IRC Server

Generato da Doxygen 1.8.13

# **Indice**

1	IRC	server		1
	1.1	Cosa i	mplmenta e cosa non implementa il server	1
	1.2	Come	installare il server	1
2	Indic	ce delle	strutture dati	3
	2.1	Struttu	re dati	3
3	Indic	ce dei fi	le	5
	3.1	Elenco	dei file	5
4	Doci	umenta	zione delle classi	7
	4.1	Riferim	enti per la struct _Channel_list	7
		4.1.1	Descrizione dettagliata	7
		4.1.2	Documentazione dei campi	7
			4.1.2.1 next	7
			4.1.2.2 payload	8
			4.1.2.3 prev	8
	4.2	Riferim	enti per la struct _User_list	8
		4.2.1	Descrizione dettagliata	8
		4.2.2	Documentazione dei campi	8
			4.2.2.1 next	9
			4.2.2.2 payload	9
			4.2.2.3 prev	9
	4.3	Riferim	enti per la struct Channel	9
		4.3.1	Descrizione dettagliata	10

ii INDICE

		4.3.2	Docume	ntazione dei campi	10
			4.3.2.1	name	10
			4.3.2.2	number_users	10
			4.3.2.3	topic	10
			4.3.2.4	users	10
	4.4	Riferim	nenti per la	struct User	11
		4.4.1	Descrizio	one dettagliata	11
		4.4.2	Docume	ntazione dei campi	11
			4.4.2.1	channels	11
			4.4.2.2	hostname	12
			4.4.2.3	$id \ \ldots \ldots \ldots \ldots \ldots$	12
			4.4.2.4	name	12
			4.4.2.5	socket	12
			4.4.2.6	socket_mutex	12
			4.4.2.7	thread	12
5	Doc	umenta	zione dei	file	13
5	<b>Doc</b> : 5.1			file file channel.h	<b>13</b>
5			nenti per il		
5		Riferim	nenti per il Descrizio	file channel.h	13
5		Riferim 5.1.1	nenti per il Descrizio	file channel.h	13 14
5		Riferim 5.1.1	Descrizion Docume 5.1.2.1	file channel.h  one dettagliata  ntazione delle ridefinizioni di tipo (typedef)	13 14 14
5		Riferim 5.1.1 5.1.2	Descrizion Docume 5.1.2.1	file channel.h  one dettagliata  ntazione delle ridefinizioni di tipo (typedef)  Channel_list	13 14 14 14
5		Riferim 5.1.1 5.1.2	Descrizion Docume 5.1.2.1 Docume	file channel.h  one dettagliata  ntazione delle ridefinizioni di tipo (typedef)  Channel_list  ntazione delle funzioni	13 14 14 14 14
5		Riferim 5.1.1 5.1.2	Descrizion Docume 5.1.2.1 Docume 5.1.3.1	file channel.h  one dettagliata  ntazione delle ridefinizioni di tipo (typedef)  Channel_list  ntazione delle funzioni  add_channel()	13 14 14 14 14
5		Riferim 5.1.1 5.1.2	Descrizion Docume 5.1.2.1 Docume 5.1.3.1 5.1.3.2	file channel.h  one dettagliata  ntazione delle ridefinizioni di tipo (typedef)  Channel_list  ntazione delle funzioni  add_channel()  add_user_to_channel()	13 14 14 14 14 14 15
5		Riferim 5.1.1 5.1.2	Descrizion Docume 5.1.2.1 Docume 5.1.3.1 5.1.3.2 5.1.3.3	file channel.h  one dettagliata  ntazione delle ridefinizioni di tipo (typedef)  Channel_list  ntazione delle funzioni  add_channel()  add_user_to_channel()  change_topic()	13 14 14 14 14 14 15 15
5		Riferim 5.1.1 5.1.2	Descrizion Docume 5.1.2.1 Docume 5.1.3.1 5.1.3.2 5.1.3.3 5.1.3.4	file channel.h  one dettagliata  ntazione delle ridefinizioni di tipo (typedef)  Channel_list  ntazione delle funzioni  add_channel()  add_user_to_channel()  change_topic()  create_channel()	13 14 14 14 14 15 15
5		Riferim 5.1.1 5.1.2	Descrizion Documer 5.1.2.1 Documer 5.1.3.1 5.1.3.2 5.1.3.3 5.1.3.4 5.1.3.5	file channel.h  one dettagliata  ntazione delle ridefinizioni di tipo (typedef)  Channel_list  ntazione delle funzioni  add_channel()  add_user_to_channel()  change_topic()  create_channel()  find_channel()	13 14 14 14 14 15 15 16 16
5		Riferim 5.1.1 5.1.2	Descrizion Docume 5.1.2.1 Docume 5.1.3.1 5.1.3.2 5.1.3.3 5.1.3.4 5.1.3.5 5.1.3.6	file channel.h  one dettagliata  ntazione delle ridefinizioni di tipo (typedef)  Channel_list  ntazione delle funzioni  add_channel()  add_user_to_channel()  change_topic()  create_channel()  find_channel()  print_list_channel()	13 14 14 14 14 15 15 16 16
5		Riferim 5.1.1 5.1.2	Descrizion Documer 5.1.2.1 Documer 5.1.3.1 5.1.3.2 5.1.3.3 5.1.3.4 5.1.3.5 5.1.3.6 5.1.3.7	file channel.h  one dettagliata  ntazione delle ridefinizioni di tipo (typedef)  Channel_list  ntazione delle funzioni  add_channel()  add_user_to_channel()  change_topic()  create_channel()  find_channel()  print_list_channel_and_users()	13 14 14 14 14 15 15 16 16 17

