begin body

TITLE

AUTHOR
Version
CREATEDATE

Sommario

Table of contents

Indice delle strutture dati

Strutture dati

Queste sono le strutture dati con una loro breve descrizione:

analisiDiSessionePuntata 3
argomento 4
client_tag 4
node_tag 5
player 5
puntate_node 6
queue 7
sessioneDiGioco 7
sessioneDiPuntate 8
vincitore 8

Indice dei file

Elenco dei file

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

clientRoulette/client_header.h (Include tutte le definizioni per il client della Roulette) 9
common/common_header.h (Include tutte le definizioni comuni al progetto) 10
serverRoulette/croupier.h (Include la definizione del routine per il thread croupier) 20
serverRoulette/player.h (Include la definizione del routine per il thread player e i suoi sotto-thread) 21
serverRoulette/queue.h (Include la definizione della struttura dati queue) 22

Documentazione delle classi

Riferimenti per la struct analisiDiSessionePuntata

```
#include <common_header.h>
```

Campi

1 **queue elencoVincitori**

2 int **numeroPerdenti**
3 int **numeroVincitori**
4 int **stato**
5 pthread_mutex_t **mutex**
6 pthread_cond_t **attesaMessaggi**

Descrizione dettagliata

analisi_puntata_t

Contiene l'analisi per una giocata fatta da tutti i giocatori connessi. Viene riazzerata ad ogni puntata.

Parametri:

elencoVincitori Lista contenente tutti i vincitori della sessione di puntate corrente
numeroPerdenti Numero di perdenti nella sessione di puntate corrente
numeroVincitori Numero di vincitori nella sessione di puntate corrente
stato Indica se il croupier ha analizzato la sessione di puntate. 0 se deve ancora analizzare la sessione, 1 se ha terminato l'analisi.
mutex Mutex associato alla sessione di analisi
attesaMessaggi Condition variable che indica l'attesa di un player per poter intraprendere la fase di scambio messaggi di congratulazioni

---

La documentazione per questa struct è stata generata a partire dal seguente file:
7 common/**common_header.h**

Riferimenti per la struct argomento

```
#include <player.h>
```

Campi

8 **queue** * **listaPuntatePrivata**
9 int **clientFd**

Descrizione dettagliata

argomento

argomento del gestore delle puntate nel server.

Parametri:

listaPuntatePrivata Contiene la lista di puntate temporanea ricevuta dal gestore
clientFd Il socket di comunicazione con il client associato al thread player

---

La documentazione per questa struct è stata generata a partire dal seguente file:
10 serverRoulette/**player.h**

Riferimenti per la struct client_tag

```
#include <common_header.h>
```

Campi

11 struct sockaddr_in **clientData**
12 int **clientFd**

Descrizione dettagliata

client_t

Contiene i dettagli di connessione per un singolo client connesso al server

Parametri:

clientData porta e indirizzo del client
clientFd socket di comunicazione tra client e thread player

La documentazione per questa struct è stata generata a partire dal seguente file:
13 common/**common_header.h**

Riferimenti per la struct node_tag

```
#include <queue.h>
```

Campi

14 struct **node** * **next**

Descrizione dettagliata

node_tag

Un nodo di una lista. Rappresenta il caso base, ovvero un nodo contenente solo il puntatore al nodo successivo. Verrà utilizzato come "superclasse" per tutti gli altri tipi di nodo

Parametri:

next Puntatore al nodo successivo

La documentazione per questa struct è stata generata a partire dal seguente file:
15 serverRoulette/**queue.h**

Riferimenti per la struct player

```
#include <common_header.h>
```

Campi

16 struct **node** * **next**
17 int **budgetPrecedente**
18 int **budgetAttuale**
19 char **nickname** [NICK_LENGTH]

20 in_port_t **portaMessaggiCongratulazioni**
21 **client_t** * **datiConnessioneClient**
22 **queue elencoPuntate**
23 int **vincitore**

