parsear
-------------------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.4 int msgDefault (char * *m_in*)

Accion por defecto para los mensajes

Parameters

m_{\leftarrow} _in	mensaje que se va a parsear
-------------------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.5 int msgEndNames (char * *m_in*)

Parsea el mensaje de ENDOFNAMES

Parameters

m_{\leftarrow} _in	mensaje que se va a parsear
-------------------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.6 int msgIsSupport (char * *m_in*)

Parsea el mensaje de ISSUPPORT

Parameters

m_{\leftarrow} _in	mensaje que se va a parsear
-------------------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.7 int msgJoin (char * *m_in*)

Parsea el mensaje de JOIN

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.8 int msgKick (char * *m_in*)

Parsea el mensaje de Kick

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.9 int msgList (char * *m_in*)

Parsea el mensaje de LIST

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.10 int msgMode (char * *m_in*)

Parsea el mensaje de MODE

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.11 int msgModeChanged (char * *m_in*)

Parsea el mensaje de que se recibe se cambia el modo de un canal

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.12 int msgMOTD (char * *m_in*)

Parsea el mensaje de MOTD

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.13 int msgMOTDEnd (char * *m_in*)

Parsea el mensaje de MOTD

Parameters

$m \leftarrow$ _in	mensaje que se va a parsear
-----------------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.14 int msgMOTDStart (char * m_in)

Parsea el mensaje de MOTD

Parameters

$m \leftarrow$ _in	mensaje que se va a parsear
-----------------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.15 int msgMyInfo (char * m_in)

Parsea el mensaje de MYINFO

Parameters

$m \leftarrow$ _in	mensaje que se va a parsear
-----------------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.16 int msgNames (char * m_in)

Parsea el mensaje de NAMES

Parameters

$m \leftarrow$ _in	mensaje que se va a parsear
-----------------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.17 int msgNickChanged (char * *m_in*)

Parsea el mensaje de NICK

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.18 int msgNoNickOrChannel (char * *m_in*)

Parsea el mensaje de NO NICK OR CHANNEL

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.19 int msgNotice (char * *m_in*)

Parsea el mensaje de NOTICE

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.20 int msgNotOperator (char * m_in)

Parsea el mensaje de You're not channel operator

Parameters

m_{in}	mensaje que se va a parsear
----------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.21 int msgPart (char * m_in)

Parsea el mensaje de Part

Parameters

m_{in}	mensaje que se va a parsear
----------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.22 int msgPing (char * m_in)

Parsea el mensaje de Ping enviado por el servidor Y ENVIA EL PONG

Parameters

m_{in}	mensaje que se va a parsear
----------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.23 int msgPrivmsg (char * m_in)

Parsea el mensaje de PRIVMSG

Parameters

m_{in}	mensaje que se va a parsear
----------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.24 int msgQuit (char * m_{in})

Parsea el mensaje de Quit

Parameters

m_{in}	mensaje que se va a parsear
----------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.25 int msgrplAway (char * m_{in})

Parsea el mensaje de Away despues de mandar un privmsg

Parameters

m_{in}	mensaje que se va a parsear
----------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.26 int msgTopicChanged (char * m_{in})

Parsea el mensaje de Ping enviado por el servidor Y ENVIA EL PONG

Parameters

m_{in}	mensaje que se va a parsear
----------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.27 int msgWelcome (char * *m_in*)

Parsea el mensaje de WELCOME

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.28 int msgWho (char * *m_in*)

Parsea el mensaje de WHO

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.29 int msgWhoisChannels (char * *m_in*)

Parsea el mensaje de WHOISCHANNELS

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.30 int msgWhoisEnd (char * *m_in*)

Parsea el mensaje de WHOISEND

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.31 int msgWhoisIdle (char * *m_in*)

Parsea el mensaje de WHOISIDLE

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.32 int msgWhoisMode (char * *m_in*)

