IRC Server

Generato da Doxygen 1.8.13

Indice

Capitolo 1

IRC server

IRC Server è un progetto sviluppato per l'esame di laboratorio di ingegneria informatica dell'università di Padova.

1.1 Cosa implmenta e cosa non implementa il server

Il server è un'implementazione molto semplificata del protocollo definitio dalla RFC1459 (https://tools.ietf.org/html/rfc1459) in particolare implementa i comandi

- NICK
- JOIN
- MODE
- WHO
- WHOIS
- PING
- PRIVMSG
- PART
- QUIT
- LIST

Il server non gestisce a differenza di quanto dichiarato nel RFC1459 (https://tools.ietf.org/html/rfc1459) le proprietà dei canali e degli utenti, non gestisce gli operatori dei canali quindi un utente non si può chiamare il comando KICK su nessun utente e non è stato implementato l'invio di file tra utenti

1.2 Come installare il server

Una volta scaricati i sorgenti del progetto eseguire i seguenti comandi

- cmake .
- make

Il primo serve a creare i file make per la compilazione, il secondo comando compila il progetto creando gli eseguibili nella cartella bin/ il progetto include anche due programmi di test per le due liste, Channel_list e User_list i sorgenti sono disponibili in test/

I test file sono stati creati con il framework Check (https://libcheck.github.io/check/)

Capitolo 2

Indice delle strutture dati

2.1 Strutture dati

Queste sono le strutture dati con una loro breve descrizione:

Capitolo 3

Indice dei file

3.1 Elenco dei file

Questo è un elenco dei file documentati con una loro breve descrizione:

Capitolo 4

Documentazione delle classi

4.1 Riferimenti per la struct _Channel_list

Struttura per collezionare i canali nel server.

`#include <channel.h>`

Campi

- struct _Channel_list * prev
  puntatore alla cella precedente della lista
- struct _Channel_list * next
  puntatore alla cella successiva della lista
- Channel * payload
  il canale da memorizzare nella lista

4.1.1 Descrizione dettagliata

Struttura per collezionare i canali nel server.

Definizione alla linea 60 del file channel.h.

4.1.2 Documentazione dei campi

4.1.2.1 next

`struct _Channel_list* _Channel_list::next`

puntatore alla cella successiva della lista

Definizione alla linea 62 del file channel.h.

4.1.2.2 payload

```
Channel* _Channel_list::payload
```

il canale da memorizzare nella lista

Definizione alla linea 63 del file channel.h.

4.1.2.3 prev

```
struct _Channel_list* _Channel_list::prev
```

puntatore alla cella precedente della lista

Definizione alla linea 61 del file channel.h.

La documentazione per questa struct è stata generata a partire dal seguente file:

- channel.h

4.2 Riferimenti per la struct _User_list

Lista concatenata per gestire gli utenti.

```
#include <user.h>
```

Campi

- struct _User_list * next
  puntatore alla cella successiva della lista
- struct _User_list * prev
  puntatore alla cella precedente della lista
- User * payload
  l'utente da memorizzare nella lista

4.2.1 Descrizione dettagliata

Lista concatenata per gestire gli utenti.

Implementazione di una lista doppiamente concatenata per gestire gli utenti connessi al server

Definizione alla linea 64 del file user.h.

4.2.2 Documentazione dei campi

4.2.2.1 next

```
struct _User_list* _User_list::next
```

puntatore alla cella successiva della lista

Definizione alla linea 65 del file user.h.

4.2.2.2 payload

```
User* _User_list::payload
```

l'utente da memorizzare nella lista

Definizione alla linea 67 del file user.h.

4.2.2.3 prev

```
struct _User_list* _User_list::prev
```

puntatore alla cella precedente della lista

Definizione alla linea 66 del file user.h.

La documentazione per questa struct è stata generata a partire dal seguente file:

- user.h

4.3 Riferimenti per la struct Channel

Struttura per gestire i cananli.

```
#include <channel.h>
```

Campi

- string name
  il nome del canale
- string topic
  il topic del canale, non ancora utilizzato, per sviluppi futuri
- int number_users
  conta il numero di utenti presenti nel canale
- User_list * users
  la lista degli utenti presenti nel canale

4.3.1 Descrizione dettagliata

Struttura per gestire i cananli.

Definizione alla linea 27 del file channel.h.

4.3.2 Documentazione dei campi

4.3.2.1 name

```
string Channel::name
```

il nome del canale

Definizione alla linea 28 del file channel.h.

