Chatterbox

1

Generato da Doxygen 1.8.8

Gio 7 Feb 2019 19:03:35

Indice

Capitolo 1

Chatterbox

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 data

body del messaggio

`#include <message.h>`

4.1.1 Descrizione dettagliata

body del messaggio

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

- message.h

4.2 Riferimenti per la struct header

header del messaggio

`#include <message.h>`

4.2.1 Descrizione dettagliata

header del messaggio

header della parte dati

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

- message.h

4.3 Riferimenti per la struct HISTORY

Struttura history dei messaggi.

`#include <history.h>`

4.3.1 Descrizione dettagliata

Struttura history dei messaggi.

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

- history.h

4.4 Riferimenti per la struct history_t

Diagramma di collaborazione per history_t:

Campi

- message_t ** **msgs**
- int **head**
- int **dim**
- int **dimMax**
- pthread_mutex_t **mtx**

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

- history.h

4.5 Riferimenti per la struct icl_entry_s

Diagramma di collaborazione per icl_entry_s:

Campi

- void * **key**
- void * **data**
- struct icl_entry_s * **next**

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

- icl_hash.h

4.6 Riferimenti per la struct icl_hash_s

Diagramma di collaborazione per icl_hash_s:

Campi

- pthread_mutex_t * **mutexes**
- int **nbuckets**
- int **nentries**
- int **nsections**
- icl_entry_t ** **buckets**
- unsigned int(* **hash_function**)(void *)
- int(* **hash_key_compare**)(void *, void *)

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

- icl_hash.h

4.7 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

4.8 Riferimenti per la struct message_data_t

Diagramma di collaborazione per 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

4.9 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

4.10 Riferimenti per la struct message_t

Diagramma di collaborazione per 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

4.11 Riferimenti per la struct messaggio

tipo del messaggio

`#include <message.h>`

4.11.1 Descrizione dettagliata

tipo del messaggio

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

- message.h

4.12 Riferimenti per la struct Node

`#include <queue.h>`

Diagramma di collaborazione per Node:

Campi

- void * **data**
- struct Node * **next**

4.12.1 Descrizione dettagliata

Elemento della coda.

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

- queue.h

4.13 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

4.14 Riferimenti per la struct Queue

`#include <queue.h>`

Diagramma di collaborazione per Queue:

Campi

- Node_t * **head**
- Node_t * **tail**
- unsigned long **qlen**

4.14.1 Descrizione dettagliata

Struttura dati coda.

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

- queue.h

4.15 Riferimenti per la struct serverConf

Parametri di configurazione del server.

`#include <parser.h>`

Campi

- char **UnixPath** [MAX_LINESIZE]
- char **DirName** [MAX_LINESIZE]
- char **StatFileName** [MAX_LINESIZE]
- int **MaxConnections**
- int **ThreadsInPool**
- int **MaxMsgSize**
- int **MaxFileSize**
- int **MaxHistMsgs**

4.15.1 Descrizione dettagliata

Parametri di configurazione del server.

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

- parser.h

4.16 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

4.17 Riferimenti per la struct user

Struttura utente registrato.

`#include <user.h>`

4.17.1 Descrizione dettagliata

Struttura utente registrato.

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

- user.h

4.18 Riferimenti per la struct user_online

Struttura utente online.

`#include <user.h>`

4.18.1 Descrizione dettagliata

Struttura utente online.

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

- user.h

4.19 Riferimenti per la struct user_online_t

Campi

- char **name** [MAX_NAME_LENGTH+1]
- int **fd**

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

- user.h

4.20 Riferimenti per la struct user_t

Diagramma di collaborazione per user_t:

Campi

- char **name** [MAX_NAME_LENGTH+1]
- int **fd**
- history_t * **history**

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

- user.h

4.21 Riferimenti per la struct users_db

Struttura dati utilizzata dal server.

`#include <user.h>`

4.21.1 Descrizione dettagliata

Struttura dati utilizzata dal server.

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

- user.h

4.22 Riferimenti per la struct users_db_t

Diagramma di collaborazione per users_db_t:

Campi

- icl_hash_t * **db**
- user_online_t * **users_online**
- int **n_users_online**
- int **history_size**
- int **max_connections**

