**RELAZIONE PER PROGETTO DI LABORATORIO DI SISTEMI OPERATIVI**

**Prof. FINZI Alberto**

**STUDENTI**

**PICCOLO Giuseppe**

**PERRETTA Alberto N86002596**

**TRACCIA A**

**1)GUIDA PER L’USO**

In questo progetto è stato sviluppato un gioco in cui i giocatori si possono spostare in una mappa in cui ci sono degli obiettivi da raccogliere e da depositare in opportuni luoghi a seconda del colore. Questo gioco ha tutte le caratteristiche richieste dalla traccia “a” e che verranno approfondite più in avanti ma c’è da precisare che bisogna usare i comandi “W” e “S” per muoversi rispettivamente in basso ed in alto ed “Invio” per scegliere le opzioni date nelle schermate che riguardano i menù del programma in questione.

**COMPILAZIONE**

Per avviare il gioco bisogna aprire il terminale nella directory del progetto, dare i permessi di esecuzione ai file **compile\_server.sh** e **compile\_client.sh** e successivamente bisogna eseguirli.

**ESECUZIONE SERVER E CLIENT**

Dopo l’esecuzione dei file, nella stessa directory verranno creati altri due file ovvero **client.out** e **server.out** che avranno già i permessi di esecuzione. Per prima cosa va eseguito il file **server.out** con un parametro che indicherà la porta in cui il server rimarrà in ascolto delle richieste del client. Quando si inserisce il parametro per il server bisogna tenere in considerazione che va messo n valore numerico superiore a 1023 perché le porte da 0 a 1023 sono riservate al sistema (Well Known). Se l’avvio del file viene effettuato correttamente allora comparirà il messaggio “Connessione al Server riuscita”.

Dopo ciò bisogna aprire un altro terminale ed eseguire anche il file **client.out** con due parametri questa volta: il primo è **l’indirizzo IP** a cui il client deve connettersi (se ci si connette dallo stesso computer bisogna mettere 0) ed il secondo è la stessa **porta del server** passata come parametro nella precedente esecuzione. Se anche questo passaggio sarà andato a buon fine si aprirà la schermata in cui si possono effettuare le azioni per l’accesso al gioco.

**SCHERMATA DI ACCESSO**

L’azione di **login** permette di accedere al gioco inserendo **account** e **password** di un giocatore già precedentemente registrato. Altrimenti se un utente vuole registrarsi può benissimo premere il tasto **Invio** su **Crea Account**, inserire **Username** e **Password**, per poi accedere con le credenziali appena inserite sempre nella sezione **Login**. Tutte le credenziali degli utenti registrati, vengono salvate in un file apposito che servirà per controllare l’inserimento delle credenziali nei successivi accessi. Ovviamente questa schermata permette anche all’utente che non vuole giocare, di uscire tramite l’opzione **Esci**. Se l’utente tenta di entrare inserendo dati diversi da quelli registrati nella sezione **Crea Account** allora il terminale segnalerà l’errore facendoti ritornare nella schermata di accesso. In seguito ad un inserimento di dati corretti da parte di un utente, uscirà il messaggio “Login effettuato con successo” e premendo il tasto **Invio** si accederà alla successiva schermata.

**SCHERMATA HOME**

In quest’ultima schermata possiamo avviare una partita premendo su **Gioca** anche se prima di effettuare la prima partita è consigliato andare nella sezione **Comandi** per l’appunto visualizzare i comandi da utilizzare nel gioco. Infine per uscire da una partita basta premere il tasto **Esc** e successivamente la schermata ti permette anche di effettuare il **Logout** portandoti nella schermata di accesso.

**2)COMUNICAZIONE CLIENT-SERVER**

Il client ed il server comunicano tramite **Socket TCP** come richiesto nella traccia "a". Per far sì che il client istauri una connessione con il server, viene utilizzata una funzione specifica chiamata **get\_connessione\_server**,implementata nei files **utility\_comunicazione\_client.h** e **utility\_comunicazione\_client.c**.

Come questa, ci sono molteplici funzioni che vengono usate per inoltrare e gestire le richieste tra Client e Server, tutte raggruppate nelle directory **utility\_comunicazione\_client** e **utility\_comunicazione\_server**. Le richieste che il client può mandare al server sono definite nell'enum **Messaggio\_client** definito nel file **messaggio\_client.h**, mentre il server potrà inviare le info che sono richieste dal client oppure potrà mandare dei messaggi al client che sono definiti nell'enum **Messaggio\_server** implementato nel file **messaggio\_server.h**.

Per la gestione di una richiesta, nel file **server.c**, che sarebbe il file in cui è implementato il server, la funzione **esegui\_server** attende nel while loop che un client si connetta tramite la funzione **accept**, poi crea un **Thread** che si occuperà di gestire la richiesta del client appena connesso, la funzione che fa da thread è **thread\_richiesta\_client**. A questo **Thread** tramite la struct **Thread\_args** definita nel file server.c, verranno passati, come parametro, il socket del client appena connesso, l'hostname del client, l'indirizzo ip del client e una variabile di tipo **Gestione\_gioco\_server**, questa struct è definita nel file **gestione\_gioco\_server.h** e si occupa di tutta la logica di gioco del server.

Il thread inizialmente chiama la funzione **kick\_off\_account\_disconnessi**, il suo funzionamento verrà spiegato dettagliatamente nel punto 3, che disconnetterà i giocatori inattivi, successivamente chiamerà la funzione **get\_richiesta\_dal\_client.** Questa funziona e definita e implementata nei files **utility\_comunicazione\_server.h** e **utility\_comunicazione\_server.c**, il suo scopo è quello di ricevere la richiesta del client. Una volta ricevuta questa richiesta, essa viene passata alla funzione **gestisci\_richiesta\_client**. In quest'ultima funzione, a seconda della richiesta inviata, verrà chiamata la funzione appropriata per soddisfare la richiesta mandata dal client. Dopo aver soddisfatto la richiesta del client e quindi, dopo che la computazione della funzione **gestisci\_richiesta\_client** termini, il thread chiude la connessione con il client tramite la funzione **close** e il thread termina l'esecuzione. Mentre il loop della funzione **esegui\_server**, dopo aver creato il thread, itera di nuovo questo processo.

C’è da notare che in una partita di questo gioco possono collegarsi al Server anche due giocatori e quindi due Client, questo qui è il numero massimo di client che si possono connettere al server, se si tenta di connettere un terzo client, quest'ultimo riceverà un errore quando tenterà di comunicare con il server. Questa scelta è stata decisa perché maggiore è il numero di client che si connettono, maggiore sarà il tempo di latenza del gioco, siccome il protocollo **TCP** ogni qualvolta che avviene una connessione con un client, ma anche una disconnessione sempre con un client, il protocollo utilizzarà la tecnica del Three Way Handshake, questo comporta un aumento della latenza.

Siccome possono connettersi due Client alla volta, in qualche modo il server deve capire con quali dei due sta comunicando, per fare ciò, ogni volta che un utente tramite client esegue l'accesso del gioco, il server assegna a quest'ultimo un identificativo rappresentato tramite un numero intero. A richieste successive di questo client, esso dovrà anche mandare il proprio identificativo. Così il server avrà la capacità di capire con chi sta comunicando. Quando un client si disconnetterà, quindi quando esso invierà una richiesta di logout, il server cancellerà l'identificativo di questo client.