DistributedMemory

1.0

Generato da Doxygen 1.5.8

Fri Feb 4 17:36:39 2011

Indice

1 Indice delle strutture dati **1**

1.1 Strutture dati 1

2 Indice dei file **3**

2.1 Elenco dei file 3

3 Documentazione delle classi **5**

3.1 Riferimenti per la struct block 5

3.1.1 Descrizione dettagliata 5

3.1.2 Documentazione dei campi 6

3.1.2.1 ID 6

3.1.2.2 pointer 6

3.1.2.3 b_acc 6

3.1.2.4 cl_ptr 6

3.1.2.5 succ 6

3.2 Riferimenti per la struct block_list 7

3.2.1 Descrizione dettagliata 7

3.2.2 Documentazione dei campi 7

3.2.2.1 ID 7

3.2.2.2 pointer 7

3.2.2.3 associated 7

3.2.2.4 succ 7

3.3 Riferimenti per la struct client_list 8

3.3.1 Descrizione dettagliata 8

3.3.2 Documentazione dei campi 8

3.3.2.1 cl_ID 8

3.3.2.2 valid 8

3.3.2.3 lock 8

3.3.2.4 succ 8

3.4 Riferimenti per la classe Library . . . 9
3.4.1 Descrizione dettagliata . . . 10
3.4.2 Documentazione dei costruttori e dei distruttori . . . 10
3.4.2.1 Library . . . 10
3.4.3 Documentazione delle funzioni membro . . . 10
3.4.3.1 find_server . . . 10
3.4.3.2 find_block . . . 10
3.4.3.3 create_socket . . . 10
3.4.3.4 client_connect . . . 11
3.4.3.5 read_config_file . . . 11
3.4.3.6 dm_init . . . 11
3.4.3.7 dm_block_map . . . 11
3.4.3.8 dm_block_unmap . . . 11
3.4.3.9 dm_block_update . . . 12
3.4.3.10 dm_block_write . . . 12
3.4.3.11 dm_block_wait . . . 12
3.4.4 Documentazione dei campi . . . 12
3.4.4.1 head . . . 12
3.5 Riferimenti per la classe Server . . . 13
3.5.1 Descrizione dettagliata . . . 14
3.5.2 Documentazione dei costruttori e dei distruttori . . . 14
3.5.2.1 Server . . . 14
3.5.3 Documentazione delle funzioni membro . . . 14
3.5.3.1 add_block . . . 14
3.5.3.2 find_block . . . 14
3.5.3.3 find_client . . . 14
3.5.3.4 add_client . . . 15
3.5.3.5 delete_client . . . 15
3.5.3.6 change_valid . . . 15
3.5.3.7 client_unblock . . . 15
3.5.3.8 port_control . . . 15
3.5.3.9 read_config_file . . . 15
3.5.3.10 block_map . . . 15
3.5.3.11 block_unmap . . . 16
3.5.3.12 block_update . . . 16
3.5.3.13 block_write . . . 16

3.5.3.14 block_wait . . . 16
3.5.4 Documentazione dei campi . . . 16
3.5.4.1 head . . . 16
3.6 Riferimenti per la struct server_list . . . 18
3.6.1 Descrizione dettagliata . . . 18
3.6.2 Documentazione dei campi . . . 18
3.6.2.1 ip_add . . . 18
3.6.2.2 port . . . 18
3.6.2.3 cl_sk . . . 18
3.6.2.4 connected . . . 19
3.6.2.5 block_head . . . 19
3.6.2.6 succ . . . 19

4 Documentazione dei file **21**
4.1 Riferimenti per il file client_library.h . . . 21
4.1.1 Documentazione delle definizioni . . . 22
4.1.1.1 BACKLOG . . . 22
4.1.1.2 MAX_LINE_LEN . . . 22
4.1.1.3 DIMBLOCK . . . 22
4.2 Riferimenti per il file server.h . . . 23
4.2.1 Documentazione delle definizioni . . . 24
4.2.1.1 BACKLOG . . . 24
4.2.1.2 MAX_LINE_LEN . . . 24
4.2.1.3 DIMBLOCK . . . 24
4.2.2 Documentazione delle variabili . . . 24
4.2.2.1 SERVER_IP_ADD . . . 24
4.2.2.2 SERVER_PORT . . . 24
4.2.2.3 config_file_path . . . 24
4.2.2.4 line . . . 24
4.2.2.5 m_acc . . . 24

