SISTEMI OPERATIVI

A CURA DI:

Bottega marica
Loffredo Francesca

Sommario

Comandi di esecuzione

F4Server

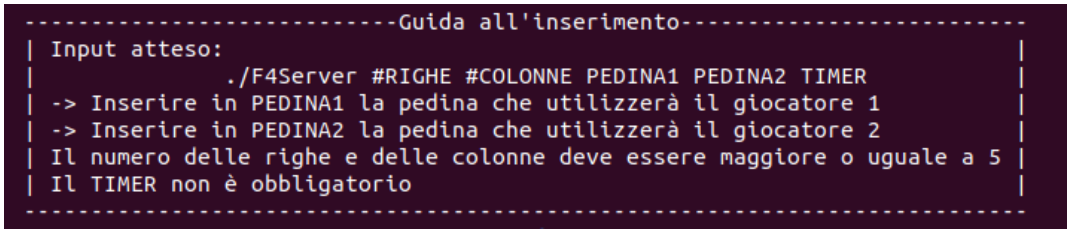
La riga di esecuzione è:

./F4Server #righe #colonne #pedina1 #pedina2 #timer

Il parametro "#timer" indica un tempo massimo in cui i giocatori devono scegliere la mossa, nel caso in cui non venga effettuata l'altro giocatore vince la partita a tavolino.

Se il campo #timer è nullo, la partita procederà senza l'implementazione del timer.

Se non vengono forniti i parametri corretti viene stampata una guida per mostrare il corretto modo per eseguire il comando.



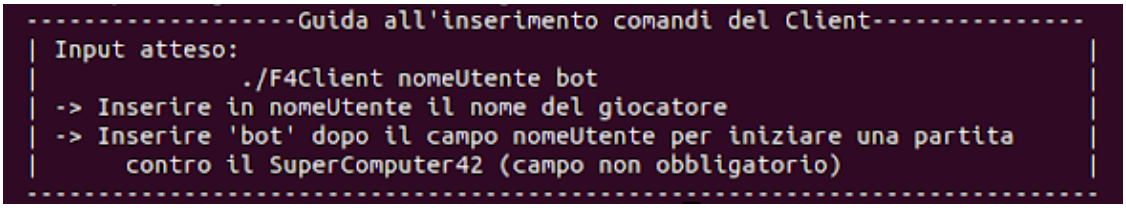
F4Client

La riga di esecuzione è:

./F4Client nomeUtente

Un client può giocare in modo automatico, questo avviene tramite la riga di esecuzione:

./F4Client nomeUtente bot



Semafori – Semaphore (sem)

SEMAFORI PER LA GESTIONE CLIENT-SERVER

Il semaforo **SINC** viene utilizzato per la sincronizzazione tra client e server
Il semaforo **INS** viene utilizzato per attendere l'inserimento da parte del server della mossa scelta dal giocatore.
Il semaforo **B** viene utilizzato per far si che il client aspetti che il server finisca l'esecuzione del ciclo.
Il semaforo **MUTEX** viene utilizzato per la mutuaesclusione.

Il semaforo **TERM**, inizializzato a 0, viene utilizzato per la terminazione del gioco e per rimuovere le IPC, i semafori, la memoria condivisa e le code di messaggi utilizzati all'interno del codice.

Memoria condivisa – Shared memory (shm)

Nel gioco vengono utilizzate due memorie condivise, una per la gestione dei dati necessari per l'esecuzione del gioco raggruppati in una struttura dati e un'altra per condividere un array che rappresenta la griglia di gioco, la quale viene aggiornata dal server dopo ogni mossa e stampata dal client dopo ogni giocata per mostrare l'andamento del gioco.

```
struct dati{
    int nColonne;   //numero di colonne della griglia di gioco;
    int nRighe;     //numero di righe della griglia di gioco;
    int timer;  //tempo in cui i giocatori devono effettuare la mossa;
    char pedina[2];   //array in cui vengono salvate le pedine dei due giocatori
    int indirizzamento[2];   //array utilizzato per indirizzare i processi nelle
                               rispettive funzioni;
    int turno[2];    //array usato per gestire i turni dei due giocatori e per
                       identificare il turno quando il gioco termina;
    int pidClient[2];    //array contenente i pid dei due giocatori adoperati per
                           l'abbandono da parte di uno dei due client;
    int fineGioco;    //serve per gestire i vari modi per cui la partita termina;
    int giocoAutomatico;    //indica se il client entra nella modalità gioco
                              automatico
    int mossaBot;   //viene condiviso il numero random per indicare la colonna
                      scelta
};
```

Funzionamento del programma

FUNZIONE GIOCO DEL SERVER

La funzione gioco viene utilizzata per l'inserimento della pedina nella griglia di gioco (shm), dopo che il giocatore ha effettuato la mossa (scelta della colonna), la quale viene comunicata al server tramite la coda di messaggi (msq). Inoltre, controlla la terminazione del gioco in caso di vittoria o pareggio.