4.3.2.2 number_users

```
int Channel::number_users
```

conta il numero di utenti presenti nel canale

Definizione alla linea 30 del file channel.h.

4.3.2.3 topic

```
string Channel::topic
```

il topic del canale, non ancora utilizzato, per sviluppi futuri

Definizione alla linea 29 del file channel.h.

4.3.2.4 users

```
User_list* Channel::users
```

la lista degli utenti presenti nel canale

Definizione alla linea 31 del file channel.h.

La documentazione per questa struct è stata generata a partire dal seguente file:

- channel.h

4.4 Riferimenti per la struct User

Struttura per gestire gli utenti.

`#include <user.h>`

Campi

- string name
  nome utente
- string hostname
  l'hostname da cui l'utente si collega
- string channels [15]
  la lista dei nomi dei canali a cui l'utente è connesso
- int id
  l'identificativo univoco
- int socket
  il socket dell'utente
- pthread_mutex_t socket_mutex
  semaforo per il socket
- pthread_t thread
  il thread associato all'utente

4.4.1 Descrizione dettagliata

Struttura per gestire gli utenti.

L'utente è un'entità con nome e identificativo univoci, un indicatore per il canale corrente (curr_channel) e il socket da cui l'utente si connette

Definizione alla linea 30 del file user.h.

4.4.2 Documentazione dei campi

4.4.2.1 channels

`string User::channels[15]`

la lista dei nomi dei canali a cui l'utente è connesso

Definizione alla linea 33 del file user.h.

4.4.2.2 hostname

```
string User::hostname
```

l'hostname da cui l'utente si collega

Definizione alla linea 32 del file user.h.

4.4.2.3 id

```
int User::id
```

l'identificativo univoco

Definizione alla linea 34 del file user.h.

4.4.2.4 name

```
string User::name
```

nome utente

Definizione alla linea 31 del file user.h.

4.4.2.5 socket

```
int User::socket
```

il socket dell'utente

Definizione alla linea 35 del file user.h.

4.4.2.6 socket_mutex

```
pthread_mutex_t User::socket_mutex
```

semaforo per il socket

Definizione alla linea 36 del file user.h.

4.4.2.7 thread

```
pthread_t User::thread
```

il thread associato all'utente

Definizione alla linea 37 del file user.h.

La documentazione per questa struct è stata generata a partire dal seguente file:

- user.h

Capitolo 5

Documentazione dei file

5.1 Riferimenti per il file channel.h

File per gestire i cananli del server.

```
#include "user.h"
#include <stdlib.h>
```

Strutture dati

- struct Channel

  Struttura per gestire i cananli.
- struct _Channel_list

  Struttura per collezionare i canali nel server.

Definizioni

- #define **__STRING__**
- #define **CHANNEL**(OBJ) ((Channel∗)(OGJ))
- #define **CHANNEL_LIST**(OBJ) ((Channel_list∗)(OBJ))
- #define **MAXLINE** 4096

Ridefinizioni di tipo (typedef)

- typedef char ∗ **string**
- typedef struct _Channel_list Channel_list

  Struttura per collezionare i canali nel server.

Funzioni

- Channel * create_channel (string name, string topic)
  Crea un canale dato un nome ed un argomento.
- void change_topic (Channel *c, string topic)
  Cambia il topic.
- int add_user_to_channel (Channel **c, User *u)
  Aggiunge un utente al canale.
- int add_channel (Channel_list **list, Channel *c)
  Aggiunge un canale alla lista.
- Channel * remove_channel (Channel_list **list, string name)
  Rimuove un canale dalla lista.
- Channel * find_channel (Channel_list *list, string name)
  Trova un canale nella lista canali.
- void print_list_channel (Channel_list *list)
  Stampa la lista dei canali.
- void print_users_in_channel (Channel *c)
  Stampa la lista utenti per il canale selezionato.
- void print_list_channel_and_users (Channel_list *list)
  Stampa la lista degli utenti in ogni canale.

5.1.1 Descrizione dettagliata

File per gestire i cananli del server.

Autore

Edoardo Vanin

Data

17 Giugno 2017

5.1.2 Documentazione delle ridefinizioni di tipo (typedef)

5.1.2.1 Channel_list

```
typedef struct _Channel_list Channel_list
```

Struttura per collezionare i canali nel server.

5.1.3 Documentazione delle funzioni

5.1.3.1 add_channel()

```
int add_channel (
            Channel_list ** list,
            Channel * c )
```

Aggiunge un canale alla lista.