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 chatty.c

File principale del server chatterbox.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <assert.h>
#include <string.h>
#include <signal.h>
#include <pthread.h>
#include <errno.h>
#include <sys/stat.h>
#include <sys/select.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <sys/mman.h>
#include <sys/un.h>
#include <fcntl.h>
#include "connections.h"
#include "ops.h"
#include "queue.h"
#include "parser.h"
#include "icl_hash.h"
#include "user.h"
#include "util.h"
#include "stats.h"
```

Grafo delle dipendenze di inclusione per chatty.c:

Definizioni

- #define **_POSIX_C_SOURCE** 200809L
- #define **NBUCKETS** 1024

Funzioni

- void ∗ sigHandler (void ∗arg)

  Funzione eseguita dal thread che gestisce i segnali.
- char ∗ setDir (char ∗filePath)

  Aggiunge DirName al path filePath.

- int register_op (message_t msg_receved, int client_fd)

  Gestisce la richiesta di registrazione di un nickname.
- int connect_op (message_t msg_receved, int client_fd)

  Gestisce la richiesta di connessione di un client.
- int posttxt_op (message_t msg_receved, int client_fd)

  Gestisce la richiesta di invio di un messaggio testuale ad un nickname.
- int posttxtall_op (message_t msg_receved, int client_fd)

  Gestisce l'invio di un messaggio testuale a tutti gli utenti registrati.
- int postfile_op (message_t msg_receved, int client_fd)

  Gestisce la richiesta di invio di un
  file ad un nickname.
- int getfile_op (message_t msg_receved, int client_fd)

  Gestisce la richiesta di recupero di un
  file inviato da un altro nickname.
- int getprevmsgs_op (message_t msg_receved, int client_fd)

  Gestisce la richiesta di recupero degli ultimi messaggi inviati al client.
- int usrlist_op (message_t msg_receved, int client_fd)

  Gestisce la richiesta della lista di tutti i nickname connessi.
- int unregister_op (message_t msg_receved, int client_fd)

  Gestisce la richiesta di deregistrazione di un nickname.
- int disconnect_op (message_t msg_receved, int client_fd)

  Gestisce la richiesta di disconnessione del client.
- int handler (message_t msg_receved, int client_fd)

  Gestisce le richieste dei client.
- void * thread_worker (void *arg)

  Funzione eseguita dai thread presenti nel pool.
- int **main** (int argc, char *argv[])

Variabili

- struct statistics **chattyStats** = { 0,0,0,0,0,0,0 }
- struct serverConf **configuration** = { {'\0', '\0', '\0', 0, 0, 0, 0, 0} }
- sigset_t **sigset**
- Queue_t * **q**
- pthread_t * **threadPool**
- pthread_t **sigTread**
- users_db_t * **users_db** = NULL
- fd_set **set**

5.1.1 Descrizione dettagliata

File principale del server chatterbox.

membox Progetto del corso di LSO 2017/2018

Dipartimento di Informatica Università di Pisa Docenti: Prencipe, Torquati

Autore

Federico Germinario 545081

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

5.1.2 Documentazione delle funzioni

5.1.2.1 int connect_op (message_t *msg_receved,* int *client_fd*)

Gestisce la richiesta di connessione di un client.

connect_op

Parametri

msg_receved	messaggio ricevuto dal client
client_fd	descrittore della connessione

Restituisce

0 successo, -1 fallimento

5.1.2.2 int disconnect_op (message_t *msg_receved,* int *client_fd*)

Gestisce la richiesta di disconnessione del client.

disconnect_op

Parametri

msg_receved	messaggio ricevuto dal client
client_fd	descrittore della connessione

Restituisce

0 successo, -1 fallimento

5.1.2.3 int getfile_op (message_t *msg_receved,* int *client_fd*)

Gestisce la richiesta di recupero di un

file inviato da un altro nickname.

getfile_op

Parametri

msg_receved	messaggio ricevuto dal client
client_fd	descrittore della connessione

Restituisce

0 successo, -1 fallimento

5.1.2.4 int getprevmsgs_op (message_t *msg_receved,* int *client_fd*)

Gestisce la richiesta di recupero degli ultimi messaggi inviati al client.

getprevmsgs_op

Parametri

msg_receved	messaggio ricevuto dal client
client_fd	descrittore della connessione

Restituisce

0 successo, -1 fallimento

5.1.2.5 int handler (message_t *msg_receved,* int *client_fd*)

Gestisce le richieste dei client.

handler

Parametri

msg_receved	messaggio ricevuto dal client
client_fd	descrittore della connessione

Restituisce

0 successo, -1 fallimento

5.1.2.6 int postfile_op (message_t *msg_receved,* int *client_fd*)

Gestisce la richiesta di invio di un

file ad un nickname.

postfile_op

Parametri

msg_receved	messaggio ricevuto dal client
client_fd	descrittore della connessione

Restituisce

0 successo, -1 fallimento

5.1.2.7 int posttxt_op (message_t *msg_receved,* int *client_fd*)

Gestisce la richiesta di invio di un messaggio testuale ad un nickname.

posttxt_op

Parametri

msg_receved	messaggio ricevuto dal client
client_fd	descrittore della connessione

Restituisce

0 successo, -1 fallimento

5.1.2.8 int posttxtall_op (message_t *msg_receved,* int *client_fd*)

Gestisce l'invio di un messaggio testuale a tutti gli utenti registrati.

posttxtall_op

Parametri

msg_receved	messaggio ricevuto dal client
client_fd	descrittore della connessione

Restituisce

0 successo, -1 fallimento

5.1.2.9 int register_op (message_t *msg_receved,* int *client_fd*)

Gestisce la richiesta di registrazione di un nickname.

register_op

Parametri

msg_receved	messaggio ricevuto dal client
client_fd	descrittore della connessione

Restituisce

0 successo, -1 fallimento

5.1.2.10 char* setDir (char * *filePath*)

Aggiunge DirName al path filePath.

setDir

Parametri

filePath	

Restituisce

DirName concatenato con filePath

5.1.2.11 void* sigHandler (void * *arg*)

Funzione eseguita dal thread che gestisce i segnali.

sigHandler

Parametri

arg	

Restituisce

null

5.1.2.12 void* thread_worker (void * *arg*)

Funzione eseguita dai thread presenti nel pool.

thread_worker

Parametri

arg	thread id

Restituisce

null

5.1.2.13 int unregister_op (message_t *msg_receved,* int *client_fd*)

Gestisce la richiesta di deregistrazione di un nickname.

unregister_op

Parametri

msg_receved	messaggio ricevuto dal client
client_fd	descrittore della connessione

Restituisce

0 successo, -1 fallimento

5.1.2.14 int usrlist_op (message_t *msg_receved,* int *client_fd*)

Gestisce la richiesta della lista di tutti i nickname connessi.

usrlist_op

Parametri

msg_receved	messaggio ricevuto dal client
client_fd	descrittore della connessione

Restituisce

0 successo, -1 fallimento

5.2 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>
```

Grafo delle dipendenze di inclusione per client.c:

Strutture dati

- struct operation_t

Definizioni

- #define **_POSIX_C_SOURCE** 200809L

Funzioni

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

5.2.1 Descrizione dettagliata

Semplice client di test.

