TESINA SISTEMI OPERATIVI

Domenico Azzarito – Matricola 0312790

Anno Accademico 2022/2023

SPECIFICHE DEL PROGETTO

Il progetto consiste in un servizio di scambio messaggi supportato tramite un server concorrente.

Il servizio deve accettare messaggi provenienti da client (ospitati generalmente su macchine diverse da quella in cui 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 contiene i campi Destinatario, Oggetto e Testo. Inoltre, il servizio potrà essere utilizzato solo da utenti autorizzati attraverso un meccanismo di autenticazione.

Il sistema operativo per cui è stato scritto il servizio è il sistema LINUX.

DESCRIZIONE DEL SISTEMA

Un utente per accedere al servizio deve effettuare la registrazione o l'autenticazione. In caso di esito positivo, viene stampato a schermo il menu con le funzioni di lettura, spedizione e cancellazione dei messaggi e la funzione che permette di eseguire il logout. Dopo l'esecuzione di ciascuna funzione, escluso il logout, si ritorna al menu principale.

Connessione tra Server e Client

Il processo principale del server rimane in ascolto per nuove connessioni. Dopo aver accettato una nuova connessione, il processo principale verifica se è stato raggiunto il numero massimo di connessioni permesse contemporaneamente. In caso di esito positivo viene chiusa la connessione, altrimenti essa viene mandata in gestione ad un thread che si occuperà di interagire con il client. Tale thread è settato come "detached" in modo tale che alla propria chiusura le sue risorse vengano rilasciate dal sistema senza l'attesa di un "pthread_join ()". Inoltre, per evitare di mantenere connessioni morte, ogni thread imposta un timeout di 300 secondi sul proprio socket attraverso l'opzione SO_RCVTIMEO. La scelta del timeout è arbitraria.

Sincronizzazione e comunicazione tra Server e Client

Il Client e Server comunicano tramite protocollo TCP nel dominio AF_INET su porta 8080. La sincronizzazione delle due applicazioni avviene tramite un set di codici consistenti in stringhe di lunghezza fissata "lenght_code" (10 caratteri) che permettono di comunicare cosa si sta inviando e

la lunghezza di ciò che si sta inviando. Se nel Client si cerca di modificare tale set, il Server procede alla chiusura della connessione.

La ricezione e l'invio dei dati avviene attraverso le funzioni "read_from_sock" e "write_on_sock" che verificano di leggere o scrivere i dati nella loro interezza. In caso di disconnessione da parte del Client o del Server, le funzioni incorrono in un errore che le porta a terminare la connessione.

Prima di inviare qualsiasi dato diverso da un codice, si invia la sua lunghezza attraverso una stringa creata attraverso un processo di padding. Esso genera una stringa contente il numero della lunghezza del dato da inviare e tanti caratteri '#' quanti necessari per arrivare a una lunghezza prefissata. In questo caso, si è scelto lunghezza 4 in quanto non è necessario inviare stringhe più lunghe di 9999 caratteri.

Gestione segnali Server e Client

Il Server è settato in modo tale da ignorare tutti i segnali tranne quelli legati a operazioni illegali come SIG_ILL o SIG_SEGV. In questo modo può essere arrestato esternamente solo tramite SIGKILL.
Il Client è settato in modo tale da gestire i segnali SIGINT, SIGQUIT, SIGTERM e SIGPIPE. Quando vengono ricevuti i primi tre segnali si esegue lo stesso handler che esegue la disconnessione e terminazione del client. In caso di SIGPIPE, viene comunicato all'utente la disconnessione del server e viene terminato il processo.
I segnali SIGINT, SIGQUIT e SIGTERM sono stati lasciati per permettere all'utente di terminare il processo in modo volontario durante una delle operazioni di invio, lettura o eliminazione dei messaggi, senza dover per forza tornare al menu di scelta.

