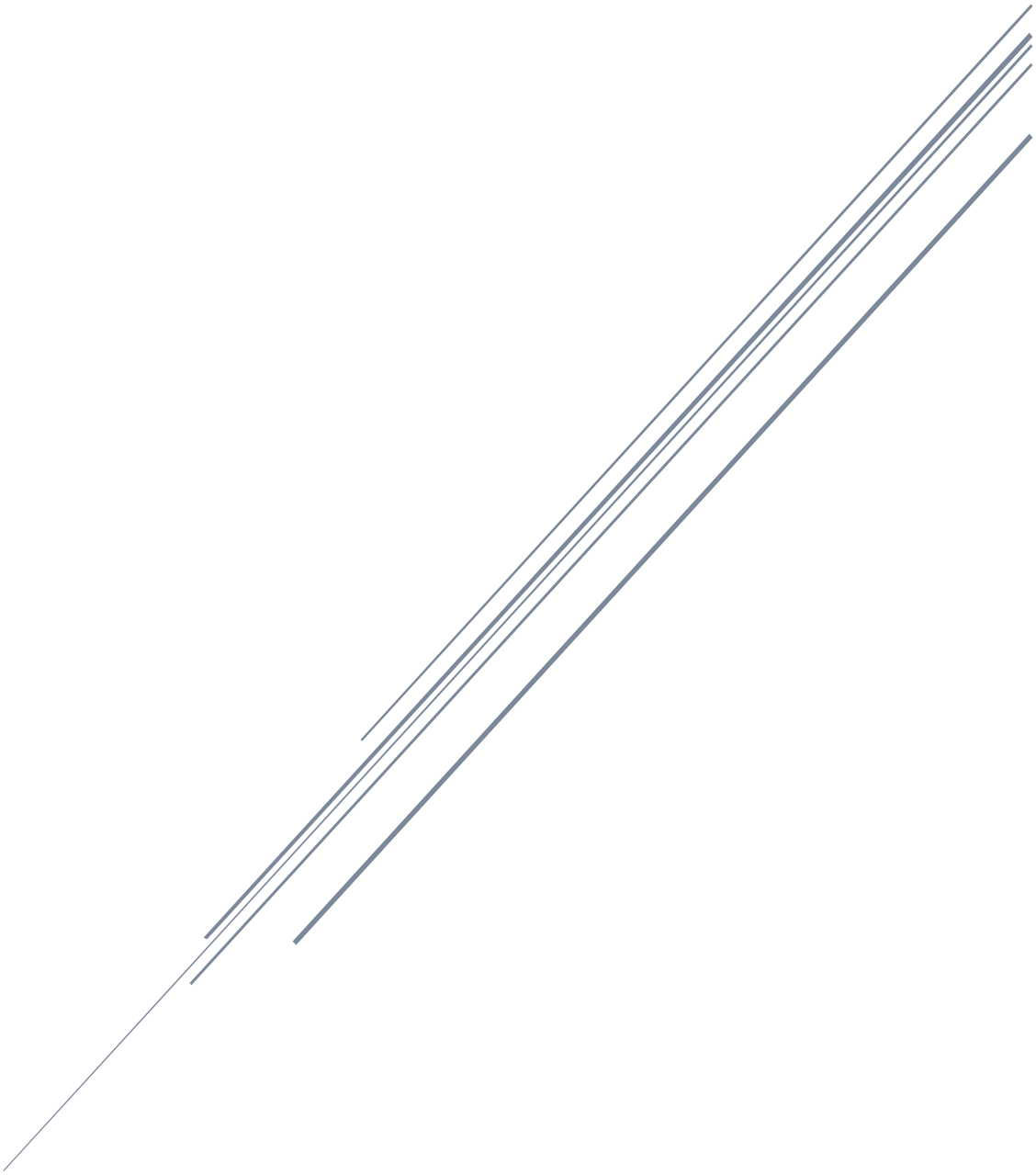
RELAZIONE PROGETTO CHATTY

Sistemi Operativi e Laboratorio 2017/2018

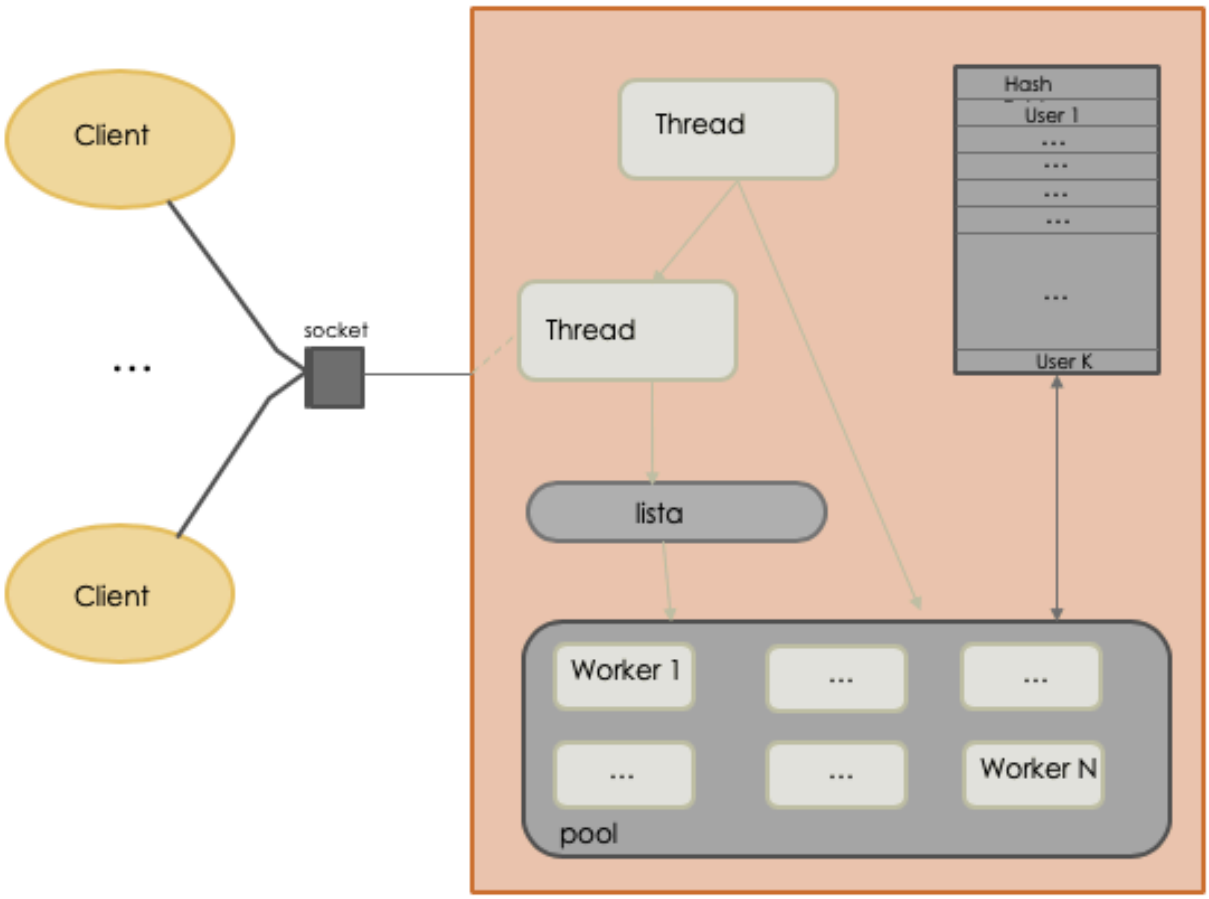


Lorenzo Arcidiacono
Corso B, Matr. 534235

Sommario

1. Architettura

Il sistema ha un'architettura di tipo Client-Server.
Il server è di tipo multi-thread in modo da poter gestire contemporaneamente più richieste da client diversi, mantenendo delle performance migliori rispetto ad un server sequenziale.
Di seguito è mostrata una semplificazione dello schema dell'architettura del server, dei suoi componenti e delle comunicazioni tra questi, che saranno approfonditi nei capitoli successivi.



2. Funzionalità dei Thread

Il processo, oltre al thread Main, ha anche un thread Master e un pool di thread Worker:

- Thread Main: viene lanciato all'inizio del processo, installa il gestore dei segnali, alloca e inizializza le strutture dati globali, fa il parsing del file di configurazione per settare le variabili di configurazione, inizializza il socket, crea il thread Master e il pool di thread Worker e, alla chiusura del server, si occupa di liberare la memoria;

- Thread Master: è il thread che si mette in ascolto sul socket tramite una select, aggiunge il file descriptor dei nuovi clients al set di quelli già presenti e scrive nella lista i descrittori di quelli che hanno richiesto delle operazioni;

- Pool di thread Worker: Ogni Worker estrae dalla lista il descrittore di una connessione e si occupa quindi di leggere la prima richiesta e portarla a termine; una volta eseguita rimette il file descriptor nel set e ne estrae un altro dalla lista.
  Ogni thread si occupa dunque di una sola richiesta del client e comunica a questi il risultato dell'operazione.

3. Gestione dei Segnali

All'avvio del processo il thread Main installa un gestore dei segnali personalizzato; in particolare, viene ignorato il SIGPIPE per evitare che il server possa essere chiuso dalla lettura di un file descriptor di un client terminato.
SIGINT, SIGTERM, SIGQUIT vengono gestiti in modo da non terminare immediatamente il processo, ma, settando delle apposite variabili, di terminare il thread Master e i Worker e dare modo al Main di liberare la memoria prima di terminare.
Alla ricezione di SIGUSR1 vengono stampate le statistiche come richiesto dalla specifica.