5.3 Riferimenti per il file config.h

File contenente alcune define con valori massimi utilizzabili.

Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

Definizioni

- #define **MAX_NAME_LENGTH** 32

Ridefinizioni di tipo (typedef)

- typedef int **make_iso_compilers_happy**

5.3.1 Descrizione dettagliata

File contenente alcune define con valori massimi utilizzabili.

5.4 Riferimenti per il file history.c

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include "config.h"
#include "history.h"
#include "util.h"
```

Grafo delle dipendenze di inclusione per history.c:

Funzioni

- history_t ∗ createHistory (int MaxHistMsg)

  Crea una nuova history.

- int destroyHistory (history_t *history)

  Dealloca le strutture dati della history.
- int insertMsg (history_t *history, message_t *msg)

  Inserisce un messaggio nella history in mutua escusione.
- int outMsg (history_t *history, message_t ***msg_list)

  Estrae tutti i messaggi presenti.

5.4.1 Descrizione dettagliata

Autore

Federico Germinario 545081

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

5.4.2 Documentazione delle funzioni

5.4.2.1 history_t* createHistory (int *MaxHistMsg*)

Crea una nuova history.

createHistory

Parametri

MaxHistMsgs	dimensione massima history

Restituisce

puntatore alla nuova history

5.4.2.2 int destroyHistory (history_t * *history*)

Dealloca le strutture dati della history.

Dealloca le strutture della history.

destroyHistory

Parametri

history	puntatore history

Restituisce

0 successo, -1 fallimento

5.4.2.3 int insertMsg (history_t * *history,* message_t * *msg*)

Inserisce un messaggio nella history in mutua escusione.

insertMsg

Parametri

history	puntatore alla history dove inserire il messaggio
msg	puntatore al messaggio da inserire

Restituisce

0 successo , -1 fallimento

5.4.2.4 int outMsg (history_t * *history*, message_t *** *msg_list*)

Estrae tutti i messaggi presenti.

outMsg

Parametri

history	puntatore history
msg_list	puntatore dove memorizzare i messagi estratti

Restituisce

numero messaggi da leggere, -1 fallimento

5.5 Riferimenti per il file history.h

History circolare con concorrenza.

```
#include "message.h"
#include <stdio.h>
#include <pthread.h>
```

Grafo delle dipendenze di inclusione per history.h: Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

Strutture dati

- struct history_t

Funzioni

- history_t * createHistory (int MaxHistMsgs)

  Crea una nuova history.
- int destroyHistory (history_t *history)

  Dealloca le strutture della history.
- int insertMsg (history_t *history, message_t *msg)

  Inserisce un messaggio nella history in mutua escusione.
- int outMsg (history_t *history, message_t ***msg_list)

  Estrae tutti i messaggi presenti.

5.5.1 Descrizione dettagliata

History circolare con concorrenza.

Autore

Federico Germinario 545081

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

5.5.2 Documentazione delle funzioni

5.5.2.1 history_t* createHistory (int *MaxHistMsg*)

Crea una nuova history.

createHistory

Parametri

MaxHistMsgs	dimensione massima history

Restituisce

puntatore alla nuova history

5.5.2.2 int destroyHistory (history_t * *history*)

Dealloca le strutture della history.

destroyHistory

Parametri

history	puntatore history

Restituisce

0 successo, -1 fallimento

Dealloca le strutture della history.

destroyHistory

Parametri

history	puntatore history

Restituisce

0 successo, -1 fallimento

5.5.2.3 int insertMsg (history_t * *history*, message_t * *msg*)

Inserisce un messaggio nella history in mutua escusione.

insertMsg

Parametri

history	puntatore alla history dove inserire il messaggio
msg	puntatore al messaggio da inserire

Restituisce

0 successo , -1 fallimento

5.5.2.4 int outMsg (history_t * *history*, message_t *** *msg_list*)

Estrae tutti i messaggi presenti.

outMsg

Parametri

history	puntatore history
msg_list	puntatore dove memorizzare i messagi estratti

Restituisce

numero messaggi da leggere, -1 fallimento

5.6 Riferimenti per il file icl_hash.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <assert.h>
#include <limits.h>
#include <math.h>
#include "icl_hash.h"
```

Grafo delle dipendenze di inclusione per icl_hash.c:

Definizioni

- #define **BITS_IN_int** (sizeof(int) * CHAR_BIT)
- #define **THREE_QUARTERS** ((int) ((BITS_IN_int * 3) / 4))
- #define **ONE_EIGHTH** ((int) (BITS_IN_int / 8))
- #define **HIGH_BITS** (~((unsigned int)(~0) >> ONE_EIGHTH))

Funzioni

- int lock_hash_section (icl_hash_t *ht, void *key)

  Prende la m.e. della sezione della tabella relativa a key.
- int unlock_hash_section (icl_hash_t *ht, void *key)

  Rilascia la m.e. della sezione della tabella relativa a key.
- int lock_hash (icl_hash_t *ht)

  Prende la m.e. dell' intera tabella hash.
- int unlock_hash (icl_hash_t *ht)

  Rilascia la m.e. dell' intera tabella hash.
- icl_hash_t * icl_hash_create (int nbuckets, unsigned int(*hash_function)(void *), int(*hash_key_compare)(void *, void *))

  Create a new hash table.