Gestione dei messaggi

Il messaggio consiste in una struct di tre stringhe di lunghezza prestabilita che corrispondo al "mittente", "oggetto" e "testo". Il Server per ogni utente registrato crea due file, uno per salvare i messaggi e l'altro per salvare un indice in cui registro il numero di messaggi e gli offset di inizio e fine di ciascun messaggio del primo file. Per permettere di gestire l'indice si utilizza una struct apposita che viene caricata con i dati presi dal file indice e, in caso di aggiornamento, sovrascritta sul file stesso.

Le possibili operazioni elementari sono:

Inserimento messaggio
Viene scritto il messaggio alla fine del file dei messaggi e aggiornato il file indice.

Visualizzazione messaggio
Considerando i messaggi numerati in ordine crescente dal primo messaggio scritto sul file, con un ID si seleziona il messaggio che si vuole visualizzare e attraverso l'indice si ricava l'offset da cui partire per recuperarlo dal file dei messaggi.

Eliminazione messaggio
Considerando i messaggi numerati in ordine crescente dal primo messaggio scritto sul file, con un ID si seleziona il messaggio che si vuole eliminare e attraverso l'indice si ricava l'offset da cui

partire per eliminarlo. La funzione che il sistema propone, dopo aver recuperato il numero di bytes che occupava il messaggio nel file, crea un nuovo indice in cui carica gli stessi offset del vecchio indice per ogni messaggio precedente a quello da eliminare. Per ogni messaggio successivo, gli offset vengono diminuiti del numero di bytes da eliminare e caricati nel nuovo indice, il quale viene sovrascritto sul file apposito. In seguito, si crea un nuovo file su cui vengono caricati i messaggi precedenti e successivi a quello da eliminare, eliminando il vecchio file utilizzato per la loro archiviazione.

Per permettere la concorrenza, il sistema utilizza una tabella hash costituita da un array le cui entry sono puntatori a una struct contenente la stringa corrispondente all'username dell'utente e un puntatore a un mutex che permette di gestire l'accesso concorrente al file dei messaggi e dell'indice legati all'utente stesso. Tale struct viene inizializzata al momento della registrazione dell'utente.

Le tre operazioni elementari vengono sfruttate lato server per permettere la visualizzazione, l'invio e l'eliminazione di un messaggio attraverso i dati trasmessi dal client. Inoltre, nell'inserimento dell'oggetto e del testo del messaggio, se si vuole finire prima del limite di lunghezza stabilito, si deve digitare in una nuova linea la parola "END" e andare a capo. Se invece mentre si digita si superano i limiti prestabiliti, il sistema tronca il testo immesso e chiede se va bene troncato o lo si vuole ridigitare interamente.

Meccanismo di autenticazione

Il sistema permette l'autenticazione attraverso un username, di lunghezza limitata e composto di caratteri alfanumerici minuscoli, e una password di lunghezza limitata generata dal sistema stesso al momento della registrazione.

Quando avviene la registrazione l'utente consegna un username al sistema che, dopo aver verificato se disponibile e accettabile, chiede all'utente di inserire la lunghezza della password desiderata. Se il numero inserito è nel range prestabilito, il sistema prosegue con la generazione di un sale di lunghezza prestabilita e di una password utilizzando per entrambi due set di caratteri prestabiliti. Dopo aver ritornato la password in chiaro all'utente, il sistema genera attraverso la funzione "crypt_r()" la password criptata. Il sistema permette di modificare prima dell'avvio la tecnica di crittografia modificando la variabile CODE_CRYPT. In questo caso si è scelto l'algoritmo SHA-512 per una maggior sicurezza.

La scelta del set di caratteri della password e sale e dei limiti sulle lunghezze di username, password e sale è arbitraria. In questo caso per la password è stata scelta una lunghezza compresa tra i 12 e i 20 caratteri per aumentarne la sicurezza e scoraggiare possibili cifrature. Per quanto riguarda il set di caratteri del sale, la scelta rispetta le descrizioni date dal manuale man per la funzione "crypt_r()".