Parsea el mensaje de WHOISIDLE

Parameters

<i>m_in</i>	mensaje que se va a parsear
-------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.33 int msgWhoisServer (char * *m_in*)

Parsea el mensaje de WHOISSERVER

Parameters

$m \leftarrow$ _in	mensaje que se va a parsear
-----------------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.34 int msgWhoisUser (char * m_in)

Parsea el mensaje de WHOISUSER

Parameters

$m \leftarrow$ _in	mensaje que se va a parsear
-----------------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.5.2.35 int msgYourHost (char * m_in)

Parsea el mensaje de YOURHOST

Parameters

$m \leftarrow$ _in	mensaje que se va a parsear
-----------------------	-----------------------------

Returns

0 si OK
-1 ERROR

4.6 /home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-userCommands.h File Reference

```
#include "../includes/G-2301-04-P2-messages.h"
```

Include dependency graph for G-2301-04-P2-userCommands.h: This graph shows which files directly or indirectly include this file:

Typedefs

- typedef int(* [pUCommand](#)) (char *m_in, SSL *ssl)
Definimos el tipo funcion de comando de usuario.

Functions

- int [userCommandDefault](#) (char *command, SSL *ssl)
- int [userCommandNames](#) (char *command, SSL *ssl)
- int [userCommandList](#) (char *command, SSL *ssl)
- int [userCommandJoin](#) (char *command, SSL *ssl)
- int [userCommandNick](#) (char *command, SSL *ssl)
- int [userCommandWhois](#) (char *command, SSL *ssl)
- int [userCommandPrivmsg](#) (char *command, SSL *ssl)
- int [userCommandMode](#) (char *command, SSL *ssl)
- int [userCommandKick](#) (char *command, SSL *ssl)
- int [userCommandPart](#) (char *command, SSL *ssl)
- int [userCommandQuit](#) (char *command, SSL *ssl)
- int [userCommandAway](#) (char *command, SSL *ssl)
- int [userCommandBack](#) (char *command, SSL *ssl)
- int [userCommandTopic](#) (char *command, SSL *ssl)

Variables

- [pUCommand UserCommands](#) [64]
Array de punteros a las funciones de comandos de usuario.

4.6.1 Detailed Description

Modulo de comandos de usuario. Contiene funciones de alto nivel para la gestion de comandos que ejecuta el cliente.

Author

Antonio Amor Mourelle <antonio.amor@estudiante.uam.es>
 Esther Lopez Ramos <esther.lopezramos@estudiante.uam.es>
 Mario Santiago Yepes <mario.santiagoy@estudiante.uam.es>

4.6.2 Function Documentation

4.6.2.1 int userCommandAway (char * command, SSL * ssl)

Parsea el comando de usuario AWAY crea el mensaje para el servidor y lo envia

Parameters

<i>command</i>	commando de usuario que se va a enviar
<i>ssl</i>	estructura de ssl. En caso de no usar la capa segura debe valer NULL

Returns

0 si OK
-1 ERROR

4.6.2.2 int userCommandBack (char * *command*, SSL * *ssl*)

Parsea el comando de usuario BACK

Parameters

<i>command</i>	commando de usuario que se va a enviar
<i>ssl</i>	estructura de ssl. En caso de no usar la capa segura debe valer NULL

Returns

0 si OK
-1 ERROR

4.6.2.3 int userCommandDefault (char * *command*, SSL * *ssl*)

Accion por defecto para los mensajes

Parameters

<i>command</i>	commando de usuario que se va a enviar
<i>ssl</i>	estructura de ssl. En caso de no usar la capa segura debe valer NULL

Returns

0 si OK
-1 ERROR

4.6.2.4 int userCommandJoin (char * *command*, SSL * *ssl*)

Parsea el comando de usuario JOIN crea el mensaje para el servidor y lo envia

Parameters

<i>command</i>	commando de usuario que se va a enviar
<i>ssl</i>	estructura de ssl. En caso de no usar la capa segura debe valer NULL

