**RELAZIONE DEL PROGETTO DI RETI 1**

A.A. 2018/2019 Vercelli

Sviluppato da: Alessandro Fatone – 20019780

# DESCRIZIONE DEL SOFTWARE

Il software è una semplice applicazione di rete, costituita da un programma server ed un programma client, in cui il programma server calcola la distribuzione dei caratteri alfanumerici (ovvero crea, per ogni carattere memorizzato, un contatore) che compongono un messaggio fornito dal client, che utilizzerà vari comandi per poterli inviare al server e ricevere le eventuali risposte. I comandi che il client utilizza e che il server riesce a riconoscere sono:

* TEXT – Per inviare i messaggi di testo;
* HIST – Per ricevere l’istogramma dei caratteri;
* EXIT – Per eseguire il comando HIST e chiudere la connessione;
* QUIT – Per chiudere la connessione;

L’applicazione è realizzata in linguaggio C ed è implementa tramite le librerie di sistema POSIX; inoltre utilizza i protocolli IPv4 e TCP per scambiarsi i messaggi, e tali messaggi non superano la dimensione di 512 bytes.

Il programma server, una volta eseguito, rimane sempre attivo per gestire le richieste del client, però è in grado di rispondere soltanto ad un programma client per volta.

# FUNZIONALITÀ DEL CLIENT

Il client, per comunicare con il server, ha bisogno di conoscere il suo indirizzo IP e il numero di porta su cui attende le comunicazioni. Quando si sarà connesso, il client riceverà un messaggio di benvenuto dal server, nel formato “OK START <testo>”, che stamperà a schermo (il client si occuperà di rimuovere il protocollo “OK START” e di stampare solo il messaggio di testo, sia per il messaggio di benvenuto che per tutti gli altri messaggi che riceverà), insieme all’ora e alla data di connessione. Dopodiché mostrerà una descrizione del comportamento del programma e le sue istruzioni d’uso. Esso avrà quattro opzioni a disposizione dell’utente (comandi), che sono:

* Inserimento del testo: tramite il comando TEXT il client invia al server il testo da analizzare, insieme ad un contatore che rappresenta il numero di caratteri alfanumerici presenti nel testo. Il messaggio inviato dovrà rispettare un determinato formato (devono essere presenti almeno due spazi, uno che separa il comando dal testo e l’altro che separa il testo dal contatore, e deve terminare con un invio), e lo stesso vale per i singoli componenti che compongono il messaggio (il comando deve essere scritto in maiuscolo, il contatore deve essere maggiore di zero e composto sempre da tre cifre e non si possono inserire più di 500 caratteri nel testo di un messaggio). Il comando può essere eseguito molteplici volte per analizzare testi di grandi dimensioni (però non potrà essere eseguito all’infinito in quanto i contatori del server sono limitati).

Esempio di un comando TEXT:

Client: TEXT CIAO A TUTTI 010

Server: 10

La risposta che il client riceverà dal server, nel caso non fossero presenti errori, conterrà il contatore del messaggio (nel caso dell’esempio, la risposta sarà 10);

* Analisi del testo: tramite il comando HIST il client invia al server la richiesta dell’istogramma dei caratteri. Come per il comando precedente, anche HIST deve rispettare un certo formato (deve essere scritto in maiuscolo e terminare con un invio) e può essere ripetuto molteplici volte per poter analizzare i caratteri dei nuovi testi inseriti successivamente all’istogramma.

Esempio di un comando HIST:

Client: HIST

Server: A:2 C:1 I:2 O:1 T:3 U:1

Server: END

Il server invia il messaggio di risposta contenete i caratteri e il numero di volte che sono stati inseriti nel/i comando/i TEXT (l’esempio utilizza il messaggio inviato precedentemente: “TEXT CIAO A TUTTI 010”) e un messaggio di testo finale (END) che conclude il comando.

Il server può inviare più di un istogramma nel caso esso non rientri nella dimensione dei messaggi;

* Chiusura programma (con analisi del testo): tramite il comando EXIT il client esegue il comando di HIST e, una volta terminato, la connessione con il server e il programma stesso vengono chiusi.

Come per HIST, anche EXIT deve essere scritto nello stesso formato (maiuscolo e invio finale).

