Università del Piemonte Orientale - Vercelli - Reti

RELAZIONE PROGETTO

Programmi sviluppati da Alessandro Fatone

DESCRIZIONE

Il software è una semplice applicazione di rete composta da un programma client ed un programma server, realizzata in linguaggio C e testata su Linux Ubuntu 22.04 LTS. Il progetto si basa su una versione semplificata di Wordle, che è un gioco dove bisogna indovinare una parola attraverso una serie di indizi. Il server genera una parola casuale mentre l'utente deve indovinare la parola nei tentativi a disposizione.
Per poter comunicare il client ed il server utilizzano i protocolli IPv4 e TCP, e per identificare ed elaborare i messaggi le applicazioni inseriscono, all'inizio di ogni messaggio, un protocollo specifico. Inoltre, per garantire l'interoperabilità tra programmi client e server analoghi, il formato del messaggio deve rispettare alcune regole precise: deve avere tutti i caratteri del protocollo in maiuscolo, deve terminare sempre con il carattere d'invio (\n) e non può superare il limite massimo di 256 caratteri.
Dopo aver terminato i tentativi, o nel caso di resa del giocatore, la connessione tra client e server viene terminata. In presenza di un eventuale errore, la connessione viene terminata immediatamente senza aspettare la fine del gioco.

SPECIFICHE DEL CLIENT

Il client ha bisogno di conoscere l'indirizzo IP e la porta del server; per questo motivo, per l'esecuzione del programma, sono necessari gli argomenti "./client <indirizzo server> <numero porta>". Se il client riesce a creare il socket ed a connettersi con il server, esso richiamerà ciclicamente due funzioni: writemex e readmex. Il client inizialmente si aspetterà di ricevere un messaggio di benvenuto dal server e perciò eseguirà soltanto la funzione readmex, mentre successivamente verranno richiamate entrambe le funzioni, per scrivere ed inviare la parola e per leggere ed elaborare il messaggio ricevuto. Una volta richiamate, le funzioni eseguono le seguenti operazioni:

- Writemex: riceve in input cmex (utilizzato per stampare il numero del tentativo attuale) e simpleSocket (per inviare il messaggio al server), e richiede all'utente di inserire una parola di cinque caratteri. La parola deve rispettare la lunghezza prestabilita (massimo 5 caratteri) e i caratteri inseriti devono essere esclusivamente alfabetici. In caso contrario il client stampa a schermo un messaggio di errore e sollecita l'utente a inserire una nuova parola (tramite ciclo do-while). Una volta terminati i controlli, il contenuto inserito viene salvato in un buffer nel formato "WORD <parola>\n"; invece, se è stata digitata la parola "fine" (con caratteri minuscoli o maiuscoli), nel buffer verrà salvato il messaggio "QUIT\n". A fine funzione, il messaggio del buffer verrà inviato al server e la funzione ritornerà il valore di endsignal al main del programma. Se endsignal assume il valore 1 significa che la funzione non è riuscita ad inviare il messaggio al server, e perciò si dovrà terminare l'esecuzione del programma. In caso contrario, endsignal sarà pari a 0 e il gioco proseguirà normalmente;

- Readmex: come per writemex la funzione riceve in input cmex e simpleSocket, e quest'ultimo viene utilizzato per leggere il messaggio dal server. Se il client non riesce a ricevere il messaggio o se esso supera il limite di caratteri prestabiliti, la variabile di ritorno endsignal assumerà il valore 1. Altrimenti viene richiamata la funzione checkmex per controllare e stampare il messaggio del server, e a sua volta checkmex restituirà un valore pari a 0 o ad 1 che verrà salvato sempre nella variabile endsignal;