Capitolo 1

Indice delle strutture dati

1.1 Strutture dati

Queste sono le strutture dati con una loro breve descrizione:

block (Struttura dati del blocco di memoria: block) 5
block_list (Struttura dati del blocco di memoria: block_list) 7
client_list (Struttura dati della lista di client associati al blocco: client_list) 8
Library (Classe Library) 9
Server (Classe Server) 13
server_list (Struttura dati della lista di server: server_list) 18

Capitolo 2

Indice dei file

2.1 Elenco dei file

Questo è un elenco di tutti i file con una loro breve descrizione:

client_library.h 21
server.h 23

Capitolo 3

Documentazione delle classi

3.1 Riferimenti per la struct block

Struttura dati del blocco di memoria: block.

`#include <server.h>`

Campi

- int ID

  ID del blocco allocato in memoria.

- void * pointer

  Puntatore alla prima cella di memoria in cui è memorizzato il blocco.

- pthread_mutex_t b_acc

  Semaforo per la gestione della mutua esclusione sul blocco.

- client_list * cl_ptr

  Puntatore alla testa della lista dei client connessi al server.

- block * succ

  Puntatore al blocco successivo.

3.1.1 Descrizione dettagliata

Struttura dati del blocco di memoria: block.

La struttura dati contiene l'ID del blocco allocato in memoria, il puntatore alla prima cella di memoria in cui è memorizzato il blocco, il semaforo per la gestione della mutua esclusione sul blocco, il puntatore alla testa della lista dei client connessi al server, il puntatore al blocco successivo.

3.1.2 Documentazione dei campi

3.1.2.1 int block::ID

ID del blocco allocato in memoria.

3.1.2.2 void* block::pointer

Puntatore alla prima cella di memoria in cui è memorizzato il blocco.

3.1.2.3 pthread_mutex_t block::b_acc

Semaforo per la gestione della mutua esclusione sul blocco.

3.1.2.4 client_list* block::cl_ptr

Puntatore alla testa della lista dei client connessi al server.

3.1.2.5 block* block::succ

Puntatore al blocco successivo.

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

- server.h

3.2 Riferimenti per la struct block_list

Struttura dati del blocco di memoria: block_list.

`#include <client_library.h>`

Campi

- int ID
  ID del blocco di memoria.
- void * pointer
  Puntatore alla prima cella di memoria in cui è memorizzato il blocco.
- bool associated
  Variabile per gestire l'associazione tra blocco di memoria e indirizzo di memoria locale.
- block_list * succ
  Puntatore al blocco successivo.

3.2.1 Descrizione dettagliata

Struttura dati del blocco di memoria: block_list.

La struttura dati contiene l'ID del blocco di memoria, il puntatore alla prima cella di memoria in cui è memorizzato il blocco, la variabile booleana per gestire l'associazione tra blocco di memoria e indirizzo di memoria locale, il puntatore al blocco successivo.

3.2.2 Documentazione dei campi

3.2.2.1 int block_list::ID

ID del blocco di memoria.

3.2.2.2 void* block_list::pointer

Puntatore alla prima cella di memoria in cui è memorizzato il blocco.

3.2.2.3 bool block_list::associated

Variabile per gestire l'associazione tra blocco di memoria e indirizzo di memoria locale.

3.2.2.4 block_list* block_list::succ

Puntatore al blocco successivo.

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

- client_library.h

3.3 Riferimenti per la struct client_list

Struttura dati della lista di client associati al blocco: client_list.

`#include <server.h>`

Campi

- int cl_ID
  Identificativo del client.
- bool valid
  Variabile per la gestione della validità del blocco associato al client.
- bool lock
  Variabile per la gestione del client in attesa che la copia locale diventi valida.
- client_list * succ
  Puntatore al client successivo connesso al server.

3.3.1 Descrizione dettagliata

Struttura dati della lista di client associati al blocco: client_list.

La struttura dati contiene l'identificativo del client, la variabile booleana per la gestione della validità del blocco associato al client, la variabile booleana per la gestione del client in attesa che la copia locale diventi valida, il puntatore al client successivo connesso al server.

3.3.2 Documentazione dei campi

3.3.2.1 int client_list::cl_ID

Identificativo del client.