Returns

0 si OK
-1 ERROR

4.6.2.5 int userCommandKick (char * *command*, SSL * *ssl*)

Parsea el comando de usuario KICK crea el mensaje para el servidor y lo envia

Parameters

<i>command</i>	commando de usuario que se va a enviar
<i>ssl</i>	estructura de ssl. En caso de no usar la capa segura debe valer NULL

Returns

0 si OK
-1 ERROR

4.6.2.6 int userCommandList (char * *command*, SSL * *ssl*)

Parsea el comando de usuario LIST crea el mensaje para el servidor y lo envia

Parameters

<i>command</i>	commando de usuario que se va a enviar
<i>ssl</i>	estructura de ssl. En caso de no usar la capa segura debe valer NULL

Returns

0 si OK
-1 ERROR

4.6.2.7 int userCommandMode (char * *command*, SSL * *ssl*)

Parsea el comando de usuario MODE crea el mensaje para el servidor y lo envia

Parameters

<i>command</i>	commando de usuario que se va a enviar
<i>ssl</i>	estructura de ssl. En caso de no usar la capa segura debe valer NULL

Returns

0 si OK
-1 ERROR

4.6.2.8 int userCommandNames (char * *command*, SSL * *ssl*)

Parsea el comando de usuario NAMES crea el mensaje para el servidor y lo envia

Parameters

<i>command</i>	commando de usuario que se va a enviar
<i>ssl</i>	estructura de ssl. En caso de no usar la capa segura debe valer NULL

Returns

0 si OK
-1 ERROR

4.6.2.9 int userCommandNick (char * *command*, SSL * *ssl*)

Parsea el comando de usuario nick crea el mensaje para el servidor y lo envia

Parameters

<i>command</i>	commando de usuario que se va a enviar
<i>ssl</i>	estructura de ssl. En caso de no usar la capa segura debe valer NULL

Returns

0 si OK
-1 ERROR

4.6.2.10 int userCommandPart (char * *command*, SSL * *ssl*)

Parsea el comando de usuario PART crea el mensaje para el servidor y lo envia

Parameters

<i>command</i>	commando de usuario que se va a enviar
<i>ssl</i>	estructura de ssl. En caso de no usar la capa segura debe valer NULL

Returns

0 si OK
-1 ERROR

4.6.2.11 int userCommandPrivmsg (char * *command*, SSL * *ssl*)

Parsea el comando de usuario PRIVMSG crea el mensaje para el servidor y lo envia

Parameters

<i>command</i>	commando de usuario que se va a enviar
<i>ssl</i>	estructura de ssl. En caso de no usar la capa segura debe valer NULL

Returns

0 si OK
-1 ERROR

4.6.2.12 int userCommandQuit (char * *command*, SSL * *ssl*)

Parsea el comando de usuario QUIT crea el mensaje para el servidor y lo envia

Parameters

<i>command</i>	commando de usuario que se va a enviar
<i>ssl</i>	estructura de ssl. En caso de no usar la capa segura debe valer NULL

Returns

0 si OK
-1 ERROR

4.6.2.13 int userCommandTopic (char * *command*, SSL * *ssl*)

Parsea el comando de usuario TOPIC

Parameters

<i>command</i>	commando de usuario que se va a enviar
<i>ssl</i>	estructura de ssl. En caso de no usar la capa segura debe valer NULL

Returns

0 si OK
-1 ERROR

4.6.2.14 int userCommandWhois (char * *command*, SSL * *ssl*)

Parsea el comando de usuario PRIVMSG crea el mensaje para el servidor y lo envia

Parameters

<i>command</i>	commando de usuario que se va a enviar
<i>ssl</i>	estructura de ssl. En caso de no usar la capa segura debe valer NULL

Returns

0 si OK
-1 ERROR

4.7 /home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P3-redes2.h File Reference

```
#include "G-2301-04-P1-socket.h"
#include "G-2301-04-P1-command.h"
#include "G-2301-04-P2-files.h"
#include "G-2301-04-P2-audio.h"
#include "G-2301-04-P2-messages.h"
#include "G-2301-04-P2-userCommands.h"
#include "G-2301-04-P3-ssl.h"
Include dependency graph for G-2301-04-P3-redes2.h:
```

4.7.1 Detailed Description

Modulo de redes2. Contiene todas las funciones de la libreria implementada.

