chatterbox

2.0

Generato da Doxygen 1.8.14

Indice

1 Indice delle strutture dati

1.1 Strutture dati

Queste sono le strutture dati con una loro breve descrizione:

2 Indice dei file

2.1 Elenco dei file

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

3 Documentazione delle classi

3.1 Riferimenti per la struct _conf

Campi

- char * **unix_path**

- unsigned long **max_con**
- unsigned long **nthreads**
- unsigned long **max_msg**
- unsigned long **max_size**
- unsigned long **max_history**
- char * **down_path**
- char * **stats**

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

- parser.h

3.2 Riferimenti per la struct _group

Campi

- char **name** [MAX_NAME_LENGTH+1]
- pthread_mutex_t * **group_mtx**
- user_group * **group**
- UT_hash_handle **hh**

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

- user.h

3.3 Riferimenti per la struct _history

lista history utente

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

Campi

- message_t * **data**
- struct _history * **next**

3.3.1 Descrizione dettagliata

lista history utente

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

- user.h

3.4 Riferimenti per la struct _job

struttura contenente informazioni lavoro da svolgere

`#include <task.h>`

Campi

- unsigned long **fd**
- message_t * **msg**
- task_manager * **tsk**
- UT_hash_handle **hh**

3.4.1 Descrizione dettagliata

struttura contenente informazioni lavoro da svolgere

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

- task.h

3.5 Riferimenti per la struct _t_pool

Campi

- pthread_mutex_t **lock**
- pthread_cond_t **cond**
- pthread_t * **threads**
- task * **coda**
- int **nthreads**
- int **dim_coda**
- int **head**
- int **tail**
- int **count**
- int **shutdown**
- int **avviati**

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

- tpool.c

3.6 Riferimenti per la struct _task

Campi

- int(* **fun**)(job *)
- void * **arg**

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

- tpool.c

3.7 Riferimenti per la struct _task_manager

struttura per la gestione dei task

`#include <task.h>`

Campi

- pthread_mutex_t * **mtx_poll**
- pthread_mutex_t * **mtx_tm**
- int **epollfd**
- job * **jobs**
- userdata * **ud**

3.7.1 Descrizione dettagliata

struttura per la gestione dei task

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

- task.h

3.8 Riferimenti per la struct _user

struttura contenente informazioni utente

`#include <user.h>`

Campi

- char **nick** [MAX_NAME_LENGTH+1]
- history * **history**
- unsigned long **hc**
- unsigned long **fd**
- UT_hash_handle **hh**

3.8.1 Descrizione dettagliata

struttura contenente informazioni utente

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

- user.h

3.9 Riferimenti per la struct _user_group

struttura contenente informazioni utente di un gruppo

`#include <user.h>`

Campi

- char **nick** [MAX_NAME_LENGTH+1]
- user * **user**
- UT_hash_handle **hh**

3.9.1 Descrizione dettagliata

struttura contenente informazioni utente di un gruppo

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

- user.h

3.10 Riferimenti per la struct _user_list

lista utenti

`#include <user.h>`

Campi

- char **nick** [MAX_NAME_LENGTH+1]
- unsigned long **fd**
- struct _user_list * **next**

3.10.1 Descrizione dettagliata

lista utenti

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

- user.h

3.11 Riferimenti per la struct _usersdata

struttura contenente utenti registrati

`#include <user.h>`

Campi

- pthread_mutex_t * **mtx**
- user * **users**
- unsigned long **connessi**
- conf * **cf**
- group * **groups**
- struct statistics * **stats**
- pthread_mutex_t * **stats_mtx**
- pthread_mutex_t * **his_mtx**

3.11.1 Descrizione dettagliata

struttura contenente utenti registrati

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

- user.h

3.12 Riferimenti per la struct conf

configurazione server

```
#include <parser.h>
```

3.12.1 Descrizione dettagliata

configurazione server

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

- parser.h

3.13 Riferimenti per la struct data

body del messaggio

```
#include <message.h>
```

3.13.1 Descrizione dettagliata

body del messaggio

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

- message.h

3.14 Riferimenti per la struct header

header del messaggio

`#include <message.h>`

3.14.1 Descrizione dettagliata

header del messaggio

header della parte dati

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

- message.h

3.15 Riferimenti per la struct message_data_hdr_t

Campi

- char **receiver** [MAX_NAME_LENGTH+1]
- unsigned int **len**

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

- message.h

3.16 Riferimenti per la struct message_data_t

Campi

- message_data_hdr_t **hdr**
- char * **buf**

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

- message.h

3.17 Riferimenti per la struct message_hdr_t

Campi

- op_t **op**
- char **sender** [MAX_NAME_LENGTH+1]

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

- message.h

3.18 Riferimenti per la struct message_t

Campi

- message_hdr_t **hdr**
- message_data_t **data**

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

- message.h

3.19 Riferimenti per la struct messaggio

tipo del messaggio

`#include <message.h>`

3.19.1 Descrizione dettagliata

tipo del messaggio

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

- message.h

3.20 Riferimenti per la struct operation_t

Campi

- char * **sname**
- char * **rname**
- op_t **op**
- char * **msg**
- long **size**
- long **n**

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

- client.c

3.21 Riferimenti per la struct statistics

Campi

- unsigned long **nusers**
- unsigned long **nonline**
- unsigned long **ndelivered**
- unsigned long **nnotdelivered**
- unsigned long **nfiledelivered**
- unsigned long **nfilenotdelivered**
- unsigned long **nerrors**

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

- stats.h

3.22 Riferimenti per la struct UT_hash_bucket

Campi

- struct UT_hash_handle * **hh_head**
- unsigned **count**
- unsigned **expand_mult**