- void * icl_hash_find (icl_hash_t *ht, void *key)

  Search for an entry in a hash table.
- int icl_hash_insert (icl_hash_t *ht, void *key, void *data)

  Insert an item into the hash table.
- int icl_hash_delete (icl_hash_t *ht, void *key, void(*free_key)(void *), void(*free_data)(void *))

  Free one hash table entry located by key (key and data are freed using functions).
- int icl_hash_destroy (icl_hash_t *ht, void(*free_key)(void *), void(*free_data)(void *))

  Free hash table structures (key and data are freed using functions).

5.6.1 Descrizione dettagliata

Dependency free hash table implementation.

Autore

Jakub Kurzak

5.6.2 Documentazione delle funzioni

5.6.2.1 icl_hash_t* icl_hash_create (int *nbuckets*, unsigned int(*)(void *) *hash_function*, int(*)(void *, void *) *hash_key_compare*)

Create a new hash table.

icl_hash_create

Parametri

in	*nbuckets*	– number of buckets to create
in	*hash_function*	– pointer to the hashing function to be used
in	*hash_key_compare*	– pointer to the hash key comparison function to be used

Restituisce

pointer to new hash table.

5.6.2.2 int icl_hash_delete (icl_hash_t * *ht*, void * *key*, void(*)(void *) *free_key*, void(*)(void *) *free_data*)

Free one hash table entry located by key (key and data are freed using functions).

icl_hash_delete

Parametri

ht	– the hash table to be freed
key	– the key of the new item
free_key	– pointer to function that frees the key
free_data	– pointer to function that frees the data

Restituisce

0 on success, -1 on failure.

5.6.2.3 int icl_hash_destroy (icl_hash_t * *ht,* void(*)(void *) *free_key,* void(*)(void *) *free_data*)

Free hash table structures (key and data are freed using functions).

icl_hash_destroy

Parametri

ht	– the hash table to be freed
free_key	– pointer to function that frees the key
free_data	– pointer to function that frees the data

Restituisce

0 on success, -1 on failure.

5.6.2.4 void* icl_hash_find (icl_hash_t * *ht,* void * *key*)

Search for an entry in a hash table.

icl_hash_find

Parametri

ht	– the hash table to be searched
key	– the key of the item to search for

Restituisce

pointer to the data corresponding to the key. If the key was not found, returns NULL.

5.6.2.5 int icl_hash_insert (icl_hash_t * *ht,* void * *key,* void * *data*)

Insert an item into the hash table.

icl_hash_insert

Parametri

ht	– the hash table
key	– the key of the new item
data	– pointer to the new item's data

Restituisce

0 successo, -1 utente gia registrato, -2 inserimento fallito

5.6.2.6 int lock_hash (icl_hash_t * *ht*)

Prende la m.e. dell' intera tabella hash.

lock_hash

Parametri

ht	tabella hash

Restituisce

-1 fallimento, 0 successo

5.6.2.7 int lock_hash_section (icl_hash_t * *ht,* void * *key*)

Prende la m.e. della sezione della tabella relativa a key.

lock_hash_section

Parametri

ht	tabella hash
key	chiave appartenente alla sezione su cui prendere la mutua esclusione

Restituisce

-1 fallimento, 0 successo

5.6.2.8 int unlock_hash (icl_hash_t * *ht*)

Rilascia la m.e. dell' intera tabella hash.

unlock_hash

Parametri

ht	tabella hash

Restituisce

-1 fallimento, 0 successo

5.6.2.9 int unlock_hash_section (icl_hash_t * *ht*, void * *key*)

Rilascia la m.e. della sezione della tabella relativa a key.

unlock_hash_section

Parametri

ht	tabella hash
key	chiave appartenente alla sezione su cui rilasciare la mutua esclusione

Restituisce

-1 fallimento, 0 successo

5.7 Riferimenti per il file icl_hash.h

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

Grafo delle dipendenze di inclusione per icl_hash.h: Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

Strutture dati

- struct icl_entry_s
- struct icl_hash_s

Definizioni

- #define **icl_hash_foreach**(ht, dp, code)
- #define **icl_hash_foreach_mutex**(ht, dp, code)

Ridefinizioni di tipo (typedef)

- typedef struct icl_entry_s **icl_entry_t**
- typedef struct icl_hash_s **icl_hash_t**

Funzioni

- int lock_hash_section (icl_hash_t *ht, void *key)

  Prende la m.e. della sezione della tabella relativa a key.
- int unlock_hash_section (icl_hash_t *ht, void *key)

  Rilascia la m.e. della sezione della tabella relativa a key.
- int lock_hash (icl_hash_t *ht)

  Prende la m.e. dell' intera tabella hash.
- int unlock_hash (icl_hash_t *ht)

  Rilascia la m.e. dell' intera tabella hash.
- icl_hash_t * icl_hash_create (int nbuckets, unsigned int(*hash_function)(void *), int(*hash_key_compare)(void *, void *))

  Create a new hash table.
- void * icl_hash_find (icl_hash_t *, void *)

  Search for an entry in a hash table.
- int icl_hash_insert (icl_hash_t *, void *, void *)

  Insert an item into the hash table.
- int icl_hash_destroy (icl_hash_t *, void(*)(void *), void(*)(void *))

  Free hash table structures (key and data are freed using functions).
- int icl_hash_delete (icl_hash_t *ht, void *key, void(*free_key)(void *), void(*free_data)(void *))

  Free one hash table entry located by key (key and data are freed using functions).

