Relazione Sistemi Operativi

Daniele Ferrarelli - Marco Ferri

1 Introduzione

La tesina assegnata richiede la realizzazione di un servizio "talk" via internet gestito tramite server per client residenti, in generale, su macchine diverse. I client possono connettersi per scambiare messaggi solo con altri che non sono già impegnati in un'altra conversazione. Un client oltre a iniziare una nuova conversazione deve essere in grado di scollegarsi da quella corrente e rendersi disponibile per nuove connessioni o per disconnettersi completamente dal servizio. Il tutto deve essere gestito da un server che si occupa delle connessioni e delle liste dei client attivi e di quelli disponibili. Il servizio è accessibile tramite un account, che viene creato in fase di registrazione.

2 Scelte di progettazione

La nostra realizzazione si è focalizzata sul rendere il client più semplice possibile e far gestire tutte le operazioni al server. Le connessioni tra client non sono dirette, ma avvengono tramite il server che ha il ruolo di "ponte" tra di essi. Ogni messaggio di una conversazione infatti non viene mandato direttamente al client destinatario, ma al server che lo inoltra ad esso. Questa scelta è stata dettata dal voler realizzare un'unica connessione del client al server tramite due socket, che simula la connessioni dirette tra client. Tutti i socket che vengono creati appartengono alla famiglia di indirizzi AF_INET con protocollo standard TCP per permettere connessioni tra macchine diverse attraverso la rete internet. Tutte le operazioni su file e lista sono sincronizzate tramite due semafori.

Il server mantiene le informazioni dei client connessi tramite una lista collegata. Il nodo della lista relativo ad un client viene generato al momento della sua prima connessione al server e rimosso al momento della sua disconnessione dal servizio. In ogni nodo viene memorizzato il nome utente, il socket relativo alla trasmissione di comandi, quello realtivo allo scabio di messaggi, il puntatore al client target relativo alla conversazione corrente e lo stato che identifica se il client è disponibile o è già impegnato in una conversazione. Abbiamo realizzato inoltre alcune funzioni di supporto utili per la gestione della lista collegata che hanno agevolato operazioni più complesse. Il main thread del server accetta connessioni in un loop infinito e quando ne riceve una crea un thread che gestisce la registrazione ed il login. Le coppie di username e password degli utenti registrati vengono memorizzati sul file "credenziali.txt". Nella fase di registrazione si chiede username

e password, accertandosi che lo username non sia già esistente, e vengono memorizzati su file. Nella fase di accettazione di socket viene applicato un controllo per verificare che i due socket appartengano allo stesso client.

Il client è costituito da 3 thread, uno per la gestione dei comandi provenienti dal server, uno per i messaggi di altri client ed uno per inviare messaggi e comandi. La fase di accesso al servizio viene gestita da una funzione che permette la registrazione o il login. Sulle password viene applicata una funzione hash (non realizzata da noi) in modo da non conservarle in chiaro sul file. La conversazione viene aperta tramite comando `connect` e si instaura nel momento in cui il thread che legge i messaggi riceve il primo messaggio.

3 Manuale d'uso

Codici sorgente del progetto:

- server.c
- client.c
- Makefile

Passi per l'utilizzo:

1. Compilare i programmi usando il comando `make` sulla shell unix
2. Avviare il server scrivendo `./server [PORT]` sulla shell, dove *[PORT]* è il numero di porta da assegnare al server
3. Avviare i vari client scrivendo `./client [IP] [PORT]`, dove *[IP]* è l'indirizzo IP del server (espresso come indirizzo numerico o come nome del server) e *[PORT]* è il numero di porta del server.

Gestione SIGINT:

- **server** chiude tutti i socket, rilascia la memoria riservata alla lista e termina tutti i client connessi
- **client** termina la conversazione corrente

Comandi del client:

- `log` avvia la procedura di login
- `reg` avvia la procedura di registrazione
- `help` stampa i comandi disponibili
- `list` stampa tutti gli utenti collegati al server
- `avl` stampa gli utenti disponibili per una conversazione
- `connect [Username]` permette di connettersi all'utente *[Username]*
- `quit` termina la connessione con il server e chiude il client