PROGETTO LABORATORIO SISTEMI OPERATIVI

Firme: Valerio Panzera & Antonio Vanacore
Progetto: Traccia A

INDICE

Introduzione ……pag.2

Dettagli progetto …….pag.3

Guida all'uso..……..……..pag.4

Comunicazione..……..pag.7

Codice Server …….pag.11

Codice Client ……..pag.22

INTRODUZIONE

La seguente documentazione si pone l’obbiettivo di chiarire qualsiasi possibile dubbio sullo scopo del progetto, ed inoltre di illustrare tramite informazioni dirette (come le immagini) ed indirette i vari passi della programmazione e dell’utilizzo del software.

Il progetto si pone l’obbiettivo di realizzare un gioco il cui ambiente verrà dato da una matrice di caratteri, in cui gli utenti avranno la possibilità di spostarsi di un passo per volta nelle direzioni cardinali. Lo scopo di ogni utente sarà quello di raccogliere i pacchetti (denotati con la lettera ‘x’), e di depositarli nella locazione indicata così da acquisire più punti possibili, facendo però attenzione a non inceppare negli ostacoli invisibili. Ogni utente potrà introdursi nel gioco in qualsiasi momento della partita. Il termine della partita sarà scandito dal Timer a tempo 0h 0m 0s, esso sarà avviato nel momento in cui il primo utente farà l’accesso al gioco; ogni partita dura quindici minuti ed al finire del tempo si riavvierà una nuova partita. A fine partita sarà giudicato vincitore colui che avrà punteggio maggiore.
Il progetto prevede un sistema di registrazione e di login, ove l’utente potrà riempire i campi “username” e “password” nelle apposite barre. Le registrazioni saranno salvate in uno specifico file di testo; così come anche le azioni di qualsiasi utente verranno registrate in un secondo file di testo, dal momento in cui il client inizierà a comunicare col server.

DETTAGLI PROGETTO

Il progetto avrà come principale scopo quello di rappresentare un sistema client/server che comunica attraverso i socket. Il sistema consente a più utenti di collegarsi al server operando in modo concorrente fra loro grazie all'uso di threads che permettendo la computazione di diverse operazioni in contemporanea;

Dalla parte del server per gestire le scelte, oltre che a dei consueti if-else e diverse variabili di visibilità globale, si utilizzerà un thread contenente un consistente switch-case. Questo permetterà al server di inviare le informazioni giuste al client. Inoltre sarà presente un altro thread che si occuperà di scandire il tempo e gestire situazioni di “tempo esaurito” durante il gioco.

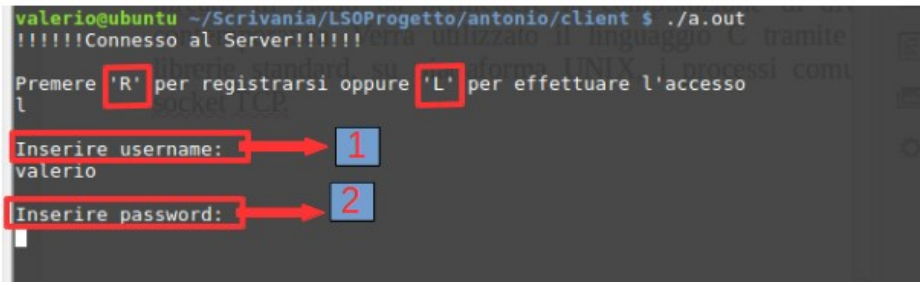
Nel client invece sono presenti due threads con due compiti distinti. Il primo si occupa di ricevere i dati in arrivo dal server e stampare a video le informazioni necessarie all'utente, mentre il secondo di leggere gli input dati dall'utente e inviarli al server.

Il linguaggio utilizzato sarà C tramite l’uso di librerie standard, su piattaforma UNIX/LINUX, mentre i processi comunicheranno tramite socket TCP.

GUIDA ALL'USO

LOGIN

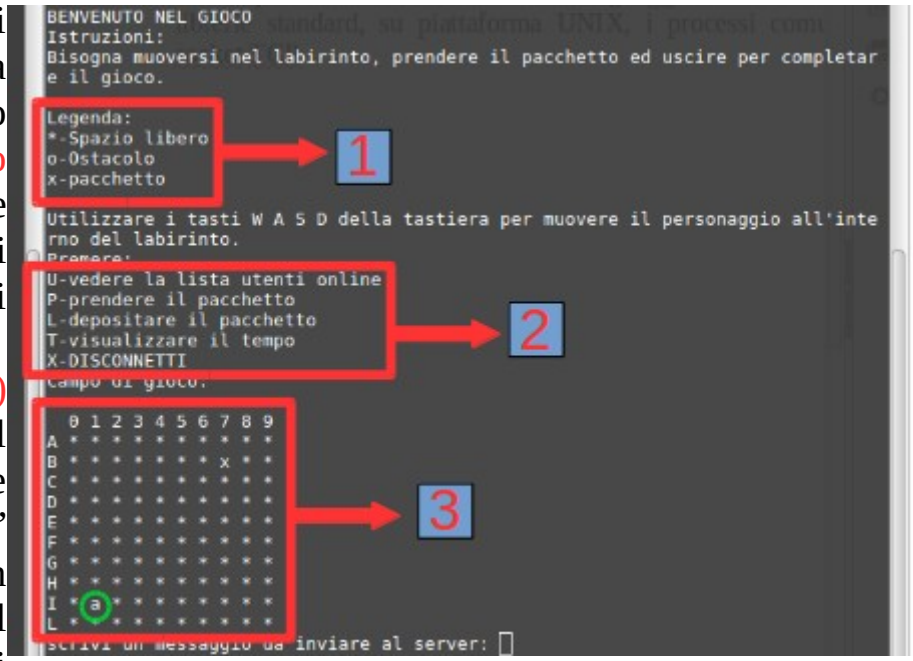
Una volta volta avviato il gioco vi comparirà una semplice schermata di login ove vi verrà chiesto di premere il tasto ‘R’ in caso voleste registrarvi, oppure di premere il tasto ‘L’ per fare il login.



Una volta premuto ‘L’, dovrete inserire username (1) e password (2). Se effettuerete il login in maniera errata, allora sarete ammoniti da un messaggio.

CAMPO GIOCO

Se avrete compilato i campi precedenti in maniera corretta allora avrete accesso al gioco. Il campo di gioco (3) ha una forma quadrata e sarà possibile muoversi all’interno di esso tramite i tasti WASD.



Vi apparirà la Legenda (1) con tutte le spiegazioni sul campo: come noterete dall’immagine gli ‘*’ indicano gli spazi liberi in cui potrete effettuare il movimento, però state attenti perché all’interno del campo sono nascosti degli ostacoli ‘o’ che vi rallenteranno. I pacchetti da raccogliere saranno indicati con la lettera ‘x’, mentre voi sarete indicati da un lettera minuscola dell’alfabeto italiano, in questo specifico caso dalla ‘a’.