5.7.1 Descrizione dettagliata

Header file for icl_hash routines.

5.7.2 Documentazione delle definizioni

5.7.2.1 #define icl_hash_foreach(*ht, dp, code*)

Valore:

```
int tmpint;                                                                 \
    icl_entry_t *tmpent;                                                        \
    char *kp;                                                                   \
    for (tmpint=0;tmpint<ht->nbuckets; tmpint++){                               \
        for (tmpent=ht->buckets[tmpint];                                        \
             tmpent!=NULL&&((kp=tmpent->key)!=NULL)&&((dp=tmpent->data)!=NULL); \
             tmpent=tmpent->next){                                              \
                code                                                            \
            };                  \
    }
```

5.7.2.2 #define icl_hash_foreach_mutex(*ht, dp, code*)

Valore:

```
int tmpint;                                                                    \
    icl_entry_t *tmpent;                                                       \
    char *kp;                                                                  \
    for (tmpint=0;tmpint<ht->nbuckets; tmpint++){                              \
        unsigned int ind_mutex = tmpint % ht->nsections;                       \
        pthread_mutex_lock (&(ht->mutexes[ind_mutex]));                        \
        for (tmpent=ht->buckets[tmpint];                                       \
             tmpent!=NULL&&((kp=tmpent->key)!=NULL)&&((dp=tmpent->data)!=NULL); \
             tmpent=tmpent->next){                                             \
                code                                                           \
            }pthread_mutex_unlock (&(ht->mutexes[ind_mutex]));                 \
    }
```

5.7.3 Documentazione delle funzioni

5.7.3.1 icl_hash_t* icl_hash_create (int *nbuckets*, unsigned int(*)(void *) *hash_function*, int(*)(void *, void *) *hash_key_compare*)

Create a new hash table.

icl_hash_create

Parametri

in	*nbuckets*	– number of buckets to create
in	*hash_function*	– pointer to the hashing function to be used
in	*hash_key_compare*	– pointer to the hash key comparison function to be used

Restituisce

pointer to new hash table.

5.7.3.2 int icl_hash_delete (icl_hash_t * *ht*, void * *key*, void(*)(void *) *free_key*, void(*)(void *) *free_data*)

Free one hash table entry located by key (key and data are freed using functions).

icl_hash_delete

Parametri

ht	– the hash table to be freed
key	– the key of the new item
free_key	– pointer to function that frees the key
free_data	– pointer to function that frees the data

Restituisce

0 on success, -1 on failure.

5.7.3.3 int icl_hash_destroy (icl_hash_t * *ht*, void(*)(void *) *free_key*, void(*)(void *) *free_data*)

Free hash table structures (key and data are freed using functions).

icl_hash_destroy

Parametri

ht	– the hash table to be freed
free_key	– pointer to function that frees the key
free_data	– pointer to function that frees the data

Restituisce

0 on success, -1 on failure.

5.7.3.4 void* icl_hash_find (icl_hash_t * *ht,* void * *key*)

Search for an entry in a hash table.

icl_hash_find

Parametri

ht	– the hash table to be searched
key	– the key of the item to search for

Restituisce

pointer to the data corresponding to the key. If the key was not found, returns NULL.

5.7.3.5 int icl_hash_insert (icl_hash_t * *ht,* void * *key,* void * *data*)

Insert an item into the hash table.

icl_hash_insert

Parametri

ht	– the hash table
key	– the key of the new item
data	– pointer to the new item's data

Restituisce

0 successo, -1 utente gia registrato, -2 inserimento fallito

5.7.3.6 int lock_hash (icl_hash_t * *ht*)

Prende la m.e. dell' intera tabella hash.

lock_hash

Parametri

ht	tabella hash

Restituisce

-1 fallimento, 0 successo

5.7.3.7 int lock_hash_section (icl_hash_t * *ht,* void * *key*)

Prende la m.e. della sezione della tabella relativa a key.

lock_hash_section

Parametri

ht	tabella hash
key	chiave appartenente alla sezione su cui prendere la mutua esclusione

Restituisce

-1 fallimento, 0 successo

5.7.3.8 int unlock_hash (icl_hash_t * *ht*)

Rilascia la m.e. dell' intera tabella hash.

unlock_hash

Parametri

ht	tabella hash

Restituisce

-1 fallimento, 0 successo

5.7.3.9 int unlock_hash_section (icl_hash_t * *ht*, void * *key*)

Rilascia la m.e. della sezione della tabella relativa a key.

unlock_hash_section

Parametri

ht	tabella hash
key	chiave appartenente alla sezione su cui rilasciare la mutua esclusione

Restituisce

-1 fallimento, 0 successo

5.8 Riferimenti per il file message.h

Contiene il formato del messaggio.

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

Grafo delle dipendenze di inclusione per message.h: Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

Strutture dati

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

5.8.1 Descrizione dettagliata

Contiene il formato del messaggio.

5.9 Riferimenti per il file ops.h

Contiene i codici delle operazioni di richiesta e risposta.

Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

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 }

5.9.1 Descrizione dettagliata

Contiene i codici delle operazioni di richiesta e risposta.

5.9.2 Documentazione dei tipi enumerati

5.9.2.1 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

5.10 Riferimenti per il file parser.c

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

Grafo delle dipendenze di inclusione per parser.c:

Funzioni

- int parsing (char *path, struct serverConf *conf)

  Parsing file di configurazione.