3.3.2.2 bool client_list::valid

Variabile per la gestione della validità del blocco associato al client.

3.3.2.3 bool client_list::lock

Variabile per la gestione del client in attesa che la copia locale diventi valida.

3.3.2.4 client_list* client_list::succ

Puntatore al client successivo connesso al server.

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

- server.h

3.4 Riferimenti per la classe Library

Classe Library.

`#include <client_library.h>`

Membri pubblici

- Library ()

  Library(): costruttore della classe Library.

- void dm_init (char *config_file)

  void dm_init(char config_file): funzione pubblica che inizializza la libreria.*

- int dm_block_map (int ID, void *address)

  int dm_block_map(int ID, void address): funzione pubblica che stabilisce l'associazione tra blocco di memoria e indirizzo di memoria locale.*

- void dm_block_unmap (int ID)

  void dm_block_unmap(int ID): funzione pubblica che segnala al server che il client non è più interessato al blocco ID.

- int dm_block_update (int ID)

  int dm_block_update(int ID): funzione pubblica che aggiorna il contenuto della copia locale con il contenuto globale sul server.

- int dm_block_write (int ID)

  int dm_block_write(int ID): funzione pubblica che sincronizza il contenuto locale del blocco con il contenuto globale sul server.

- int dm_block_wait (int ID)

  int dm_block_wait(int ID): funzione pubblica che controlla la validità della copia locale del blocco.

Membri privati

- server_list * find_server (int id)

  *server_list * find_server(int id): funzione privata che restituisce il server contenente il blocco identificato da id.*

- block_list * find_block (int id, server_list *sl)

  *block_list * find_block(int id, server_list *sl): funzione privata che restituisce il blocco identificato da id.*

- void create_socket ()

  void create_socket(): funzione privata che crea i socket necessari per mettersi in connessione con i server.

- int client_connect (char *ip_add, int port, int cl_sk, server_list *sl)

  *int client_connect(char *ip_add, int port, int cl_sk, server_list *sl): funzione privata che effettua la connessione del client al server.*

- void read_config_file (char *config_file)

  void read_config_file(char config_file): funzione privata che legge il file di configurazione.*

Attributi privati

- server_list * head

  Puntatore alla testa della lista dei server.

3.4.1 Descrizione dettagliata

Classe Library.

La classe implementa le strutture dati e le funzioni di interfaccia richieste dalle specifiche. Utilizza, inoltre, diverse funzioni private per svolgere le operazioni necessarie ai metodi richiesti.

3.4.2 Documentazione dei costruttori e dei distruttori

3.4.2.1 Library::Library ()

Library(): costruttore della classe Library.

Il costruttore inizializza il puntatore alla testa della lista dei server a NULL.

3.4.3 Documentazione delle funzioni membro

3.4.3.1 server_list* Library::find_server (int *id*) [private]

server_list * find_server(int id): funzione privata che restituisce il server contenente il blocco identificato da id.

La funzione scorre la lista di server, contestualmente scorre la lista di blocchi appesi al server e restiuisce il puntatore al server che contiene il blocco che ha id come identificativo. Restituisce NULL se il server non è presente in lista.

3.4.3.2 block_list* Library::find_block (int *id*, server_list * *sl*) [private]

block_list * find_block(int id, server_list *sl): funzione privata che restituisce il blocco identificato da id.

La funzione scorre la lista di blocchi appesi al server puntato da sl e restituisce il puntatore al blocco identificato da id. Restituisce NULL se il blocco non è presente in lista.

3.4.3.3 void Library::create_socket () [private]

void create_socket(): funzione privata che crea i socket necessari per mettersi in connessione con i server.

La funzione scorre la lista di server e crea un socket per ogni elemento presente in lista.

3.4.3.4 int Library::client_connect (char * *ip_add*, int *port*, int *cl_sk*, server_list * *sl*) [private]

int client_connect(char *ip_add, int port, int cl_sk, server_list *sl): funzione privata che effettua la connessione del client al server.

La funzione inizializza le strutture dati necessarie , effettua la connessione verso il server avente indirizo IP ip_add e porta port, setta la variabile booleana connected a 1. La funzione ritorna 1 in caso di successo e -1 in caso di errore.