Esempio di un comando EXIT:

Client: HIST

Server: A:2 C:1 I:2 O:1 T:3 U:1

Server: END

Server: <messaggio di testo di fine connessione>

L’unica differenza rispetto all’esempio di HIST è che il client riceve un unico messaggio contenente il messaggio finale di HIST e il messaggio di testo di EXIT che annuncia la fine della connessione;

* Abbandono programma (senza analisi del testo): tramite il comando QUIT il client chiude semplicemente la connessione, ricevendo dal server solo la risposta di fine connessione, e il programma viene terminato.

Esempio di un comando EXIT:

Client: QUIT

Server: <messaggio di testo di fine connessione>

Il comando rispetta lo stesso formato dei precedenti comandi di HIST e EXIT.

Nel programma client possono verificarsi diversi errori, che possono essere stampati direttamente dal client o ricevuti dal server, e questi possono essere:

* Errori di comunicazione: questi errori avvengono se il client non riesce a stabilire una connessione con il server (errore sul socket, errore di connessione) o nel caso questa venga interrotta (errore di comunicazione);

* Errori di sintassi: questi errori invece sono inviati dal server perché i messaggi che ha ricevuto non rispettano il formato prestabilito (errori di scrittura sul comando, sul testo, sul contatore etc.);

* Errori sul contatore: essi vengono ricevuti dal server e riguardano il contatore del comando TEXT (il contatore è minore di uno oppure il suo valore non rispecchia il numero di caratteri alfanumerici presenti nel testo);

* Errori di overflow: errori che coinvolgono sempre il comando TEXT e possono avvenire o nel client (errore sul testo di input, dove esso supera il numero di caratteri prestabiliti dai messaggi) oppure nel server (overflow del/i contatore/i nel server, quando uno o più contatori di caratteri alfanumerici superano i valori di memorizzazione previsti).

Quando questi errori si verificano il programma client chiude la connessione (a meno che l’errore non riguardi la creazione del socket, perché in quel caso non viene neanche aperta) con il server e termina la sua esecuzione.

# FUNZIONALITÀ DEL SERVER

Il server, per comunicare con il client, deve attendere ed “ascoltare”, attraverso una specifica porta, che il programma client invii una richiesta di comunicazione. Quando il server riceve questa richiesta e riesce a stabilire una connessione con il client, esso invia un messaggio di benvenuto composto dal protocollo “OK START” più un messaggio di benvenuto che verrà stampato dal client una volta ricevuto. Dopodiché il programma server rimarrà in attesa di ricevere il messaggio che il client gli spedirà. Una volta che lo avrà ricevuto, il server verificherà la presenza del comando all’inizio del messaggio, controllerà che il suo formato sia corretto (lettere maiuscole e nessun errore di scrittura) e, se è in grado di riconoscerlo, utilizzerà dei valori positivi per rappresentare i comandi più facilmente. Dopodiché, in base a quei valori, il server svolgerà specifiche funzioni per realizzare le azioni previste dal comando. Le funzioni sono richiamate in base ai seguenti comandi:

• Per il comando TEXT il server richiama una funzione di controllo per verificare la sintassi del messaggio ricevuto. Nel messaggio (che sarà scritto nel formato “TEXT <testo>

<contatore>”) verrà verificata la presenza degli spazi separatori (tra il comando e il testo e tra il testo e il contatore) e dell’invio finale, la presenza di un testo che contenga almeno un carattere alfanumerico e la presenza di un contatore (scritto con 3 cifre). Inoltre verrà anche testata la validità del contatore (infatti, all’interno della funzione, ci sarà un ciclo che conterà tutti i caratteri alfanumerici presenti nel messaggio e, una volta terminato, essa confronterà il contatore del ciclo con quello del messaggio).

Esempio di un messaggio che rispetta tutti i controlli sopraccitati:

TEXT CIAO A TUTTI 010

Se il messaggio rispetta tutti i controlli allora verrà chiamata una seconda funzione che si occuperà di salvare i caratteri alfanumerici. Essa avrà il compito di confrontare ogni singolo carattere del messaggio con il contenuto di un array di strutture, soltanto se il carattere preso in campione sia alfanumerico. Esso verrà cercato all’interno del vettore (per verificare se quel carattere è già stato salvato precedentemente) e se la ricerca avrà successo la funzione dovrà solo incrementare il contatore presente in quella posizione dell’array (il contatore non deve aver raggiunto il suo massimo valore); nel caso contrario il carattere verrebbe memorizzato nella prima posizione libera e il suo contatore verrebbe incrementato di uno. Infine la funzione esegue un algoritmo di insertion sort per ordinare i caratteri presenti nel vettore, e invia al client un messaggio composto dal protocollo OK TEXT più il contatore alfanumerico del messaggio.