Parametri

il	puntatore alla lista di canali da aggiornare
il	canale da aggiungere

Restituisce

-1 se il canale NON è stato inserito correttamente nella lista
diverso da -1 se il canale è stato inserito correttamente

Definizione alla linea 39 del file channel.c.

5.1.3.2 add_user_to_channel()

```
int add_user_to_channel (
            Channel ** c,
            User * u )
```

Aggiunge un utente al canale.

Parametri

c	il canale a cui aggiungere l'utente
u	l'utente da aggiungere al canale

Restituisce

-1 se l'utente NON è stato inserito correttamente nella lista
diverso da -1 se l'utente è stato inserito correttamente

Definizione alla linea 33 del file channel.c.

5.1.3.3 change_topic()

```
void change_topic (
            Channel * c,
            string topic )
```

Cambia il topic.

Parametri

topic	l'argomento da cambiare

Definizione alla linea 20 del file channel.c.

5.1.3.4 create_channel()

```
Channel* create_channel (
            string name,
            string topic )
```

Crea un canale dato un nome ed un argomento.

Parametri

name	il nome del canale
topic	l'argomento del canale, può eventualmente essere nullo non influisce sulla creazione effettiva del canale

Restituisce

il puntatore all'area occupata dal canale appena creato

Definizione alla linea 5 del file channel.c.

5.1.3.5 find_channel()

```
Channel* find_channel (
            Channel_list * list,
            string name )
```

Trova un canale nella lista canali.

Parametri

list	puntatore alla lista di canali
name	il nome del canale

Restituisce

il puntatore al canale se trovato altrimenti ritorna puntatore nullo

a differenza della funzione remove_channel non elimina la referenza nella lista di canali

Definizione alla linea 65 del file channel.c.

5.1.3.6 print_list_channel()

```
void print_list_channel (
            Channel_list * list )
```

Stampa la lista dei canali.

Parametri

list	la lista da stampare

Definizione alla linea 116 del file channel.c.

5.1.3.7 print_list_channel_and_users()

```
void print_list_channel_and_users (
            Channel_list * list )
```

Stampa la lista degli utenti in ogni canale.

Parametri

lista	la lista che si vuole stampare

Definizione alla linea 133 del file channel.c.

5.1.3.8 print_users_in_channel()

```
void print_users_in_channel (
            Channel * c )
```

Stampa la lista utenti per il canale selezionato.

Parametri

c	il canale per di cui si vuole stampare la lista utenti

Definizione alla linea 125 del file channel.c.

5.1.3.9 remove_channel()

```
Channel* remove_channel (
            Channel_list ** list,
            string name )
```

Rimuove un canale dalla lista.

Parametri

list	puntatore doppio alla lista di canali, la lista viene modificata
name	il nome del canale

Restituisce

il puntatore al canale se trovato, altrimenti puntatore nullo

La funzione cerca il canale e se lo trova restituire un puntatore alla zona di memora contenente l'oggetto e contemporaneamente lo rimuove dalla lista ma NON libera l'area di memoria occupata

Definizione alla linea 82 del file channel.c.

5.2 Riferimenti per il file commands.h

Definisco i comandi utenti permessi come stringhe, i comandi sono fanno riferimento alla documentazione ufficiale (RFC)

Variabili

- char * **DCC**
- char * **WHO**
- char * **NICK**
- char * **USER**
- char * **LIST**
- char * **JOIN**
- char * **PART**
- char * **MODE**
- char * **PING**
- char * **PONG**
- char * **QUIT**
- char * **WHOIS**
- char * **NOTICE**
- char * **PRIVMSG**
- char * **MACDATA**
- char * **MACHORA**
- char * **MACTEMPERATURA**
- char * **TOPIC**

5.2.1 Descrizione dettagliata

Definisco i comandi utenti permessi come stringhe, i comandi sono fanno riferimento alla documentazione ufficiale (RFC)

Autore

Edoardo Vanin

Data

23 Giugno 2017

5.3 Riferimenti per il file errors.h

Definisco gli errori possibili, fanno riferimento alla documentazione ufficiale (RFC1459)