3.4.3.5 void Library::read_config_file (char * *config_file*) [private]

void read_config_file(char* config_file): funzione privata che legge il file di configurazione.

La funzione legge il contenuto del file di configurazione, crea la lista di server in base al numero di righe presenti nel file effettuando l'inserimento di ogni nuovo elemento in testa, per ogni elemento della lista di server crea una lista di blocchi associati al server in base ai blocchi presenti nel file di configurazione effettuando l'inserimento di ogni nuovo elemento in testa, inizializza le variabili della struttura relativa al blocco per ogni elemento della lista di blocchi.

3.4.3.6 void Library::dm_init (char * *config_file*)

void dm_init(char* config_file): funzione pubblica che inizializza la libreria.

La funzione carica dal file config_file l'associazione tra ID dei blocchi e server richiamando la read_-config_file, apre una serie di socket per permettere al client di mettersi in connessione con i server richiamando la create_socket.

3.4.3.7 int Library::dm_block_map (int *ID*, void * *address*)

int dm_block_map(int ID, void* address): funzione pubblica che stabilisce l'associazione tra blocco di memoria e indirizzo di memoria locale.

La funzione trova il server che contiene il blocco ID richiamando la find_server, trova il blocco interessato richiamando la find_block, effettua un controllo sulla variabile booleana connected e se il client non risulta connesso al server, effettua la connessione richiamando la client_connect. Successivamente invia il comando relativo all'operazione dal svolgere al server, invia l'ID del blocco interessato al server, richiede la copia del blocco a partire dall'indirizzo address, segnala che il blocco è associato ad un indirizzo di memoria locale settando la variabile booleana associated a 1. La funzione restituisce 1 in caso di successo e -1 in caso di errore (non è possibile effettuare la connessione, il server non risponde, il blocco è gia associato ad un altro indirizzo di memoria locale).

3.4.3.8 void Library::dm_block_unmap (int *ID*)

void dm_block_unmap(int ID): funzione pubblica che segnala al server che il client non è più interessato al blocco ID.

La funzione trova il server che contiene il blocco ID richiamando la find_server, trova il blocco interessato richiamando la find_block, invia il comando relativo all'operazione dal svolgere al server, invia l'ID del blocco interessato al server, segnala che il blocco non è più associato ad un indirizzo di memoria locale resettando la variabile boolena associated a 0. La funzione fallisce se il blocco non era stato preventivamente associato ad un indirizzo di memoria locale.

3.4.3.9 int Library::dm_block_update (int *ID*)

int dm_block_update(int ID): funzione pubblica che aggiorna il contenuto della copia locale con il contenuto globale sul server.

La funzione trova il server che contiene il blocco ID richiamando la find_server, trova il blocco interessato richiamando la find_block, invia il comando relativo all'operazione dal svolgere al server, invia l'ID del blocco interessato al server, riceve il controllo sulla validità della copia locale del blocco dal server e se la copia locale risulta ancora valida non effettua alcuna copia, altrimenti, aggiorna il contenuto della copia locale con quello della copia globale. La funzione restituisce 1 in caso di successo e -1 in caso di errore (il server non risponde, il blocco non era stato associato ad alcun indirizzo di memoria locale).

3.4.3.10 int Library::dm_block_write (int *ID*)

int dm_block_write(int ID): funzione pubblica che sincronizza il contenuto locale del blocco con il contenuto globale sul server.

La funzione trova il server che contiene il blocco ID richiamando la find_server, trova il blocco interessato richiamando la find_block, invia il comando relativo all'operazione dal svolgere al server, invia l'ID del blocco interessato al server, riceve il controllo sulla validità della copia locale del blocco dal server e se la copia locale risulta valida sincronizza il contenuto locale del blocco con il contenuto globale sul server. La funzione ritorna con errore (con codici di errori diversi) nel caso in cui il blocco non era stato associato ad alcun indirizzo di memoria locale, la copia locale del blocco risulta invalida, il server non risponde. La funzione restituisce 1 in caso di successo.

3.4.3.11 int Library::dm_block_wait (int *ID*)

int dm_block_wait(int ID): funzione pubblica che controlla la validità della copia locale del blocco.