Author

Antonio Amor Mourelle <antonio.amor@estudiante.uam.es>
 Esther Lopez Ramos <esther.lopezramos@estudiante.uam.es>
 Mario Santiago Yepes <mario.santiagoy@estudiante.uam.es>

4.8 /home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P3-ssl.h File Reference

```
#include <stdio.h>
#include <openssl/err.h>
#include <openssl/ssl.h>
#include "G-2301-04-P1-socket.h"
Include dependency graph for G-2301-04-P3-ssl.h: This graph shows which files directly or indirectly include this file:
```

Functions

- `SSL_CTX * inicializar_nivel_SSL (char *ca_cert, char *clserv_pem)`
Realiza todas las llamadas necesarias para que la aplicación pueda usar la capa segura SSL.
- `SSL_CTX * fijar_contexto_SSL (char *ca_cert, char *clserv_cert)`
Inicializa el contexto que será utilizado para la creación de canales seguros mediante SSL.
- `SSL * conectar_canal_seguro_SSL (SSL_CTX *ctx, int sck)`
Dado un contexto SSL y un descriptor de socket obtiene un canal seguro SSL iniciando el proceso de handshake con el otro extremo.
- `SSL * aceptar_canal_seguro_SSL (SSL_CTX *ctx, int sck)`
Dado un contexto SSL y un descriptor de socket esta función se queda esperando hasta recibir un handshake por parte del cliente.
- `int evaluar_post_connectar_SSL (const SSL *ssl)`
comprobará una vez realizado el handshake que el canal de comunicación se puede considerar seguro
- `int enviar_datos_SSL (SSL *ssl, char *buffer, int nbytes)`
Envia datos a traves de la conexion ssl preestablecida.
- `int recibir_datos_SSL (SSL *ssl, char *buffer, int nbytes)`
Recibe datos a traves de la conexion ssl preestablecida.
- `void cerrar_canal_SSL (SSL *ssl, SSL_CTX *ctx)`
Libera los recursos reservados para la capa ssl.

4.8.1 Detailed Description

Modulo de ssl. Contiene funciones de alto nivel para la gestion de uncliente en modo seguridad ssl.

Author

Antonio Amor Mourelle <antonio.amor@estudiante.uam.es>

Esther Lopez Ramos <esther.lopezramos@estudiante.uam.es>

Mario Santiago Yepes <mario.santiagoy@estudiante.uam.es>

4.8.2 Function Documentation

4.8.2.1 SSL* aceptar_canal_seguro_SSL (SSL_CTX * *ctx*, int *sck*)

Dado un contexto SSL y un descriptor de socket esta función se queda esperando hasta recibir un handshake por parte del cliente.

Parameters

<i>ctx</i>	contexto de la aplicacion
<i>sck</i>	descriptor del socket

Returns

puntero a una estructura ssl con la conexion creada
NULL en caso de error

4.8.2.2 void cerrar_canal_SSL (SSL * *ssl*, SSL_CTX * *ctx*)

Libera los recursos reservados para la capa ssl.

Parameters

<i>ssl</i>	puntero a la estructura de conexión ssl
<i>ctx</i>	contexto creado para la conexion ssl

4.8.2.3 SSL* conectar_canal_seguro_SSL (SSL_CTX * *ctx*, int *sck*)

Dado un contexto SSL y un descriptor de socket obtiene un canal seguro SSL iniciando el proceso de handshake con el otro extremo.