In basso, dopo la prima legenda potrai trovarne una seconda (2), ove sono spiegati i vari tasti da utilizzare.

MULTIGIOCATORE

Il gioco è programmato per uno o più utenti ed ogni giocatore può collegarsi in qualsivoglia momento. Come te, anche gli altri giocatori hanno i tuoi medesimi obbiettivi; quindi depositare quanti più pacchi possibili in modo da incrementare il proprio punteggio.

Premendo il tasto ‘U’ puoi rimanere aggiornato sui vari giocatori online tramite una lista (1).

TEMPO

Ogni partita ha una durata massima di 15min e ricordati sempre che puoi rimanere aggiornato sul tempo residuo (1) tramite il tasto ‘T’.

COMUNICAZIONE

COLLEGAMENTO CLIENT-SERVER

La fase di collegamento è la prima fase della comunicazione che andremo a vedere una volta avviato il programma. Una volta avviato il Server, esso farà presente la sua disponibilità a ricevere Client. D'altro canto il client dovrà specificare l'IP e la porta del Server tramite riga di comando, così come illustrato in figura.

```
valerio@ubuntu ~/Scrivania/LSOProgetto/Server/src $ ./a.out
In attesa di richieste da parte di un client...
```

```
valerio@ubuntu ~/Scrivania/LSOProgetto/Client/src $ ./a.out <IP> <porta>
```

LOGIN

La fase di login è la fase che viene eseguita subito dopo la fase di collegamento. In particolar modo, in questa fase distinguiamo due parti: la registrazione dettata dalla scelta 'R', ed l'accesso tramite la pressione del tasto 'L'.

```
179
180 if(a=='R'){
181     write(socketClientDescriptor, &a, sizeof(a));                    1
182     printf("\nScegliere l'username: \n");
183     scanf("%s", username);
184     k=strlen(username);
185     username[k] = '-';
186     username[k+1] = '\0';
187
188         printf("\nScegliere una password: \n");
189         scanf("%s", password);
190
191     printf("\nRipetere password: \n");
192     scanf("%s", confronto);
193
194     if(strcmp(password,confronto) == 0){                             2
195         char t='Y';
196         write(socketClientDescriptor, &t, sizeof(t));
197         strcat(username,password);
198         write(socketClientDescriptor, username, sizeof(username));
199     }else{
200             char t='N';
201             printf("\nConfronto password errato, ripetere!\n");
202             write(socketClientDescriptor, &t, sizeof(t));            3
203     }
```

Nella fase di registrazione, come vediamo dall'immagine, per prima cosa il Client avvisa il Server della scelta dell'utente tramite una write() (1), in secondo luogo fa un confronto (2) dei dati immessi nel campo "Password" dall'utente, in modo da impedire errori accidentali di digitazione, se ciò avvenisse bisogna avvisare (3) il Server che dovrà fermare la procedura ed avviarne una nuova.

Similmente anche nel caso di accesso dobbiamo prima avvisare (4) il Server della scelta fatta, per adattarlo alla risposta adeguata.

```
206
207         write(socketClientDescriptor, &a, sizeof(a));                4
208         printf("\nInserire username: \n");
209         scanf("%s", username);
210
211         k = strlen(username);
212         username[k] = '-';
213         username[k+1] = '\0';
214         printf("\nInserire password: \n");
215         scanf("%s", password);
216         strcat(username,password);
217         write(socketClientDescriptor, username, sizeof(username));   5
218         read(socketClientDescriptor,&proseguo,sizeof(proseguo));
219
220 if(proseguo==0){
221     printf("\nDati errati: username o password errati!\n");
222 }
223
224     }else{
225         printf("\nScelta errata, ripetere!\n");
226     }
227
228 }//fine while del sistema di registrazione
```

Come evidenziato dalle immagini, nella riga in cui vengono inviati (5) i dati inerenti alla registrazione, si è preferito accorpare le stringhe di "username" e "password", per evitare accodamenti di

messaggi che potenzialmente rendono meno robusto il sistema di comunicazione.

Dal canto suo, il Server, per prima cosa, si occuperà di leggere la scelta dell'utente tramite una read() (6) così da entrare nella corrispondente condizione. Nel caso di 'L', allora si occuperà di un controllo, mediante la funzione check() (7), nell'apposito file di testo per verificare l'autenticità dell'utente e nel caso il controllo abbia riscontro positivo, verrà inviato (8) un valore di tipo 'flag' che darà accesso all'utente alla fase successiva.
Nel caso l'utente abbia scelto di registrarsi allora, tramite la funzione registra() (9), verrà salvato l'username nel file precedentemente citato.

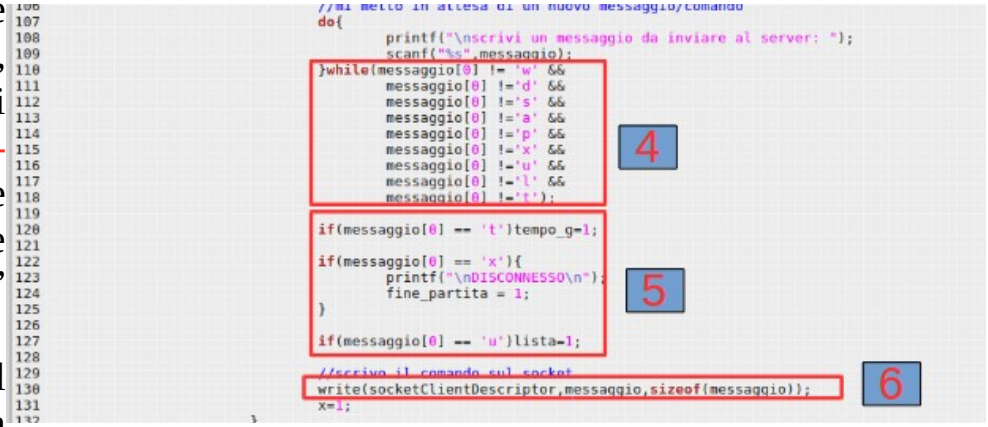
GIOCO

L'intero sistema di comunicazione durante la fase di gioco, è orientato dal lato Server, da un consistente switch-case. Esso potrebbe essere definito il nucleo di questa fase. Ma nonostante la sua dimensione, esso è facilmente riassumibile in poche parti in quanto i "case" di movimento, che costituiscono una buona parte delle scelte, sono simili.
Ogni case di movimento, ha come prima istruzione due funzioni (1) molto simili fra loro che hanno lo scopo 'trovare' il personaggio corrente così da riportarne le coordinate che serviranno in seguito per entrare nel corrispondente if-else. Gli if-else sono facilmente intuibili poiché, in sintesi, si occupano di spostare il personaggio da una casella del campo di gioco, ad un altra. Nel caso la cella in questione sia diversa da '*' o da 'x', il movimento del personaggio sarà nullo.