La funzione trova il server che contiene il blocco ID richiamando la find_server, trova il blocco interessato richiamando la find_block, invia il comando relativo all'operazione dal svolgere al server, invia l'ID del blocco interessato al server, riceve il controllo sulla validità della copia locale del blocco dal server e se la copia locale risulta invalida ritorna immediatamente, altrimenti, resta in attesa di ricevere dal server il segnale che notifica che la copia locale è diventata invalida. La funzione restituisce 1 in caso di successo e -1 in caso di errore (il server non risponde, il blocco non era stato associato ad alcun indirizzo di memoria locale).

3.4.4 Documentazione dei campi

3.4.4.1 server_list* Library::head [private]

Puntatore alla testa della lista dei server.

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

- client_library.h

3.5 Riferimenti per la classe Server

Classe Server.

`#include <server.h>`

Membri pubblici

- Server ()

  Server(): costruttore della classe Server.

- int port_control (int port)

  int port_control(int port): funzione pubblica che effettua il controllo sulla porta associata al server.

- void read_config_file (const char *config_file)

  *read_config_file(const char *config_file): funzione pubblica che legge il file di configurazione.*

- int block_map (int sk)

  int block_map(int sk): funzione pubblica che stabilisce l'associazione tra client e blocco.

- int block_unmap (int sk)

  int block_unmap(int sk): funzione pubblica che cancella l'associazione tra client e blocco.

- int block_update (int sk)

  int block_update(int sk): funzione pubblica che invia il contenuto del blocco al client.

- int block_write (int sk)

  int block_write(int sk): funzione pubblica che aggiorna il contenuto della copia globale del blocco con quello della copia locale.

- int block_wait (int sk)

  int block_wait(int sk): funzione pubblica che controlla la validità della copia locale del blocco.

Membri privati

- void add_block (int id_block)

  void add_block(int id_block): funzione privata che crea e inserisce un nuovo blocco in lista.

- block * find_block (int id)

  *block * find_block(int id): funzione privata che restituisce il blocco identificato da id.*

- client_list * find_client (int client_id, block *bk_elem)

  *client_list * find_client(int client_id, block* bk_elem): funzione privata che restituisce il client identificato da client_id.*

- void add_client (int client_id, block *bk_elem)

  *void add_client(int client_id, block *bk_elem): funzione privata che crea e inserisce un nuovo client in lista.*

- void delete_client (int client_id, block *bk_elem)

*void delete_client(int client_id, block *bk_elem): funzione privata che elimina un client dalla lista di client.*

- void change_valid (int client_id, block *bk_elem)

  *void change_valid(int client_id, block *bk_elem): funzione privata che modifica la validità della copia locale del blocco.*

- int client_unblock (int client_id, block *bk_elem)

  *int client_unblock(int client_id, block *bk_elem): funzione privata che sblocca la wait.*

Attributi privati

- block * head

  Puntatore alla testa della lista di blocchi.

3.5.1 Descrizione dettagliata

Classe Server.

La classe server implementa le strutture dati e i metodi richiesti dalle specifiche lato server. Utilizza, inoltre, diverse funzioni private per svolgere le operazioni necessarie ai metodi richiesti.

3.5.2 Documentazione dei costruttori e dei distruttori

3.5.2.1 Server::Server ()

Server(): costruttore della classe Server.

Il costruttore inizializza il puntatore alla testa della lista di blocchi a NULL.

3.5.3 Documentazione delle funzioni membro

3.5.3.1 void Server::add_block (int *id_block*) [private]

void add_block(int id_block): funzione privata che crea e inserisce un nuovo blocco in lista.

La funzione crea la lista di blocchi effetuando l'inserimento di un nuovo elemento in testa alla lista, inizializza i parametri della struttura dati block e alloca lo spazio di memoria necessario.

3.5.3.2 block* Server::find_block (int *id*) [private]

block * find_block(int id): funzione privata che restituisce il blocco identificato da id.

La funzione scorre la lista di blocchi e restituisce il puntatore al blocco che ha id come identificativo. Restituisce NULL se il blocco identificato da id non è presente in lista.

3.5.3.3 client_list* Server::find_client (int *client_id*, block * *bk_elem*) [private]

client_list * find_client(int client_id, block* bk_elem): funzione privata che restituisce il client identificato da client_id.

La funzione scorre la lista di client appesa al blocco puntato da bk_elem e restituisce il puntatore al client che ha client_id come identificativo. Restituisce NULL se il client identificato da client_id non è presente in lista.

