**PROGETTO DI SISTEMI OPERATIVI**

**2021**

**SPECIFICHE DEL PROGETTO:**

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.

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

Per progetti misti Unix/Windows è a scelta quale delle due applicazioni sviluppare per uno dei due sistemi.

**SVILUPPO:**

**Server**

Per prima cosa ho dovuto gestire il parse della command line; quindi ho utilizzato la funzione parseCmd() per leggere le stringhe di argv successive alle stringhe “-p” e “-a”, effettuando qualche controllo per verificare la correttezza di quanto inserito al lancio del programma. Successivamente ho gestito il log in di un utente. Per questo, quando il server viene lanciato ha una prima fase di inizializzazione in cui crea il file log/log.txt che serve per salvare tutti gli utenti registrati con le loro relative password. Inoltre, viene creata la directory data/ utile per salvare i file contenenti i messaggi ricevuti dai vari utenti. Per comunicare con il client ho utilizzato una connessione di tipo TCP, per non perdere nessun pacchetto nella comunicazione, con un server di tipo sequenziale, quindi ho utilizzato il seguente schema:

socket() get socket

bind() bind socket to Network address and port

listen() open socket to accept connections

while(1){

accept() accept connection from client

read() read message from client

if()

else if() …

}

Quindi, formattando la prima parte del messaggio ricevuto (controllo la prima parte per capire quale richiesta fa il client e come devo utilizzare la seconda parte) si entra in uno dei casi del if/else if. Per la gestione del log in e della registrazione utilizzo delle funzioni ausiliarie:

-verifica\_username() controllo che l’utente sia registrato e invio al client il messaggio di risposta, affermativa o negativa.

-verifica\_password() controllo se la password associata all’utente sia quella inserita dal client

-addUtente() aggiungo un utente, quindi scrivo una nuova riga sul file log/log.txt

-addPasswd() aggiungo la nuova password relativa all’utente appena registarato

Negli altri rami else if, trovo 3 funzionalità principali, quella di salvare un nuovo messaggio sul file data/nome\_utente.txt, quello di mostrare tutti i messaggi ricevuti da un utente e quello di cancellare uno dei messaggi. Per leggere bene i vari file, utilizzo dei caratteri di formattazione, come ad esempio il carattere ‘$’. Infatti le righe del file contenete i messaggi ricevuti saranno del tipo: Mittente:<Nome>$Oggetto:<oggetto>$<data e ora>$Testo:<testo>.

Perciò per leggere dal file, mi basterà scorrere e contare quante volte trovo il carattere ‘$’ per capire quale parte del messaggio sto analizzando.

**Client:**

Anche il client inizia con una prima parte dedicata al parsing della command line; utilizzo anche qui la funzione ausiliaria parseCmd() che però ricerca unicamente la stringa “-p” e controlla che quanto inserito sia corretto. Successivamente mi occupo del log in, mediante l’utilizzo di due funzioni log\_in() e verify\_password(). In queste funzioni lo schema di comunicazione con il server è:

socket() get socket

connect() open a connection

write() send to the server

read() read the answer from the server

Se la risposta della read riporta un qualche tipo di errore, allora la procedura di login ricomincia. Successivamente all’autenticazione, ho sviluppato un menù, attraverso l’utilizzo di alcune macro che portano in vari punti del programma. Per ogni macro troviamo un ciclo while ed uno schema di comunicazione con il server identica allo schema sopra riportato. Le macro sono:

-mex: serve per inviare al server un nuovo messaggio, contenente i vari campi (oggetto, mittente, destinatario, testo)

-show: serve per chiedere al server di mostrare tutti i messaggi ricevuti dall’utente. Questa feature è fondamentale sia per mostrare i messaggi ricevuti, sia per scegliere quale messaggio eliminare.

-end: serve a fare il log off dell’utente. (questo avviene anche se il programma viene chiuso con il segnale SIGINT)