Descrizione dettagliata

player_t

Nodo della lista sessioneDiGioco::elencoGiocatori. Contiene i dettagli su un giocatore connesso al server

Parametri:

next Puntatore al successivo nodo nella lista
budgetPrecedente Budget del giocatore prima dell'analisi della sessione di puntate da parte del croupier
budgetAttuale Budget reale del giocatore, modificato durante l'analisi della sessione di puntate da parte del croupier
nickname Nickname del giocatore
portaMessaggiCongratulazioni Porta sulla quale il client del giocatore è posto in ascolto per accettare i messaggi di congratulazione
datiConnessioneClient Contiene i dati di connessione del client, porta e indirizzo IP
elencoPuntate Lista contenente tutte le puntate del giocatore per la sessione corrente
vincitore Indica se il giocatore ha vinto la puntata corrente; 1 se ha vinto, 0 se ha perso

---

La documentazione per questa struct è stata generata a partire dal seguente file:
24 common/**common_header.h**

Riferimenti per la struct puntate_node

```
#include <common_header.h>
```

Campi

25 struct **node** * **next**
26 int **numeroPuntato**
27 int **tipoPuntata**
28 int **sommaPuntata**

Descrizione dettagliata

puntata_t

Nodo della lista puntate. Contiene i dettagli della puntata.

Parametri:

next Puntatore al successivo nodo nella lista
numeroPuntato Se la puntata è di tipo numerico, contiene il numero puntato dal giocatore
tipoPuntata Contiene il tipo di puntata. Può assumere i valori:
29 -1 -> Dispari;
30 -2 -> Pari;
31 >=0 <=36 -> Numero

sommaPuntata Somma puntata dal giocatore

La documentazione per questa struct è stata generata a partire dal seguente file:
32 common/**common_header.h**

Riferimenti per la struct queue

`#include <queue.h>`

Campi

33 **node** * **head**
34 **node** * **tail**

Descrizione dettagliata

queue

Struttura dati contenente due nodi. Uno rappresenterà la testa, l'altro la coda della lista.

Parametri:

head Testa della coda
tail Coda

La documentazione per questa struct è stata generata a partire dal seguente file:
35 serverRoulette/**queue.h**

Riferimenti per la struct sessioneDiGioco

`#include <common_header.h>`

Campi

36 **queue elencoGiocatori**
37 pthread_mutex_t **mutex**
38 pthread_cond_t **attesaRiempimentoListaPuntate**
39 pthread_cond_t **attesaAlmenoUnGiocatore**
40 int **giocatoriConnessi**
41 int **giocatoriChePuntano**

Descrizione dettagliata

sessione_gioco_t

Contiene le informazioni su una sessione di gioco della Roulette In ogni momento dell'esecuzione ne esiste una ed una sola istanza chiamata sessioneGiocoCorrente

Parametri:

elencoGiocatori Lista contenente tutti i giocatori connessi
mutex Mutex associato alla sessione di gioco
attesaRiempimentoListaPuntate Condition variable che indica l'attesa del croupier per il riempimento

della lista puntate da parte dei player
attesaAlmenoUnGiocatore Condition variable che indica l'attesa da parte del croupier per la connessione di un numero minimo di giocatori
giocatoriConnessi Numero di giocatori connessi al server
giocatoriChePuntano Numero di giocatori che partecipano alla puntata corrente

---

La documentazione per questa struct è stata generata a partire dal seguente file:
42 common/**common_header.h**

Riferimenti per la struct sessioneDiPuntate

```
#include <common_header.h>
```

Campi

43 pthread_mutex_t **mutex**
44 pthread_cond_t **aperte**
45 pthread_cond_t **chiuse**
46 pthread_cond_t **attesaCroupier**
47 int **stato**

Descrizione dettagliata

sessione_puntate_t

Contiene le informazioni sullo stato delle puntate della sessione corrente. In ogni momento dell'esecuzione ne esiste una ed una sola istanza chiamata sessionePuntateCorrente