3.5.3.4 void Server::add_client (int *client_id*, block * *bk_elem*) [private]

void add_client(int client_id, block *bk_elem): funzione privata che crea e inserisce un nuovo client in lista.

La funzione crea la lista di client effettuando l'inserimento di un nuovo elemento in testa alla lista e inizializza i parametri della struttura dati client_list.

3.5.3.5 void Server::delete_client (int *client_id*, block * *bk_elem*) [private]

void delete_client(int client_id, block *bk_elem): funzione privata che elimina un client dalla lista di client.

La funzione scorre la lista di client appesa al blocco puntato da bk_elem, trova il client identificato da client_id e lo elimina dalla lista.

3.5.3.6 void Server::change_valid (int *client_id*, block * *bk_elem*) [private]

void change_valid(int client_id, block *bk_elem): funzione privata che modifica la validità della copia locale del blocco.

La funzione scorre la lista di client appesa al blocco puntato da bk_elem, trova il client identificato da client_id e pone la variabile booleana valid relativa al client pari a 0 (copia locale invalida).

3.5.3.7 int Server::client_unblock (int *client_id*, block * *bk_elem*) [private]

int client_unblock(int client_id, block *bk_elem): funzione privata che sblocca la wait.

La funzione scorre la lista di client appesa al blocco puntato da bk_elem, trova il client identificato da client_id e se questo risulta essere bloccato sulla wait (ovvero se ha la variabile booleana lock pari a 1), lo sblocca e resetta il booleano lock relativo al client a 0.

3.5.3.8 int Server::port_control (int *port*)

int port_control(int port): funzione pubblica che effettua il controllo sulla porta associata al server.

La funzione effettua il controllo sulla porta associata al server per impedire l'utilizzo di porte destinate ad altri scopi.

3.5.3.9 void Server::read_config_file (const char * *config_file*)

read_config_file(const char *config_file): funzione pubblica che legge il file di configurazione.

La funzione legge il contenuto del file di configurazione, ne memorizza le righe, controlla indirizzo IP e porta, e in caso di corrispondenza richiama la add_block.

3.5.3.10 int Server::block_map (int *sk*)

int block_map(int sk): funzione pubblica che stabilisce l'associazione tra client e blocco.

La funzione funzione riceve l'id del blocco interessato dal client identificato da sk, trova il blocco richiamando la find_block, stabilisce l'associazione blocco-client richiamando la add_client, invia il contenuto del blocco al client. Tutte le operazioni sul blocco sono effettuate in mutua esclusione. La funzione ritorna 1 in caso di successo e -1 in caso di errore.

3.5.3.11 int Server::block_unmap (int *sk*)

int block_unmap(int sk): funzione pubblica che cancella l'associazione tra client e blocco.

La funzione riceve l'id del blocco interessato dal client identificato da sk, trova il blocco richiamando la find_block, elimina l'associazione client-blocco richiamando la delete_client. Tutte le operazioni sul blocco sono effettuate in mutua esclusione. La funzione ritorna 1 in caso di successo e -1 in caso di errore.

3.5.3.12 int Server::block_update (int *sk*)

int block_update(int sk): funzione pubblica che invia il contenuto del blocco al client.

La funzione riceve l'id del blocco interessato dal client identificato da sk, trova il blocco richiamando la find_block, trova il client assocciato al blocco richiamando la find_client, effettua un controllo sulla validità della copia locale e manda il valore del booleano valid al client. Se la copia locale risulta ancora valida termina le operazioni, altrimenti manda il contenuto del blocco al client e segnala la copia locale come valida settando il valore della variabile booleana valid a 1. Tutte le operazioni sul blocco sono effettuate in mutua esclusione. La funzione ritorna 1 in caso di successo e -1 in caso di errore.

3.5.3.13 int Server::block_write (int *sk*)

int block_write(int sk): funzione pubblica che aggiorna il contenuto della copia globale del blocco con quello della copia locale.

La funzione riceve l'id del blocco interessato dal client identificato da sk, trova il blocco richiamando la find_block, trova il client associato al blocco richiamando la find_client, effettua un controllo sulla validità della copia locale e manda il valore della variabile booleana valid al client. Se la copia locale risulta invalida termina le operazioni, altrimenti aggiorna il contenuto della copia globale del blocco con quello della copia locale inviatagli dal client, rende la copia locale invalida richiamando la change_valid, invia al client il segnale di sbloccarsi sulla wait richiamando la client_unblock. Tutte le operazioni sul blocco sono effettuate in mutua esclusione. La funzione ritorna 1 in caso di successo e -1 in caso di errore.

