Servidor y cliente IRC

Generated by Doxygen 1.8.11

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

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 nuesta aplicaión.

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 especifico. 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

sck	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

sck	descriptor del socket
port	puerto con el protocolo de nivel de aplicacion
max_clients	numero maximo de clientes esperando a ser atendidos
p	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. Específica para clientes.

Parameters

sck	identificador de fichero con el socket
host_name	url o direccion IP del servidor
port	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

p	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

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

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

sender	nick del cliente que envia audio
msg	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

args	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

args	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

nick	nick del emisor
msg	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

args	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

fs	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

m_in	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

m_in	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↩ _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↩ _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↩ _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↩ _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

m_in	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

m_in	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

m_in	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

m_in	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

m_in	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

m_in	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_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_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_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_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

m_in	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

m_in	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

m_in	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

m_in	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

m_in	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

m_in	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

m_in	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

m_in	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

m_in	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_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_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_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

command	commando de usuario que se va a enviar
ssl	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

command	commando de usuario que se va a enviar
ssl	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

command	commando de usuario que se va a enviar
ssl	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

command	commando de usuario que se va a enviar
ssl	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

command	commando de usuario que se va a enviar
ssl	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

command	commando de usuario que se va a enviar
ssl	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

command	commando de usuario que se va a enviar
ssl	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

command	commando de usuario que se va a enviar
ssl	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

command	commando de usuario que se va a enviar
ssl	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

command	commando de usuario que se va a enviar
ssl	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

command	commando de usuario que se va a enviar
ssl	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

command	commando de usuario que se va a enviar
ssl	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

command	commando de usuario que se va a enviar
ssl	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

command	commando de usuario que se va a enviar
ssl	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 apli- cación 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

ctx	contexto de la aplicacion
sck	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

ssl	puntero a la estructura de conexión ssl
ctx	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

ctx	contexto de la aplicacion
sck	descriptor del socket

Returns

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

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

Envia datos a traves de la conexion ssl preestablecida.

Parameters

ssl	puntero a la estructura de conexión ssl
buffer	cadena con los datos enviados
nbytes	numero de bytes que se envian

Returns

-1 en caso de error
0 correcto

4.8.2.5 int evaluar_post_connectar_SSL (const SSL * *ssl*)

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

Parameters

ssl	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

ca_cert	nombre del certificado de la CA
clserv_cert	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 apli- cación pueda usar la capa segura SSL.

Parameters

ca_cert	nombre del certificado de la CA
clserv_pem	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 traves de la conexion ssl preestablecida.

Parameters

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

Returns

<=0 en caso de error
0 correcto

Index