Variabili

- char * **ERR_NOSUCHNICK**
- char * **ERR_NOSUCHSERVER**
- char * **ERR_NOSUCHCHANNEL**
- char * **ERR_CANNOTSENDTOCHAN**
- char * **ERR_TOOMANYCHANNELS**
- char * **ERR_WASNOSUCHNICK**
- char * **ERR_TOOMANYTARGETS**
- char * **ERR_NORECIPIENT**
- char * **ERR_NOTEXTTOSEND**
- char * **ERR_NOTOPLEVEL**
- char * **ERR_WILDTOPLEVEL**
- char * **ERR_UNKNOWNCOMMAND**
- char * **ERR_NOMOTD**
- char * **ERR_NOADMININFO**
- char * **ERR_FILEERROR**
- char * **ERR_NONICKNAMEGIVEN**
- char * **ERR_ERRONEUSNICKNAME**
- char * **ERR_NICKNAMEINUSE**
- char * **ERR_NICKCOLLISION**
- char * **ERR_USERNOTINCHANNEL**
- char * **ERR_NOTONCHANNEL**
- char * **ERR_USERONCHANNEL**
- char * **ERR_NOLOGIN**
- char * **ERR_SUMMONDISABLED**
- char * **ERR_USERSDISABLED**
- char * **ERR_NOTREGISTERED**
- char * **ERR_NEEDMOREPARAMS**
- char * **ERR_ALREADYREGISTRED**
- char * **ERR_NOPERMFORHOST**
- char * **ERR_PASSWDMISMATCH**
- char * **ERR_YOUREBANNEDCREEP**
- char * **ERR_KEYSET**
- char * **ERR_CHANNELISFULL**
- char * **ERR_UNKNOWNMODE**
- char * **ERR_INVITEONLYCHAN**
- char * **ERR_BANNEDFROMCHAN**
- char * **ERR_BADCHANNELKEY**
- char * **ERR_NOPRIVILEGES**
- char * **ERR_CHANOPRIVSNEEDED**
- char * **ERR_CANTKILLSERVER**
- char * **ERR_NOOPERHOST**
- char * **ERR_UMODEUNKNOWNFLAG**
- char * **ERR_USERSDONTMATCH**

5.3.1 Descrizione dettagliata

Definisco gli errori possibili, fanno riferimento alla documentazione ufficiale (RFC1459)

Autore

Edoardo Vanin

Data

23 Giugno 2017

5.4 Riferimenti per il file main.c

```
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <errno.h>
#include <unistd.h>
#include "user.h"
#include "channel.h"
#include "utils.h"
#include "user_thread.h"
```

Funzioni

- int **main** (int argc, char const *argv[ ])

5.4.1 Descrizione dettagliata

Autore

Edoardo Vanin edoardo.vanin.1@studenti.unipd.it

Data

17 Giugno 2017

5.5 Riferimenti per il file recieve_commands.h

File contenente le funzioni da svolgere ogni volta che ricevo un comando.

```
#include <unistd.h>
#include "commands.h"
#include "user.h"
#include "channel.h"
#include "utils.h"
#include "text.h"
#include "errors.h"
#include "response.h"
```

Funzioni

- void recieve_nick (User *u, User_list *list, char *name)
  Cambia il nickname.
- void recieve_user (User *u)
  Risponde con un messaggio di benvenuto.
- void recieve_join (User *u, char *name)
  Entra in un canale.
- void recieve_mode (User *u, char *umode)
  Mode.
- void recieve_who (User *u, char *query)
  Who.
- void recieve_whois (User *u, char *query)
  Whois.
- void recieve_ping (User *u, char *ping_message)
  Ping.
- void recieve_privmsg (User *u, char *message)
  Privmsg.
- void recieve_part (User *u, char *parameter)
  Part.
- void recieve_quit (User *u, char *parameter)
  Quit.
- void recieve_list (User *u)
  List.

5.5.1 Descrizione dettagliata

File contenente le funzioni da svolgere ogni volta che ricevo un comando.

Autore

Edoardo Vanin

Data

23 Giugno 2017

5.5.2 Documentazione delle funzioni

5.5.2.1 recieve_join()

```
void recieve_join (
            User * u,
            char * name )
```

Entra in un canale.

Parametri

u	l'utente che ha chiamato il comando
name	il nome del canale in cui entrare

la funzione entra sempre in un canale, se non esiste lo crea, se esiste aggiunge l'utente alla lista utenti del canale

Definizione alla linea 71 del file recieve_commands.c.

5.5.2.2 recieve_list()

```
void recieve_list (
            User * u )
```

List.

Parametri

u	l'utente che ha chiamato il comando

ritorna la lista di canali presenti nel server

Definizione alla linea 487 del file recieve_commands.c.