Parameters

<i>ctx</i>	contexto de la aplicacion
<i>sck</i>	descriptor del socket

Returns

puntero a una estructura ssl con la conexión creada
NULL en caso de error

4.8.2.4 int enviar_datos_SSL (SSL * *ssl*, char * *buffer*, int *nbytes*)

Envía datos a través de la conexión ssl preestablecida.

Parameters

<i>ssl</i>	puntero a la estructura de conexión ssl
<i>buffer</i>	cadena con los datos enviados
<i>nbytes</i>	numero de bytes que se envían

Returns

-1 en caso de error
0 correcto

4.8.2.5 int evaluar_post_conectar_SSL (const SSL * *ssl*)

comprobará una vez realizado el handshake que el canal de comunicación se puede considerar seguro

Parameters

<i>ssl</i>	puntero a la estructura de conexión ssl
------------	-----------------------------------------

Returns

-1 en caso de error
0 correcto

4.8.2.6 SSL_CTX* fijar_contexto_SSL (char * *ca_cert*, char * *clserv_cert*)

Inicializa el contexto que será utilizado para la creación de canales seguros mediante SSL.

Parameters

<i>ca_cert</i>	nombre del certificado de la CA
<i>clserv_cert</i>	ruta del certificado de la CA

Returns

contexto creado
NULL en caso de error

4.8.2.7 SSL_CTX* inicializar_nivel_SSL (char * *ca_cert*, char * *clserv_pem*)

Realiza todas las llamadas necesarias para que la aplicación pueda usar la capa segura SSL.

Parameters

<i>ca_cert</i>	nombre del certificado de la CA
<i>clserv_pem</i>	ruta del certificado de la CA

Returns

contexto creado
NULL en caso de error

4.8.2.8 int recibir_datos_SSL (SSL * *ssl*, char * *buffer*, int *nbytes*)

Recibe datos a través de la conexión ssl preestablecida.

Parameters

<i>ssl</i>	puntero a la estructura de conexión ssl
<i>buffer</i>	cadena con los datos recibidos
<i>nbytes</i>	numero de bytes máximos que se pueden recibir

Returns

<=0 en caso de error
0 correcto

Index

/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P1-command.h, [9](#)
/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P1-socket.h, [9](#)
/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-audio.h, [12](#)
/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-files.h, [14](#)
/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-messages.h, [15](#)
/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P2-userCommands.h, [27](#)
/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P3-redes2.h, [33](#)
/home/esther/Escritorio/REDES-II/SSL/includes/G-2301-04-P3-ssl.h, [33](#)

_, [5](#)

sck, [5](#)

ssl, [5](#)

acceptSocket

G-2301-04-P1-socket.h, [10](#)

aceptar_canal_seguro_SSL

G-2301-04-P3-ssl.h, [34](#)

audioArgs, [6](#)

ip, [6](#)

port, [6](#)

audioChat

G-2301-04-P2-audio.h, [13](#)

audioRecv

G-2301-04-P2-audio.h, [13](#)

audioSend

G-2301-04-P2-audio.h, [13](#)

bindSocket

G-2301-04-P1-socket.h, [10](#)

cerrar_canal_SSL

G-2301-04-P3-ssl.h, [34](#)

conectar_canal_seguro_SSL

G-2301-04-P3-ssl.h, [34](#)

connectClientSocket

G-2301-04-P1-socket.h, [11](#)

data

fileSender_args, [7](#)

enviar_datos_SSL

G-2301-04-P3-ssl.h, [35](#)

evaluar_post_conectar_SSL

G-2301-04-P3-ssl.h, [35](#)

fijar_contexto_SSL

G-2301-04-P3-ssl.h, [35](#)

fileDialog

G-2301-04-P2-files.h, [14](#)

fileReceiver

G-2301-04-P2-files.h, [15](#)

fileReceiver_args, [6](#)

length, [7](#)

path, [7](#)

sckF, [7](#)

fileSender

G-2301-04-P2-files.h, [15](#)

fileSender_args, [7](#)

data, [7](#)

length, [7](#)

sck, [7](#)