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

- uthash.h

3.23 Riferimenti per la struct UT_hash_handle

Campi

- struct UT_hash_table * **tbl**
- void * **prev**
- void * **next**
- struct UT_hash_handle * **hh_prev**
- struct UT_hash_handle * **hh_next**
- void * **key**
- unsigned **keylen**
- unsigned **hashv**

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

- uthash.h

3.24 Riferimenti per la struct UT_hash_table

Campi

- UT_hash_bucket * **buckets**
- unsigned **num_buckets**
- unsigned **log2_num_buckets**
- unsigned **num_items**
- struct UT_hash_handle * **tail**
- ptrdiff_t **hho**
- unsigned **ideal_chain_maxlen**
- unsigned **nonideal_items**
- unsigned **ineff_expands**
- unsigned **noexpand**
- uint32_t **signature**

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

- uthash.h

4 Documentazione dei file

4.1 Riferimenti per il file chatty.c

file principale del server chatterbox

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/epoll.h>
#include "parser.h"
#include "stats.h"
#include "connections.h"
#include "task.h"
#include "tpool.h"
#include "message.h"
#include "user.h"
```

Definizioni

- #define **_POSIX_C_SOURCE** 200809L
- #define **MAX_EVENTS** 60

Funzioni

- void **signals** ()
- void **stop_server** ()
- void **get_stats** ()
- void * **print_stats** (void *arg)
- int **main** (int argc, char *argv[])

Variabili

- int **stop** = 1
- char * **statspath**
- int **alert** = 0
- pthread_mutex_t * **stats_mtx**
- struct statistics **chattyStats** = { 0,0,0,0,0,0,0 }
- conf **cf** = {NULL, 0, 0, 0, 0, 0, NULL, NULL}

4.1.1 Descrizione dettagliata

file principale del server chatterbox

Autore

Valerio Besozzi 543685 Si dichiara che il contenuto di questo file e' in ogni sua parte opera originale dell'autore

4.2 Riferimenti per il file client.c

Semplice client di test.

```
#include <time.h>
#include <errno.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <connections.h>
#include <ops.h>
```

Strutture dati

- struct operation_t

Definizioni

- #define **_POSIX_C_SOURCE** 200809L

Funzioni

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

4.2.1 Descrizione dettagliata

Semplice client di test.

4.3 Riferimenti per il file config.h

File contenente alcune define con valori massimi utilizzabili.

Definizioni

- #define **MAX_NAME_LENGTH** 32

Ridefinizioni di tipo (typedef)

- typedef int **make_iso_compilers_happy**

4.3.1 Descrizione dettagliata

File contenente alcune define con valori massimi utilizzabili.

4.4 Riferimenti per il file listener.c

file utilizzato per testcase

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

Funzioni

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

4.4.1 Descrizione dettagliata

file utilizzato per testcase

Autore

Valerio Besozzi Si dichiara che il contenuto di questo file e' in ogni sua parte opera originale dell'autore

4.5 Riferimenti per il file message.h

file contenente definizioni base, strutture e funzioni per gestione messaggi

```
#include <stdlib.h>
#include <assert.h>
#include <string.h>
#include <config.h>
#include <ops.h>
```

Strutture dati

- struct message_hdr_t
- struct message_data_hdr_t
- struct message_data_t
- struct message_t

4.5.1 Descrizione dettagliata

file contenente definizioni base, strutture e funzioni per gestione messaggi

Autore

Valerio Besozzi 543685 Si dichiara che il contenuto di questo file e' in ogni sua parte opera originale dell'autore

4.6 Riferimenti per il file ops.h

Contiene i codici delle operazioni di richiesta e risposta.

Tipi enumerati (enum)

- enum op_t {
  REGISTER_OP = 0, CONNECT_OP = 1, POSTTXT_OP = 2, POSTTXTALL_OP = 3,
  POSTFILE_OP = 4, GETFILE_OP = 5, GETPREVMSGS_OP = 6, USRLIST_OP = 7,
  UNREGISTER_OP = 8, DISCONNECT_OP = 9, CREATEGROUP_OP = 10, ADDGROUP_OP = 11,
  DELGROUP_OP = 12, OP_OK = 20, **TXT_MESSAGE** = 21, **FILE_MESSAGE** = 22,
  OP_FAIL = 25, **OP_NICK_ALREADY** = 26, **OP_NICK_UNKNOWN** = 27, **OP_MSG_TOOLONG** = 28,
  OP_NO_SUCH_FILE = 29, **OP_END** = 100 }

4.6.1 Descrizione dettagliata

Contiene i codici delle operazioni di richiesta e risposta.

Autore

Valerio Besozzi 543685 Si dichiara che il contenuto di questo file e' in ogni sua parte opera originale dell'autore

4.6.2 Documentazione dei tipi enumerati

4.6.2.1 op_t

```
enum op_t
```

Valori del tipo enumerato

CONNECT_OP	richiesta di registrazione di un ninckname
POSTTXT_OP	richiesta di connessione di un client
POSTTXTALL_OP	richiesta di invio di un messaggio testuale ad un nickname o groupname
POSTFILE_OP	richiesta di invio di un messaggio testuale a tutti gli utenti
GETFILE_OP	richiesta di invio di un file ad un nickname o groupname
GETPREVMSGS_OP	richiesta di recupero di un file
USRLIST_OP	richiesta di recupero della history dei messaggi
UNREGISTER_OP	richiesta di avere la lista di tutti gli utenti attualmente connessi
DISCONNECT_OP	richiesta di deregistrazione di un nickname o groupname
CREATEGROUP_OP	richiesta di disconnessione
ADDGROUP_OP	richiesta di creazione di un gruppo
DELGROUP_OP	richiesta di aggiunta ad un gruppo
OP_OK	richiesta di rimozione da un gruppo

4.7 Riferimenti per il file parser.c

contiene implementazioni funzioni che permettono la lettura del file di configurazione

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

Funzioni

- int load_conf (char *path, conf *cf)
  legge il file di configurazione

4.7.1 Descrizione dettagliata

contiene implementazioni funzioni che permettono la lettura del file di configurazione

Autore

Valerio Besozzi 543685 Si dichiara che il contenuto di questo file e' in ogni sua parte opera originale dell'autore

4.7.2 Documentazione delle funzioni

4.7.2.1 load_conf()