3.5.3.14 int Server::block_wait (int *sk*)

int block_wait(int sk): funzione pubblica che controlla la validità della copia locale del blocco.

La funzione riceve l'id del blocco interessato dal client identificato da sk, trova il blocco richiamando la find_block, trova il client associato al blocco richiamando la find_client, manda al client il valore della variabile booleana valid e se quest ultimo risulta pari a 1, setta la variabile booleana lock a 1 segnalando che il client è bloccato sulla wait.

3.5.4 Documentazione dei campi

3.5.4.1 block* Server::head [private]

Puntatore alla testa della lista di blocchi.

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

- server.h

3.6 Riferimenti per la struct server_list

Struttura dati della lista di server: server_list.

`#include <client_library.h>`

Campi

- char ip_add [15]

  Indirizzo IP del server a cui si connette il client.

- int port

  Porta associata all'indirizzo IP.

- int cl_sk

  Descrittore del socket lato client.

- bool connected

  Variabile per la gestione della connessione client-server.

- block_list * block_head

  Puntatore alla testa della lista di blocchi associati ad ogni server.

- server_list * succ

  Puntatore all'elemento successivo della lista di server a cui il client è connesso.

3.6.1 Descrizione dettagliata

Struttura dati della lista di server: server_list.

La struttura dati contiene l'indirizzo IP del server a cui si connette il client, la porta associata all'indirizzo IP, il descrittore del socket lato client, la variabile booleana per la gestione della connessione client-server, il puntatore alla testa della lista di blocchi associati ad ogni server, il puntatore all'elemento successivo della lista dei server a cui il client si è connesso.

3.6.2 Documentazione dei campi

3.6.2.1 char server_list::ip_add[15]

Indirizzo IP del server a cui si connette il client.

3.6.2.2 int server_list::port

Porta associata all'indirizzo IP.

3.6.2.3 int server_list::cl_sk

Descrittore del socket lato client.

3.6.2.4 bool server_list::connected

Variabile per la gestione della connessione client-server.

3.6.2.5 block_list* server_list::block_head

Puntatore alla testa della lista di blocchi associati ad ogni server.

3.6.2.6 server_list* server_list::succ

Puntatore all'elemento successivo della lista di server a cui il client è connesso.

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

- client_library.h

Capitolo 4

Documentazione dei file

4.1 Riferimenti per il file client_library.h

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <stdlib.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <strings.h>
#include <pthread.h>
```

Strutture dati

- struct block_list

  Struttura dati del blocco di memoria: block_list.

- struct server_list

  Struttura dati della lista di server: server_list.

- class Library

  Classe Library.

Definizioni

- #define BACKLOG 5

  Numero massimo di richieste di connessione che si possono accodare.

- #define MAX_LINE_LEN 1024

Lunghezza massima di ogni riga del file di configurazione.

- #define DIMBLOCK 128∗sizeof(char)

  Dimensione del blocco di memoria in bytes.

4.1.1 Documentazione delle definizioni

4.1.1.1 #define BACKLOG 5

Numero massimo di richieste di connessione che si possono accodare.

4.1.1.2 #define MAX_LINE_LEN 1024

Lunghezza massima di ogni riga del file di configurazione.

4.1.1.3 #define DIMBLOCK 128∗sizeof(char)

Dimensione del blocco di memoria in bytes.

4.2 Riferimenti per il file server.h

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <stdlib.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <strings.h>
#include <pthread.h>
```

Strutture dati

- struct client_list

  Struttura dati della lista di client associati al blocco: client_list.

- struct block

  Struttura dati del blocco di memoria: block.

- class Server

  Classe Server.

Definizioni

- #define BACKLOG 5

  Numero massimo di richieste di connessione che si possono accodare.

- #define MAX_LINE_LEN 1024

  Lunghezza massima di ogni riga del file di configurazione.

- #define DIMBLOCK 128∗sizeof(char)

  Dimensione del blocco di memoria in bytes.

Variabili

- char ∗ SERVER_IP_ADD

  Indirizzo IP del server.