G-2301-04-P1-socket.h

acceptSocket, [10](#)

bindSocket, [10](#)

connectClientSocket, [11](#)

openSocket, [11](#)

receiveData, [11](#)

sendData, [12](#)

G-2301-04-P2-audio.h

audioChat, [13](#)

audioRecv, [13](#)

audioSend, [13](#)

G-2301-04-P2-files.h

fileDialog, [14](#)

fileReceiver, [15](#)

fileSender, [15](#)

G-2301-04-P2-messages.h

msgAway, [17](#)

msgBack, [17](#)

msgCreated, [17](#)

msgDefault, [18](#)

msgEndNames, [18](#)

msgIsSupport, [18](#)

msgJoin, [19](#)

msgKick, [19](#)

msgList, [19](#)

msgMOTDEnd, [20](#)

msgMOTDStart, [21](#)

msgMOTD, [20](#)

msgMode, [19](#)

msgModeChanged, [20](#)

msgMyInfo, [21](#)

msgNames, [21](#)

msgNickChanged, [22](#)

- msgNoNickOrChannel, [22](#)
- msgNotOperator, [22](#)
- msgNotice, [22](#)
- msgPart, [23](#)
- msgPing, [23](#)
- msgPrivmsg, [23](#)
- msgQuit, [24](#)
- msgTopicChanged, [24](#)
- msgWelcome, [25](#)
- msgWho, [25](#)
- msgWhoisChannels, [25](#)
- msgWhoisEnd, [25](#)
- msgWhoisIdle, [26](#)
- msgWhoisMode, [26](#)
- msgWhoisServer, [26](#)
- msgWhoisUser, [27](#)
- msgYourHost, [27](#)
- msgRplAway, [24](#)
- G-2301-04-P2-userCommands.h
 - userCommandAway, [28](#)
 - userCommandBack, [29](#)
 - userCommandDefault, [29](#)
 - userCommandJoin, [29](#)
 - userCommandKick, [29](#)
 - userCommandList, [30](#)
 - userCommandMode, [30](#)
 - userCommandNames, [30](#)
 - userCommandNick, [31](#)
 - userCommandPart, [31](#)
 - userCommandPrivmsg, [31](#)
 - userCommandQuit, [32](#)
 - userCommandTopic, [32](#)
 - userCommandWhois, [32](#)
- G-2301-04-P3-ssl.h
 - aceptar_canal_seguro_SSL, [34](#)
 - cerrar_canal_SSL, [34](#)
 - conectar_canal_seguro_SSL, [34](#)
 - enviar_datos_SSL, [35](#)
 - evaluar_post_conectar_SSL, [35](#)
 - fijar_contexto_SSL, [35](#)
 - inicializar_nivel_SSL, [35](#)
 - recibir_datos_SSL, [36](#)
- inicializar_nivel_SSL
 - G-2301-04-P3-ssl.h, [35](#)
- ip
 - audioArgs, [6](#)
- length
 - fileReceiver_args, [7](#)
 - fileSender_args, [7](#)
- msgAway
 - G-2301-04-P2-messages.h, [17](#)
- msgBack
 - G-2301-04-P2-messages.h, [17](#)
- msgCreated
 - G-2301-04-P2-messages.h, [17](#)
- msgDefault
 - G-2301-04-P2-messages.h, [18](#)
- msgEndNames
 - G-2301-04-P2-messages.h, [18](#)
- msgIsSupport
 - G-2301-04-P2-messages.h, [18](#)
- msgJoin
 - G-2301-04-P2-messages.h, [19](#)
- msgKick
 - G-2301-04-P2-messages.h, [19](#)
- msgList
 - G-2301-04-P2-messages.h, [19](#)
- msgMOTDEnd
 - G-2301-04-P2-messages.h, [20](#)
- msgMOTDStart
 - G-2301-04-P2-messages.h, [21](#)
- msgMOTD
 - G-2301-04-P2-messages.h, [20](#)
- msgMode
 - G-2301-04-P2-messages.h, [19](#)
- msgModeChanged
 - G-2301-04-P2-messages.h, [20](#)
- msgMyInfo
 - G-2301-04-P2-messages.h, [21](#)
- msgNames
 - G-2301-04-P2-messages.h, [21](#)
- msgNickChanged
 - G-2301-04-P2-messages.h, [22](#)
- msgNoNickOrChannel
 - G-2301-04-P2-messages.h, [22](#)
- msgNotOperator
 - G-2301-04-P2-messages.h, [22](#)
- msgNotice
 - G-2301-04-P2-messages.h, [22](#)
- msgPart
 - G-2301-04-P2-messages.h, [23](#)
- msgPing
 - G-2301-04-P2-messages.h, [23](#)
- msgPrivmsg
 - G-2301-04-P2-messages.h, [23](#)
- msgQuit
 - G-2301-04-P2-messages.h, [24](#)
- msgTopicChanged
 - G-2301-04-P2-messages.h, [24](#)
- msgWelcome
 - G-2301-04-P2-messages.h, [25](#)
- msgWho
 - G-2301-04-P2-messages.h, [25](#)
- msgWhoisChannels
 - G-2301-04-P2-messages.h, [25](#)
- msgWhoisEnd
 - G-2301-04-P2-messages.h, [25](#)
- msgWhoisIdle
 - G-2301-04-P2-messages.h, [26](#)
- msgWhoisMode
 - G-2301-04-P2-messages.h, [26](#)
- msgWhoisServer
 - G-2301-04-P2-messages.h, [26](#)
- msgWhoisUser
 - G-2301-04-P2-messages.h, [26](#)