```
int load_conf (
            char * path,
            conf * cf )
```

legge il file di configurazione

load_conf

Parametri

path	locazione file configurazione

4.8 Riferimenti per il file parser.h

contiene le funzioni che permettono la lettura del file di configurazione

Strutture dati

- struct _conf

Definizioni

- #define **MAXBUF** 1024
- #define **DIV** "= \t\r\n\v\f"
- #define **CHECK_TOK**(token)

Ridefinizioni di tipo (typedef)

- typedef struct _conf **conf**

Funzioni

- int load_conf (char *path, conf *cf)
  legge il file di configurazione

4.8.1 Descrizione dettagliata

contiene le funzioni che permettono la lettura del file di configurazione

Autore

Valerio Besozzi 543685 Si dichiara che il contenuto di questo file e' in ogni sua parte opera originale dell'autore

4.8.2 Documentazione delle definizioni

4.8.2.1 CHECK_TOK

```
#define CHECK_TOK(
                token )
```

Valore:

```
if(token == NULL) { \
      "); \
                                  } \
                                                    printf("Errore nel file di configurazione\n
                                                    return -2; \
```

4.8.3 Documentazione delle funzioni

4.8.3.1 load_conf()

```
int load_conf (
            char * path,
            conf * cf )
```

legge il file di configurazione

load_conf

Parametri

path	locazione file configurazione

4.9 Riferimenti per il file stats.h

file contenente funzioni di utilita' per le statistiche

```
#include <stdio.h>
#include <time.h>
```

Strutture dati

- struct statistics

4.9.1 Descrizione dettagliata

file contenente funzioni di utilita' per le statistiche

Autore

Valerio Besozzi 543685 Si dichiara che il contenuto di questo file e' in ogni sua parte opera originale dell'autore

4.10 Riferimenti per il file task.c

file contenente implementazioni funzioni per la gestione ed esecuzione dei task

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <signal.h>
#include <fcntl.h>
#include <pthread.h>
```

```
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/epoll.h>
#include <stats.h>
#include <message.h>
#include <connections.h>
#include <user.h>
#include <parser.h>
#include <task.h>
```

Funzioni

- task_manager * new_manager (int epollfd, userdata *ud)
  crea la struttura per la gestione dei task
- void del_task (task_manager *tsk)
  elimina la struttura per la gestione dei task
- job * new_job (int fd, task_manager *tsk)
  crea un nuovo job
- void del_job (job *jobd)
  elimina job
- int read_task (job *jobr)
  legge richiesta client
- int write_task (job *jobw)
  esegue richiesta client
- int exec_task (task_manager *tsk, message_t *s_msg, int fd, userdata *ud)
  esegue operazioni per soddisfare richiesta del client

4.10.1 Descrizione dettagliata

file contenente implementazioni funzioni per la gestione ed esecuzione dei task

Autore

Valerio Besozzi 543685 Si dichiara che il contenuto di questo file e' in ogni sua parte opera originale dell'autore

4.10.2 Documentazione delle funzioni

4.10.2.1 del_job()

```
void del_job (
            job * jobd )
```

elimina job

del_job

Parametri

jobd	job da eliminare

4.10.2.2 del_task()

```
void del_task (
            task_manager * tsk )
```

elimina la struttura per la gestione dei task

del_task

Parametri

tsk	puntatore alla struttura task_manager da eliminare

4.10.2.3 exec_task()

```
int exec_task (
            task_manager * tsk,
            message_t * s_msg,
            int fd,
            userdata * ud )
```

esegue operazioni per soddisfare richiesta del client

exec_task

Parametri

tsk	puntatore alla struttura _task_manager in uso
s_msg	messaggio contenente richiesta client
fd	fd del client che ha inviato la richiesta
ud	puntatore alla struttura _userdata in cui operare

Restituisce

0 se la richiesta da parte del client e' stata eseguita correttamente, -1 altrimenti

4.10.2.4 new_job()

```
job* new_job (
            int fd,
            task_manager * tsk )
```

crea un nuovo job

new_job

Parametri

fd	fd relativo al client oggetto del job
tsk	puntatore alla struttura task_manager in uso

Restituisce

puntatore alla struttura _job creata

4.10.2.5 new_manager()

```
task_manager* new_manager (
              int epollfd,
              userdata * ud )
```

crea la struttura per la gestione dei task

new_manager

Parametri

epollfd	fd relativo all'istanza epoll in uso
ud	puntatore alla struttura _userdata in uso

Restituisce

puntatore alla struttura task_manager creata

4.10.2.6 read_task()

```
int read_task (
              job * jobr )
```

legge richiesta client

read_task

Parametri

jobr	struttura _job contenente informazioni richiesta del client

Restituisce

0 se la richiesta da parte del client e' stata letta e registrata, -1 altrimenti

4.10.2.7 write_task()

```
int write_task (
            job * jobw )
```

esegue richiesta client

write_task

Parametri

jobr	struttura _job contenente informazioni richiesta del client

Restituisce

0 se la richiesta da parte del client e' stata eseguita correttamente, -1 altrimenti