5.5.2.3 recieve_mode()

```
void recieve_mode (
            User * u,
            char * umode )
```

Mode.

Parametri

u	l'utente che ha chiamato il comando

Definizione alla linea 193 del file recieve_commands.c.

5.5.2.4 recieve_nick()

```
void recieve_nick (
            User * u,
```

```
            User_list * list,
            char * name )
```

Cambia il nickname.

Parametri

u	l'utente che ha chiamato il comando
list	la lista utenti da cui andare a cercare u
name	il nuovo nickname

Definizione alla linea 3 del file recieve_commands.c.

5.5.2.5 recieve_part()

```
void recieve_part (
            User * u,
            char * parameter )
```

Part.

Parametri

u	l'utente che ha chiamato il comando
parameter	la stringa che contiene il canale da abbandonare e il messaggio di leave

la funzione non gestisce, a differenza di quanto indicato dalla RFC1459 l'abbandono multiplo

Definizione alla linea 382 del file recieve_commands.c.

5.5.2.6 recieve_ping()

```
void recieve_ping (
            User * u,
            char * ping_message )
```

Ping.

Parametri

u	l'utente che ha chiamato il comando
ping_message	l'identificativo del cliente che ha chiamato il comando Ping

a differenza della specifica nel RFC1459 non supporta la gestione di server multipli o chiamata di ping verso utenti del sistema

Definizione alla linea 335 del file recieve_commands.c.

5.5.2.7 recieve_privmsg()

```
void recieve_privmsg (
            User * u,
            char * message )
```

Privmsg.

Parametri

u	l'utente che ha chiamato il comando
message	il messaggio da inviare al canale

la funzione non gestisce lo scambio di messaggi tra utenti ma sono nei canali

Definizione alla linea 345 del file recieve_commands.c.

5.5.2.8 recieve_quit()

```
void recieve_quit (
            User * u,
            char * parameter )
```

Quit.

Parametri

u	l'utente che ha chiamato il comando
parameter	il leave message

comando quit, disconnnette l'utente dal server e rimuove la sua presenza dalle liste utente e canale

Definizione alla linea 447 del file recieve_commands.c.

5.5.2.9 recieve_user()

```
void recieve_user (
            User * u )
```

Risponde con un messaggio di benvenuto.

Parametri

u	l'utente che ha chiamato il comando

Definizione alla linea 57 del file recieve_commands.c.

5.5.2.10 recieve_who()

```
void recieve_who (
            User * u,
            char * query )
```

Who.

Parametri

u	l'utente che ha chiamato il comando
query	la sintassi per il comando WHO

si possono distinguere tre casi in cui agisce il comando WHO il primo senza query ed invia al chiamante tutte le informazioni di tutti gli utenti che non sono negli stessi canali di u il secondo con parametro un canale ed invia ad u le informazioni di tutt gli utenti del canale (u escluso) infine il terzo caso in cui la funzione WHO viene chiamata su un utente e invia ad u nickname e hostname dell'utente cercato

Definizione alla linea 212 del file recieve_commands.c.

5.5.2.11 recieve_whois()

```
void recieve_whois (
            User * u,
            char * query )
```

Whois.

Parametri

u	l'utente che ha chiamato il comando
query	l'utente su vui si vuole chiamare il WHOIS

invia le stesse informazioni di WHO username solo con la formattazione di una chiamata WHOIS

Definizione alla linea 269 del file recieve_commands.c.

5.6 Riferimenti per il file response.h

File contenente le dichiarazioni di tutte le risposte possibili.

