# UNIVERSITA' di VERONA
# DIPARTIMENTO di INFORMATICA

# CORSO DI LAUREA in BIOINFORMATICA

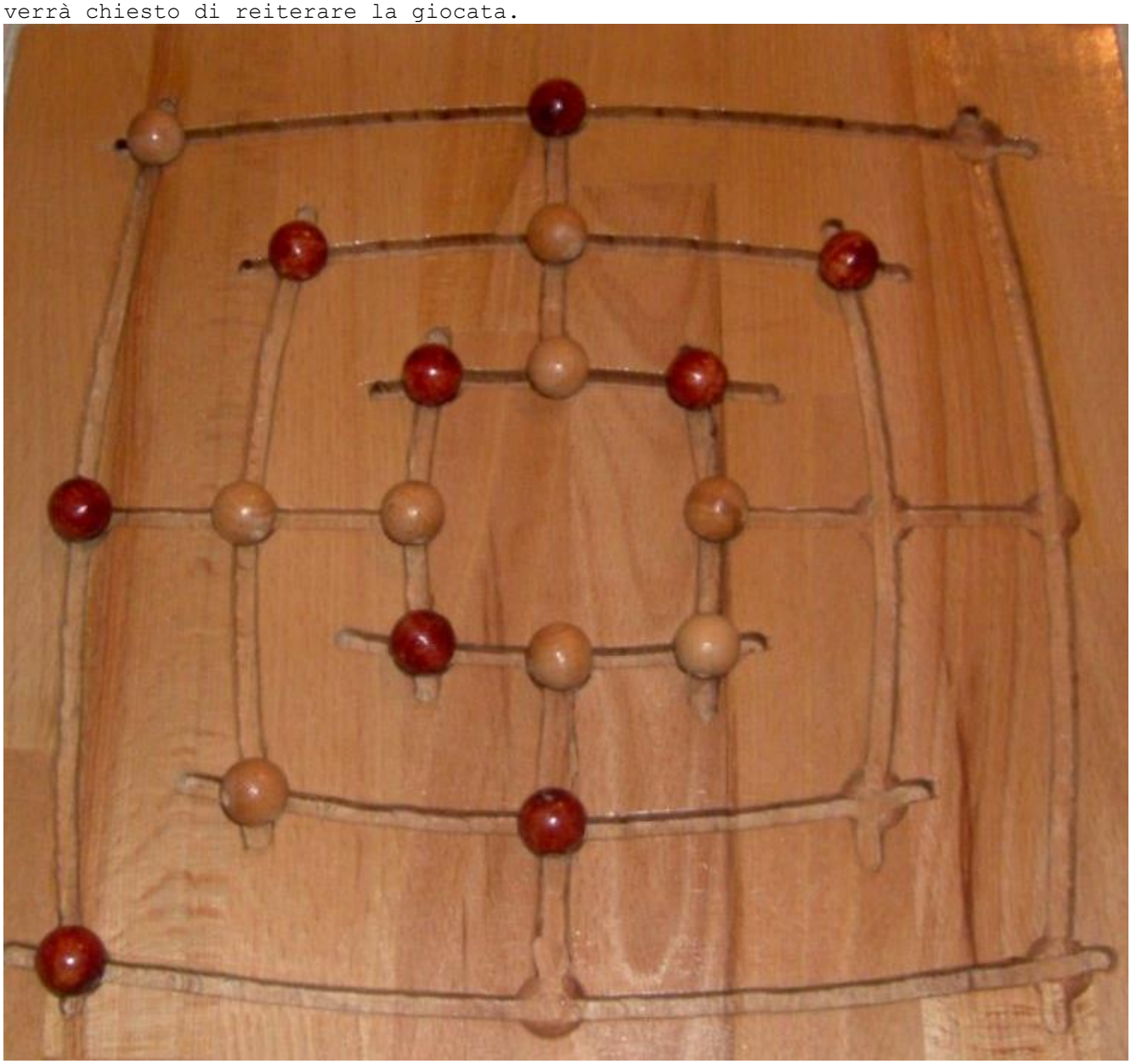
Elementi di sistemi operativi
2021/2022

**CONSEGNA – ELEARNING: entro 22.00 del 20 Gennaio 2022**
**21 Dicembre 2021 REV 1.0 <- In caso di presenza di errori fa fede l'ultima revisione**