4.11 Riferimenti per il file task.h

file contenente funzioni per la gestione ed esecuzione dei task

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/epoll.h>
#include <user.h>
```

Strutture dati

- struct _task_manager

  struttura per la gestione dei task
- struct _job

  struttura contenente informazioni lavoro da svolgere

Ridefinizioni di tipo (typedef)

- typedef struct _task_manager **task_manager**
- typedef struct _job **job**

Funzioni

- task_manager * new_manager (int epollfd, userdata *ud)
  crea la struttura per la gestione dei task
- void del_task (task_manager *tsk)
  elimina la struttura per la gestione dei task
- job * new_job (int fd, task_manager *tsk)
  crea un nuovo job
- void del_job (job *jobd)
  elimina job
- int read_task (job *jobr)
  legge richiesta client
- int write_task (job *jobw)
  esegue richiesta client
- int exec_task (task_manager *tsk, message_t *s_msg, int fd, userdata *ud)
  esegue operazioni per soddisfare richiesta del client

4.11.1 Descrizione dettagliata

file contenente funzioni per la gestione ed esecuzione dei task

Autore

Valerio Besozzi 543685 Si dichiara che il contenuto di questo file e' in ogni sua parte opera originale dell'autore

4.11.2 Documentazione delle funzioni

4.11.2.1 del_job()

```
void del_job (
            job * jobd )
```

elimina job

del_job

Parametri

jobd	job da eliminare

4.11.2.2 del_task()

```
void del_task (
            task_manager * tsk )
```

elimina la struttura per la gestione dei task

del_task

Parametri

tsk	puntatore alla struttura task_manager da eliminare

4.11.2.3 exec_task()

```
int exec_task (
            task_manager * tsk,
            message_t * s_msg,
            int fd,
            userdata * ud )
```

esegue operazioni per soddisfare richiesta del client

exec_task

Parametri

tsk	puntatore alla struttura _task_manager in uso
s_msg	messaggio contenente richiesta client
fd	fd del client che ha inviato la richiesta
ud	puntatore alla struttura _userdata in cui operare

Restituisce

0 se la richiesta da parte del client e' stata eseguita correttamente, -1 altrimenti

4.11.2.4 new_job()

```
job* new_job (
            int fd,
            task_manager * tsk )
```

crea un nuovo job

new_job

Parametri

fd	fd relativo al client oggetto del job
tsk	puntatore alla struttura task_manager in uso

Restituisce

puntatore alla struttura _job creata

4.11.2.5 new_manager()

```
task_manager* new_manager (
            int epollfd,
            userdata * ud )
```

crea la struttura per la gestione dei task

new_manager

Parametri

epollfd	fd relativo all'istanza epoll in uso
ud	puntatore alla struttura _userdata in uso

Restituisce

puntatore alla struttura task_manager creata

4.11.2.6 read_task()

```
int read_task (
            job * jobr )
```

legge richiesta client

read_task

Parametri

jobr	struttura _job contenente informazioni richiesta del client

Restituisce

0 se la richiesta da parte del client e' stata letta e registrata, -1 altrimenti

4.11.2.7 write_task()

```
int write_task (
            job * jobw )
```

esegue richiesta client

write_task

Parametri

jobr	struttura _job contenente informazioni richiesta del client

Restituisce

0 se la richiesta da parte del client e' stata eseguita correttamente, -1 altrimenti

4.12 Riferimenti per il file tpool.c

contiene implementazioni funzioni per la creazione, eliminazione e gestione del threadpool

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <errno.h>
#include <unistd.h>
#include <tpool.h>
```

Strutture dati

- struct _task
- struct _t_pool

Ridefinizioni di tipo (typedef)

- typedef struct _task **task**

Funzioni

- int **tpool_free** (t_pool *pool)
- t_pool * tpool_create (int nthreads, int dim_coda)
  crea threadpool
- int tpool_add (t_pool *pool, int(*fun)(job *), void *arg)
  aggiunge task
- int tpool_delete (t_pool *pool)
  elimina threadpool

4.12.1 Descrizione dettagliata

contiene implementazioni funzioni per la creazione, eliminazione e gestione del threadpool

Autore

Valerio Besozzi 543685 Si dichiara che il contenuto di questo file e' in ogni sua parte opera originale dell'autore

4.12.2 Documentazione delle funzioni

4.12.2.1 tpool_add()

```
int tpool_add (
            t_pool * pool,
            int(*)(job *) fun,
            void * arg )
```

aggiunge task

tpool_add

Parametri

pool	puntatore alla struttura t_pool in cui aggiungere task
(∗fun)(job	∗) funzione da chiamare
arg	argomento per la funzione da chiamare

Restituisce

0 se il task e' stato aggiunto correttamente, -1 altrimenti

4.12.2.2 tpool_create()

```
t_pool* tpool_create (
             int nthreads,
             int max_con )
```

crea threadpool

tpool_create

Parametri

nthreads	numero thread da creare
max_con	numero massimo connessioni

Restituisce

puntatore alla struttura t_pool creata

4.12.2.3 tpool_delete()

```
int tpool_delete (
             t_pool * pool )
```

elimina threadpool

tpool_delete

Parametri

pool	puntatore alla struttura t_pool da eliminare

Restituisce

0 se il threadpool e' stato eliminato, -1 altrimenti

4.13 Riferimenti per il file tpool.h

contiene funzioni per la creazione, eliminazione e gestione del threadpool

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <pthread.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/select.h>
#include <user.h>
#include <task.h>
```

Definizioni

- #define **CLEAN_UP**(pool)

Ridefinizioni di tipo (typedef)

- typedef struct _t_pool **t_pool**

Funzioni

- t_pool * tpool_create (int nthreads, int max_con)
  crea threadpool
- int tpool_add (t_pool *pool, int(*fun)(job *), void *arg)
  aggiunge task
- int tpool_delete (t_pool *pool)
  elimina threadpool

4.13.1 Descrizione dettagliata

contiene funzioni per la creazione, eliminazione e gestione del threadpool

Autore

Valerio Besozzi 543685 Si dichiara che il contenuto di questo file e' in ogni sua parte opera originale dell'autore

4.13.2 Documentazione delle definizioni

4.13.2.1 CLEAN_UP