Variabili

- char * **RPL_NONE**
- char * **RPL_USERHOST**
- char * **RPL_ISON**
- char * **RPL_AWAY**
- char * **RPL_UNAWAY**
- char * **RPL_NOWAWAY**
- char * **RPL_WHOISUSER**
- char * **RPL_WHOISSERVER**
- char * **RPL_WHOISOPERATOR**
- char * **RPL_WHOISIDLE**
- char * **RPL_ENDOFWHOIS**
- char * **RPL_WHOISCHANNELS**
- char * **RPL_WHOWASUSER**
- char * **RPL_ENDOFWHOWAS**
- char * **RPL_LISTSTART**
- char * **RPL_LIST**
- char * **RPL_LISTEND**
- char * **RPL_CHANNELMODEIS**
- char * **RPL_NOTOPIC**
- char * **RPL_TOPIC**
- char * **RPL_INVITING**
- char * **RPL_SUMMONING**
- char * **RPL_VERSION**
- char * **RPL_WHOREPLY**
- char * **RPL_ENDOFWHO**
- char * **RPL_NAMREPLY**
- char * **RPL_ENDOFNAMES**
- char * **RPL_LINKS**
- char * **RPL_ENDOFLINKS**
- char * **RPL_BANLIST**
- char * **RPL_ENDOFBANLIST**
- char * **RPL_INFO**
- char * **RPL_ENDOFINFO**
- char * **RPL_MOTDSTART**
- char * **RPL_MOTD**
- char * **RPL_ENDOFMOTD**
- char * **RPL_YOUREOPER**
- char * **RPL_REHASHING**
- char * **RPL_TIME**
- char * **RPL_USERSSTART**
- char * **RPL_USERS**
- char * **RPL_ENDOFUSERS**
- char * **RPL_NOUSERS**
- char * **RPL_TRACELINK**
- char * **RPL_TRACECONNECTING**
- char * **RPL_TRACEHANDSHAKE**
- char * **RPL_TRACEUNKNOWN**
- char * **RPL_TRACEOPERATOR**
- char * **RPL_TRACEUSER**
- char * **RPL_TRACESERVER**
- char * **RPL_TRACENEWTYPE**
- char * **RPL_TRACELOG**
- char * **RPL_STATSLINKINFO**

- char * **RPL_STATSCOMMANDS**
- char * **RPL_STATSCLINE**
- char * **RPL_STATSNLINE**
- char * **RPL_STATSILINE**
- char * **RPL_STATSKLINE**
- char * **RPL_STATSYLINE**
- char * **RPL_ENDOFSTATS**
- char * **RPL_STATSLLINE**
- char * **RPL_STATSUPTIME**
- char * **RPL_STATSOLINE**
- char * **RPL_STATSHLINE**
- char * **RPL_UMODEIS**
- char * **RPL_LUSERCLIENT**
- char * **RPL_LUSEROP**
- char * **RPL_LUSERUNKNOWN**
- char * **RPL_LUSERCHANNELS**
- char * **RPL_LUSERME**
- char * **RPL_ADMINME**
- char * **RPL_ADMINLOC1**
- char * **RPL_ADMINLOC2**
- char * **RPL_ADMINEMAIL**

5.6.1 Descrizione dettagliata

File contenente le dichiarazioni di tutte le risposte possibili.

Autore

Edoardo Vanin

Data

24 Giugno 2017

5.7 Riferimenti per il file text.h

File contenente le dichiarazioni delle variabili contenenti i testi predefiniti.

Variabili

- char * **WELCOME_01**
- char * **WELCOME_02**
- char * **WELCOME_03**
- char * **WELCOME_USER_01**
- char * **WELCOME_USER_02**
- char * **SERVER_NAME**
- char * **SERVER_INFO**
- char * **DEFAULT_PONG**
- char * **ENDOFWHOIS**
- char * **ENDOFWHO**
- char * **ENDOFNAMES**
- char * **LIST_END**
- char * **LIST_HEADER**
- char * **DCC_START**
- char * **DCC_END**
- char * **NONICKNAMEGIVEN**

5.7.1 Descrizione dettagliata

File contenente le dichiarazioni delle variabili contenenti i testi predefiniti.

Autore

Edoardo Vanin

Data

24 Giugno 2017

5.8 Riferimenti per il file user.h

File per gestire gli utenti collegati al server.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
```

Strutture dati

- struct User

  Struttura per gestire gli utenti.
- struct _User_list

  Lista concatenata per gestire gli utenti.

Definizioni

- #define **__STRING__**
- #define **USER**(OBJ) ((User∗)(OBJ))
- #define **USER_LIST**(OBJ) ((User_list∗)(OBJ))

Ridefinizioni di tipo (typedef)

- typedef char ∗ **string**
- typedef struct _User_list User_list

  Lista concatenata per gestire gli utenti.

Funzioni

- User * create_user (string name, string hostname, int id, int socket)
  Inizializza un oggetto utente.
- int change_name (User **u, string name)
  Cambia il nome all'utente.
- int add_user (User_list **list, User *u)
  Aggiunge un utente alla lista.
- User * remove_user_by_id (User_list **list, int id)
  Rimuove un utente dalla lista dato l'identificatore.
- User * remove_user_by_username (User_list **list, string username)
  Rimuove un utente dalla lista dato lo username.
- User * find_by_id (User_list *list, int id)
  Trova un utende usando il suo identificativo.
- User * find_by_username (User_list *list, string username)
  Trova un utende usando il suo identificativo.
- int print_list (User_list *list)
  Stampa la lista utenti.

5.8.1 Descrizione dettagliata

File per gestire gli utenti collegati al server.

Autore

Edoardo Vanin

Data

12 Giugno 2017

5.8.2 Documentazione delle ridefinizioni di tipo (typedef)

5.8.2.1 User_list

