UNIVERSITÀ DI PISA

Relazione Chatterbox

Sistemi Operativi e Laboratorio 2017/2018

Docente: Massimo Torquati

Autore: Nicolò Maio

Indice

1. Struttura progetto

Il progetto chatterbox prevede l'implementazione del lato server di una chat in cui gli utenti siano in grado di registrarsi, inviare messaggi di testo, e file.

2. Scelte del progetto

Sono state utilizzate e implementate diverse strutture dati per portare a termine il progetto ovvero:

2.1 Tabella hash

Per gestire le varie registrazioni degli utenti ho utilizzato la tabella hash consigliata dal docente in classe ovvero icl_hash, le cui implementazioni sono presenti nel file icl_hash.c. La dimensione massima della tabella hash (MAX_DIM_HASH) è modificabile nel file config.h.

2.2 Struct history

La struct history è composta da due puntatori testa e fine, da un contatore msg_in_hist e da un puntatore a puntatore di message_t che definisce un array di struct di messaggi. Tale struttura è implementata nel file history.c e dichiarata in history.h

2.3 Vettore user_list

Il vettore user_list è usato per contenere le varie informazioni degli utenti online ovvero nome, fd e relativa variabile di mutua esclusione per poter inviare messaggi all'fd online.

Il vettore user_list e le sue relative funzioni sono stati implementati nel file user.c e dichiarati nel file user.h.

2.4 Coda richieste

L'implementazione della coda delle richieste dei vari client è stata effettuata seguendo le indicazioni, viste a lezione, di una semplice comunicazione produttore-consumatore. Su di essa fa la push degli fd delle varie richieste il thread listener e da essa fanno la pop i vari thread worker che eseguono le richieste dei client. La coda delle richieste è implementata nel file queue.c e dichiarata nel file queue.h.

2.5 Struct Config

La struttura Config è stata usata per contenere le varie informazioni lette al momento del parsing del file di configurazione chatty.conf1 o chatty.conf2 presenti nella cartella DATA. Essa è stata implementata nel file parset.c e dichiarata nel file parset.h.

3. Struttura del codice

3.1 Logica della divisione su più file e librerie

Il file chatty.c rappresenta il file centrale del progetto in cui è presente la funzione main del server. In esso sono presenti anche le implementazioni delle funzioni dei thread listener e worker. Inoltre, sono presenti le implementazioni delle funzioni dichiarate nel file mylib.h fra cui le varie funzioni usate per poter eseguire le richieste dei client. Per ogni struttura dati è stato implementato un file .c e una libreria .h in cui sono dichiarate le varie funzioni usate per modificare o leggere la struttura dati relativa. Inoltre, è stato implementato il file stats.c per poter implementare la lettura e la scrittura delle statistiche in mutua esclusione dichiarate nel file stats.h.

4 Iterazione fra i processi/threads

4.1 Protocolli di comunicazione

Per la comunicazione fra server e client ho implementato come richiesto dalla consegna le funzioni dichiarate in connections.h e implementate in connections.c

Per la comunicazione fra il thread listener e il pool di threads worker è stata utilizzata la coda di richieste. Il thread listener fa la push dei vari fd dei client che si connettono e da essa i worker fanno la pop per leggere gli fd e il messaggio ricevuto dal client per poi poter eseguire la richiesta. Una volta portata a termine la richiesta del client, il

suo fd viene rimandato al listener mediante pipe. Il listener dalla pipe legge e reinserisce l'fd letto nel set su cui lavora la select.

4.2 Protocollo di terminazione

Il protocollo di terminazione si avvia quando il server riceve un SIGINT o SIGTERM o SIGQUIT. Ricevendo uno dei segnali di terminazione, il thread listener fa la push del valore -1 nella coda di richieste. Così facendo tutti i worker vengono svegliati e terminano leggendo il -1. Nella parte del main in cui viene eseguito il protocollo di terminazione vengono fatte le join di tutti i thread, viene liberata la memoria con l'eliminazione delle varie strutture dati implementate e infine viene eliminato il file del socket ovvero chatty_socket.

4.3 Gestione della concorrenza

Tutte le operazioni fatte sulle strutture dati presenti nel progetto sono fatte in mutua esclusione per evitare problemi di concorrenza fra i vari thread worker e listener.