Per riconoscere la cella del pacchetto, si è deciso di salvare i dati del pacchetto nel momento della creazione e di confrontali tramite un if (2) con le coordinate del personaggio acquisite nell'istruzione precedente.

Una volta effettuato l'aggiornamento della matrice, il Server si occuperà di aggiornare il Client delle modfiche con una write() (3) a fine delle condizioni if-else.

Il lato Client, durante l'invio dei messaggi, utilizza un sistema di controllo dato da un do-while (4) che impedisce all'utente di inviare messaggi non "previsti" al Server.



In seguito, se il messaggio inviato, non ha il fine di muovere il personaggio verrà salvata la richiesta dell'utente tramite delle variabili globali (5), che hanno il fine di ricevere il messaggio correttamente, come vedremo nella figura successiva. Infine il messaggio, viene semplicemente inviato con una write() (6).

La stampa del campo di gioco è designata ad un considerevole ciclo for() (7), che conta fino a centoventuno iterazioni per una matrice 11*11.

Tale ciclo è suddiviso in diversi if-else che gestiscono in maniera minuziosa ogni singolo carattere della cella della matrice. In maniera particolare notiamo l'if-else a riga 57 che si occupa di stampare un carattere diverso da quello ricevuto, in quanto, come visto nel lato Server, le celle con carattere 'i' sono ostacoli 'o' non nascosti.

Infine il lato Client si conclude con la

distinzione dei messaggi da leggere tramite delle condizioni di if-else (8). Ogni condizione dipende dalle variabili booleane globali che vengono settate nel momento in cui il Client invia il messaggio, così da predisporlo ai prossimi messaggi inviati dal Server; in quanto tali messaggi variano di tipologia: dal semplice carattere ad interi.