5.10.1 Descrizione dettagliata

Autore

Federico Germinario 545081

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

5.10.2 Documentazione delle funzioni

5.10.2.1 int parsing (char * *path*, struct serverConf * *conf*)

Parsing file di configurazione.

parsing

Parametri

path	path file da leggere
conf	puntatore alla struttura dati dove memorizzare i parametri letti

Restituisce

0 successo, -1 fallimento

5.11 Riferimenti per il file parser.h

Parser file di configurazione server.

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

Grafo delle dipendenze di inclusione per parser.h: Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

Strutture dati

- struct serverConf

  Parametri di configurazione del server.

Definizioni

- #define **MAX_LINESIZE** 1024

Funzioni

- int parsing (char *path, struct serverConf *conf)

  Parsing file di configurazione.

5.11.1 Descrizione dettagliata

Parser file di configurazione server.

Autore

Federico Germinario 545081

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

5.11.2 Documentazione delle funzioni

5.11.2.1 int parsing (char * *path*, struct serverConf * *conf*)

Parsing file di configurazione.

parsing

Parametri

path	path file da leggere
conf	puntatore alla struttura dati dove memorizzare i parametri letti

Restituisce

0 successo, -1 fallimento

5.12 Riferimenti per il file queue.c

File di implementazione dell'interfaccia per la coda.

```
#include <stdlib.h>
#include <assert.h>
#include <unistd.h>
#include <pthread.h>
#include <stdio.h>
#include <queue.h>
```

Grafo delle dipendenze di inclusione per queue.c:

Funzioni

- Queue_t * initQueue ()
- void deleteQueue (Queue_t *q)
- int push (Queue_t *q, void *data)
- void * pop (Queue_t *q)
- unsigned long length (Queue_t *q)

5.12.1 Descrizione dettagliata

File di implementazione dell'interfaccia per la coda.

5.12.2 Documentazione delle funzioni

5.12.2.1 void deleteQueue (Queue_t * *q*)

Cancella una coda allocata con initQueue. Deve essere chiamata da da un solo thread (tipicamente il thread main).

Parametri

q	puntatore alla coda da cancellare

5.12.2.2 Queue_t* initQueue ( )

Alloca ed inizializza una coda. Deve essere chiamata da un solo thread (tipicamente il thread main).

Parametri

nrow	numero righe
numero	colonne

Valori di ritorno

NULL	se si sono verificati problemi nell'allocazione (errno settato)
q	puntatore alla coda allocata

5.12.2.3 unsigned long length (Queue_t * *q*)

Ritorna la lunghezza della coda. L'accesso non è in mutua esclusione !

Valori di ritorno

lunghezza	della coda.

5.12.2.4 void* pop (Queue_t * *q*)

Estrae un dato dalla coda.

Valori di ritorno

data	puntatore al dato estratto.

5.12.2.5 int push (Queue_t * *q*, void * *data*)

Inserisce un dato nella coda.

Parametri

data	puntatore al dato da inserire

Valori di ritorno

0	se successo
-1	se errore (errno settato opportunamente)

5.13 Riferimenti per il file user.c

```
#include <stdio.h>
#include <string.h>
#include <pthread.h>
#include <stdlib.h>
#include <errno.h>
#include "icl_hash.h"
#include "user.h"
#include "config.h"
#include "util.h"
```

Grafo delle dipendenze di inclusione per user.c:

Definizioni

- #define **DEFAULT_NBUCKETS_HASH** 1024

Funzioni

- int add_user_online (users_db_t *users_db, char *name, int fd)

  Aggiunge un utente all' array degli utenti connessi in mutua esclusione.
- int delete_user_online (users_db_t *users_db, char *name)

  Elimina un utente dall' array degli utenti connessi in mutua esclusione.
- void free_data (void *user)

  Dealloca risorse del campo data della hash table.
- users_db_t * users_db_create (int nbuckets, int max_connections, int history_size)

  Inizializza le strutture dati del server.
- int users_db_destroy (users_db_t *users_db)

  Distrugge le strutture dati del server.
- int register_user (users_db_t *users_db, char *name)

  Registrazione utente.
- int unregister_user (users_db_t *users_db, char *name)

  Deregistrazione utente.
- int get_users_online (users_db_t *users_db, char **list_online)

  Lista utenti online.
- int connect_user (users_db_t *users_db, char *name, int fd)

  Connessione utente.
- int disconnect_user (users_db_t *users_db, char *name)

  Disconnesione utente utilizzando il suo nome.
- int disconnect_user_fd (users_db_t *users_db, int fd)

  Disconnesione utente utilizzando il suo file descriptor.
- user_t * get_user (users_db_t *users_db, char *name)

  Restituisce la struttura dell' utente name.
- history_t * history_sender (users_db_t *users_db, char *name)

  Restitisce la history di name.

5.13.1 Descrizione dettagliata

Autore

Federico Germinario 545081

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

5.13.2 Documentazione delle funzioni

5.13.2.1 int add_user_online (users_db_t * users_db, char * name, int fd)

Aggiunge un utente all' array degli utenti connessi in mutua esclusione.

Aggiunge un utente all' array degli utenti connessi in mutua escusione.

add_user_online

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente da inserire nell' array

Restituisce

0 successo, -1 fallimento

5.13.2.2 int connect_user (users_db_t * users_db, char * name, int fd)

Connessione utente.

connect_user

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente da connettere param fd file descriptor dell'utente

Restituisce

0 successo, -1 fallimento, -2 utente non registrato, -3 utente gia connesso

5.13.2.3 int delete_user_online (users_db_t * users_db, char * name)

Elimina un utente dall' array degli utenti connessi in mutua esclusione.

