UNIVERSITÀ di VERONA
DIPARTIMENTO di INFORMATICA
CORSO DI LAUREA IN INFORMATICA

29 APRILE 2024 REV 1.0 <- In caso di presenza di errori fa fede l'ultima revisione pubblicata sull'E-Learning

ELABORATO FINALE SYSTEM CALL

Utilizzo delle system call per la programmazione di sistema.

Scrivere un'applicazione in linguaggio **"C"** che sfruttando le **system call** SYSTEMV viste a lezione implementi il gioco "**TRIS**", il gioco si basa sulle regole del classico gioco da tavolo, implementato con alcune varianti e in grado di funzionare in ambiente UNIX/LINUX da 2 utenti.

1. LE REGOLE DEL GIOCO

Il gioco si svolge tra due giocatori, su un campo rettangolare di dimensione **3x3** dove ogni giocatore **a turno indica le coordinate dove inserire** il proprio simbolo che andrà a posizionarsi nella posizione indicata (qualora la posizione risulti occupata si rimanda ad apposita sezione).

	X	
	O	O
X	X	O

Il giocatore che avrà vinto sarà il primo giocatore che sarà riuscito ad allineare 3 propri simboli.
Il tris potrà avvenire senza interruzione in orizzontale, verticale o diagonale.

Le system call che dovranno obbligatoriamente essere utilizzate saranno: la gestione dei processi figli, la memoria condivisa, i semafori e i segnali tra processi, resta facoltà dello studente utilizzare ulteriori system call viste a lezione.

2. FUNZIONAMENTO BASE DEL GIOCO

L'applicazione dovrà essere composta almeno da due eseguibili:

- **TriServer** che si dovrà occupare di inizializzare il gioco, (predisporre, ad esempio, l'area di memoria condivisa semafori etc.) e di arbitrare la partita tra i 2 giocatori, indicando a ogni mossa se qualcuno ha vinto.

- **TriClient** che si occupa di far giocare il singolo giocatore, raccogliendo la mossa del giocatore e visualizzando il "quadro" di gioco.

TriServer
Il server dovrà prevedere quando viene eseguito la possibilità di definire il timeout per poter eseguire la mossa di gioco, (se il valore di timeout è 0 non vi è timeout), quindi la riga di esecuzione sarà:

```
./F4Server 10 O X
```

Che andrà a generare un'area di gioco, con un timeout di 10 secondi, il primo giocatore posizionerà O, mentre il secondo X.

I parametri aggiuntivi, nel nostro caso **"O X"** (puramente di esempio) saranno le forme dei due gettoni uno per ogni giocatore, utilizzati nella partita. I simboli scelti andranno necessariamente comunicati ai 2 giocatori (client) in quanto saranno utilizzati da loro nel gioco).
L'esecuzione del server senza parametri (o con un numero di parametri inferiori al necessario) comporterà la stampa a video di un messaggio di aiuto per mostrare il modo corretto per eseguire il server.
Il server dovrà gestire eventuali errori di esecuzione dovuti alla presenza precedente di campi di gioco (memoria condivisa e/o semafori), dovrà inoltre terminare in modo corretto e coerente qualora venga premuto due volte di seguito il comando CTRL-C indicando alla prima pressione che una seconda pressione comporta la terminazione del gioco. Per corretto e coerente, si intende che dovrà avvisare i processi dei giocatori che stanno giocando che il gioco è stato terminato dall'esterno (si consiglia di usare un segnale) e dovrà rimuovere in modo corretto le eventuali IPC utilizzate (memoria condivisa e semafori).
È compito del server "arbitrare" la partita, ovvero sarà il server a decidere dopo ogni giocata se il giocatore che ha giocato ha vinto o meno la partita, il server dovrà segnalare di conseguenza ai client anche se hanno vinto o meno.
Il server dovrà inoltre notificare ai client, quando non sono più possibili inserimenti di gettoni (MATRICE di gioco PIENA) che la partita è finita alla pari.
Quando uno dei due giocatori ha vinto è a scelta del candidato decidere se terminare la partita per tutti o, per esempio, proporre ulteriori giocate.

F4Client
Il client dovrà supportare alcune opzioni in fase di esecuzione, la prima è il nome del giocatore, quindi la riga di esecuzione sarà:

```
./TriClient nomeUtente
```

Una volta lanciato il client, rimarrà in attesa che venga "trovato il secondo giocatore" dopo di che il gioco potrà iniziare (è opportuno che al client venga notificato la ricerca di un giocatore per proseguire).
Il client si occupa di stampare a ogni turno "la matrice" di gioco aggiornata, e di chiedere al giocatore su quale colonna intende inserire il proprio gettone. Si noti che i client NON imbrogliano, ma di fatto devono segnalare al giocatore se la POSIZIONE prescelta non è utilizzabile perché già piena. È discrezione del candidato, decidere eventuali "penalità" se si inserisce il gettone in una posizione già occupata (ad esempio si potrebbe passare il turno all'altro giocatore, se si posiziona in un posto già occupato).
La pressione di CTRL-C sul client andrà gestita, di fatto verrà considerato che il giocatore perde "per abbandono" della partita; quindi, il client dovrà prima di terminare notificare la chiusura anticipata della partita al server (che a sua volta notificherà al secondo giocatore la vittoria per abbandono dell'altro giocatore).
È opzionale la gestione di eventuali altri segnali (come, per esempio, la chiusura del terminale dove era in esecuzione uno dei client o il server).