INDICE iii

5.2	Riferim	nenti per il 1	file commands.h	. 18
	5.2.1	Descrizio	one dettagliata	. 18
5.3	Riferim	nenti per il t	file errors.h	. 19
	5.3.1	Descrizio	one dettagliata	. 20
5.4	Riferim	nenti per il t	file main.c	. 20
	5.4.1	Descrizio	one dettagliata	. 20
5.5	Riferim	nenti per il t	file recieve_commands.h	. 20
	5.5.1	Descrizio	one dettagliata	. 21
	5.5.2	Documer	ntazione delle funzioni	. 21
		5.5.2.1	recieve_join()	. 21
		5.5.2.2	recieve_list()	. 22
		5.5.2.3	recieve_mode()	. 22
		5.5.2.4	recieve_nick()	. 22
		5.5.2.5	recieve_part()	. 24
		5.5.2.6	recieve_ping()	. 24
		5.5.2.7	recieve_privmsg()	. 25
		5.5.2.8	recieve_quit()	. 25
		5.5.2.9	recieve_user()	. 25
		5.5.2.10	recieve_who()	. 26
		5.5.2.11	recieve_whois()	. 26
5.6	Riferim	nenti per il t	file response.h	. 26
	5.6.1	Descrizio	one dettagliata	. 28
5.7	Riferim	nenti per il f	file text.h	. 28
	5.7.1	Descrizio	one dettagliata	. 29
5.8	Riferim	nenti per il t	file user.h	. 29
	5.8.1	Descrizio	one dettagliata	. 30
	5.8.2	Documer	ntazione delle ridefinizioni di tipo (typedef)	. 30
		5.8.2.1	User_list	. 30
	5.8.3	Documer	ntazione delle funzioni	. 30
		5.8.3.1	add_user()	. 30