CODICE SERVER

```
/*
 ============================================================================
 Name        : Server.c
 Author      : Antonio Vanacore & Valerio Panzera
 Version     :
 Copyright   : Your copyright notice
 Description : Uso dei Socket TCP in C
 ============================================================================
 */

//compilare usando gcc nomefile.c -lpthread
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <errno.h>
#include <pthread.h>
#include <time.h>

#define MAX 11 //dimensione matrice
#define PORTA 1205 //porta sulla quale il server è in ascolto

typedef struct{
	int descrittore;
	char indirizzo[100];
}parametriClient;

parametriClient p[100]; //variabile globale che rappresenta tutti i client che si connettono
int numeroClient=0; //variabile globale,indice dei client connessi
int clientConnessi=0; //var globale che indica attualmente il numero di client connessi
char campoGioco[MAX][MAX]; //variabile globale, tutti i client lavorano sullo stesso campo
char personaggi[]={'a','b','c','d','e','f','g','h','l','m','n','p','q','r','s','t','u','v','z'};
int personaggioCorrente=0; //indica il personaggio a cui siamo arrivati
int ore = 0, minuti = 1, secondi = 45; //var globale tempo
char username_globali[124]; //ci inserisco gli user dei giocatori online
int q = 0; //indice variabile username_globali
int vittoriaVettore[100];
int punteggioClient[100];//ci memorizzo i punteggi dei vari client in gioco

void creaCampo(char m[MAX][MAX]);
void creaOstacoli(char m[MAX][MAX]);
void creaPacco(char m[MAX][MAX]);
char creaPersonaggio(char m[MAX][MAX]);
int ipos(char m[][MAX],char personaggio);
int jpos(char m[][MAX],char personaggio);
int check(char utente[]);
void registra(char utente[]);
int cercaPersonaggio(char personaggio);
void logging_log(char ip[]);
void logging_exit(char ip[]);
char *salva_username(char username[]);
void cancella_online(char username[]);
void logging_posato(char username[], int i, int j, int punteggio);
void logging_preso(char username[], int i, int j, int punteggio);
void resetPersonaggiConnessi();
void inviaVittoria(int id);
void azzeraPunteggi();
void inviaVittoriaTempo();

void *timerThread(void *arg){
	while(1){
		if(secondi < 0){
			secondi = 59;
			--minuti;
```

```
                }
                if(minuti < 0){
                        inviaVittoriaTempo();
                        minuti=1;
                        secondi=45;
                        resetPersonaggiConnessi();
                        azzeraPunteggi();
                }
                sleep(1);
                --secondi;
        }
}

void *gestioneClientThread(void * arg){
                char buffer[121];
                int idClient=numeroClient-1; //client associato a qst thread
                char username[30];
                char nome[20];
                int esiste=0;//controllo dei dati utenti
                char accesso;
                int proseguo =0;//sistema di registrazione
                        char personaggioClient;//personaggio associato a client in questione
                        int locazioneI=0;
                        int locazioneJ=0; //coordinate della locazione di arrivo di un pacco
                        char ultimoCarattere = '*'; //variabile temporanea per la disconnessione
                        int disconnesso=0;
                        char personaggiConnessi[MAX]; //vettore contenente tutti i caratteri connessi
                        int k=0;
                        punteggioClient[idClient]=0; //numero pacchi posati
                        char *user;

                        printf("**Richiesta connessione dal client %s\n",p[idClient].indirizzo);

                /////REGISTRAZIONE/////
                while(proseguo==0){
                        read(p[idClient].descrittore,&accesso,sizeof(accesso));
                   printf("-[%s]Opzione di accesso selezionata: '%c'\n",p[idClient].indirizzo,accesso);
                   if(accesso == 'L'){
                        read(p[idClient].descrittore,username,sizeof(username));
                      proseguo=check(username);
                      write(p[idClient].descrittore,&proseguo, sizeof(proseguo));
                   }else if(accesso == 'R'){
                        char t;
                      read(p[idClient].descrittore, &t,sizeof(t));
                      if(t=='Y'){
                        read(p[idClient].descrittore,username,sizeof(username));
                         registra(username);
                      }
                    }else{
                          printf("\nPremere 'L' o 'R'!\n");
                     }
                }//fine while del sistema di registrazione

                printf("***Connessione riuscita con %s\n",p[idClient].indirizzo);

                //inserisco il nome del personaggio nella variabile globale
                user=salva_username(username);
                for(int q=0;q<20;q++){
                        nome[q]=user[q];
                }

                //genero personaggio e pacco,ogni client è diverso e salvo il carattere del client
                        personaggioClient=creaPersonaggio(campoGioco);
                        creaPacco(campoGioco);

                        printf("-[%s]ID client: %d\n",p[idClient].indirizzo,idClient);
                        printf("-[%s]User: %s\n",p[idClient].indirizzo,nome);
                        printf("-[%s]Personaggio assegnato al client: %c\n",p[idClient].indirizzo,personaggioClient);
                        printf("*Client connessi: %d\n",clientConnessi);

                        logging_log(nome);
                        vittoriaVettore[idClient]=0; //lo pongo a zero altrimenti se entro
                                                                //subito dopo che qlkn ha vinto mi
                                                        //appare il quadro di sconfitta

                        //dopo la connessione con il client invio subito il
                        //quadro di gioco il tempo  e il punteggio attuale solo al client connesso
                        write(p[idClient].descrittore,&minuti,1);
                        write(p[idClient].descrittore,&secondi,1);
```

```
            write(p[idClient].descrittore,&punteggioClient[idClient], 1);
            write(p[idClient].descrittore,&vittoriaVettore[idClient],1);
            write(p[idClient].descrittore,campoGioco,121);

            printf("Attendo richieste...\n");

while(disconnesso==0){
                //poi dopo aspetto di leggere il porssimo comando
                read(p[idClient].descrittore,buffer,1);
                printf("\n-[%s][ID: %d]Messaggio ricevuto: %s \n",p[idClient].indirizzo,idClient,buffer);

                int i,j; //coordinate del del personaggio
                int paccoI,paccoJ; //coordinate del pacco
                int scelta;

                scelta = buffer[0];

                //se il comando è minuscolo, allora lo rendo maiuscolo
                if(scelta >= 'a' && scelta <= 'z'){
                        scelta = scelta-32;
                }

                switch(scelta){
                        case 'A':
                                i = ipos(campoGioco,personaggioClient);
                                j = jpos(campoGioco,personaggioClient);
        if(campoGioco[i][j-1] == '*'){
                                        if((i==paccoI && j==paccoJ) ){
                                                ultimoCarattere = campoGioco[i][j-1];