- Checkmex: è la funzione più complessa del programma, che si occupa di controllare e stampare il messaggio ricevuto. La funzione viene richiamata all'interno di readmex quando essa riceve con successo un messaggio dal server, e perciò come imput checkmex avrà, oltre alla variabile cmex, una stringa buffer (che contiene il messaggio ricevuto) e una variabile returnStatus (che contiene il numero totale dei caratteri del messaggio). All'inizio viene effettuato un controllo su cmex, perché se la variabile ha valore 0 la funzione si aspetta un messaggio di benvenuto. In questo caso il messaggio ricevuto deve avere la seguente struttura: "OK N(N) <messaggio>\n". Per verificare la struttura del messaggio vengono eseguiti vari controlli. Come già accennato la lunghezza massima deve essere pari o inferiore a 256 caratteri, deve terminare con il carattere d'invio e il protocollo iniziale deve contenere caratteri maiuscoli; oltre a ciò, devono essere presenti due spazi: uno che separa il protocollo dai tentativi e l'altro che separa i tentativi dal testo. Bisogna anche effettuare controlli sul numero dei tentativi totali (N(N)), perché esso può avere 1 o 2 cifre a seconda del suo valore, che va tra un range di 6 e 10. Se il messaggio passa tutti i controlli la funzione stampa ogni carattere del testo, altrimenti viene stampato a schermo un messaggio di errore e la variabile endsignal assume il valore 1. Se la variabile cmex è maggiore di zero significa che il ciclo writemex-readmex è già stato eseguito ed il gioco è attualmente in corso. In questo caso il contenuto del messaggio ricevuto varia a seconda di ciò che il client ha inviato al server tramite la funzione writemex. Il messaggio, infatti, potrebbe iniziare con il protocollo OK, END, ERR o QUIT. Quando comincia con OK il messaggio potrebbe essere "OK PERFECT\n" (parola indovinata) oppure "OK N(N) <segni>\n" (stringa di segni). In entrambi i casi si controlla che sia presente il carattere spazio dopo OK e il carattere d'invio alla fine del messaggio; dopodiché si controlla la dimensione dei messaggi. Se il messaggio ha undici caratteri e corrisponde a "OK PERFECT\n", viene stampato a schermo il messaggio di congratulazioni per aver indovinato la parola ed endsignal viene impostato ad 1. Invece se il messaggio non corrisponde a "PERFECT" esso può avere sia 11 o 12 caratteri a causa della struttura "OK N(N) <segni>\n", e di conseguenza vengono eseguiti ulteriori controlli per verificare la correttezza del messaggio. Come per il messaggio di benvenuto, si verifica il numero delle cifre del tentativo attuale (1 o 2 cifre) e la presenza dello spazio separatore tra i tentativi e la stringa di caratteri. Viene, inoltre, eseguito un confronto tra il numero del tentativo attuale e cmex, e successivamente si controlla, tramite ciclo while, che tutti i caratteri della stringa (<segni>) corrispondano a "-", "+" o "*". Similmente vengono eseguiti controlli analoghi con END e ERR. Infatti, si controlla sempre la presenza dello spazio separatore dopo il protocollo e il carattere d'invio finale presente nel messaggio, e se esso corrisponde ad "ERR <testo>\n" si imposta un puntatore che viene utilizzato alla fine della funzione per stampare il messaggio. Mentre se il messaggio è "END N(N) <parola>\n" si eseguono gli stessi controlli eseguiti su "OK N(N) <segni>\n", con le uniche differenze che i caratteri totali saranno 12 o 13 e il ciclo while andrà a controllare che tutti i caratteri della parola siano alfabetici. Con il messaggio "QUIT <messaggio>\n" basta fare i primi due controlli di base, cioè quello dello spazio separatore e dell'invio finale. Per tutti i protocolli precedenti la variabile endsignal assumerà il valore 0 solo per "OK N(N) <segni>\n", invece per tutti gli altri casi il valore sarà pari ad 1. Alla fine della funzione viene stampato a schermo il contenuto specifico del messaggio grazie all'utilizzo di alcune variabili che sono state impostate dopo il riconoscimento del messaggio stesso. In particolare, se il protocollo era OK verrà stampata la stringa dei segni, se era END o QUIT viene stampata la parola, e se il messaggio cominciava con ERR solo il messaggio di errore viene visualizzato a schermo. Infine, come per le altre funzioni, viene ritornata la variabile endsignal alla funzione main al fine di continuare o interrompere l'esecuzione del gioco.