```
#define CLEAN_UP(
              pool )
```

Valore:

```
if(pool) {                                  \
                                    tpool_free(pool);                       \
                                }                                           \
                                return NULL;                                \
```

4.13.3 Documentazione delle funzioni

4.13.3.1 tpool_add()

```
int tpool_add (
             t_pool * pool,
             int(*)(job *) fun,
             void * arg )
```

aggiunge task

tpool_add

Parametri

pool	puntatore alla struttura t_pool in cui aggiungere task
(∗fun)(job	∗) funzione da chiamare
arg	argomento per la funzione da chiamare

Restituisce

0 se il task e' stato aggiunto correttamente, -1 altrimenti

4.13.3.2 tpool_create()

```
t_pool* tpool_create (
             int nthreads,
             int max_con )
```

crea threadpool

tpool_create

Parametri

nthreads	numero thread da creare
max_con	numero massimo connessioni

Restituisce

puntatore alla struttura t_pool creata

4.13.3.3 tpool_delete()

```
int tpool_delete (
            t_pool * pool )
```

elimina threadpool

tpool_delete

Parametri

pool	puntatore alla struttura t_pool da eliminare

Restituisce

0 se il threadpool e' stato eliminato, -1 altrimenti

4.14 Riferimenti per il file user.c

file contenente implementazioni funzioni per operare sulle strutture per la gestione degli utenti

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <sys/epoll.h>
#include <unistd.h>
#include <uthash.h>
#include <utlist.h>
#include <config.h>
#include <message.h>
#include <user.h>
#include <parser.h>
#include <stats.h>
#include <connections.h>
```

Funzioni

- userdata * new_udata (conf *cf, struct statistics *chattyStats)

  crea la struttura per la gestione degli utenti
- void del_udata (userdata *ud)

  elimina la struttura per la gestione degli utenti
- int reg_user (userdata *ud, char *nick)

  registra utente
- int con_user (userdata *ud, char *nick, unsigned long fd)

connette utente

- int del_user (userdata *ud, char *nick)

  elimina utente
- int logout_user (userdata *ud, char *nick)

  disconnette utente
- int brutal_logout (userdata *ud, int fd, int epollfd, pthread_mutex_t *mtx)

  disconnette utente dato il suo fd
- void all_user (userdata *ud, ulist **list)

  restituisce lista utenti connessi
- void del_list (ulist **list)

  elimina lista utenti connessi
- int save_msg (userdata *ud, char *nick, message_t *msg)

  salva msg nella history di un utente non connesso
- void del_history (userdata *ud, history **list)

  elimina history utente
- user * find_user (userdata *ud, char *nick)

  cerca un utente
- history * find_history (userdata *ud, char *nick)

  cerca history utente
- void **print_list** (ulist *list)
- void **print_users** (userdata *ud)
- void stats (int reg, int con, int msg_delivered, int msg_waiting, int file_delivered, int file_waiting, int err, userdata *ud)

  aggiorna statistiche
- group * new_group (char *name)

  crea una struttura di tipo _group
- void del_group (group *gdata)

  elimina una struttura di tipo _group
- int reg_user_group (group *gdata, user *new)

  aggiunge un utente ad un gruppo
- int del_user_group (group *gdata, user *del)

  elimina un utente da un gruppo
- int find_user_group (group *gdata, char *nick)

  verifica la presenza di un utente in un gruppo
- group * find_group (userdata *ud, char *name)

  cerca un gruppo tra i gruppi creati
- int add_group (userdata *ud, char *name)

  crea e registra un gruppo in ud

4.14.1 Descrizione dettagliata

file contenente implementazioni funzioni per operare sulle strutture per la gestione degli utenti

Autore

Valerio Besozzi 543685 Si dichiara che il contenuto di questo file e' in ogni sua parte opera originale dell'autore

4.14.2 Documentazione delle funzioni

4.14.2.1 add_group()

```
int add_group (
            userdata * ud,
            char * name )
```

crea e registra un gruppo in ud

add_group

Parametri

ud	struttura _userdata dove e' presente hashmap con i gruppi
name	nome del gruppo da creare e registrare

Restituisce

0 se il gruppo e' stato creato e registrato correttamente, -1 altrimenti

4.14.2.2 all_user()

```
void all_user (
            userdata * ud,
            ulist ** list )
```

restituisce lista utenti connessi

all_user

Parametri

ud	puntatore alla struttura userdata dove cercare utente
list	lista dove salvare utenti

4.14.2.3 brutal_logout()

```
int brutal_logout (
            userdata * ud,
            int fd,
            int epollfd,
            pthread_mutex_t * mtx )
```

disconnette utente dato il suo fd

brutal_user

Parametri

ud	puntatore alla struttura userdata in cui disconnettere utente
fd	descrittore della connessione relativo all'utente da disconnettere
epollfd	fd relativo all'istanza epoll in uso
mtx	mutex

Restituisce

0 se la disconnessione e' avvenuta con successo, -1 altrimenti

4.14.2.4 con_user()

```
int con_user (
            userdata * ud,
            char * nick,
            unsigned long fd )
```

connette utente

con_user

Parametri

ud	puntatore alla struttura userdata in cui connettere utente
nick	nome dell'utente da connettere
fd	fd relativo all'utente da connettere

Restituisce

0 se la connessione e' avvenuta con successo, -1 altrimenti

4.14.2.5 del_group()

```
void del_group (
            group * gdata )
```

elimina una struttura di tipo _group

del_group

Parametri

gdata	il gruppo da eliminare

4.14.2.6 del_history()

```
void del_history (
            userdata * ud,
            history ** list )
```

elimina history utente

del_history

Parametri

ud	puntatore alla struttura userdata in cui e' presente utente
list	lista da eliminare