La conservazione dei dati sensibili (username, sale e password criptata) avviene in un file di testo. Tale file è strutturato in modo tale che ogni riga rappresenta un utente e contiene i dati separati da uno spazio. Ogni volta che avvio il sistema carico i dati dal file e li inserisco in una tabella hash apposita. Ogni entry della tabella hash è una struct che contiene le stringhe username, sale e password. Ogni volta che avviene una registrazione si aggiunge una entry alla tabella hash e si aggiunge una riga al file di testo. Ogni volta che avviene un login, si verifica attraverso la tabella hash se l'utente è registrato e se la password inserita è corretta.

Per permettere una corretta concorrenza durante la registrazione si utilizza un'altra tabella hash composta da un array di stringhe che contiene i nomi che sono in corso di registrazione. In questo modo mentre registro un username, impedisco ad altri utenti di usarlo per la propria registrazione. Se il primo utente effettua una disconnessione durante la registrazione, non completandola, gli altri utenti hanno nuovamente la possibilità di usare lo stesso username del primo utente.

Gestione delle Tabelle Hash

Il sistema utilizza una funzione di doppio hashing per calcolare l'indice associato a ciascuna chiave che consiste in una stringa. Tale funzione è formulata come segue

$$\mathrm{H(i,k)} = \big(\mathrm{h_1(k)} + \mathrm{i} * \mathrm{h_2(k)}\big) \,\%\, \mathrm{num\_key}$$

con $h_1(k) = k \,\%\, \mathrm{num\_key}$ e $h_2(k) = \big(k \,\%\, (\mathrm{num\_key} - 2)\big) + 1$. Il valore passato alla funzione per num_key, affinché la funzione stessa sia affidabile, deve essere scelto tra i numeri primi distanti da potenze di 2 e di 10.

Per quanto riguarda le tabelle hash "authTable" e "userTable", il numero massimo di chiavi "MAX_KEY" è lo stesso e deve essere modificato per aumentare l'efficienza della funzione di hashing quando il numero di utenti registrati è maggiore della metà delle chiavi totali. Il sistema è programmato per mandare un avviso quando tale limite è raggiunto.

Per quanto riguarda la tabella hash "tempUserNameTable", il numero massimo di chiavi "MAX_KEY_TEMP" deve essere il minimo numero che soddisfi la condizione di "num_key" e che sia maggiore del doppio del numero di utenti possibili online contemporaneamente. Il sistema è programmato per uscire nel caso in cui si setta "MAX_KEY_TEMP" minore del doppio di "MAX_USERS_ONLINE".

COMPILAZIONE ED ESECUZIONE

Il Client e il Server vengono forniti in due cartelle differenti, ognuno con il proprio Makefile.

Il Server è diviso in 4 file sorgente e 5 file header:

- Il file sorgente main.c contiene tutto il necessario per descrivere il processo principale del server;
- Il file sorgente mainFunctions.c contiene le funzioni principali per eseguire l'autenticazione, per gestire le funzioni di invio, visualizzazione ed eliminazione dei messaggi e le funzioni di comunicazione tra sockets.
  Ad esso corrisponde un file header che ne permette la comunicazione con il file main.c e che contiene i codici per la comunicazione con il client;
- Il file sorgente authFunctions.c contiene le funzioni utilizzate per gestire la tabella hash di autenticazione, per generare la password e il sale e verificarle insieme all'username.
  Ad esso corrisponde un file header che ne permette il collegamento con il file mainFucntions.c;
- Il file sorgente messageFunctions.c contiene le funzioni utilizzate per gestire la tabella hash associata ai file dei messaggi e degli indici, l'inserimento, l'eliminazione e la visualizzazione di un messaggio presente su un file dei messaggi, il caricamento e l'aggiornamento del file indice;

Ad esso corrisponde un file header che ne permette il collegamento con il file mainFucntions.c;
- Il file header globalVar.h è contenuto in tutti i file sorgente, descrivendo tutte le strutture e le costanti utilizzate da tutti i file sorgente.