Parametri:

mutex Mutex associato alla sessione delle puntate
aperte Condition variable che indica l'attesa per l'apertura delle puntate
chiuse Condition variable che indica l'attesa per la chiusura delle puntate
attesaCroupier Condition variable utilizzata per simulare lo scadere del tempo
stato Informa sullo stato delle puntate. 1 significa puntate aperte, 0 significa puntate chiuse

---

La documentazione per questa struct è stata generata a partire dal seguente file:
48 common/**common_header.h**

Riferimenti per la struct vincitore

```
#include <common_header.h>
```

Campi

49 struct **node** * **next**
50 int **portaMessaggiCongratulazioni**
51 struct sockaddr_in **indirizzoIp**

Descrizione dettagliata

vincitore_t

Elemento della lista vincitori.

Parametri:

next Puntatore al successivo nodo nella lista
portaMessaggiCongratulazioni Contiene la porta su cui il client associato è in ascolto per i messaggi di congratulazioni
indirizzoIp Contiene l'indirizzo del client associato

La documentazione per questa struct è stata generata a partire dal seguente file:
52 common/**common_header.h**

Documentazione dei file

Riferimenti per il file clientRoulette/client_header.h

Include tutte le definizioni per il client della Roulette.

Funzioni

53 int **parse_bet** (char *puntataStr, int *sommaPuntata, int *tipoPuntata, int *numeroPuntato)
54 char * **tipoPuntataTestuale** (int tipo)

Descrizione dettagliata

Include tutte le definizioni per il client della Roulette.

Gruppo 7

Data:

Gennaio 2011

Documentazione delle funzioni

int parse_bet (char * *puntataStr*, int * *sommaPuntata*, int * *tipoPuntata*, int * *numeroPuntato*)

parse_bet

Analizza una puntata. Riconosce se è stata fatta una puntata di tipo Pari/Dispari oppure su un numero

Parametri:

puntataStr L'input dell'utente da linea di comando. Deve essere uno di questi tre tipi:
55 D:<X>;
56 P:<X>;
57 <X>:<Y>

sommaPuntata [out] Somma puntata dal giocatore
tipoPuntata [out] Tipo di puntata effettuata

numeroPuntato [out] Numero puntato dal giocatore

Restituisce:

parse_bet

Analizza una puntata. Riconosce se è stata fatta una puntata di tipo Pari/Dispari oppure su un numero

char* tipoPuntataTestuale (int *tipo*)

tipoPuntataTestuale

Restituisce la stringa relativa al tipo di puntata

Parametri:

tipo Può assumere i valori:
58 -1 -> Dispari
59 -2 -> Pari
60 >=0 <=36 -> Numerico

Restituisce:

La stringa corrispondente al tipo di puntata

Riferimenti per il file common/common_header.h

Include tutte le definizioni comuni al progetto.

```
#include <arpa/inet.h>
#include <ctype.h>
#include <errno.h>
#include <fcntl.h>
#include "../serverRoulette/player.h"
#include <pthread.h>
#include "../serverRoulette/queue.h"
#include <signal.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <wait.h>
#include "../serverRoulette/player.h"
```

Strutture dati

61 struct **client_tag**
62 struct **sessioneDiPuntate**
63 struct **sessioneDiGioco**
64 struct **player**

struct **puntate_node**
struct **vincitore**
struct **analisiDiSessionePuntata**

Definizioni

#define **MAXBUF** 4096
#define **MAX_BUDGET** 500
#define **NICK_LENGTH** 100
#define **IP_ADDRESS_LENGTH** 15

Ridefinizioni di tipo (typedef)

typedef struct **client_tag client_t**
typedef struct **sessioneDiPuntate sessione_puntate_t**
typedef struct **sessioneDiGioco sessione_gioco_t**
typedef struct **player player_t**
typedef struct **puntate_node puntata_t**
typedef struct **vincitore vincitore_t**
typedef struct **analisiDiSessionePuntata analisi_puntata_t**