4.14.2.7 del_list()

```
void del_list (
            ulist ** list )
```

elimina lista utenti connessi

del_list

Parametri

list	lista da eliminare

4.14.2.8 del_udata()

```
void del_udata (
            userdata * ud )
```

elimina la struttura per la gestione degli utenti

del_udata

Parametri

ud	puntatore alla struttura userdata da eliminare

4.14.2.9 del_user()

```
int del_user (
            userdata * ud,
            char * nick )
```

elimina utente

del_user

Parametri

ud	puntatore alla struttura userdata in cui eliminare utente
nick	nome dell'utente da eliminare

Restituisce

0 se l'eliminazione e' avvenuta con successo, -1 altrimenti

4.14.2.10 del_user_group()

```
int del_user_group (
            group * gdata,
            user * del )
```

elimina un utente da un gruppo

del_user_group

Parametri

gdata	il gruppo in cui c'e' l'utente da rimuovere
del	utente da rimuovere dal gruppo

Restituisce

0 se utente e' stato eliminato correttamente, -1 in caso di errore

4.14.2.11 find_group()

```
group* find_group (
            userdata * ud,
            char * name )
```

cerca un gruppo tra i gruppi creati

find_group

Parametri

ud	struttura _userdata dove e' presente hashmap con i gruppi
name	nome del gruppo da cercare

Restituisce

puntatore alla struttura _group relativo al gruppo cercato, NULL altrimenti

4.14.2.12 find_history()

```
history* find_history (
            userdata * ud,
            char * nick )
```

cerca history utente

find_history

Parametri

ud	puntatore alla struttura userdata in cui cercare history utente
nick	nome utente da cercare

Restituisce

history se la history utente e' stata trovata, null altrimenti

4.14.2.13 find_user()

```
user* find_user (
            userdata * ud,
            char * nick )
```

cerca un utente

find_user

Parametri

ud	puntatore alla struttura userdata in cui cercare utente
nick	nome utente da cercare

Restituisce

user se l'utente e' stato trovato, null altrimenti

4.14.2.14 find_user_group()

```
int find_user_group (
            group * gdata,
            char * nick )
```

verifica la presenza di un utente in un gruppo

find_user_group

Parametri

gdata	il gruppo in cui cercare l'utente
nick	nickname dell'utente da cercare

Restituisce

0 se utente e' stato trovato, -1 in caso di errore

4.14.2.15 logout_user()

```
int logout_user (
            userdata * ud,
            char * nick )
```

disconnette utente

logout_user

Parametri

ud	puntatore alla struttura userdata in cui disconnettere utente
nick	nome dell'utente da disconnettere

Restituisce

0 se la disconnessione e' avvenuta con successo, -1 altrimenti

4.14.2.16 new_group()

```
group* new_group (
            char * name )
```

crea una struttura di tipo _group

new_group

Parametri

name	il nome del gruppo da creare

Restituisce

puntatore alla struttura _group appena creata

4.14.2.17 new_udata()

```
userdata* new_udata (
            conf * cf,
            struct statistics * chattyStats )
```

crea la struttura per la gestione degli utenti

new_udata

Parametri

cf	struttura contenente configurazioni
chattyStats	struttura contenente statistiche

Restituisce

puntatore alla struttura userdata creata

4.14.2.18 reg_user()

```
int reg_user (
            userdata * ud,
            char * nick )
```

registra utente

reg_user

Parametri

ud	puntatore alla struttura userdata in cui aggiungere utente
nick	nome dell'utente da registrare

Restituisce

0 se la registrazione e' avvenuta con successo, -1 altrimenti

4.14.2.19 reg_user_group()

```
int reg_user_group (
            group * gdata,
            user * new )
```

aggiunge un utente ad un gruppo

reg_user_group

Parametri

gdata	il gruppo in cui aggiungere l'utente
new	utente da aggiungere al gruppo

Restituisce

0 se utente e' stato aggiunto correttamente, -1 in caso di errore

4.14.2.20 save_msg()

```
int save_msg (
            userdata * ud,
            char * nick,
            message_t * msg )
```

salva msg nella history di un utente non connesso

save_msg

Parametri

ud	puntatore alla struttura userdata in cui e' presente utente
nick	nome utente del destinatario del msg
msg	msg da salvare

Restituisce

0 se la disconnessione e' avvenuta con successo, -1 altrimenti

4.14.2.21 stats()

```
void stats (
            int reg,
            int con,
            int msg_delivered,
            int msg_waiting,
            int file_delivered,
            int file_waiting,
            int err,
            userdata * ud )
```

aggiorna statistiche

stats

Parametri

ud	puntatore alla struttura userdata in cui e' presente la struttura stats da aggiornare

4.15 Riferimenti per il file user.h

file contenente definizione delle strutture per la gestione degli utenti e funzioni per operare su esse

```
#include <string.h>
#include <pthread.h>
#include <uthash.h>
#include <utlist.h>
#include <config.h>
#include <message.h>
#include <parser.h>
#include <stats.h>
#include <sys/socket.h>
#include <sys/select.h>
#include <sys/epoll.h>
```

Strutture dati

- struct _history

  lista history utente
- struct _user_list

  lista utenti
- struct _user

  struttura contenente informazioni utente
- struct _user_group

  struttura contenente informazioni utente di un gruppo
- struct _group
- struct _usersdata

  struttura contenente utenti registrati

Ridefinizioni di tipo (typedef)

- typedef struct _history **history**
- typedef struct _user_list **ulist**
- typedef struct _user **user**
- typedef struct _user_group **user_group**
- typedef struct _group **group**
- typedef struct _usersdata **userdata**