iv INDICE

		5.8.3.2	change_name()	31
		5.8.3.3	create_user()	31
		5.8.3.4	find_by_id()	32
		5.8.3.5	find_by_username()	32
		5.8.3.6	print_list()	33
		5.8.3.7	remove_user_by_id()	33
		5.8.3.8	remove_user_by_username()	33
5.9	Riferim	enti per il i	file user_thread.h	34
	5.9.1	Descrizio	one dettagliata	34
	5.9.2	Documer	ntazione delle funzioni	34
		5.9.2.1	user_thread()	34
5.10	Riferim	enti per il i	file utils.h	35
	5.10.1	Descrizio	one dettagliata	36
	5.10.2	Documer	ntazione delle definizioni	36
		5.10.2.1	MAXUSER	36
	5.10.3	Documer	ntazione delle funzioni	36
		5.10.3.1	send_all_user_info()	36
		5.10.3.2	send_channel_info()	36
		5.10.3.3	send_user_info()	37
	5.10.4	Documer	ntazione delle variabili	37
		5.10.4.1	count	37
		5.10.4.2	main_channel_list	38
		5.10.4.3	main_channel_list_mutex	38
		5.10.4.4	main_user_list	38
		5.10.4.5	main_user_list_mutex	38

Indice

39

## 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/)

2 IRC server

# Capitolo 2

# Indice delle strutture dati

## 2.1 Strutture dati

Queste sono le strutture dati con una loro breve descrizione:

_Channe	_list	
	Struttura per collezionare i canali nel server	7
_User_lis	st end of the control	
	Lista concatenata per gestire gli utenti	8
Channel		
	Struttura per gestire i cananli	ç
User		
	Struttura per gestire gli utenti	11

Indice delle strutture dati

# Capitolo 3

# Indice dei file

## 3.1 Elenco dei file

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

channel.c	??
channel.h	
File per gestire i cananli del server	13
commands.c	??
commands.h	
Definisco i comandi utenti permessi come stringhe, i comandi sono fanno riferimento alla documentazione ufficiale (RFC)	18
errors.c	??
Definisco gli errori possibili, fanno riferimento alla documentazione ufficiale (RFC1459)	19
main.c	20
recieve_commands.c	??
File contenente le funzioni da svolgere ogni volta che ricevo un comando	20
response.c	??
response.h	
File contenente le dichiarazioni di tutte le risposte possibili	26
text.c	??
File contenente le dichiarazioni delle variabili contenenti i testi predefiniti	28
user.c	??
File per gestire gli utenti collegati al server	29
user_thread.c	??
File per gestire i comandi utente	34
utils.c	??
File contenente funzioni e/o variabili globali	35

6 Indice dei file

## 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()

Aggiunge un canale alla lista.

#### Parametri

	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()

Aggiunge un utente al canale.

#### Parametri

С	il canale a cui aggiungere l'utente
и	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()

Cambia il topic.

## Parametri

topic l'argomento da cambiare

Definizione alla linea 20 del file channel.c.

#### 5.1.3.4 create\_channel()

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()

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()

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()

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()

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()

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 \* PRIVMSGchar \* 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)
void recieve_who (User *u, char *query)
void recieve_whois (User *u, char *query)
     Whois.
• void recieve_ping (User *u, char *ping_message)
• void recieve_privmsg (User *u, char *message)
• void recieve_part (User *u, char *parameter)

    void recieve_quit (User *u, char *parameter)

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()

Entra in un canale.

## Parametri

и	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 ( {\tt 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()