Funzioni

void **err_abort** (int code, char *text)
int **open_socket** (struct sockaddr_in self, short int server_port)
void **gestisci_puntata_numero** (int estratto, **puntata_t** *puntata, **player_t** ***player**)
void **gestisci_puntata_pari** (int estratto, **puntata_t** *puntata, **player_t** ***player**)
void **gestisci_puntata_dispari** (int estratto, **puntata_t** *puntata, **player_t** ***player**)
void **aumenta_budget** (int moltiplicatore, **puntata_t** *puntata, **player_t** ***player**)
struct timespec **calcola_intervallo** (int intervallo)
int **Socket** (int domain, int type, int protocol)
void **Bind** (int sockfd, const struct sockaddr *addr, socklen_t addrlen)
void **Listen** (int sockfd, int backlog)
int **Accept** (int sockfd, struct sockaddr *addr, socklen_t *addrlen)
void **Connect** (int sockfd, const struct sockaddr *addr, socklen_t addrlen)
void **Close** (int fildes)
void * **Malloc** (size_t size)
void **Pthread_mutex_init** (pthread_mutex_t *mutex, const pthread_mutexattr_t *attr)
void **Pthread_cond_init** (pthread_cond_t *cond, const pthread_condattr_t *attr)
void **Pthread_create** (pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine)(void *), void *arg)
void **Pthread_cancel** (pthread_t thread)
void **Pthread_mutex_lock** (pthread_mutex_t *mutex)
void **Pthread_mutex_unlock** (pthread_mutex_t *mutex)
void **Pthread_cond_wait** (pthread_cond_t *cond, pthread_mutex_t *mutex)
int **Pthread_cond_timedwait** (pthread_cond_t *cond, pthread_mutex_t *mutex, const struct timespec *abstime)
void **Pthread_cond_broadcast** (pthread_cond_t *cond)
void **Pthread_cond_signal** (pthread_cond_t *cond)
ssize_t **Write** (int fd, const void *buf, size_t count)
ssize_t **Read** (int fd, void *buf, size_t count)

Variabili

sessione_gioco_t sessioneGiocoCorrente
sessione_puntate_t sessionePuntateCorrente
analisi_puntata_t analisiSessionePuntata
const char **messaggioPuntateAperte** []

109 ssize_t **lenMessaggioPuntateAperte**
110 const char **messaggioPuntateChiuse** []
111 ssize_t **lenMessaggioPuntateChiuse**
112 int **numeroMinimoGiocatori**

Descrizione dettagliata

Include tutte le definizioni comuni al progetto.

Gruppo 7

Data:

Gennaio 2011

Documentazione delle definizioni

#define IP_ADDRESS_LENGTH 15

Lunghezza di un indirizzo IP in formato ASCII

#define MAX_BUDGET 500

Massimo budget ammesso dal server //TODO integrarlo nella giocata

#define MAXBUF 4096

Massima lunghezza di un buffer di testo

#define NICK_LENGTH 100

Lunghezza massima di un nickname per un giocatore

Documentazione delle ridefinizioni di tipo (typedef)

typedef struct analisiDiSessionePuntata analisi_puntata_t

analisi_puntata_t

Contiene l'analisi per una giocata fatta da tutti i giocatori connessi. Viene riazzerata ad ogni puntata.

Parametri:

elencoVincitori Lista contenente tutti i vincitori della sessione di puntate corrente
numeroPerdenti Numero di perdenti nella sessione di puntate corrente
numeroVincitori Numero di vincitori nella sessione di puntate corrente
stato Indica se il croupier ha analizzato la sessione di puntate. 0 se deve ancora analizzare la sessione, 1 se ha terminato l'analisi.
mutex Mutex associato alla sessione di analisi
attesaMessaggi Condition variable che indica l'attesa di un player per poter intraprendere la fase di scambio messaggi di congratulazioni

typedef struct client_tag client_t

client_t

Contiene i dettagli di connessione per un singolo client connesso al server

Parametri:

clientData porta e indirizzo del client
clientFd socket di comunicazione tra client e thread player

typedef struct player player_t

player_t

Nodo della lista sessioneDiGioco::elencoGiocatori. Contiene i dettagli su un giocatore connesso al server

Parametri:

next Puntatore al successivo nodo nella lista
budgetPrecedente Budget del giocatore prima dell'analisi della sessione di puntate da parte del croupier
budgetAttuale Budget reale del giocatore, modificato durante l'analisi della sessione di puntate da parte del croupier
nickname Nickname del giocatore
portaMessaggiCongratulazioni Porta sulla quale il client del giocatore è posto in ascolto per accettare i messaggi di congratulazione
datiConnessioneClient Contiene i dati di connessione del client, porta e indirizzo IP
elencoPuntate Lista contenente tutte le puntate del giocatore per la sessione corrente
vincitore Indica se il giocatore ha vinto la puntata corrente; 1 se ha vinto, 0 se ha perso

typedef struct puntate_node puntata_t

puntata_t

Nodo della lista puntate. Contiene i dettagli della puntata.

Parametri:

next Puntatore al successivo nodo nella lista
numeroPuntato Se la puntata è di tipo numerico, contiene il numero puntato dal giocatore
tipoPuntata Contiene il tipo di puntata. Può assumere i valori:
113 -1 -> Dispari;
114 -2 -> Pari;
115 >=0 <=36 -> Numero

sommaPuntata Somma puntata dal giocatore

typedef struct sessioneDiGioco sessione_gioco_t

sessione_gioco_t

Contiene le informazioni su una sessione di gioco della Roulette In ogni momento dell'esecuzione ne esiste una ed una sola istanza chiamata sessioneGiocoCorrente

Parametri:

elencoGiocatori Lista contenente tutti i giocatori connessi
mutex Mutex associato alla sessione di gioco
attesaRiempimentoListaPuntate Condition variable che indica l'attesa del croupier per il riempimento della lista puntate da parte dei player
attesaAlmenoUnGiocatore Condition variable che indica l'attesa da parte del croupier per la connessione di un numero minimo di giocatori
giocatoriConnessi Numero di giocatori connessi al server
giocatoriChePuntano Numero di giocatori che partecipano alla puntata corrente

typedef struct sessioneDiPuntate sessione_puntate_t

sessione_puntate_t

Contiene le informazioni sullo stato delle puntate della sessione corrente. In ogni momento dell'esecuzione ne esiste una ed una sola istanza chiamata sessionePuntateCorrente

Parametri:

mutex Mutex associato alla sessione delle puntate
aperte Condition variable che indica l'attesa per l'apertura delle puntate
chiuse Condition variable che indica l'attesa per la chiusura delle puntate
attesaCroupier Condition variable utilizzata per simulare lo scadere del tempo
stato Informa sullo stato delle puntate. 1 significa puntate aperte, 0 significa puntate chiuse

typedef struct vincitore vincitore_t

vincitore_t

Elemento della lista vincitori.

Parametri:

next Puntatore al successivo nodo nella lista
portaMessaggiCongratulazioni Contiene la porta su cui il client associato è in ascolto per i messaggi di congratulazioni
indirizzoIp Contiene l'indirizzo del client associato

Documentazione delle funzioni

int Accept (int *sockfd*, struct sockaddr * *addr*, socklen_t * *addrlen*)

Accept

Wrapper function per la system call accept. Termina il programma se si verifica un errore

Parametri:

sockfd
addr
addrlen

void aumenta_budget (int *moltiplicatore*, puntata_t * *puntata*, player_t * *player*)

aumenta_budget

Aumenta il budget del giocatore di un dato moltiplicatore

Parametri:

moltiplicatore Moltiplicatore del budget
puntata Puntata //TODO possibile passare un int invece di tutta la puntata
player //TODO possibile passare il budget invece del giocatore

Aumenta il budget del giocatore, dato un moltiplicatore

Parametri:

moltiplicatore
puntata
player

void Bind (int *sockfd*, const struct sockaddr * *addr*, socklen_t *addrlen*)

Bind

Wrapper function per la system call bind. Termina il programma se si verifica un errore

Parametri:

sockfd
addr
addrlen

struct timespec calcola_intervallo (int *intervallo*) [read]

Calcola l'intervallo di attesa del croupier

Parametri:

intervallo

Restituisce:

la struttura timespec contenente il tempo di fine attesa

void Close (int *fildes*)

Close

Wrapper function per la system call close. Termina il programma se si verifica un errore

Parametri:

fildes

void Connect (int *sockfd*, const struct sockaddr * *addr*, socklen_t *addrlen*)

Connect

Wrapper function per la system call connect. Termina il programma se si verifica un errore

Parametri:

sockfd
addr
addrlen

void err_abort (int *code*, char * *text*)

err_abort

Stampa sullo standard error un messaggio contenente l'errore, il file che l'ha causato, la riga e la spiegazione.

Parametri:

code il codice d'errore (il valore di ritorno per le funzioni pthread, la variabile errno per le altre syscall)
text Una stringa che spiega l'errore

Stampa sullo standard error un messaggio contenente l'errore, il file che l'ha causato, la riga e la spiegazione.

Parametri:

code il codice d'errore (il valore di ritorno per le funzioni pthread, la variabile errno per le altre syscall)
text Una stringa che spiega l'errore

void gestisci_puntata_dispari (int *estratto*, puntata_t * *puntata*, player_t * *player*)

gestisci_puntata_dispari

Gestisce una puntata di tipo dispari, aumentando il budget del giocatore nel caso in cui sia

vincente.

Parametri:

estratto Numero estratto nella sessione di gioco corrente
puntata Puntata effettuata dal giocatore
player Giocatore che ha effettuato la puntata

Controlla se una puntata del tipo D:<N> è vincente

Parametri:

estratto
puntata
player

void gestisci_puntata_numero (int *estratto*, puntata_t * *puntata*, player_t * *player*)

gestisci_puntata_numero

Gestisce una puntata di tipo numerico, aumentando il budget del giocatore nel caso in cui sia vincente.

Parametri:

estratto Numero estratto nella sessione di gioco corrente
puntata Puntata effettuata dal giocatore
player Giocatore che ha effettuato la puntata

Controlla se una puntata del tipo <N>:<M> è vincente

Parametri:

estratto
puntata
player

void gestisci_puntata_pari (int *estratto*, puntata_t * *puntata*, player_t * *player*)

gestisci_puntata_pari

Gestisce una puntata di tipo pari, aumentando il budget del giocatore nel caso in cui sia vincente.

Parametri:

estratto Numero estratto nella sessione di gioco corrente
puntata Puntata effettuata dal giocatore
player Giocatore che ha effettuato la puntata

Controlla se una puntata del tipo P:<N> è vincente

Parametri:

estratto
puntata
player

void Listen (int *sockfd*, int *backlog*)

Listen

Wrapper function per la system call listen. Termina il programma se si verifica un errore

Parametri:

sockfd
addr
addrlen

void* Malloc (size_t *size*)

Malloc

Wrapper function per la system call malloc. Termina il programma se si verifica un errore

Parametri:

mutex
attr

int open_socket (struct sockaddr_in *self*, short int *server_port*)

open_socket

Apre un socket descriptor, gli assegna un nome e si mette in ascolto.

Parametri:

self struttura che conterrà le informazioni relative al server
server_port porta sulla quale mettere in ascolto il server

Apre un socket, e si mette in ascolto su di esso

Parametri:

self
server_port

Restituisce:

il socket aperto

void Pthread_cancel (pthread_t *thread*)

Pthread_cancel

Wrapper function per la system call pthread_cancel. Termina il programma se si verifica un errore

Parametri:

thread

void Pthread_cond_broadcast (pthread_cond_t * *cond*)

Phtread_cond_broadcast

Wrapper function per la system call pthread_cond_broadcast. Termina il programma se si verifica un errore

Parametri:

cond

void Pthread_cond_init (pthread_cond_t * *cond*, const pthread_condattr_t * *attr*)

Pthread_cond_init

Wrapper function per la system call pthread_cond_init. Termina il programma se si verifica un errore

Parametri:

cond
attr

void Pthread_cond_signal (pthread_cond_t * *cond*)

Pthread_cond_signal

Wrapper function per la system call pthread_cond_signal. Termina il programma se si verifica un errore

Parametri:

cond

int Pthread_cond_timedwait (pthread_cond_t * *cond*, pthread_mutex_t * *mutex*, const struct timespec * *abstime*)

Pthread_cond_timedwait

Wrapper function per la system call pthread_cond_timedwait. Termina il programma se si verifica un errore

Parametri:

cond

void Pthread_cond_wait (pthread_cond_t * *cond*, pthread_mutex_t * *mutex*)

Pthread_cond_wait

Wrapper function per la system call pthread_cond_wait. Termina il programma se si verifica un errore

Parametri:

cond
mutex

void Pthread_create (pthread_t * *thread*, const pthread_attr_t * *attr*, void *(*)(void *) *start_routine*, void * *arg*)

Pthread_create

Wrapper function per la system call pthread_create. Termina il programma se si verifica un errore

Parametri:

thread

void Pthread_mutex_init (pthread_mutex_t * *mutex*, const pthread_mutexattr_t * *attr*)

Pthread_mutex_init

Wrapper function per la system call pthread_mutex_init. Termina il programma se si verifica un errore

Parametri:

thread
attr
start_routine
arg

void Pthread_mutex_lock (pthread_mutex_t * *mutex*)

Pthread_mutex_lock

Wrapper function per la system call pthread_mutex_lock. Termina il programma se si verifica un errore

Parametri:

mutex

void Pthread_mutex_unlock (pthread_mutex_t * *mutex*)

Pthread_mutex_unlock

Wrapper function per la system call pthread_mutex_unlock. Termina il programma se si verifica un errore

Parametri:

mutex

ssize_t Read (int *fd*, void * *buf*, size_t *count*)

Read

Wrapper function per la system call read. Termina il programma se si verifica un errore

Parametri:

fd
buf
count

Restituisce:

int Socket (int *domain*, int *type*, int *protocol*)

Socket

Wrapper function per la system call socket. Termina il programma se si verifica un errore

Parametri:

domain
type
protocol

Restituisce:

ssize_t Write (int *fd*, const void * *buf*, size_t *count*)

Write

Wrapper function per la system call write. Termina il programma se si verifica un errore

Parametri:

fd
buf
count

Restituisce:

Documentazione delle variabili

analisi_puntata_t analisiSessionePuntata

analisiSessionePuntata

Gestisce l'analisi della sessione di puntate corrente

ssize_t lenMessaggioPuntateAperte

lenMessaggioPuntateAperte

Dimensione del messaggio di puntate aperte

ssize_t lenMessaggioPuntateChiuse

lenMessaggioPuntateChiuse

Dimensione del messaggio di puntate chiuse

const char messaggioPuntateAperte[]

messaggioPuntateAperte

Messaggio che il thread player invia al client per indicare l'apertura delle puntate

Costanti

const char messaggioPuntateChiuse[]

messaggioPuntateChiuse

Messaggio che il thread player invia al client per indicare la chiusura delle puntate

int numeroMinimoGiocatori

numeroMinimoGiocatori

Numero minimo di giocatori connessi che il server attende per iniziare il gioco

sessione_gioco_t sessioneGiocoCorrente

sessioneGiocoCorrente

Gestisce la sessione di gioco corrente

sessione_puntate_t sessionePuntateCorrente

sessionePuntateCorrente

Gestisce la sessione di puntate corrente

Riferimenti per il file serverRoulette/croupier.h

Include la definizione del routine per il thread croupier.