SPECIFICHE DEL SERVER

Il programma server deve essere avviato da terminale con gli argomenti "./server <numero porta> <numero tentativi>". Dopo aver verificato il numero di argomenti viene avviata la funzione control_argv, o la funzione control_maxtry se è stato inserito un numero di tentativi facoltativo. Entrambe le funzioni controllano che l'utente abbia inserito solo caratteri in cifre, e che il valore della porta sia compreso tra 1024 e 65535. In control_maxtry si applicano gli stessi controlli al numero dei tentativi, con la differenza che quel valore deve essere compreso tra 6 e 10. Il valore di default è 6 se il numero dei tentativi non viene specificato, e come per il client tutti gli argomenti inseriti devono essere separati da uno spazio in modo da essere identificati correttamente. Una volta avviato, il programma rimane sempre attivo al fine di gestire le richieste di connessione, ma può gestirne soltanto una alla volta. Esso attende ed "ascolta", attraverso la porta inserita in input, che il programma client invii una richiesta di connessione. Quando il server la riceve e riesce a stabilire una connessione, esso selezionerà casualmente una delle 256 parole (a 5 caratteri) a sua disposizione ed avvierà un numero di cicli pari al valore di maxtry (variabile in cui viene salvato il numero dei tentativi). Nel primo ciclo il server invierà al client il messaggio di benvenuto nel formato "OK N(N) <messaggio>\n" (N(N) equivale al numero dei tentativi), mentre per tutti gli altri cicli il programma server attenderà la risposta del client, che potrà essere una parola o un messaggio di "QUIT". Una volta che l'avrà ricevuta, il server analizzerà la risposta del client tramite la funzione checkmex. Se il messaggio contiene effettivamente una parola essa verrà confrontata con quella generata dal server tramite la funzione wordscmp, altrimenti in caso di errore o di resa del giocatore il gioco verrà terminato e la connessione sarà interrotta. In ogni caso al termine delle funzioni il server invierà una risposta adeguata al client in base al risultato elaborato. Nello specifico, i risultati ottenuti dalle funzioni Checkmex e Wordscmp sono frutto delle seguenti operazioni:

- Checkmex: riceve in input il mainbuffer contenente il messaggio, insieme alla stringa word e alla variabile simpleChildSocket. La funzione si aspetta due tipi di messaggi strutturati nel seguente modo: "WORD <parola>\n" o "QUIT\n". Per entrambi i messaggi vengono eseguiti i classici controlli, ovvero il controllo delle maiuscole del protocollo, la presenza dell'invio finale e la dimensione in caratteri del messaggio (5 per QUIT e 11 per WORD). Nel caso di QUIT, una volta conclusi i controlli, basterà semplicemente scrivere nel buffer di risposta la stringa word, ovvero la parola che si doveva indovinare. Per il protocollo WORD invece bisogna eseguire altri 2 controlli: bisognerà verificare la presenza dello spazio tra il protocollo e la parola, e tramite un ciclo while si dovrà analizzare ogni carattere per verificare che i caratteri della parola siano tutti alfabetici. Se WORD passa tutti i controlli la variabile endsignal rimarrà a 0, mentre in caso di QUIT o di errore il valore verrà impostato ad 1 e di conseguenza il server invierà al client, tramite l'ausilio della variabile simpleChildSocket, il buffer di risposta contenente un messaggio finale o un messaggio di errore. Infine, endisignal verrà utilizzata come variabile di ritorno al main, e se il suo valore sarà pari a 1 la funzione principale non dovrà eseguire Wordscmp, e di conseguenza il gioco e la connessione termineranno;

- Wordscmp: confronta la parola ricevuta dal client con quella selezionata casualmente. Per eseguire il confronto la funzione riceve in input le stringhe mainbuffer e word, insieme a maxtry, ntry, e simpleChildSocket (per verificare il numero dei tentativi a disposizione e per inviare il messaggio di risposta al client). Inizialmente viene inizializzata una stringa a 5 caratteri (ssign) tutti pari a "-", dopodiché viene eseguito l'algoritmo per confrontare le parole. L'algoritmo è ottimizzato, infatti normalmente si eseguirebbero n^2 cicli (25) per confrontare tutti i caratteri, mentre in questo caso se ne eseguono molti di meno. Ogni volta che saranno presenti 2 caratteri uguali all'interno delle parole verrà scritto il carattere "+" nella stringa ssign, mentre se i caratteri uguali si trovano nella stessa posizione il carattere in ssign verrà sovrascritto con "*". I caratteri maiuscoli e minuscoli sono considerati uguali, ed il confronto viene eseguito tra il valore assoluto della loro sottrazione ed il numero 32 (che corrisponde alla distanza tra i caratteri maiuscoli e minuscoli in ASCII). Dopo aver controllato tutti i caratteri la stringa dei segni sarà composta dai caratteri "-" "+" "*", e la posizione dei segni corrisponderà ai caratteri della parola inviata dal client. Se le due parole coincidono la variabile countchar, utilizzata per contare il numero di "*" nell'algoritmo, sarà pari a 5 e perciò il server invierà al client il messaggio "OK PERFECT\n" ed endsignal verrà impostato ad 1. In caso contrario verranno confrontate le variabili ntry e maxtry per verificare il numero dei tentativi a disposizione. Se i tentativi non sono ancora terminati il programma server invierà il messaggio "OK N(N) <segni>\n", altrimenti verrà inviato "END N(N) <parola>\n" e il valore 1 sarà assegnato alla variabile endsignal. Come per tutte le altre funzioni, endsignal verrà ritornato al main ed il gioco e la connessione continueranno solo se la variabile avrà valore 0.