- G-2301-04-P2-messages.h, [27](#)
- msgYourHost
 - G-2301-04-P2-messages.h, [27](#)
- msgRplAway
 - G-2301-04-P2-messages.h, [24](#)
- openSocket
 - G-2301-04-P1-socket.h, [11](#)
- path
 - fileReceiver_args, [7](#)
- port
 - audioArgs, [6](#)
- receiveData
 - G-2301-04-P1-socket.h, [11](#)
- recibir_datos_SSL
 - G-2301-04-P3-ssl.h, [36](#)
- sck
 - _, [5](#)
 - fileSender_args, [7](#)
- sckF
 - fileReceiver_args, [7](#)
- sendData
 - G-2301-04-P1-socket.h, [12](#)
- ssl
 - _, [5](#)
- userCommandAway
 - G-2301-04-P2-userCommands.h, [28](#)
- userCommandBack
 - G-2301-04-P2-userCommands.h, [29](#)
- userCommandDefault
 - G-2301-04-P2-userCommands.h, [29](#)
- userCommandJoin
 - G-2301-04-P2-userCommands.h, [29](#)
- userCommandKick
 - G-2301-04-P2-userCommands.h, [29](#)
- userCommandList
 - G-2301-04-P2-userCommands.h, [30](#)
- userCommandMode
 - G-2301-04-P2-userCommands.h, [30](#)
- userCommandNames
 - G-2301-04-P2-userCommands.h, [30](#)
- userCommandNick
 - G-2301-04-P2-userCommands.h, [31](#)
- userCommandPart
 - G-2301-04-P2-userCommands.h, [31](#)
- userCommandPrivmsg
 - G-2301-04-P2-userCommands.h, [31](#)
- userCommandQuit
 - G-2301-04-P2-userCommands.h, [32](#)
- userCommandTopic
 - G-2301-04-P2-userCommands.h, [32](#)
- userCommandWhois
 - G-2301-04-P2-userCommands.h, [32](#)