4. Comunicazione tra Processi e tra Thread

La comunicazione tra client e server avviene tramite un socket AF_UNIX, dal momento che lavorano sulla stessa macchina. Le comunicazioni avvengono attraverso lo scambio di messaggi codificati come *Message*.

La comunicazione tra thread, in particolare Master e Worker, avviene tramite l'inserimento da parte del Master dei file descriptor che hanno delle richieste nella lista; questi saranno consegnati a un thread del pool che eseguirà l'operazione richiesta.

5. Concorrenza

Essendo il server multi-thread si è reso necessaria un'attenta gestione della concorrenza.
Tutte le varibili globali che sono utilizzate in lettura e/o scrittura da più di un thread sono lette o modificate previa lock della relativa variabile di mutua esclusione.
Le funzioni di set e get della lista sono gestite tramite una variabile di mutua esclusione unica per tutta la lista e due variabili condizionali in caso di lista vuota o piena.
Più interessante è la gestione della mutua esclusione nel caso della tabella hash: per mantenere delle performance migliori del server è stata implementata una gestione "lettori e scrittori" di blocchi della tabella hash; nel caso di tabella hash piccola i blocchi corrispondo alle bucket in relazione uno a uno, nel caso di dimensioni più grandi i blocchi corrisponderanno a più bucket.
Questo permette a più thread di lavorare contemporaneamente sulla tabella mantenendone tuttavia la consistenza.

6. Strutture Dati

- Hash: tabella hash con liste di trabocco che viene usata come memoria di tutti gli utenti registrati, ogni entry della tabella ha come chiave un valore calcolato a partire dal nickname dell'utente tramite un'opportuna funzione hash, un elemento di tipo User in cui vengono salvate le informazioni dell'utente e un puntatore al successivo elemento della lista di trabocco;

- User: struttura utilizzata per salvare le informazioni di un particolare utente, il suo file descriptor, il suo status (online o offline) e la history con gli ultimi messaggi ricevuti. In particolar modo la history implementa un array di message_t con una gestione della memoria brutale, una volta arrivati alla massima dimensione della history si sovrascrivono i messaggi più vecchi;

- Lista: struttura di comunicazione tra il thread Master e i Worker, implementa un' array sincronizzato di interi, per gestire in maniera sicura la concorrenza;

- Set: fd_set dei file descriptor attivi è utilizzato dal Master tramite una select per capire quali Client hanno delle richieste. Poiché è utilizzato sia dal Master che dal Pool si è reso necessario l'utilizzo di una variabile di mutua esclusione per la lettura e la scrittura del Set;

- Configuration: struttura che salva in memoria i valori passati attraverso il file di configurazione;

- Statistics: struttura che mantiene le statistiche del server in maniera thread safe; queste statistiche vengono scritte sul file selezionato in fase di configurazione all'arrivo del SIGUSR1;

- Message: codifica del tipo di messaggi che vengono scambiati tra clients e server, suddivisa in header in cui è salvato il mittente e l'operazione, e in data in cui viene scritto il messaggio e il ricevente.

7. Gestione della Memoria

Per evitare possibili letture di memoria non inizializzata o sporca, per ogni allocazione di memoria sullo heap, viene eseguita una memset della memoria a 0.

Alla fine di ogni operazione si libera tutta la memoria che non verrà più utilizzata e alla chiusura del processo viene liberata la memoria allocata per le strutture dati globali.

8. Accesso ai Files

Per inizializzare il server, il file di configurazione passato come argomento viene parsato per poter settare nel modo corretto le variabili di configurazione.

Alla ricezione di SIGUSR1 il server scrive nel file scelto durante la configurazione, le statistiche delle operazioni avvenute fino a quel momento.

9. Divisione in File

Data la complessità del programma sono state create più librerie per poter separare le funzionalità richieste e le varie strutture dati:

- Parsing del file di configurazione e inizializzazione delle relative variabili;
- Gestione della comunicazione tra clients e server;
- Tabella hash degli utenti registrati;
- Lista per la comunicazione tra il thread master e il pool;
- Gestione del singolo utente;
- Gestione della stampa delle statistiche;
- Gestione dei messaggi;
- Operazioni disponibili ed eventuali risposte.

Ogni file '.c' chiama le librerie necessarie alle sue operazioni e svolge i compiti richiesti dal server.

10. Testing e Debugging

Il codice è stato testato su una macchina virtuale con installato Linux nella distribuzione Xubuntu fornita durante il corso.

In fase di debugging è stato usato principalmente il programma Valgrind per l'analisi della memoria; in maniera minore è stato usato anche il software GDB per l'analisi degli errori a tempo di esecuzione.

11. Difficoltà incontrate

È risultato particolarmente difficile la gestione di una mutua esclusione efficiente e consistente per la tabella hash, soprattutto nella gestione delle funzioni che hanno bisogno di leggere/scrivere tutti i blocchi della tabella.