                                                campoGioco[i][j] = 'x';
                                                campoGioco[i][j-1] = personaggioClient;
            }else{
              ultimoCarattere = campoGioco[i][j-1];
              campoGioco[i][j] = '*';
              campoGioco[i][j-1] = personaggioClient;
            }
                                }else if(campoGioco[i][j-1] == 'x'){
                                        ultimoCarattere = campoGioco[i][j-1];
                                        paccoI = i, paccoJ = j-1;
            campoGioco[i][j] = '*';
                                        campoGioco[i][j-1] = personaggioClient;
                                 }else if(campoGioco[i][j-1] == 'o'){
                                        campoGioco[i][j-1] = 'i';
                                 }

                                write(p[idClient].descrittore,&minuti,1);
                                write(p[idClient].descrittore,&secondi,1);
                                write(p[idClient].descrittore,&punteggioClient[idClient], 1);
                                write(p[idClient].descrittore,&vittoriaVettore[idClient],1);
                                write(p[idClient].descrittore,campoGioco, 121);
                                printf("-[%s]Matrice inviata\n",p[idClient].indirizzo);

                                break;

                        case 'D':
                                i = ipos(campoGioco,personaggioClient);
                                j = jpos(campoGioco,personaggioClient);
        if(campoGioco[i][j+1] == '*'){
                                        if((i==paccoI && j==paccoJ) ){
                                                ultimoCarattere = campoGioco[i][j+1];
                                                campoGioco[i][j] = 'x';
                                                campoGioco[i][j+1] = personaggioClient;
            }else{
              ultimoCarattere = campoGioco[i][j+1];
              campoGioco[i][j] = '*';
              campoGioco[i][j+1] = personaggioClient;
            }
                                }else if(campoGioco[i][j+1] == 'x'){
                                        ultimoCarattere = campoGioco[i][j+1];
                                        paccoI = i, paccoJ = j+1;
            campoGioco[i][j] = '*';
                                        campoGioco[i][j+1] = personaggioClient;
                                 }else if(campoGioco[i][j+1] == 'o'){
                                        campoGioco[i][j+1] = 'i';
                                 }

                                write(p[idClient].descrittore,&minuti,1);
                                write(p[idClient].descrittore,&secondi,1);
                                write(p[idClient].descrittore,&punteggioClient[idClient], 1);
```

```
                        write(p[idClient].descrittore,&vittoriaVettore[idClient],1);
                        write(p[idClient].descrittore,campoGioco, 121);
                        printf("-[%s]Matrice inviata\n",p[idClient].indirizzo);

                        break;

                case 'W':
                        i = ipos(campoGioco,personaggioClient);
                        j = jpos(campoGioco,personaggioClient);
if(campoGioco[i-1][j] == '*'){
                                if((i==paccoI && j==paccoJ) ){
                                        ultimoCarattere = campoGioco[i-1][j];
                                        campoGioco[i][j] = 'x';
                                        campoGioco[i-1][j] = personaggioClient;
   }else {
      ultimoCarattere = campoGioco[i-1][j];
       campoGioco[i][j] = '*';
                                        campoGioco[i-1][j] = personaggioClient;
    }
                        }else if(campoGioco[i-1][j] == 'x'){
                                ultimoCarattere = campoGioco[i-1][j];
                                paccoI = i-1, paccoJ = j;
   campoGioco[i][j] = '*';
                                campoGioco[i-1][j] = personaggioClient;
                         }else if(campoGioco[i-1][j] == 'o'){
                                campoGioco[i-1][j] = 'i';
                         }

                        write(p[idClient].descrittore,&minuti,1);
                        write(p[idClient].descrittore,&secondi,1);
                        write(p[idClient].descrittore,&punteggioClient[idClient], 1);
                        write(p[idClient].descrittore,&vittoriaVettore[idClient],1);
                        write(p[idClient].descrittore,campoGioco, 121);
                        printf("-[%s]Matrice inviata\n",p[idClient].indirizzo);

                        break;

                case 'S':
                        i = ipos(campoGioco,personaggioClient);
                        j = jpos(campoGioco,personaggioClient);
if(campoGioco[i+1][j] == '*'){
                                if((i==paccoI && j==paccoJ) ){
                                        ultimoCarattere = campoGioco[i+1][j];
                                        campoGioco[i][j] = 'x';
                                        campoGioco[i+1][j] = personaggioClient;
   }else {
      ultimoCarattere = campoGioco[i+1][j];
      campoGioco[i][j] = '*';
                                        campoGioco[i+1][j] = personaggioClient;
    }
                        }else if(campoGioco[i+1][j] == 'x'){
                                ultimoCarattere = campoGioco[i+1][j];
                                paccoI = i+1, paccoJ = j;
   campoGioco[i][j] = '*';
                                campoGioco[i+1][j] = personaggioClient;
                         }else if(campoGioco[i+1][j] == 'o'){
                                campoGioco[i+1][j] = 'i';
                         }

                        write(p[idClient].descrittore,&minuti,1);
                        write(p[idClient].descrittore,&secondi,1);
                        write(p[idClient].descrittore,&punteggioClient[idClient], 1);
                        write(p[idClient].descrittore,&vittoriaVettore[idClient],1);
                        write(p[idClient].descrittore,campoGioco, 121);

                        printf("-[%s]Matrice inviata\n",p[idClient].indirizzo);

                        break;

                case 'P':
                        i = ipos(campoGioco,personaggioClient);
                        j = jpos(campoGioco,personaggioClient);
if(i==paccoI && j == paccoJ){
      printf("-[%s]Pacco preso!\n",p[idClient].indirizzo);
      logging_preso(nome, i, j, punteggioClient[idClient]);
   paccoI=paccoJ=0;
   do{
      locazioneI=rand()%(10)+1;
       locazioneJ=rand()%(10)+1;
```

```
   }while(campoGioco[locazioneI][locazioneJ]== 'o' );

}else{
      printf("-[%s]Nessun pacco da prendere\n",p[idClient].indirizzo);
}

write(p[idClient].descrittore,&minuti,1);
write(p[idClient].descrittore,&secondi,1);
write(p[idClient].descrittore,&punteggioClient[idClient], 1);
                        write(p[idClient].descrittore,&vittoriaVettore[idClient],1);
if( vittoriaVettore[idClient] != 1 )write(p[idClient].descrittore,&locazioneI, 1);
if( vittoriaVettore[idClient] != 1 )write(p[idClient].descrittore,&locazioneJ, 1);
write(p[idClient].descrittore,campoGioco, 121);

printf("-[%s]Matrice inviata\n",p[idClient].indirizzo);

                                break;

                        case 'L':
                                i = ipos(campoGioco,personaggioClient);