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 ( {\tt User} \, * \, u ,
```

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

Cambia il nickname.

## Parametri

и	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()

## Part.

## Parametri

и	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()

## Ping.

## Parametri

и	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()

## Privmsg.

#### Parametri

и	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()

## Quit.

## Parametri

и	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()

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()

Who.

#### Parametri

и	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()

Whois.

#### Parametri

и	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()

Aggiunge un utente alla lista.

## Parametri

list puntatore doppio alla lista di utenti, la li	sta 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()

Cambia il nome all'utente.

#### Parametri

и	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()

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

Generato da Doxygen

#### 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()

Trova un utende usando il suo identificativo.

#### Parametri

list	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()

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()

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()

Rimuove un utente dalla lista dato l'identificatore.

#### Parametri

list puntatore doppio alla lista di utenti,		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()

Rimuove un utente dalla lista dato lo username.

#### **Parametri**

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

#### 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()

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()

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

#### Parametri

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

Definizione alla linea 80 del file utils.c.

## 5.10.3.3 send\_user\_info()

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

#### Parametri

u l'utente che ha chiamato la funzione WHC	
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**

_Channel_list, 7 next, 7 payload, 7 prev, 8 _User_list, 8 next, 8 payload, 9 prev, 9	user.h, 32 find_by_username user.h, 32 find_channel channel.h, 16 hostname User, 11
add channel	id
channel.h, 14	User, 12
add_user	MANUOED
user.h, 30	MAXUSER
add_user_to_channel	utils.h, 36
channel.h, 15	main.c, 20
ahanga nama	main_channel_list
change_name user.h, 31	utils.h, 37 main_channel_list_mutex
change topic	utils.h, 38
channel.h, 15	main_user_list
Channel, 9	utils.h, 38
name, 10	main_user_list_mutex
number_users, 10	utils.h, 38
topic, 10	, , , , ,
users, 10	name
channel.h, 13	Channel, 10
add_channel, 14	User, 12
add_user_to_channel, 15	next
change_topic, 15	_Channel_list, 7
Channel_list, 14	_User_list, 8
create_channel, 16	number_users
find_channel, 16	Channel, 10
print_list_channel, 16	payload
print_list_channel_and_users, 17	_Channel_list, 7
print_users_in_channel, 17	User list, 9
remove_channel, 17	prev
Channel_list	Channel list, 8
channel.h, 14	_User_list, 9
channels	print list
User, 11	user.h, 32
commands.h, 18	print_list_channel
count utils.h, 37	channel.h, 16
create_channel	print_list_channel_and_users
channel.h, 16	channel.h, 17
create_user	print_users_in_channel
user.h, 31	channel.h, 17
223, 0	
errors.h, 19	recieve_commands.h, 20
find by id	recieve_join, 21
find_by_id	recieve_list, 22

40 INDICE ANALITICO

recieve mode, 22	id, 12
_ ,	•
recieve_nick, 22	name, 12
recieve_part, 24	socket, 12
recieve_ping, 24	socket_mutex, 12
recieve_privmsg, 24	thread, 12
recieve_quit, 25	user.h, 29
recieve_user, 25	add_user, 30
recieve_who, 26	change_name, 31
recieve_whois, 26	create_user, 31
recieve_join	find_by_id, 32
recieve_commands.h, 21	find_by_username, 32
recieve_list	print_list, 32
recieve_commands.h, 22	remove_user_by_id, 33
recieve_mode	remove_user_by_username, 33
recieve_commands.h, 22	User_list, 30
recieve_nick	User_list
recieve commands.h, 22	user.h, 30
recieve part	user_thread
recieve_commands.h, 24	user_thread.h, 34
recieve ping	user_thread.h, 34
recieve_commands.h, 24	user_thread, 34
recieve_privmsg	users
recieve_commands.h, 24	Channel, 10
	utils.h, 35
recieve_quit	
recieve_commands.h, 25	count, 37
recieve_user	MAXUSER, 36
recieve_commands.h, 25	main_channel_list, 37
recieve_who	main_channel_list_mutex, 38
recieve_commands.h, 26	main_user_list, 38
recieve_whois	main_user_list_mutex, 38
recieve_commands.h, 26	send_all_user_info, 36
remove_channel	send_channel_info, 36
channel.h, 17	send_user_info, 37
remove_user_by_id	
user.h, 33	
remove_user_by_username	
user.h, 33	
response.h, 26	
•	
send_all_user_info	
utils.h, 36	
send_channel_info	
utils.h, 36	
send_user_info	
utils.h, 37	
socket	
User, 12	
socket_mutex	
User, 12	
text.h, 28	
thread	
User, 12	
Channel 10	
Channel, 10	
User, 11	
channels, 11	
hostname, 11	