Elimina un utente dall' array degli utenti connessi in mutua escusione.

delete_user_online

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente da eliminare dall' array

Restituisce

0 successo, -1 fallimento

5.13.2.4 int disconnect_user (users_db_t * users_db, char * name)

Disconnesione utente utilizzando il suo nome.

disconnect_user

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente da disconnettere

Restituisce

0 successo, -1 fallimento, -2 utente non registrato, -3 utente gia disconnesso.

5.13.2.5 int disconnect_user_fd (users_db_t * *users_db*, int *fd*)

Disconnesione utente utilizzando il suo file descriptor.

disconnect_user_fd

Parametri

users_db	puntatore alla struttura dati del server
fd	file descriptor dell' utente da disconnettere

Restituisce

0 successo, -1 fallimento, -2 utente non registrato, -3 utente gia disconnesso.

5.13.2.6 void free_data (void * *user*)

Dealloca risorse del campo data della hash table.

free_data

Parametri

user	puntatore all' utente da deallocare

5.13.2.7 user_t* get_user (users_db_t * *users_db*, char * *name*)

Restituisce la struttura dell' utente name.

get_user

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente di cui si vuole recuperare la struttura

Restituisce

puntatore alla struttura dati dell' utente

5.13.2.8 int get_users_online (users_db_t * *users_db*, char ** *list_online*)

Lista utenti online.

get_users_online

Parametri

users_db	puntatore alla struttura dati del server
list_online	puntatore alla lista dove andare a scrivere i nomi degli utenti online

Restituisce

>= 0 numero utenti online, -1 fallimento

5.13.2.9 history_t* history_sender (users_db_t * *users_db,* char * *name*)

Restitisce la history di name.

history_sender

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente di cui si vuole recuperare la history

Restituisce

puntatore alla history

5.13.2.10 int register_user (users_db_t * *users_db,* char * *name*)

Registrazione utente.

register_user

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente da registrare

Restituisce

0 successo, -1 utente gia' registrato, -2 fallimento

5.13.2.11 int unregister_user (users_db_t * *users_db,* char * *name*)

Deregistrazione utente.

unregister_user

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente da deregistrare

Restituisce

0 successo, -1 utente non registrato, -2 fallimento

5.13.2.12 users_db_t* users_db_create (int *nbuckets,* int *max_connections,* int *history_size*)

Inizializza le strutture dati del server.

users_db_create

Parametri

nbuckets	dimensione tabella hash utenti registrati
maxconnections	numero massimo di connessioni gestite dal server
history_size	numero massimo di messaggi che il server 'ricorda' per ogni client

Restituisce

puntatore al nuovo users_db

5.13.2.13 int users_db_destroy (users_db_t * *users_db*)

Distrugge le strutture dati del server.

users_db_destroy

Parametri

users_db	puntatore alla struttura dati da distruggere

Restituisce

0 successo, -1 fallimento

5.14 Riferimenti per il file user.h

Libreria contenente la struttura dati del server e le relative funzioni per la sua gestione.

```
#include <stdio.h>
#include "icl_hash.h"
#include "config.h"
#include "history.h"
```

Grafo delle dipendenze di inclusione per user.h: Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

Strutture dati

- struct user_t
- struct user_online_t
- struct users_db_t

Funzioni

- int add_user_online (users_db_t *users_db, char *name, int fd)

  Aggiunge un utente all' array degli utenti connessi in mutua escusione.
- int delete_user_online (users_db_t *users_db, char *name)

  Elimina un utente dall' array degli utenti connessi in mutua escusione.
- users_db_t * users_db_create (int nbuckets, int max_connections, int history_size)

  Inizializza le strutture dati del server.
- int users_db_destroy (users_db_t *users_db)

  Distrugge le strutture dati del server.
- int register_user (users_db_t *users_db, char *name)

  Registrazione utente.
- int unregister_user (users_db_t *users_db, char *name)

Deregistrazione utente.

- int get_users_online (users_db_t *users_db, char **list_online)

  Lista utenti online.
- int connect_user (users_db_t *users_db, char *name, int fd)

  Connessione utente.
- int disconnect_user (users_db_t *users_db, char *name)

  Disconnesione utente utilizzando il suo nome.
- int disconnect_user_fd (users_db_t *users_db, int fd)

  Disconnesione utente utilizzando il suo file descriptor.
- user_t * get_user (users_db_t *users_db, char *name)

  Restituisce la struttura dell' utente name.
- history_t * history_sender (users_db_t *users_db, char *name)

  Restitisce la history di name.

5.14.1 Descrizione dettagliata

Libreria contenente la struttura dati del server e le relative funzioni per la sua gestione.