Ritornerà alla funzione principale il valore 1, che verrà utilizzato dal programma server per mettersi in attesa del prossimo comando (dato che il comando era TEXT). Nel caso si fossero verificati degli errori la funzione principale riceverebbe come valore 0, e perciò chiuderebbe la connessione con il client, mettendosi in attesa per una nuova richiesta di collegamento;

* Per il comando HIST il server richiama una funzione per memorizzare i caratteri presenti nell’array di strutture in uno o più buffer che, successivamente, insieme ad altri messaggi, verranno inviati al client al termine della funzione stessa. Ogni buffer inizia con il protocollo “OK HIST”, seguito da uno spazio, seguito dal <carattere>:<contatore> e uno spazio per separare il dato successivo; il buffer, alla fine, terminerà sempre con un invio. In esso vengono memorizzati solo i caratteri alfanumerici presenti nella struttura, che vengono prelevati uno ad uno, e vengono controllati in base alla loro dimensione. Infatti all’interno della struttura sono presenti dei controlli che contano il numero di caratteri e soprattutto il numero di cifre presenti nel contatore, in modo tale da poter confrontare la dimensione totale (dei caratteri) di un dato con il puntatore del buffer. Se la loro somma è inferiore alla dimensione massima del buffer, il dato viene salvato nel buffer; altrimenti, per evitare che il dato venga “tagliato” o corrotto, il buffer viene inviato al client, resettato, e riutilizzato per memorizzare i dati successivi.

Al termine della funzione viene inviato al client un messaggio finale (OK HIST END\n) che rappresenta la fine delle operazioni eseguite dal comando, e che utilizzerà per interrompere il ciclo di lettura dei buffer. “OK HIST END\n” sarà l’unico messaggio inviato al client nel caso il vettore di strutture non contenga caratteri alfanumerici.

Come per il comando TEXT, questa funzione ritornerà il valore 1 che verrà utilizzato dal programma server per mettersi in attesa del prossimo comando;

* Per il comando EXIT il server richiama la stessa funzione usata per il comando HIST, con la differenza che, al termine di essa, verrà richiamata una seconda funzione (errorexitquit) per inviare al client il messaggio finale di HIST più quello di EXIT (OK HIST END\nOK EXIT <testo>\n), che il client utilizzerà per terminare la connessione; il server, a sua volta, chiuderà la connessione dopo aver inviato l’ultimo messaggio.

In questo caso la funzione ritornerà 0 perché il programma server deve attendere una nuova connessione, non più un nuovo comando;

* Per il comando QUIT il server richiama la funzione che gestisce gli errori e le disconnessioni, e tramite essa invia il messaggio “OK QUIT <testo>\n” al client e termina la connessione. Questa funzione, oltre a gestire il comando EXIT e QUIT, gestisce tutti gli errori che possono verificarsi nel server, che sono: errori del socket, di bind, di connessione (errori interni del server), di sintassi, del contatore errato e del suo overflow (errori riguardanti i messaggi). Come per l’EXIT, questa funzione avrà come valore di ritorno 0.

# COMPILAZIONE

Il programma client e il programma server devono essere compilati separatamente a causa della loro indipendenza.

La compilazione ed i test dell’applicazione sono stati effettuati in un ambiente Linux, più precisamente utilizzando il programma Windows Terminal.

Dopo aver compilato i programmi con successo, la loro esecuzione avviene tramite le seguenti istruzioni:

Per eseguire il client: ./client <indirizzo server> <numero porta>

Per eseguire il server: ./server <numero porta>

L’indirizzo IP, come viene citato all’inizio, deve essere scritto nel formato IPv4.

Il numero di porta non può superare il valore 65535, e alcune porte non possono essere utilizzate perché sono private.

Nel caso fossero presenti errori nei parametri di esecuzione, i programmi stampano a schermo le istruzioni dettagliate per poterli eseguire correttamente.

Nel caso si dovesse verificare un errore riguardante l’overflow dello stack (errore improbabile) suggerisco di compilare il server con l’istruzione -fno-stack-protector, in modo tale da disabilitare la protezione dello stack del terminale.

Per problemi riguardanti il firewall consiglio di utilizzare i privilegi di amministratore per eseguire con successo i programmi.