FUNZIONE GIOCA DEL CLIENT

La funzione gioca viene utilizzata per visualizzare la griglia di gioco dopo ogni giocata, per monitorare l'andamento del gioco. Ad ogni turno viene controllato se uno dei due client o il server ha abbandonato il gioco, se si il gioco termina, altrimenti prosegue con la scelta della mossa. Viene verificato se il numero della colonna è tra 1 e il numero delle colonne della griglia e se la colonna scelta non è piena. Se la mossa è valida, viene comunicata al server tramite la coda di messaggi, il quale provvederà con l'inserimento della pedina.

TERMINAZIONE PROGRAMMA

La terminazione del programma viene controllata dal server, il gioco può finire tramite:

Vittoria:

quando un giocatore allinea quattro pedine in una fila continua verticalmente, orizzontalmente o diagonalmente.

Funzioni:

```
bool vittoria_verticale(int pos, int nRighe, int nColonne, char *arr);
bool vittoria_orizzontale(int pos, int colonna_scelta, int nRighe, int
nColonne, char *arr);
bool vittoria_diagonale(int pos, int colonna_scelta, int nRighe, int
nColonne, char *arr);
```

Abbandono del client:

il giocatore può abbandonare la partita premendo i tasti "CTRL-C" oppure chiudendo il terminale, in tal caso all'altro giocatore verrà segnalato dal server la vittoria a tavolino per abbandono dell'altro giocatore.

Funzioni:

```
void abbandonoClient()
```

Abbandono del server:

il server può abbandonare la partita chiudendo il terminale o premendo due volte entro un timer stabilito (10 secondi) i tasti "CTRL-C".

la gestione dell'abbandono del server (doppio CTRL-C) viene gestita tramite un contatore e un timer, se una volta premuto CTRL-C non viene ripremuto una seconda volta entro lo scadere del timer il contatore (che tiene conto del numero dei CTRL-C premuti) viene resettato.

Funzioni:

```
void abbadonoServer();
void sigHandlerServer(int sig);
```

Scadenza del timer:

viene impostato un timer, tramite riga di esecuzione del server, entro il quale i giocatori devono scegliere la mossa, se non viene eseguita entro quel tempo viene dichiarata la vittoria a tavolino all'altro giocatore.

Parità:

tutte le caselle sono occupate e nessun giocatore vince.

Funzioni:

```
bool tabella_piena(int nRighe, int nColonne, char *arr);
```

RIMOZIONE IPC, SEM, SHM, MSQ

Le ipc, i semafori, la memoria condivisa e i messaggi vengono deallocati dal server una volta che i client sono terminati, in modo tale da non causare errori all'interno del gioco, tramite la funzione rimuoviIpc( ).

Funzioni:

```
void rimozioneIpc();
```

COMUNICAZIONE DELLA FINE DEL GIOCO

La comunicazione della fine del gioco avviene tramite le funzioni:

```
void fineGioco();
void sigHandlerAbbandono(int sig);
void sigHandlerTavolino(int sig);
void sigHandlerServer(int sig);
void sigHandlerTempo(int sig);
```

Le quali hanno lo scopo di comunicare a entrambi i client e al server come è finito il gioco.

GIOCO AUTOMATICO (bot)

Questa modalità avviene tramite una generazione casuale di un numero che rappresenta la colonna di gioco, qualora la colonna non supporti più spazi vuoti in cui inserire la pedina verrà generato un nuovo numero finché non viene trovata una colonna vuota.

Il server crea un processo figlio, il quale genera un numero casuale (indica la colonna in cui verrà inserita la pedina) inserisce la mossa in un campo della memoria condivisa (dati->mossaBot) dopodiché il processo figlio termina.

Funzioni:

```
void giocoAutomatico();
void generaMossa();
```

Schema funzionamento dei semafori nella versione gioco automatico: