Relazione Progetto Sistemi Operativi 2016/2017

Davide Montagno Bozzone 535910 – Corso A

Indice

1. Visione Generale

La struttura generale del progetto è dotata di 4 parti:

1. Una struttura Hash (accesso diretto): Tale struttura è gestita tramite la hashFunction() che ne gestice l'accesso. Ogni posizione di tale struttura è resa in mutua esclusione, in modo tale da non bloccare l'intera struttura con un'unica sezione critica.
   1.1. Ha un puntatore alla coda utenti e due variabili:
      1.1.1. Mutex
      1.1.2. Cond
   1.2. Ha metodi per
      1.2.1. Gestire lo stato degli elementi inseriti
         1.2.1.1. Utente registrato
         1.2.1.2. Cercare un utente

2. Una coda per gli utenti
   2.1. Coda utenti registrati
      2.1.1. Puntatore inizio
      2.1.2. Puntatore Fine
   2.2. Utente singolo
      2.2.1. La struttura dell'utente singolo è simile alla 2.1 con la differenza che vi è una **history** (Realizzata come lista di messaggi).
      2.2.2. Nome
   2.3. Messaggio
   2.4. Variabili di Gestione
3. Una coda per i gruppi

3.1. Mutex per mutua esclusione

4. Un array di utenti online

1.1. Organizzazione codice

param.c/param.h: All'interno di questi file (di cui uno è di interfaccia) vi è il codice per poter settare il server in modo corretto.

queue.c/queue.h: All'interno di questi file (di cui uno è di interfaccia) vi è il codice per poter salvare tutte le informazioni relative agli utenti, singolo utente e messaggi.

hash.c/hash.h: Creato per gestire una mutua esclusione più corretta.

fd_queue.c/fd_queue.h: Per una maggiore efficienza nella gestione dei dati per i clienti online. In questi due file (di cui uno di interfaccia) vengono memorizzati tutti gli fd degli utenti online, in modo tale da poter inviare direttamente le richieste a quest'ultimi se necessario.

groups.c/groups.h: File per l'implementazione dei gruppi.

connections.c/connections.h: File di protocollo per la comunicazione client server.

chatty.c: File contente le configurazioni iniziali del server, in modo tale che quest'ultimo inizi a gestire le richieste da parte dei thread. All'interno di esso vi è anche la funzione principale che mi permette di eseguire le richieste da parte dei thread e la gestione dei segnali.

test_nametest.c: Test per le strutture dati.

2. Protocollo di comunicazione

Il protocollo di comunicazione è implementato in connections.c
La comunicazione client-server comincia attraverso la funzione openConnection. Una volta stabilita la connessione un client può inviare una o più richieste al server. Ogni singola richiesta del client causerà la risposta, da parte del server, con una notifica di successo o fallimento. La struttura della richiesta del client è definta in message.h. La comunicazione termina quando il client o il server chiudono la connessione.
connections.h definisce tutte le funzioni necessarie per l'invio e la ricezione di una richiesta. In particolare:

1. sendRequest invia un message_t al file descriptor specificato
2. readMsg legge un message_t dal file descriptor specificato.

3. Gestione Multithread

Il server utilizza il thread main per accettare le connessioni in entrata tramite una select.

All'arrivo di una nuova richiesta, se non si è superato il limite di connessioni stabilito dal file di configurazione, il main accetta la richiesta e inserisce il file descriptor della connessione nella struttura works, in caso contrario risponde al client con una notifica di fallimento.
Il pool di thread quindi aspetta che la struttura works sia non vuota per prelevare in mutua esclusione un file descriptor e gestirne le richieste.
In più è stato utilizzato un thread per la gestione dei segnali.

4. Segnali

Come specificato alla fine della sezione 4, vi è un solo thread per la gestione dei segnali in attesa tramite **SIGWAIT.** I segnali gestiti sono:

1. SIGPIPE: Mascherato per tutta l'esecuzione, poichè una scrittura su un socket chiuso potrebbe bloccare l'intero server.
2. SIGUSR1
   2.1. Se nel file di configurazione è specificato il file di stastiche, il server scrive su tale file le stastiche raccolte fino al momento della richiesta tramite il segnale.
3. SIGTERM – SIGINT – SIGQUIT
   3.1. Viene resettata la variabile **cicle** definita nel main, la quale interrompe il loop nel main e esegue le istruzioni successive per la gestione di memory leak e la corretta terminazione dei thread creati

5. Gruppi

Le informazioni relative ai gruppi sono contenute nella struttura group_list. Per ogni gruppo essa salva il nome dello stesso, il nickname del proprietario, il numero di membri e la lista degli utenti che appartengono al quel gruppo.
Per evitare ambiguità, il nome del gruppo non può essere uguale al nickname di un utente.
Quando una richiesta coinvolge un gruppo, il server distribuisce la risposta alla richiesta ad ogni membro appartenente al gruppo.

Davide Montagno Bozzone