La tabella hash è divisa in più zone di mutua esclusione, il cui numero è modificabile nel file config.h modificando la variabile globale NUM_VAR_MUX_HASH. Il valore di tale variabile deve essere sempre un divisore della dimensione della tabella hash per poter far funzionare il sistema di moduli utilizzato per scorrere la tabella hash e per accedere ad una delle varie zone di mutua esclusione.

Per la struttura dati history di ogni utente è stata creata una variabile di mutua esclusione (pthread_mutex_t mia) inserita nella struttura user presente in chatty.c

usata per rappresentare ogni utente nella tabella hash. Tale variabile di mutua esclusione è stata utilizzata per svolgere ogni operazione su una history selezionata che sia operazione di scrittura o lettura.

Sulla struttura dati user_list ovvero sul vettore di utenti online è stata utilizzata, in user.c, una variabile di mutua esclusione dichiarata staticamente (mtx) la quale viene usata per poter leggere da user_list e modificare la user_list. Inoltre, essa viene usata quando si vuol inviare un messaggio ad un client online insieme alla variabile di mutua esclusione fd_on propria di ogni utente online.

Per quanto riguarda la struttura coda richieste, come già indicato, essa viene gestita come una coda su cui viene controllata una normale comunicazione produttore-consumatore. In mutua esclusione il produttore, ovvero il thread listener, inserisce gli fd. Mediante una variabile di condizionamento, il thread listener indica la presenza di una richiesta ad un singolo consumatore mediante chiamata di sistema signal. Oppure nel caso di terminazione, se il thread listener inserisce un -1, una chiamata di sistema broadcast risveglia tutti i thread in attesa. Se la coda è vuota i consumatori, ovvero i thread workers, aspettano in mutua esclusione sulla variabile di condizionamento finché non arriva una signal o una broadcast così da poter utilizzare l'fd letto dalla coda.

Infine, viene utilizzata una variabile di mutua esclusione per poter leggere e aggiornare le statistiche del server mediante le funzioni del file stats.c.

4.4 Gestione dei segnali

I segnali di terminazione SIGINT, SIGQUIT e SIGTERM vengono intercettati dal signalfd presente nel set da cui legge il listener mediante select e successivamente inserisce un -1 in coda per avviare protocollo di terminazione. Invece il segnale SIGUSR1 viene intercettato da un thread che attende mediante sigwait il suo arrivo per poter scrivere le statistiche sul file chatty_stats.txt presente nella cartella tmp.

5 Macchine su cui è stato testato il codice

Il progetto è stato testato su Ubuntu 16.04 lts e sulla macchina virtuale del corso.

6 Difficoltà incontrate

Fra le difficoltà riscontrate durante l'implementazione del progetto vi sono in particolare:

- Un problema che si verificava alcune volte durante l'esecuzione del test4 riguardava una lock sulla variabile di mutua esclusione del fd dell'utente online (fd_on). Tale variabile viene inizializzata correttamente dato che la pthread_mutex_lock su di essa restituisce 0.
- Un altro problema incontrato è stato capire come inviare l'elenco di nomi della user_list degli utenti online al client.

6.1 Semplificazioni

- Per risolvere il problema che si verificava durante l'esecuzione del test4 riguardante la lock (su fd_on), ho scelto di inviare alcuni messaggi di risposta specifici che davano errore prendendo solo la mutua esclusione sulla user_list. Queste varianti si possono notare nelle seconde versioni delle funzioni di user.c (InvioToUserOnline_2, InvioHeaderToUserNOnline_2 e Invio_Lista_Utenti_Online_2).
- Per inviare l'elenco dei nomi della user_list degli utenti online, ho optato per un invio sequenziale di più stringhe di dimensione MAX_NAME_LENGTH+1 come si può notare nella funzione Invio_Lista_Utenti_Online.

Un'altra semplificazione riguarda il file listener.c presente nell'archivio .tar ma lasciato vuoto. Il thread listener, come indicato in precedenza, è stato implementato nel file chatty.c. Infine, al momento del download di un file, se il nome di esso è già presente in tmp/chatty allora il file viene sovrascritto.