Autore

Federico Germinario 545081

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

5.14.2 Documentazione delle funzioni

5.14.2.1 int add_user_online (users_db_t * *users_db,* char * *name,* int *fd*)

Aggiunge un utente all' array degli utenti connessi in mutua escusione.

add_user_online

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente da inserire nell' array

Restituisce

0 successo, -1 fallimento

Aggiunge un utente all' array degli utenti connessi in mutua escusione.

add_user_online

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente da inserire nell' array

Restituisce

0 successo, -1 fallimento

5.14.2.2 int connect_user (users_db_t * *users_db,* char * *name,* int *fd*)

Connessione utente.

connect_user

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente da connettere param fd file descriptor dell'utente

Restituisce

0 successo, -1 fallimento, -2 utente non registrato, -3 utente gia connesso

5.14.2.3 int delete_user_online (users_db_t * *users_db*, char * *name*)

Elimina un utente dall' array degli utenti connessi in mutua escusione.

delete_user_online

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente da eliminare dall' array

Restituisce

0 successo, -1 fallimento

Elimina un utente dall' array degli utenti connessi in mutua escusione.

delete_user_online

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente da eliminare dall' array

Restituisce

0 successo, -1 fallimento

5.14.2.4 int disconnect_user (users_db_t * *users_db*, char * *name*)

Disconnesione utente utilizzando il suo nome.

disconnect_user

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente da disconnettere

Restituisce

0 successo, -1 fallimento, -2 utente non registrato, -3 utente gia disconnesso.

5.14.2.5 int disconnect_user_fd (users_db_t * *users_db*, int *fd*)

Disconnesione utente utilizzando il suo file descriptor.

disconnect_user_fd

Parametri

users_db	puntatore alla struttura dati del server
fd	file descriptor dell' utente da disconnettere

Restituisce

0 successo, -1 fallimento, -2 utente non registrato, -3 utente gia disconnesso.

5.14.2.6 user_t* get_user (users_db_t * *users_db*, char * *name*)

Restituisce la struttura dell' utente name.

get_user

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente di cui si vuole recuperare la struttura

Restituisce

puntatore alla struttura dati dell' utente

5.14.2.7 int get_users_online (users_db_t * *users_db*, char ** *list_online*)

Lista utenti online.

get_users_online

Parametri

users_db	puntatore alla struttura dati del server
list_online	puntatore alla lista dove andare a scrivere i nomi degli utenti online

Restituisce

>= 0 numero utenti online, -1 fallimento

5.14.2.8 history_t* history_sender (users_db_t * *users_db*, char * *name*)

Restitisce la history di name.

history_sender

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente di cui si vuole recuperare la history

Restituisce

puntatore alla history

5.14.2.9 int register_user (users_db_t * *users_db*, char * *name*)

Registrazione utente.

register_user

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente da registrare

Restituisce

0 successo, -1 utente gia' registrato, -2 fallimento

5.14.2.10 int unregister_user (users_db_t * *users_db*, char * *name*)

Deregistrazione utente.

unregister_user

Parametri

users_db	puntatore alla struttura dati del server
name	nome utente da deregistrare

Restituisce

0 successo, -1 utente non registrato, -2 fallimento

5.14.2.11 users_db_t* users_db_create (int *nbuckets*, int *max_connections*, int *history_size*)

Inizializza le strutture dati del server.

users_db_create

Parametri

nbuckets	dimensione tabella hash utenti registrati
maxconnections	numero massimo di connessioni gestite dal server
history_size	numero massimo di messaggi che il server 'ricorda' per ogni client

Restituisce

puntatore al nuovo users_db

5.14.2.12 int users_db_destroy (users_db_t * *users_db*)

Distrugge le strutture dati del server.

users_db_destroy

Parametri

users_db	puntatore alla struttura dati da distruggere

Restituisce

0 successo, -1 fallimento

5.15 Riferimenti per il file util.c

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

Grafo delle dipendenze di inclusione per util.c:

Funzioni

- void ∗ Malloc (size_t size)

  malloc con controllo dell'errore
- void ∗ Calloc (size_t nitems, size_t size)

  Calloc con controllo dell'errore.

5.15.1 Descrizione dettagliata

Autore

Federico Germinario 545081

5.15.2 Documentazione delle funzioni

5.15.2.1 void∗ Calloc (size_t *nitems,* size_t *size*)

Calloc con controllo dell'errore.

Calloc

Parametri

size	byte di memoria da allocare

Restituisce

puntatore al blocco di memoria allocato

5.15.2.2 void∗ Malloc (size_t *size*)

malloc con controllo dell'errore

Malloc

Parametri

size	byte di memoria da allocare

Restituisce

puntatore al blocco di memoria allocato

5.16 Riferimenti per il file util.h

Macro per gestione degli errori.

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

Grafo delle dipendenze di inclusione per util.h: Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

Definizioni

- #define **SYSCALL**(r, c, e) if((r=c)==-1) { perror(e);exit(errno); }
- #define **CHECK_MENO1**(r, c, e) if((r=c)==-1){ perror(e);return -1; }
- #define **MUTEX_BLOCK**(mtx, code)

Funzioni

- void * Malloc (size_t size)

  malloc con controllo dell'errore
- void * Calloc (size_t nitems, size_t size)

  Calloc con controllo dell'errore.

5.16.1 Descrizione dettagliata

Macro per gestione degli errori.

Autore

Federico Germinario 545081

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

5.16.2 Documentazione delle definizioni

5.16.2.1 #define MUTEX_BLOCK(*mtx, code*)

Valore:

```
pthread_mutex_lock(&(mtx));  \
    { code }                      \
    pthread_mutex_unlock(&(mtx));
```

5.16.3 Documentazione delle funzioni

5.16.3.1 void* Calloc (size_t *nitems,* size_t *size*)

Calloc con controllo dell'errore.

Calloc

Parametri

size	byte di memoria da allocare

Restituisce

puntatore al blocco di memoria allocato

5.16.3.2 void* Malloc (size_t *size*)

malloc con controllo dell'errore

Malloc

Parametri

size	byte di memoria da allocare

Restituisce

puntatore al blocco di memoria allocato

Indice analitico