3. FUNZIONALITÀ AGGIUNTIVE

3.1 Time-out per ogni mossa
Quando si lancia il server, verrà definito un numero di secondi di time-out, entro il quale ogni client deve obbligatoriamente giocare, qualora non giochi entro quel tempo, (due opzioni a scelta) o si passa la mano all'altro giocatore senza che il primo giochi, o si dichiara la partita vinta a tavolino dal secondo giocatore. La scelta della opzione di proseguimento del gioco allo scadere del tempo è lasciata al candidato.

3.1 Un client che gioca in modo automatico
Questo avviene banalmente con una generazione causale di una coordinata che rappresenta la posizione di gioco. Qualora la posizione di gioco risulti occupata, verrà generato un nuovo numero fino a poter infilare un nuovo gettone.
In questo il client verrà eseguito con una specifica opzione da riga di comando (*):

```
./TriClient nomeUtente *
```

Quando viene invocato in questo modo, il client informa il server che si duplica ed esegue una versione del client che genera una mossa casuale.

UNIVERSITÀ di VERONA
DIPARTIMENTO di INFORMATICA

CONSEGNA

CONSEGNA TRAMITE E-LEARNING ENTRO LE **23.00 DEL 17/06/2024.**
Verrà aperta specifica consegna su moodle per ogni appello che si chiuderà (perentoria) contestualmente alla medesima scadenza delle iscrizioni su ESSE3.
Verrà inviata comunicazione come avviso tramite moodle indicando l'apertura delle consegne del progetto.
Ogni singola persona (anche se parte di un gruppo) dovrà consegnare il progetto pena la non ammissione all'esame orale.

In calce a ogni file sorgente come commento deve essere riportato il seguente commento:

```
/**********************************
*Matricola
*Nome e cognome
*Data di realizzazione
**********************************/
```

Unitamente al progetto, va consegnato un file di testo (o .pdf) che contenga un piccolo manuale (1 o 2 pagine) che specifichi il funzionamento del programma e le parti che ritenete importanti.

Consegnare un unico file di archivio (a piacimento .tgz, .tar o .zip) il nome del file (per esempio) dovrà essere: `matricola.cognome.nome.tgz` → `vr123456.Drago.Nicola.tgz`.

In caso di consegna di gruppo (2+ persone), indicare nel nome del file le matricole e i cognomi di tutte le persone coinvolte:

`matricola.cognome.matricola.cognome.tgz` →
`vr123456.Drago.vr654321.Dallora.tgz`

NOTE

- Il programma deve compilare e funzionare senza errori sui computer del laboratorio Delta.
- È obbligatorio su client e server verificare che le IPC siano state create, e soprattutto rimuoverle in uscita da parte del gioco.
- È titolo preferenziale per la valutazione (ma non obbligatorio) il controllo dei casi particolari.
- La mancanza di alcuni punti richiesti comporta una "penalizzazione" sulla valutazione ma può portare a superare comunque l'esame.
- Il non consegnare l'elaborato, invece, comporta la non ammissione all'esame di laboratorio.
- La matrice di gioco deve essere implementata tassativamente in memoria condivisa, client e server vi accedono indistintamente (è onere del candidato, definire le politiche di accesso per evitare problemi come ad esempio il deadlock).

L'elaborato è personale (del singolo o del gruppo), la presentazione di elaborati identici (al di fuori del gruppo di presentazione) comporta la NULLITÀ degli elaborati UGUALI.

UNIVERSITÀ di VERONA
DIPARTIMENTO di INFORMATICA

N.B.: Tutto quanto non esplicitato in questo documento può essere implementato liberamente.

RIFERIMENTI

Nicola Drago: nicola.drago@univr.it
Nicola Dall'Ora nicola.dallora@univr.it

UNIVERSITÀ di VERONA
DIPARTIMENTO di INFORMATICA

FAQ

1. È possibile utilizzare primitive POSIX e non quelle viste a lezione?

   No, è richiesto l'utilizzo di quanto spiegato a lezione.

2. È possibile utilizzare l'header file della libreria standard del C string.h che contiene definizioni di macro?

   Si, è possibile utilizzare l'header file string.h.

3. È possibile utilizzare thread POSIX per funzionalità aggiuntive del progetto?

   Si, è possibile utilizzare thread POSIX per implementare funzionalità aggiuntive.

4. C'è una penalizzazione nell'utilizzare variabili globali?

   No, in alcuni casi le variabili globali sono necessarie.

5. Per la funzionalità aggiuntiva 3.1 'il client che gioca in modo automatico', chi deve duplicarsi è il client o il server?

   L'implementazione della funzionalità aggiuntiva 3.1 richiede che sia il server a duplicarsi e ad eseguire tramite 'exec' il client passandogli un parametro specifico in modo che il client generi le mosse in modo randomico.

6. E' possibile svolgere il progetto in un gruppo composto da 3 persone?

   Si, il progetto può essere svolto al massimo in gruppi composti da 3 persone.