                                j = jpos(campoGioco,personaggioClient);
                                int preso=0;
                                if(i==locazioneI && j==locazioneJ){
                                        printf("-[%s]Pacco lasciato!\n",p[idClient].indirizzo);
                                        preso=1;
                                        punteggioClient[idClient]++;
                                        logging_posato(nome, i, j, punteggioClient[idClient]);
                                        printf("-[%s]Punteggio attuale: %d\n",p[idClient].indirizzo,punteggioClient[idClient]);
                                        locazioneI=locazioneJ=0;
                                        if(punteggioClient[idClient]<3){
                                                creaPacco(campoGioco);
                                        }else{
                                                azzeraPunteggi();
                                                printf("-[%s]Il client %d ha vinto la partita!\n",p[idClient].indirizzo,idClient);
                                                inviaVittoria(idClient);
                                                resetPersonaggiConnessi();
                                                minuti=5;
                                                secondi=5;
                                        }
                                }else{
                                        if(locazioneI==0 && locazioneJ==0 ){
                                                preso=2;
                                                printf("-[%s]Nessun pacco da lasciare.\n",p[idClient].indirizzo);
                                        }else{
                                                preso=3;
                                                printf("-[%s]Locazione sbagliata\n",p[idClient].indirizzo);
                                        }
                                }

                                write(p[idClient].descrittore,&minuti,1);
                                write(p[idClient].descrittore,&secondi,1);
                                write(p[idClient].descrittore,&punteggioClient[idClient],1);
                                write(p[idClient].descrittore,&vittoriaVettore[idClient],1);
                                if( vittoriaVettore[idClient] != 1 && vittoriaVettore[idClient] !=2 )write(p[idClient].descrittore,&preso, 1);
                                write(p[idClient].descrittore,campoGioco, 121);

                                printf("-[%s]Matrice inviata\n",p[idClient].indirizzo);

                                preso=0;

                                break;

                        case 'U':
                                /*RICHIESTA LISTA Client connessi*/
                                write(p[idClient].descrittore,username_globali,124);
                                write(p[idClient].descrittore,&minuti,1);
                                write(p[idClient].descrittore,&secondi,1);
                                write(p[idClient].descrittore,&punteggioClient[idClient], 1);
                                write(p[idClient].descrittore,&vittoriaVettore[idClient],1);
                                write(p[idClient].descrittore,campoGioco, 121);

                                printf("-[%s]Matrice inviata\n",p[idClient].indirizzo);

                                break;

                        case 'X':
                                /*RICHIESTA DISCONNESSIONE ACCOUNT*/
                                i = ipos(campoGioco,personaggioClient);
                                j = jpos(campoGioco,personaggioClient);
```

```
                                        printf("**[%s]Richiesta disconnessione\n",p[idClient].indirizzo);
                                        campoGioco[i][j] = ultimoCarattere;
                                        campoGioco[ipos(campoGioco,'x')][jpos(campoGioco,'x')]='*';
                                        logging_exit(nome);
                                            cancella_online(username);
                                            clientConnessi--;
                                            punteggioClient[idClient]=0;
                                            disconnesso=1;
                                    printf("***[%s]Disconnesso.\n",p[idClient].indirizzo);

                                            break;

                                      default:

                                            break;

                                      }//fine switch
                                      vittoriaVettore[idClient]=0;
                                      printf("Attendo altre richieste...\n");
                }//fine while dei comandi(cioè accetto sempre nuovi comandi finchè nn finisco il gioco)
            //se è arrivata la richiesta di disconnessione chiudo la comunicazione
            close(p[idClient].descrittore);
}//fine funzioneThread

int main(int argc, char **argv) {
        struct sockaddr_in indirizzoServer; //serve per mettersi in ascolto su una determinata porta
        struct sockaddr_in indirizzoClient; //serve per contenere le informzioni sulla connessione che mi arrivano dal client
        int socketDescriptor; //informazioni del server
        int clientConnectionDescriptor; //info sul client arrivate tramite l ACCEPT
        pthread_t th1; //thread dedicato al timer di gioco
        pthread_t th2; //thread dedicato al gestione di ogni client

        //genero il campo da gioco che deve essere uguale per tutti i client
    creaCampo(campoGioco);

    //genero gli ostacoli uguali per ogni client
    creaOstacoli(campoGioco);

        //creo il socket
        socketDescriptor=socket(AF_INET,SOCK_STREAM,0);//creo il socket(famiglia,tipo,protocollo(zero per protocollo migliore disp))
        if(socketDescriptor<0)perror("Errore creazione del socket!"),exit(1);

        indirizzoServer.sin_family=AF_INET;
        indirizzoServer.sin_addr.s_addr=htonl(INADDR_ANY);
        indirizzoServer.sin_port=htons(PORTA);

        //collego il socketDescriptor alla struttura indirizzoServer in cui ci sono i dati per comunicare
        if(bind(socketDescriptor,(struct sockaddr *)&indirizzoServer,sizeof(indirizzoServer))<0)
                perror("Errore BIND:"),exit(0);

        printf("In attesa di richieste da parte di un client...\n");

        //metto in ascolto il socket(sd,lunghezzaCoda);
        listen(socketDescriptor,10);

        while(1){
                int lunghezzaClient;

                lunghezzaClient=sizeof(indirizzoClient);

                //accetto connessioni all infinito
                clientConnectionDescriptor=accept(socketDescriptor,(struct sockaddr *)&indirizzoClient,&lunghezzaClient);
                if(clientConnectionDescriptor<0){
                        perror("Errore ACCEPT del server");
                        exit(0);
                }

                //memorizzo i dati del client connesso nella struttura globale
                p[numeroClient].descrittore=clientConnectionDescriptor;
                strcpy(p[numeroClient].indirizzo,inet_ntoa(indirizzoClient.sin_addr));
                numeroClient++;
                clientConnessi++;

                //creo il thread per il timer solo dopo la connessione del primo client
                if(numeroClient==1){
                                pthread_create(&th1,NULL,timerThread, NULL);
                }

                //creo il thread per gestire il singolo client
```

```
		pthread_create(&th2,NULL,gestioneClientThread,NULL);
	}//fine while delle connessioni(cioè accetto sempre connessioni)
	return 0;
}

//pone a 0 i punteggi di tutti i client connessi
void azzeraPunteggi(){
	int i;
	for(i=0;i<100;i++)
		punteggioClient[i]=0;

}

void inviaVittoriaTempo(){
	int z;
	int punteggioMax=-1;
	int tempId[100];
	int s=0;

	//trovo il punteggio piu alto al momento
	for(z=0;z<100;z++){
		if(punteggioMax<punteggioClient[z]){
			punteggioMax=punteggioClient[z];
		}
	}

	//trovo tutti i client che hanno il punteggio = a punteggioMax
	for(z=0;z<100;z++){
		if(punteggioClient[z]== punteggioMax){
			tempId[s]=z;
			s++;
		}
	}

	//pongo gli indicatori di vittoria tutti a 3
	for(z=0;z<100;z++){
		 vittoriaVettore[z]=3; //non ha vinto
	}

	for(z=0;z<s;z++){
		vittoriaVettore[ tempId[z] ]=4;    //ha vinto
	}
}

//pongo a 1 gli indicatori di ogni client tranne qll che ha vinto
void inviaVittoria(int id){
	int x;
	for(x=0;x<100;x++){
		if(x==id){
			vittoriaVettore[x]=2;
		}else{
			vittoriaVettore[x]=1;
		}
	}
}

//genero una matrice 11x11 dove la riga e la colonna zero rappresentano le coordinate
void creaCampo(char m[MAX][MAX]){

	int i,j;

	//tutte le celle sono *
	for(i=0; i<MAX; i++){
		for(j=0; j<MAX; j++){
			m[i][j] = '*';
		}
	}

	m[0][0]=' ';
	m[1][0]='A';
	m[2][0]='B';
	m[3][0]='C';
	m[4][0]='D';
	m[5][0]='E';
	m[6][0]='F';
	m[7][0]='G';
	m[8][0]='H';
	m[9][0]='I';
	m[10][0]='L';
```

```
	m[0][1]='0';
	m[0][2]='1';
	m[0][3]='2';
	m[0][4]='3';
	m[0][5]='4';
	m[0][6]='5';
	m[0][7]='6';
	m[0][8]='7';
	m[0][9]='8';
	m[0][10]='9';

}

//genero 10 ostacoli casuali
void creaOstacoli(char m[][MAX]){

	int i;
	int j;
	int cont = 0;

	while(cont<10){
		i=rand()%(MAX-1)+0;
		j=rand()%(MAX-1)+0;
		if(m[i][j] == '*'){
			m[i][j] = 'o';
			cont++;
		}
	}
}

char creaPersonaggio(char m[][MAX]){
	int i;
	int j;
	int cont = 0;
	char temp;
	while(cont<1){
		i=rand()%(MAX-1)+1;
		j=rand()%(MAX-1)+1;
		if(m[i][j] == '*'){
			int x=0;//
			temp= personaggi[x];
			while( cercaPersonaggio(temp) > 0 ){
				x++;
				temp=personaggi[x];
			}
			cont++;
		}
	}
	return m[i][j]=temp;
}

void creaPacco(char m[][MAX]){
	int i=0,j=0;
	int cont =0;

	while(cont<1){
			i=rand()%(MAX-1)+0;
			j=rand()%(MAX-1)+0;
			if(m[i][j] == '*'){
				m[i][j] = 'x';
				cont++;
			}
	}
}

//ritorna la riga del personaggio dato
int ipos(char m[][MAX],char personaggio){
	int i;
	int j;

	for(i=0; i<MAX; i++){
		for(j=0; j<MAX; j++){
			if(m[i][j] == personaggio){
				return i;
			}

		}
	}
}
```

```
//ritorna la colonna del personaggio dato
int jpos(char m[][MAX],char personaggio){

	int i;
	int j;

	for(i=0; i<MAX; i++){
		for(j=0; j<MAX; j++){
			if(m[i][j] == personaggio){
				return j;
			}

		}
	}
}

void stampa(char m[][MAX]){

	int i;
	int j;

	for(i=0; i<MAX; i++){
		for(j=0; j<MAX; j++){
			printf(" %c ", m[i][j]);

			}
			printf("\n");
		}

}

int check(char utente[]){

	int i=0,j=0;
	FILE *fp;
	char lettura[20][20];
	int dim;

	fp=fopen("utenti.txt","r"); //apro il file in lettura
	if(fp){
		while(!feof(fp)){

			fscanf(fp, "%s", lettura[i]);
			i++;

		}//fine while della lettura file

	}else{
		printf("Errore apertura file!");
	 }//fine controllo esistenza file

	fclose(fp);


	int res = 0;
	dim = i;
	for (i = 0; i < dim; i++) //controllo riga per riga se i dati sono presenti nel file
		if (strcmp(utente, lettura[i]) == 0){
	  res = 1;
	  return res;
	}

	  return res;

}

void registra(char utente[]){
   FILE *fp;
   fp=fopen("utenti.txt","a"); //apro il file
   if(fp){
        fprintf(fp, "%s\n", utente);
   }else{
      printf("\nErrore nella scrittura del file!\n");
   }
   fclose(fp);
}

//controllo se è presente tra i personaggi nella matrice il personaggio dato
```

```
int cercaPersonaggio(char personaggio){
	int i;
	int j;
	int x=0;
	for(i=1; i<MAX; i++){
		for(j=1; j<MAX; j++){
			if(personaggio == campoGioco[i][j]){
				x=1;
			}
		}
	}
	return x;
}

//qnd viene riavviata una partita viene resettato il campo
void resetPersonaggiConnessi(){
	int i,j;
	int x=0; //contatore dei char dentro personaggi[]
	int z=0; //contatore dei char connessi
	char temp[20];

	//cerco i personaggi ancora connessi (cioè presenti in matrice)
	//e li salvo tutti in un array temporaneo
	for(x=0;x<20;x++){
		for(i=1; i<MAX; i++){
			for(j=1; j<MAX; j++){
				if(campoGioco[i][j]==personaggi[x]){
					temp[z]=personaggi[x];
					z++;
					j=100;
					i=100;
				}
			}
		}
	}
	creaCampo(campoGioco);

	//inserisco i personaggi salvati nel nuovo campo
	for(i=0;i<z;i++){
		int cI,cJ;
		int cont=0;
		do{
			cI=rand()%(MAX-1)+0;
			cJ=rand()%(MAX-1)+0;
			if(campoGioco[cI][cJ] == '*'){
				campoGioco[cI][cJ] = temp[i];
				cont++;
			}
		}while(cont<1);
	}
	creaOstacoli(campoGioco);

	for(i=0;i<z;i++){
		creaPacco(campoGioco);
	}
}

void logging_log(char username[]){
   FILE *fp;
   time_t ora;
   ora = time(NULL);
   fp=fopen("logging.txt","a"); //apro il file

   if(fp){
        fprintf(fp, "Login: %s %s\n", username,asctime(localtime(&ora)));
   }else{
      printf("\nErrore nella scrittura del file!\n");
   }
   fclose(fp);
}

void logging_exit(char username[]){
   FILE *fp;
   time_t ora;
   ora = time(NULL);
   fp=fopen("logging.txt","a"); //apro il file
   if(fp){
        fprintf(fp, "Exit: %s %s\n", username,asctime(localtime(&ora)));
   }else{
```

```
        printf("\nErrore nella scrittura del file!\n");
    }
    fclose(fp);
}

void logging_posato(char username[], int i, int j, int punteggio){

        FILE *fp;
    time_t ora;
    ora = time(NULL);
    fp=fopen("logging.txt","a"); //apro il file
    if(fp){
         fprintf(fp, "Nome utente: %s Azione: posa pacco Locazione: %d,%d Punteggio: %d\n", username,i,j,punteggio);
    }else{
        printf("\nErrore nella scrittura del file!\n");
    }
    fclose(fp);
}

void logging_preso(char username[], int i, int j, int punteggio){

        FILE *fp;
    time_t ora;
    ora = time(NULL);
    fp=fopen("logging.txt","a"); //apro il file
    if(fp){
         fprintf(fp, "Nome utente: %s Azione: prendi pacco Locazione: %d,%d Punteggio: %d\n", username,i,j,punteggio);
    }else{
        printf("\nErrore nella scrittura del file!\n");
    }
    fclose(fp);
}

char *salva_username(char username[]){
        int i=0;
        char *ptr;
        char nome[20];
        while(username[i] != '-'){
                username_globali[q] = username[i]; //array di username online

                nome[i] = username[i]; //distinguo l'username dalla password
                q++;
                i++;
        }

        nome[i] = '\0';
        username_globali[q] = ',';
        q++;
        username_globali[q] = '\0';
        ptr=nome;
        return ptr;
}

void cancella_online(char username[]){

        int i=0, j=0, tmp;
        int delete = 0;

        while(username_globali[i] != '\0' && delete != 1){ //cancella l'username dalla lista
                if(username_globali[i] == username[j]){
                        tmp = i;
                }
                while(username_globali[i] == username[j]){
                        i++;
                        j++;

                        if(username_globali[i] == ','){
                                delete = 1;
                        }
                }
                i++;
                j=0;
        }


        for(i=tmp; username_globali[i] != ','; i++){
                username_globali[i] = '*';
        }
```

```
        username_globali[i] = '*';
}
```

CODICE CLIENT

```
/*
 ============================================================================
 Name        : Client.c
 Author      : Antonio Vanacore & Valerio Panzera
 Version     :
 Copyright   : Your copyright notice
 Description : Uso dei Socket TCP in C
 ============================================================================
 */

#include <stdio.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <sys/types.h>
#include <string.h>
#include <pthread.h>

#define PORTA 1205
#define MAX 11

int socketClientDescriptor; //variabile globale,descrittore del socket locale del client
int sincro;
int paccoPreso= 0; //variabile che indica qnd è stato premuto il tasto p
int paccoLasciato= 0; ////variabile che indica qnd è stato premuto il tasto l
int lista=0;  //variabile booleana lista
int disconnesso=0;
int partiteVinte=0;

void *stampaMatriceThread(void *arg){
        char buffer[121];
        int punteggio=0;
        int vittoria=0;
        sincro=1;

        while(disconnesso==0){
                        if(sincro==1){
                                if(lista == 1){
                                        char utenti[124];//buffer usato qnd devo eggere la lista utenti
                                        read(socketClientDescriptor,utenti,124);
                                        printf("\nUtenti online: \n");
                                        for(int i=0; i<124; i++){
                                                if(utenti[i] == ',' && utenti[i] != '*'){
                                                        printf("\n");
                                                }else if(utenti[i] != '*'){
                                                        printf("%c", utenti[i]);
                                                }
                                        }
                                        printf(" \n");
                                        lista = 0;
                                }

                                //leggo il tempo restante
                                int tempo[3];
                                read(socketClientDescriptor,&tempo[1],1);
                                read(socketClientDescriptor,&tempo[2],1);
                                printf("\nTEMPO RESTANTE: %dm:%ds\n",tempo[1],tempo[2]);
                                printf("Partite vinte: %d\n",partiteVinte);

                                //leggo il punteggio del giocatore
                                read(socketClientDescriptor,&punteggio,1);
                                printf("Punteggio: %d\n",punteggio);

                                read(socketClientDescriptor,&vittoria,1);

                                if(vittoria==1){
```

```
        printf("Qualcuno ha raccolto tutti i pacchi e ha vinto!!\n");
        printf("Start nuova partita fai la tua mossa...");
        paccoPreso=0;
        paccoLasciato=0;
}

if(vittoria==2){
        printf("!!!!!Hai raccolto tutti i pacchi!!!!!\n");
        printf("!!!HAI VINTO!!!\n");
        printf("Start nuova partita fai la tua mossa...");
        partiteVinte++;
        paccoPreso=0;
        paccoLasciato=0;

}

if(vittoria==3){
        printf("!!!TEMPO SCADUTO!!!\n");
        printf("Non hai vinto, ritenta!\n");
        printf("Start nuova partita fai la tua mossa...");
        paccoPreso=0;
        paccoLasciato=0;
}

if(vittoria==4){
        printf("!!!TEMPO SCADUTO!!!\n");
        printf("Hai raccolto piu pacchi di tutti!\n");
        printf("!!!HAI VINTO!!!\n");
        printf("Start nuova partita fai la tua mossa...");
        partiteVinte++;
        paccoPreso=0;
        paccoLasciato=0;

}

//se ha cliccato p leggo le locazioni di arrivo
if(paccoPreso==1 ){
        int coordinateLocazioni[2];
        char temp;
        read(socketClientDescriptor,&coordinateLocazioni[0],1);
        read(socketClientDescriptor,&coordinateLocazioni[1],1);
        if(coordinateLocazioni[0]==0 && coordinateLocazioni[1]== 0){
                printf("Non ce nulla da raccogliere qui!");
        }else{
                printf("Pacco preso!");
                temp = coordinateLocazioni[0] ; //sto usando la i
                temp = temp+48 ; //sto usando la i
                temp=temp+16;
                if(temp==74){
                        temp=76;
                }
                printf("Destinazione-> %c%d",temp,coordinateLocazioni[1]-1);
        }
        paccoPreso=0;
}

if(paccoLasciato==1 ){
        int preso=0;
        char temp;

        read(socketClientDescriptor,&preso,1);

        if(preso==1 ){
                printf("Pacco lasciato!");
        }

        if(preso==2){
                printf("Non hai nessun pacco da lasciare!");
        }

        if(preso==3){
                printf("Non puoi lasciare qui questo pacco!");
        }

        paccoLasciato=0;
}

//leggo la matrice del gioco e visualizzo il quadro
read(socketClientDescriptor,buffer,121);
```

```
                        //stampo la matrice
                        for(int i=0; i<121; i++){
                                if(i % 11 == 0){
                                        printf("\n");
                                        if(buffer[i] == 'o'){
                                                printf("* ");
                                        }else if(buffer[i] == 'i'){
                                                printf("o ");
                                        }else{
                                                printf("%c ", buffer[i]);
                                        }
                                }else{
                                        if(buffer[i] == 'o'){
                                                printf("* ");
                                        }else if(buffer[i] == 'i'){
                                                printf("o ");
                                        }else{
                                                printf("%c ", buffer[i]);
                                        }
                                 }
                        }
                        sincro=2;
                }//fine if(x=1)
        }
        pthread_exit(0);
}

void *leggiComandoThread(void *arg){
        while(disconnesso==0){
                if(sincro==2){
                        char messaggio[1];

                        //mi metto in attesa di un nuovo messaggio/comando
                        do{
                                printf("\ninvia un comando: ");
                                scanf("%s",messaggio);
                        }while(messaggio[0] != 'w' &&
                                messaggio[0] !='d' &&
                                messaggio[0] !='s' &&
                                messaggio[0] !='a' &&
                                messaggio[0] !='p' &&
                                messaggio[0] !='x' &&
                                messaggio[0] !='u' &&
                                messaggio[0] !='l' );

                        if(messaggio[0] == 'p')paccoPreso=1;

                        if(messaggio[0] == 'l')paccoLasciato=1;

                        if(messaggio[0] == 'u')lista=1;

                        //scrivo il comando sul socket
                        write(socketClientDescriptor,messaggio,sizeof(messaggio));

                        if(messaggio[0] == 'x'){
                                printf("\n***DISCONNESSO\n");
                                disconnesso=1;
                                sincro=0;
                        }else{
                                sincro=1;
                        }

                }//fine if
        }//fine while
        pthread_exit(0);
}//fine funzioneThread

int main(int argc, char *argv[]) {
        char username[30];
        char password[10];
    char confronto[10];
    char a;//variabile su cui inserire la pressione dei tasti
    int proseguo =0;
    int k;
    struct sockaddr_in serverDescriptor; //struct in cui inserisco le info del server a cui voglio connettermi
        struct in_addr conversione;

        //poiche il client per connettersi deve conoscere ip e porta del server
```

```
    //struttura alla quale mi connetto(il server cioè)
    serverDescriptor.sin_family=AF_INET;
    serverDescriptor.sin_port=htons(atoi(argv[2])); //argv[2] è il secondo parametro passato
    serverDescriptor.sin_addr.s_addr= inet_addr(argv[1]);

    //creo un socket locale per il client
    socketClientDescriptor=socket(AF_INET,SOCK_STREAM,0);
    if(socketClientDescriptor<0)perror("Errore creazione client socket."),exit(0);

    //connetto il socket client all indirizzo ip del server
    if( connect(socketClientDescriptor,(struct sockaddr *)&serverDescriptor,sizeof(serverDescriptor)) <0)
            perror("Errore durante la connessione al server."),exit(0);

    printf("!!!!!!Connesso al Server!!!!!!\n");

    //SISTEMA DI REGISTRAZIONE
     while(proseguo==0){

            printf("\nPremere 'R' per registrarsi oppure 'L' per effettuare l'accesso\n");
       scanf(" %c", &a);

       if(a >= 'a' && a <= 'z'){
          a = a - 32;
       }

       if(a=='R'){
          write(socketClientDescriptor, &a, sizeof(a));
          printf("\nScegliere l'username: \n");
            scanf("%s", username);
            k=strlen(username);
            username[k] = '-';
            username[k+1] = '\0';
               printf("\nScegliere una password: \n");
               scanf("%s", password);
               printf("\nRipetere password: \n");
          scanf("%s", confronto);

          if(strcmp(password,confronto) == 0){
             char t='Y';
             write(socketClientDescriptor, &t, sizeof(t));
             strcat(username,password);
             write(socketClientDescriptor, username, sizeof(username));
          }else{
            char t='N';
            printf("\nConfronto password errato, ripetere!\n");
            write(socketClientDescriptor, &t, sizeof(t));
           }

       }else if(a=='L'){

            write(socketClientDescriptor, &a, sizeof(a));
            printf("\nInserire username: \n");
            scanf("%s", username);
            k = strlen(username);
            username[k] = '-';
            username[k+1] = '\0';
            printf("\nInserire password: \n");
            scanf("%s", password);
            strcat(username,password);
            write(socketClientDescriptor, username, sizeof(username));
            read(socketClientDescriptor,&proseguo,sizeof(proseguo));

if(proseguo==0){
   printf("\nDati errati: username o password errati!\n");
}

       }else{
          printf("\nScelta errata, ripetere!\n");
       }

    }//fine while del sistema di registrazione

    //INIZIO DEL GIOCO
    printf("\nBENVENUTO NEL GIOCO\n");
    printf("Istruzioni:\nBisogna muoversi nel labirinto, "
                    "prendere il pacchetto ed uscire per "
                    "completare il gioco.\n\nLegenda: \n"
                    "*-Spazio libero\no-Ostacolo\nx-pacchetto\n");
```

```
	printf("\nUtilizzare i tasti W A S D della tastiera "
			"per muovere il personaggio all'interno del "
			"labirinto.\nPremere:\nU-vedere la lista "
			"utenti online\nP-prendere il pacchetto\nL-"
			"depositare il pacchetto\nX-DISCONNETTI\n");
	printf("Campo di gioco:\n");

	pthread_t th1,th2;

	if( pthread_create(&th1, NULL, stampaMatriceThread, NULL) != 0 )
		printf("Failed to create thread 1\n");

	if( pthread_create(&th2, NULL, leggiComandoThread, NULL) != 0 )
		printf("Failed to create thread 2\n");

	pthread_join(th2,NULL);
	pthread_join(th1,NULL);
	printf("Fine Programma.\n");

	return 0;
}
```