Funzioni

- userdata * new_udata (conf *cf, struct statistics *chattyStats)

  crea la struttura per la gestione degli utenti
- void del_udata (userdata *ud)

  elimina la struttura per la gestione degli utenti
- int reg_user (userdata *ud, char *nick)

  registra utente
- int con_user (userdata *ud, char *nick, unsigned long fd)

  connette utente
- int del_user (userdata *ud, char *nick)

  elimina utente
- int logout_user (userdata *ud, char *nick)

  disconnette utente
- int brutal_logout (userdata *ud, int fd, int epollfd, pthread_mutex_t *mtx)

  disconnette utente dato il suo fd
- void all_user (userdata *ud, ulist **list)

  restituisce lista utenti connessi
- void del_list (ulist **list)

  elimina lista utenti connessi
- int save_msg (userdata *ud, char *nick, message_t *msg)

  salva msg nella history di un utente non connesso
- void del_history (userdata *ud, history **list)

  elimina history utente
- user * find_user (userdata *ud, char *nick)

  cerca un utente
- history * find_history (userdata *ud, char *nick)

  cerca history utente
- void **print_list** (ulist *list)
- void **print_users** (userdata *ud)
- void stats (int reg, int con, int msg_delivered, int msg_waiting, int file_delivered, int file_waiting, int err, userdata *ud)

  aggiorna statistiche
- group * new_group (char *name)

  crea una struttura di tipo _group
- void del_group (group *gdata)

  elimina una struttura di tipo _group
- int reg_user_group (group *gdata, user *new)

  aggiunge un utente ad un gruppo
- int del_user_group (group *gdata, user *del)

  elimina un utente da un gruppo
- int find_user_group (group *gdata, char *nick)

  verifica la presenza di un utente in un gruppo
- group * find_group (userdata *ud, char *name)

  cerca un gruppo tra i gruppi creati
- int add_group (userdata *ud, char *name)

  crea e registra un gruppo in ud

4.15.1 Descrizione dettagliata

file contenente definizione delle strutture per la gestione degli utenti e funzioni per operare su esse

Autore

Valerio Besozzi 543685 Si dichiara che il contenuto di questo file e' in ogni sua parte opera originale dell'autore

4.15.2 Documentazione delle funzioni

4.15.2.1 add_group()

```
int add_group (
            userdata * ud,
            char * name )
```

crea e registra un gruppo in ud

add_group

Parametri

ud	struttura _userdata dove e' presente hashmap con i gruppi
name	nome del gruppo da creare e registrare

Restituisce

0 se il gruppo e' stato creato e registrato correttamente, -1 altrimenti

4.15.2.2 all_user()

```
void all_user (
            userdata * ud,
            ulist ** list )
```

restituisce lista utenti connessi

all_user

Parametri

ud	puntatore alla struttura userdata dove cercare utente
list	lista dove salvare utenti

4.15.2.3 brutal_logout()

```
int brutal_logout (
            userdata * ud,
            int fd,
            int epollfd,
            pthread_mutex_t * mtx )
```

disconnette utente dato il suo fd

brutal_user

Parametri

ud	puntatore alla struttura userdata in cui disconnettere utente
fd	descrittore della connessione relativo all'utente da disconnettere
epollfd	fd relativo all'istanza epoll in uso
mtx	mutex

Restituisce

0 se la disconnessione e' avvenuta con successo, -1 altrimenti

4.15.2.4 con_user()

```
int con_user (
            userdata * ud,
            char * nick,
            unsigned long fd )
```

connette utente

con_user

Parametri

ud	puntatore alla struttura userdata in cui connettere utente
nick	nome dell'utente da connettere
fd	fd relativo all'utente da connettere

Restituisce

0 se la connessione e' avvenuta con successo, -1 altrimenti

4.15.2.5 del_group()

```
void del_group (
            group * gdata )
```

Parametri

gdata	il gruppo da eliminare

4.15.2.6 del_history()

```
void del_history (
            userdata * ud,
            history ** list )
```

elimina history utente

del_history

Parametri

ud	puntatore alla struttura userdata in cui e' presente utente
list	lista da eliminare

4.15.2.7 del_list()

```
void del_list (
            ulist ** list )
```

elimina lista utenti connessi

del_list

Parametri

list	lista da eliminare

4.15.2.8 del_udata()

```
void del_udata (
            userdata * ud )
```

elimina la struttura per la gestione degli utenti

del_udata

Parametri

ud	puntatore alla struttura userdata da eliminare

4.15.2.9 del_user()

```
int del_user (
            userdata * ud,
            char * nick )
```

elimina utente

del_user

Parametri

ud	puntatore alla struttura userdata in cui eliminare utente
nick	nome dell'utente da eliminare

Restituisce

0 se l'eliminazione e' avvenuta con successo, -1 altrimenti

4.15.2.10 del_user_group()

```
int del_user_group (
            group * gdata,
            user * del )
```

elimina un utente da un gruppo

del_user_group

Parametri

gdata	il gruppo in cui c'e' l'utente da rimuovere
del	utente da rimuovere dal gruppo

Restituisce

0 se utente e' stato eliminato correttamente, -1 in caso di errore

4.15.2.11 find_group()

```
group* find_group (
            userdata * ud,
            char * name )
```

cerca un gruppo tra i gruppi creati

find_group

Parametri

ud	struttura _userdata dove e' presente hashmap con i gruppi
name	nome del gruppo da cercare

Restituisce

puntatore alla struttura _group relativo al gruppo cercato, NULL altrimenti

4.15.2.12 find_history()

```
history* find_history (
            userdata * ud,
            char * nick )
```

cerca history utente

find_history

Parametri

ud	puntatore alla struttura userdata in cui cercare history utente
nick	nome utente da cercare

