UNIVERSITÀ di VERONA
DIPARTIMENTO di INFORMATICA
CORSO DI LAUREA IN INFORMATICA

14 APRILE 2023 REV 1.0 <- In caso di presenza di errori fa fede l'ultima revisione

ELABORATO 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 "**FORZA 4**", 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 **almeno 5x5** dove ogni giocatore **a turno lascia cadere** il proprio gettone che andrà a posarsi sul fondo (o su un gettone già presente in quella colonna).

	X			
	O	O		
X	X	O	O	X

Il giocatore che avrà vinto sarà il primo giocatore che sarà riuscito ad allineare 4 propri gettoni senza interruzione. Per semplicità si considerano allineati i gettoni **esclusivamente in verticale o orizzontale**, lasciando come opzionale la possibilità di conteggiare anche la situazione in 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:

- F4Server 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.

- F4Client che si occupa di far giocare il singolo giocatore, raccogliendo la mossa del giocatore e visualizzando il rettangolo di gioco.

F4Server
Il server dovrà prevedere quando viene eseguito la possibilità di definire la dimensione del campo di gioco (con un minimo di 5x5), quindi la riga di esecuzione sarà:

```
./F4Server 6 7 O X
```

Che andrà a generare un'area di gioco di 6 righe e 7 colonne (una matrice 6x7).

I e 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 minimo di aiuto per mostrare il modo corretto per eseguire il comando.
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 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:

```
./F4Client 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 giro "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 colonna prescelta non è utilizzabile perché già piena.
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 cosa 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 del time-out e 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 un numero che rappresenta la colonna di gioco. Qualora la colonna di gioco non supporti più spazi 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 (*):

```
./F4Client 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 XX YY 202X**
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.Visentin.tgz`

NOTE

- Inutile a dirsi, ma il programma deve compilarsi e girare senza errori.
- È obbligatorio su client e server verificare che le IPC siano state create, e soprattutto rimuoverle in uscita da parte del gioco.
- E 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.

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.

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

RIFERIMENTI

UNIVERSITÀ di VERONA
DIPARTIMENTO di INFORMATICA

Nicola Drago: nicola.drago@univr.it
Francesco Visentin: francesco.visentin@univr.it

UNIVERSITÀ di VERONA
DIPARTIMENTO di INFORMATICA

FAQ

(in aggiornamento)

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

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