```
typedef struct _User_list User_list
```

Lista concatenata per gestire gli utenti.

Implementazione di una lista doppiamente concatenata per gestire gli utenti connessi al server

5.8.3 Documentazione delle funzioni

5.8.3.1 add_user()

```
int add_user (
            User_list ** list,
            User * u )
```

Aggiunge un utente alla lista.

Parametri

list	puntatore doppio alla lista di utenti, la lista viene modificata

Restituisce

-1 se l'utente NON è stato inserito correttamente nella lista
diverso da -1 se l'utente è stato inserito correttamente

Definizione alla linea 34 del file user.c.

5.8.3.2 change_name()

```
int change_name (
            User ** u,
            string name )
```

Cambia il nome all'utente.

Parametri

u	l'utente a cui si vuole cambiare il nome
name	il nuovo nome

Restituisce

1 se riesce a cambiare il nome utente
-1 se il parametro name è nullo

Definizione alla linea 23 del file user.c.

5.8.3.3 create_user()

```
User* create_user (
            string name,
            string hostname,
            int id,
            int socket )
```

Inizializza un oggetto utente.

Parametri

name	il nome che si vuole dare all'utente
hostname	l'hostname con cui l'utente è collegato
id	l'identificativo univoco dell'utente
socket	il socket da cui l'utente si connette

Restituisce

il puntatore all'area di memoria occupata dall'utente o se non riesce a crearlo NULL

Definizione alla linea 4 del file user.c.

5.8.3.4 find_by_id()

```
User* find_by_id (
            User_list * list,
            int id )
```

Trova un utende usando il suo identificativo.

Parametri

list	puntatore alla lista utenti
id	l'identificatore dell'utente da trovare

Restituisce

il puntatore all'utene se trovato altrimenti ritorna puntatore nullo

a differenza delle funzioni remove_user_* non elimina la referenza nella lista di utenti

Definizione alla linea 60 del file user.c.

5.8.3.5 find_by_username()

```
User* find_by_username (
            User_list * list,
            string username )
```

Trova un utende usando il suo identificativo.

Parametri

list	puntatore alla lista utenti
usrname	l'identificatore dell'utente da trovare

Restituisce

il puntatore all'utene se trovato altrimenti ritorna puntatore nullo

a differenza delle funzioni remove_user_* non elimina la referenza nella lista di utenti

Definizione alla linea 77 del file user.c.

5.8.3.6 print_list()

```
int print_list (
            User_list * list )
```

Stampa la lista utenti.

Parametri

list	la lista utenti da stampare

Definizione alla linea 142 del file user.c.

5.8.3.7 remove_user_by_id()

```
User* remove_user_by_id (
            User_list ** list,
            int id )
```

Rimuove un utente dalla lista dato l'identificatore.

Parametri

list	puntatore doppio alla lista di utenti, la lista viene modificata
id	identificatore univoco dell'utente

Restituisce

il puntatore all'utente se trovato, altrimenti puntatore nullo

La funzione cerca l'utente con id passato come parametro e se lo trova restituire un puntatore alla zona di memora contenente i l'oggetto e contemporaneamente lo rimuove dalla lista ma NON libera l'area di memoria occupata

Definizione alla linea 94 del file user.c.

5.8.3.8 remove_user_by_username()

```
User* remove_user_by_username (
            User_list ** list,
            string username )
```

Rimuove un utente dalla lista dato lo username.

Parametri

list	puntatore doppio alla lista di utenti, la lista viene modificata
username	identificatore univoco dell'utente

Restituisce

il puntatore all'utente se trovato, altrimenti puntatore nullo

La funzione cerca l'utente con username passato come parametro e se lo trova restituire un puntatore alla zona di memora contenente i l'oggetto e contemporaneamente lo rimuove dalla lista ma NON libera l'area di memoria occupata

Definizione alla linea 128 del file user.c.

5.9 Riferimenti per il file user_thread.h

File per gestire i comandi utente.