- int SERVER_PORT

  Porta associata all'indirizzo IP del server.

- const char ∗ config_file_path

Path del file di configurazione.

- char line [MAX_LINE_LEN]

  Stringa di appoggio per il contenuto del file di configurazione.

- pthread_mutex_t m_acc

  Semaforo di mutua esclusione.

4.2.1 Documentazione delle definizioni

4.2.1.1 #define BACKLOG 5

Numero massimo di richieste di connessione che si possono accodare.

4.2.1.2 #define MAX_LINE_LEN 1024

Lunghezza massima di ogni riga del file di configurazione.

4.2.1.3 #define DIMBLOCK 128∗sizeof(char)

Dimensione del blocco di memoria in bytes.

4.2.2 Documentazione delle variabili

4.2.2.1 char∗ SERVER_IP_ADD

Indirizzo IP del server.

4.2.2.2 int SERVER_PORT

Porta associata all'indirizzo IP del server.

4.2.2.3 const char∗ config_file_path

Path del file di configurazione.

4.2.2.4 char line[MAX_LINE_LEN]

Stringa di appoggio per il contenuto del file di configurazione.

4.2.2.5 pthread_mutex_t m_acc

Semaforo di mutua esclusione.

Indice analitico

add_block
    Server, 14
add_client
    Server, 15
associated
    block_list, 7

b_acc
    block, 6
BACKLOG
    client_library.h, 22
    server.h, 24
block, 5
    b_acc, 6
    cl_ptr, 6
    ID, 6
    pointer, 6
    succ, 6
block_head
    server_list, 19
block_list, 7
    associated, 7
    ID, 7
    pointer, 7
    succ, 7
block_map
    Server, 15
block_unmap
    Server, 16
block_update
    Server, 16
block_wait
    Server, 16
block_write
    Server, 16

change_valid
    Server, 15
cl_ID
    client_list, 8
cl_ptr
    block, 6
cl_sk
    server_list, 18
client_connect
    Library, 10
client_library.h, 21
    BACKLOG, 22
    DIMBLOCK, 22
    MAX_LINE_LEN, 22
client_list, 8
    cl_ID, 8
    lock, 8
    succ, 8
    valid, 8
client_unblock
    Server, 15
config_file_path
    server.h, 24
connected
    server_list, 18
create_socket
    Library, 10

delete_client
    Server, 15
DIMBLOCK
    client_library.h, 22
    server.h, 24
dm_block_map
    Library, 11
dm_block_unmap
    Library, 11
dm_block_update
    Library, 11
dm_block_wait
    Library, 12
dm_block_write
    Library, 12
dm_init
    Library, 11

find_block
    Library, 10
    Server, 14
find_client
    Server, 14
find_server
    Library, 10

head

Library, 12
Server, 16

ID
block, 6
block_list, 7
ip_add
server_list, 18

Library, 9
client_connect, 10
create_socket, 10
dm_block_map, 11
dm_block_unmap, 11
dm_block_update, 11
dm_block_wait, 12
dm_block_write, 12
dm_init, 11
find_block, 10
find_server, 10
head, 12
Library, 10
read_config_file, 11
line
server.h, 24
lock
client_list, 8

m_acc
server.h, 24
MAX_LINE_LEN
client_library.h, 22
server.h, 24

pointer
block, 6
block_list, 7
port
server_list, 18
port_control
Server, 15

read_config_file
Library, 11
Server, 15

Server, 13
add_block, 14
add_client, 15
block_map, 15
block_unmap, 16
block_update, 16
block_wait, 16
block_write, 16
change_valid, 15
client_unblock, 15
delete_client, 15
find_block, 14
find_client, 14
head, 16
port_control, 15
read_config_file, 15
Server, 14
server.h, 23
BACKLOG, 24
config_file_path, 24
DIMBLOCK, 24
line, 24
m_acc, 24
MAX_LINE_LEN, 24
SERVER_IP_ADD, 24
SERVER_PORT, 24
SERVER_IP_ADD
server.h, 24
server_list, 18
block_head, 19
cl_sk, 18
connected, 18
ip_add, 18
port, 18
succ, 19
SERVER_PORT
server.h, 24
succ
block, 6
block_list, 7
client_list, 8
server_list, 19

valid
client_list, 8