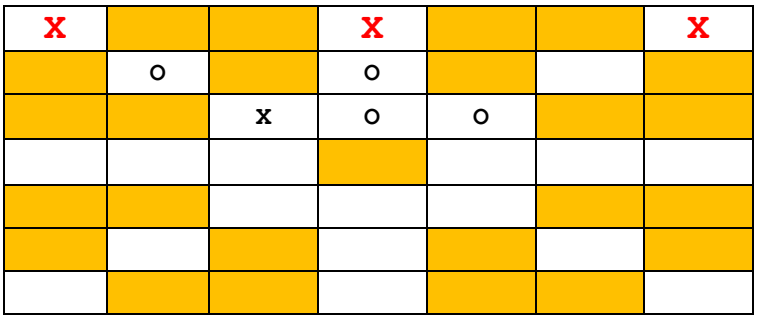
**Elaborato: 2 – SYSTEM CALL**
Utilizzo delle system call (SYSTEM V) per la programmazione di sistema.

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

**LE REGOLE DEL GIOCO:**
Il gioco si svolge tra due giocatori, su un campo quadrato di dimensione **7x7** dove ogni giocatore ***a turno posiziona*** il proprio gettone sulla scacchiera se la cella NON risulta già occupata da altra pedina, in tal caso, la mossa NON risulta lecita e verrà chiesto di reiterare la giocata.



# UNIVERSITA' di VERONA
# DIPARTIMENTO di INFORMATICA




Il giocatore che avrà vinto sarà il **primo** giocatore che sarà riuscito ad allineare 3 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 memoria condivisa, i semafori ed i segnali tra processi**, resta facoltà dello studente utilizzare ulteriori system call.

**FUNZIONAMENTO DEL GIOCO:**
L'applicazione dovrà essere composta almeno da due eseguibili:

- MulinoServer 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 ad ogni mossa se qualcuno ha vinto.

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

**MulinoServer:**
Il server dovrà prevedere quando viene eseguito la possibilità di definire la dimensione del campo di gioco (7x7), quindi la riga di esecuzione sarà:

**./MulinoServer** O X

Che andrà a generare un'area di gioco di 7 righe e 7 colonne (una matrice 7x7), è vostra discrezione utilizzare una "struttura dati" più consona o semplice per la gestione.
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).
E' 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 ad esempio proporre ulteriori giocate.

**MulinoClient:**

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

**./MulinoClient** 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 ad ogni giro "la matrice" di gioco aggiornata, e di chiedere al giocatore dove (x,y o altro sistema) intende inserire il proprio gettone. Si noti che i client NON imbrogliano, ma di fatto devono segnalare al giocatore se le coordinate prescelta non sono utilizzabili o già occupate.
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).
E' opzionale la gestione di eventuali altri segnali (come ad esempio la pressione con il mouse della X in alto della finestra).

**NOTA IMPORTANTE:**
Si consiglia agli utenti che lavorano sul server del Prof. Quaglia di proteggere da accessi indesiderati le proprie cartelle di lavoro onde evitare l'accesso al proprio progetto da parte di altri studenti malintenzionati.

**Opzioni:**

***Un client che gioca in modo automatico:***

banalmente generando ogni volta un numero casuale (o 2 coordinate) che rappresentano la posizione di gioco. Qualora la posizione 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 (*****):

**./MulinoClient** nomeUtente *****

***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.

---

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.
**L'elaborato è personale, la presentazione di elaborati identici comporta la NULLITÀ degli elaborati UGUALI.**

# UNIVERSITA' di VERONA
# DIPARTIMENTO di INFORMATICA

**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.**

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

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

Riferimenti:

Nicola Drago: nicola.drago@univr.it

Per la consegna (che andrà fatta unicamente su moodle nell'apposita sezione), per questioni di praticità, si prega di seguire le seguenti istruzioni:

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

CONSEGNA TRAMITE E-LEARNING ENTRO 22:00 del 20 Gennaio 2022

# UNIVERSITA' 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. *Dove verrà testato il progetto?*

   Il progetto verrà testato sul server messo a disposizione dal Prog. Quaglia.

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