Funzioni

116 void * **croupier** (void *arg)

Descrizione dettagliata

Include la definizione del routine per il thread croupier.

Gruppo 7

Data:
Gennaio 2011

Documentazione delle funzioni

void* croupier (void * *arg*)

croupier

Routine per il thread croupier

Parametri:
arg Argomento del croupier thread. //TODO inserire cosa ci sta

Restituisce:

Riferimenti per il file serverRoulette/player.h

Include la definizione del routine per il thread player e i suoi sotto-thread.
`#include "queue.h"`

Strutture dati

117 struct **argomento**

Ridefinizioni di tipo (typedef)

118 typedef struct **argomento argomento_gestore_puntate_t**

Funzioni

119 void * **player** (void *arg)
120 void * **gestorePuntateGiocatore** (void *arg)

Descrizione dettagliata

Include la definizione del routine per il thread player e i suoi sotto-thread.

Gruppo 7

Data:
Gennaio 2011

Documentazione delle ridefinizioni di tipo (typedef)

typedef struct argomento argomento_gestore_puntate_t

argomento

argomento del gestore delle puntate nel server.

Parametri:

listaPuntatePrivata Contiene la lista di puntate temporanea ricevuta dal gestore
clientFd Il socket di comunicazione con il client associato al thread player

---

Documentazione delle funzioni

void* gestorePuntateGiocatore (void * *arg*)

Parametri:

arg Argomento per il gestore di puntate. Di tipo argomento_gestore_puntate_t

Restituisce:

void* player (void * *arg*)

player

Routine per il thread player

Parametri:

arg Argomento del player thread. //TODO inserire cosa ci sta

Restituisce:

FUNZIONE player

==========

Riferimenti per il file serverRoulette/queue.h

Include la definizione della struttura dati queue.

Strutture dati

121 struct **node_tag**
122 struct **queue**

Ridefinizioni di tipo (typedef)

123 typedef struct **node_tag node**
124 typedef struct **queue queue**

Funzioni

125 void **queue_init** (**queue** *myroot)
126 void **queue_put** (**queue** *myroot, **node** *mynode)
127 **node** * **queue_get** (**queue** *myroot)

Descrizione dettagliata

Include la definizione della struttura dati queue.

Gruppo 7

Data:

Gennaio 2011

Documentazione delle ridefinizioni di tipo (typedef)

typedef struct node_tag node

node_tag

Un nodo di una lista. Rappresenta il caso base, ovvero un nodo contenente solo il puntatore al nodo successivo. Verrà utilizzato come "superclasse" per tutti gli altri tipi di nodo

Parametri:

next Puntatore al nodo successivo

typedef struct queue queue

queue

Struttura dati contenente due nodi. Uno rappresenterà la testa, l'altro la coda della lista.

Parametri:

head Testa della coda
tail Coda

Documentazione delle funzioni

node* queue_get (queue * *myroot*)

queue_get

Rimuove un nodo dalla coda

Parametri:

myroot Coda da cui rimuovere il nodo

Restituisce:

Puntatore al nodo rimosso

void queue_init (queue * *myroot*)

queue_init

Inizializza testa e coda a NULL

Parametri:

myroot queue da inizializzare

Queue init

void queue_put (queue * *myroot*, node * *mynode*)

queue_put

Inserisce un nodo nella coda

Parametri:

myroot Coda a cui aggiungere il nodo
mynode Nodo da aggiungere

Indice

INDEX