Tesina di Sistemi Operativi

Studente: Salvatore Licitra
Matricola: 1650021

Specifica della tesina:

Servizio di messaggistica
Realizzazione di un servizio di scambio messaggi supportato tramite un server sequenziale o concorrente (a scelta).
Il servizio deve accettare messaggi provenienti da client (ospitati in generale su macchine distinte da quella dove risiede il server) ed archiviarli.

L'applicazione client deve fornire ad un utente le seguenti funzioni:
1. Lettura tutti i messaggi spediti all'utente.
2. Spedizione di un nuovo messaggio a uno qualunque degli utenti del sistema.
3. Cancellare dei messaggi ricevuti dall'utente.

Si precisa che la specifica prevede la realizzazione sia dell'applicazione client che di quella server. Inoltre, il servizio potrà essere utilizzato solo da utenti autorizzati (deve essere quindi previsto un meccanismo di autenticazione).

Un messaggio deve contenere almeno i campi Destinatario, Oggetto e Testo.

Scelte progettuali

Tutto il progetto si basa sulla classica comunicazione tra una applicazione Server e molteplici applicazioni Client tramite il protocollo di rete TCP/IP. Il server erogherà quindi il servizio accettando in entrata le connessioni dei Client ed eseguendo i comandi richiesti dagli stessi.
Ho deciso di implementare il servizio in maniera sequenziale; ho fatto in modo che la durata della connessione tra client e server duri il minor tempo possibile, infatti tutte le interazioni tra macchina e utente, avvengono offline e, la connessione con il server avviene solo a dati processati. Inoltre per prevenire eventuali blocchi della connessione ho inserito un timer entro il quale, se la connessione è ancora aperta, ne viene forzata la chiusura.
La specifica richiede che possa usufruire di tale servizio solo un utente opportunamente autenticato, per cui verrà richiesto l'inserimento di un *username* e di una *password* all'accesso.
Le credenziali degli utenti vengono salvate in un file dal nome *"utenti_password"*, che viene consultato sia durante la creazione di un nuovo account (per controllare se l'username inserito è effettivamente disponibile) ed il login, sia durante l'invio di un messaggio, infatti non è possibile inviare un messaggio ad un utente inesistente.
I messaggi vengono anch'essi salvati in un file apposito (*"messaggi_statici"*); tale file rappresenta però solamente un archivio, infatti i messaggi inviati durante la medesima sessione del server vengono gestiti tramite l'uso di una struttura variabile.

Server

Il server inizialmente setta il proprio socket, dal quale riceverà le comunicazioni entranti dei client, impostando il proprio indirizzo e legando il port-number prelevato al momento dell'esecuzione. Dopodiché viene aperto sia il file contenete le credenziali, sia quello contenete i messaggi "archiviati" ed in entrambi i casi vengono create delle copie di backup.
I messaggi vengono mantenuti tramite una struttura dati di tipo lista collegata, tale lista è percorribile solamente in una direzione. L'inserimento dei nuovi messaggi avviene in testa, così da mantenere un ordine temporale che va dal più recente al più vecchio.
All'interno di un messaggio salvo: un codice univoco, il mittente, il destinatario, l'oggetto ed il testo vero e proprio; il codice servirà ad identificare univocamente un messaggio per permettere la sua rimozione.
A questo punto il server è pronto ad accettare le connessioni in entrata dei client, quindi entrerà in un ciclo infinito di attesa dal quale uscirà solo per chiudere il server.
Accettata la connessione parte il timer per la durata massima della stessa e si esegue il comando richiesto dal client (login, registrazione, lettura, scrittura, eliminazione). Se l'operazione non finirà in tempo, la connessione con il client verrà interrotta, ed il server ritornerà in attesa di nuove connessioni.
Il server può essere chiuso tramite combinazione "CTRL + C" o "CTRL +\"; alla chiusura il server si occuperà di salvare i messaggi contenuti nella lista nel file opportuno, e di farne una copia di backup, libererà le risorse utilizzate e terminerà l'esecuzione.
Tale comportamento verrà mantenuto anche in altri casi, in quanto si è deciso di gestire i più comuni eventi, tra cui quelli di terminazione, quali SIGINT, SIGQUIT, SIGHUP e SIGTERM, quelli causati da operazioni non consentite SIGILL e SIGSEGV, quello per il timer SIGALRM, ed infine l'evento per la perdita di connessione SIGPIPE.

Client

.

Il client prende da standard input l'indirizzo ed il numero di porta del server, all'avvio l'applicazione cercherà di instaurare una connessione con il server, l'utente potrà eseguire il log-in o la registrazione, inviando le credenziali; una volta verificate, l'utente potrà cominciare ad usare il servizio inserendo il codice numerico corrispondente alle 4 operazioni disponibili : LEGGI MESSAGGI (1), SCRIVI MESSAGGIO (2), ELIMINA MESSAGGIO(3) o LOGOUT(4).

- LEGGI: verranno stampati a schermo i messaggi ricevuti dall'utente. Prima quelli in memoria dinamica – cioè corrispondenti alla sessione in corso – dopo quelli della memoria fisica – quindi quelli dell'archivio.

- SCRIVI: verrà data la possibilità all'utente di inviare messaggi, il sistema controlla che il destinatario sia un utente facente parte il servizio ed in caso negativo comunicherà al mittente che il destinatario deve essere cambiato; l'utente potrà dunque cambiare destinatario e, a scelta, pure l'oggetto e il testo.

- ELIMINA: verranno stampati a schermo solo i messaggi contenuti in lista (quelli scritti durante la sessione del server corrente) seguiti da una stampa con corrispondenza numero messaggio stampato e codice, dopo di che l'utente può inserire il codice del messaggio da eliminare, oppure annullare tutto inserendo il numero speciale -1

- LOGOUT: l'utente viene disconnesso.

A differenza del server la chiusura del client può essere eseguita ovunque senza troppe complicazioni, preoccupandoci solo di chiudere il socket. Questa può avvenire sia tramite segnali che tramite l'inserimento del comando a video corretto.
Nel client gestiamo gli stessi segnali del server, fatta eccezione per il SIGALRM visto che non c'è alcun timer.

Manuale d'uso

Server

1. Assicurarsi di essere in un sistema operativo UNIX
2. Copiare il file "*server.c*" e, qualora esistenti, *"utenti_password"* e *"messaggi_statici"* nel direttorio in cui si preferisce eseguire il server
3. Compilare tramite il comando "gcc -o server server.c"
4. Eseguire tramite il comando "./server" seguito dal parametro "-p" e dal port-number scelto
5. Per chiudere il server basta usare la combinazione di tasti CTRL + C

Client

1. Assicurarsi di essere in un sistema operativo UNIX
2. Copiare il file "client.c" nel direttorio in cui si preferisce eseguire il client
3. Compilare tramite il comando "gcc -o client client.c"
4. Eseguire tramite il comando "./server -a" seguito dall'indirizzo del server a da "-p" seguito dal port-number
5. Da questo punto in poi seguire le indicazioni a video per eseguire le operazioni desiderate
6. Per chiudere i server digitare '0' oppure usare la combinazione di tasti CTRL + C