Restituisce

history se la history utente e' stata trovata, null altrimenti

4.15.2.13 find_user()

```
user* find_user (
            userdata * ud,
            char * nick )
```

cerca un utente

find_user

Parametri

ud	puntatore alla struttura userdata in cui cercare utente
nick	nome utente da cercare

Restituisce

user se l'utente e' stato trovato, null altrimenti

4.15.2.14 find_user_group()

```
int find_user_group (
            group * gdata,
            char * nick )
```

verifica la presenza di un utente in un gruppo

find_user_group

Parametri

gdata	il gruppo in cui cercare l'utente
nick	nickname dell'utente da cercare

Restituisce

0 se utente e' stato trovato, -1 in caso di errore

4.15.2.15 logout_user()

```
int logout_user (
            userdata * ud,
            char * nick )
```

disconnette utente

logout_user

Parametri

ud	puntatore alla struttura userdata in cui disconnettere utente
nick	nome dell'utente da disconnettere

Restituisce

0 se la disconnessione e' avvenuta con successo, -1 altrimenti

4.15.2.16 new_group()

```
group* new_group (
            char * name )
```

crea una struttura di tipo _group

new_group

Parametri

name	il nome del gruppo da creare

Restituisce

puntatore alla struttura _group appena creata

4.15.2.17 new_udata()

```
userdata* new_udata (
            conf * cf,
            struct statistics * chattyStats )
```

crea la struttura per la gestione degli utenti

new_udata

Parametri

cf	struttura contenente configurazioni
chattyStats	struttura contenente statistiche

Restituisce

puntatore alla struttura userdata creata

4.15.2.18 reg_user()

```
int reg_user (
            userdata * ud,
            char * nick )
```

registra utente

reg_user

Parametri

ud	puntatore alla struttura userdata in cui aggiungere utente
nick	nome dell'utente da registrare

Restituisce

0 se la registrazione e' avvenuta con successo, -1 altrimenti

4.15.2.19 reg_user_group()

```
int reg_user_group (
            group * gdata,
            user * new )
```

aggiunge un utente ad un gruppo

reg_user_group

Parametri

gdata	il gruppo in cui aggiungere l'utente
new	utente da aggiungere al gruppo

Restituisce

0 se utente e' stato aggiunto correttamente, -1 in caso di errore

4.15.2.20 save_msg()

```
int save_msg (
            userdata * ud,
            char * nick,
            message_t * msg )
```

salva msg nella history di un utente non connesso

save_msg

Parametri

ud	puntatore alla struttura userdata in cui e' presente utente
nick	nome utente del destinatario del msg
msg	msg da salvare

Restituisce

0 se la disconnessione e' avvenuta con successo, -1 altrimenti

4.15.2.21 stats()

```
void stats (
            int reg,
            int con,
            int msg_delivered,
            int msg_waiting,
```

```
            int file_delivered,
            int file_waiting,
            int err,
            userdata * ud )
```

aggiorna statistiche

stats

Parametri

ud	puntatore alla struttura userdata in cui e' presente la struttura stats da aggiornare

4.16 Riferimenti per il file utility.c

file contenente implementazioni funzioni per la scrittura e lettura socket

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <errno.h>
#include <utility.h>
```

Funzioni

- ssize_t readn (int fd, void ∗vptr, size_t n)
  legge fd
- ssize_t writen (int fd, const void ∗vptr, size_t n)
  scrive fd

4.16.1 Descrizione dettagliata

file contenente implementazioni funzioni per la scrittura e lettura socket

Autore

Valerio Besozzi 543685 Si dichiara che il contenuto di questo file e' in ogni sua parte opera originale dell'autore

4.16.2 Documentazione delle funzioni

4.16.2.1 readn()

```
ssize_t readn (
            int filedes,
            void * buff,
            size_t nbytes )
```

legge fd

readn

Parametri

filedes	fd da leggere
buff	puntatore al buffer dove scrivere quello che leggo
nbytes	bytes da leggere

Restituisce

bytes letti, -1 in caso di errore

4.16.2.2 writen()

```
ssize_t writen (
            int filedes,
            const void * buff,
            size_t nbytes )
```

scrive fd

writen

Parametri

filedes	fd in cui scrivere
buff	puntatore al buffer contenente quello che che scrivo
nbytes	bytes da scrivere

Restituisce

bytes scritti, -1 in caso di errore

4.17 Riferimenti per il file utility.h

file contenente funzioni per la scrittura e lettura socket

```
#include <sys/types.h>
#include <unistd.h>
```

Funzioni

- ssize_t readn (int filedes, void ∗buff, size_t nbytes)

  legge fd
- ssize_t writen (int filedes, const void ∗buff, size_t nbytes)

  scrive fd

4.17.1 Descrizione dettagliata

file contenente funzioni per la scrittura e lettura socket

Autore

Valerio Besozzi 543685 Si dichiara che il contenuto di questo file e' in ogni sua parte opera originale dell'autore

4.17.2 Documentazione delle funzioni

4.17.2.1 readn()

```
ssize_t readn (
            int filedes,
            void * buff,
            size_t nbytes )
```

legge fd

readn

Parametri

filedes	fd da leggere
buff	puntatore al buffer dove scrivere quello che leggo
nbytes	bytes da leggere

Restituisce

bytes letti, -1 in caso di errore

4.17.2.2 writen()

```
ssize_t writen (
            int filedes,
            const void * buff,
            size_t nbytes )
```

scrive fd

writen

Parametri

filedes	fd in cui scrivere
buff	puntatore al buffer contenente quello che che scrivo
nbytes	bytes da scrivere

Restituisce

bytes scritti, -1 in caso di errore

Indice analitico