```
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <errno.h>
#include <unistd.h>
#include "user.h"
#include "channel.h"
#include "utils.h"
#include "commands.h"
#include "recieve_commands.h"
```

Funzioni

- void user_thread (User *u)

  Gestice i comandi arrivati dagli utenti connessi.

5.9.1 Descrizione dettagliata

File per gestire i comandi utente.

Autore

Edoardo Vanin

Data

20 Giugno 2017

5.9.2 Documentazione delle funzioni

5.9.2.1 user_thread()

```
void user_thread (
            User * u )
```

Gestice i comandi arrivati dagli utenti connessi.

Parametri

L'utente	da gestire

la funzione principe del programma (in realtà si tratta di uno switch sul comanda arrivato) ma deve essere gestito ad hoc

Definizione alla linea 3 del file user_thread.c.

5.10 Riferimenti per il file utils.h

File contenente funzioni e/o variabili globali.

```
#include "user.h"
#include "channel.h"
#include "errors.h"
#include "response.h"
#include "text.h"
#include <pthread.h>
#include <unistd.h>
```

Definizioni

- #define MAXUSER 150

  impone un numero massimo di utenti che si possono collegare al server
- #define **MAXLINE** 4096

Funzioni

- void send_user_info (User *u, User *target, char *channel)

  Funzione per gestire il caso "utente" del comando WHO.
- void send_all_user_info (User *u)

  Funzione per gestire il caso "tutti gli utenti" del comando WHO.
- void send_channel_info (User *u, Channel *c)

  Fuzione per gestire il caso "cananle" del comando WHO.

Variabili

- int count

  contatore per la lista utenti
- User_list * main_user_list

  lista globale per contenente tutti gli utenti del presenti nel server
- pthread_mutex_t main_user_list_mutex

  semaforo per la lista utenti, evita malfunzionamenti dalla programmazione concorrente
- Channel_list * main_channel_list

  lista globale per la gestione di tutti i canali
- pthread_mutex_t main_channel_list_mutex

  semaforo per la lista cananli, evita malfunzionamenti dalla programmazione concorrente

5.10.1 Descrizione dettagliata

File contenente funzioni e/o variabili globali.

Autore

Edoardo Vanin

Data

20 Giugno 2017

5.10.2 Documentazione delle definizioni

5.10.2.1 MAXUSER

```
#define MAXUSER 150
```

impone un numero massimo di utenti che si possono collegare al server

Definizione alla linea 20 del file utils.h.

5.10.3 Documentazione delle funzioni

5.10.3.1 send_all_user_info()

```
void send_all_user_info (
            User * u )
```

Funzione per gestire il caso "tutti gli utenti" del comando WHO.

Parametri

u	l'utente che ha chiamato la funzione

Definizione alla linea 40 del file utils.c.

5.10.3.2 send_channel_info()

```
void send_channel_info (
            User * u,
            Channel * c )
```

Fuzione per gestire il caso "cananle" del comando WHO.

Parametri

u	l'utente che ha chiamato la funzione
c	il canale di cui si vogliono le informazioni

Definizione alla linea 80 del file utils.c.

5.10.3.3 send_user_info()

```
void send_user_info (
            User * u,
            User * target,
            char * channel )
```

Funzione per gestire il caso "utente" del comando WHO.

Parametri

u	l'utente che ha chiamato la funzione WHO
target	l'obbiettivo su cui chiamare WHO
il	nome del cananle

la funzione è usate in due casi d'uso diversi, la presenza del parametro channel non NULLO facilita la selezione del caso

Definizione alla linea 3 del file utils.c.

5.10.4 Documentazione delle variabili

5.10.4.1 count

```
int count
```

contatore per la lista utenti

Definizione alla linea 24 del file utils.h.

5.10.4.2 main_channel_list

```
Channel_list* main_channel_list
```

lista globale per la gestione di tutti i canali

Definizione alla linea 31 del file utils.h.

5.10.4.3 main_channel_list_mutex

```
pthread_mutex_t main_channel_list_mutex
```

semaforo per la lista cananli, evita malfunzionamenti dalla programmazione concorrente

Definizione alla linea 33 del file utils.h.

5.10.4.4 main_user_list

```
User_list* main_user_list
```

lista globale per contenente tutti gli utenti del presenti nel server

Definizione alla linea 26 del file utils.h.

5.10.4.5 main_user_list_mutex

```
pthread_mutex_t main_user_list_mutex
```

semaforo per la lista utenti, evita malfunzionamenti dalla programmazione concorrente

Definizione alla linea 28 del file utils.h.

Indice analitico