Il Client è diviso in 2 file sorgente e un file header:

- Il file sorgente main,c contiene tutto il necessario per descrivere il processo principale del client;
- Il file sorgente mainFunctions.c contiene le funzioni principali per eseguire l'autenticazione, per gestire le funzioni di invio, visualizzazione ed eliminazione dei messaggi, le funzioni di comunicazione tra sockets e la funzione per ottenere il testo e l'oggetto di un messaggio da input;
- Il file header globalVar.h mette in comunicazione il file main.c e mainFunctions.c e contiene i codici e le costanti per la comunicazione con il server.

Il Makefile associato al Server permette la creazione da ogni file sorgente di un corrispettivo file oggetto, per poi compilare l'eseguibile "server" attraverso l'unione dei file oggetto e delle librerie. In aggiunta, attraverso le seguenti opzioni si ha:

- "make build", la creazione della directory contente i file dei messaggi e degli indici e la creazione del file di autenticazione
- "make clean", l'eliminazione di tutti i file oggetti e dell'eseguibile

Il Makefile associato al Client permette la creazione da ogni file sorgente di un corrispettivo file oggetto, per poi compilare l'eseguibile "client" attraverso l'unione dei file oggetto e delle librerie. In aggiunta, attraverso la seguente opzione si ha:

- "make clean", l'eliminazione di tutti i file oggetti e dell'eseguibile

Al primo avvio del sistema bisogna andare da terminale nella cartella "Server" e digitare "make build" e "make", per poi digitare "./server" per far partire il server. Se si vuole farlo partire in background digitare "./server&". Ora che il server è avviato, si può andare da terminale nella cartella "Client" e digitare "make" per generare l'eseguibile e farlo partire con "./client" per testare il sistema.

Prima di generare l'eseguibile per il client, bisogna aprire il file "Client/main.c" e modificare IPADDRESS con la stringa contente l'indirizzo IPv4 della macchina su cui risiede il server.

MODIFICHE E MANUTENZIONE

Lato Server

Il codice raccoglie un set di costanti e variabili globali che possono essere modificate in modo tale da permettere diversi settaggi del server. Bisogna però prestare attenzione in alcuni casi:

Modifica dei codici di sincronizzazione

Nel caso in cui si modifichino i codici di sincronizzazione, bisogna modificare il set di codici associato al client; In caso contrario, si potrebbe causare l'impossibilità della connessione o la disconnessione improvvisa del client.

Modifica costanti nell'autenticazione
Nel caso si modifichino le lunghezze massime e minime dell'username e della password, bisogna procedere con la modifica di esse lato client, per evitare di rendere impossibile la registrazione.
Nel caso si modifichi l'algoritmo di crittografia bisogna aggiornare esternamente il file contente i dati sensibili per permetterne la corretta lettura dei dati già registrati. In caso contrario, le password dei dati già registrati non sarebbero mai verificabili.

Modifica costanti nella gestione messaggi
Modificando i limiti delle stringhe mittente, oggetto e testo nella struct del messaggio, bisogna aggiornare esternamente i file dei messaggi e degli indici; In caso contrario, la lettura dei messaggi dal file avverrebbe in modo errato.
La modifica del numero massimo di messaggi registrabili è sempre possibile.

Modifica pathname
Se si modifica il pathname della cartella che contiene i file dei messaggi e indici o il pathname del file di autenticazione o si hanno file mancanti, il server viene arrestato e comunica il pathname del file che non è riuscito ad aprire.

Lato Client

Qualsiasi modifica illecita lato client che porta a differenze di costanti e codici tra server e client verrà rilevata dal server che proseguirà con la disconnessione del client.