LabSO 2018-2019
Laboratorio Sistemi Operativi 1

PROGETTO LabSO 2018-2019 - Progetto Extra - Revisione I

SIMULAZIONE TORNEO "Rock Paper Scissors"

Implementare la simulazione di un torneo di "Rock Paper Scissors" (gioco "Carta-Forbici-Sasso"): il programma principale deve accettare come parametro il numero di concorrenti "c" (da 8 ad almeno 16 ma anche oltre, entro i limiti della macchina): se sono più di 8 si deve simulare un "torneo all'italiana andata-e-ritorno" (scontri diretti tra tutti i concorrenti a coppie per due volte) creando in sequenza "g" processi (le "giornate") ciascuno dei quali deve generare in parallelo "s" processi, uno per ciascuno scontro diretto (con eventuali turni di "riposo" ad esempio per un numero dispari di concorrenti). In ogni scontro si giocano 10 "lanci" al termine dei quali il punteggio può anche essere in parità. Ogni concorrente acquisisce 0 punti in caso di sconfitta, 1 per il pareggio e 2 in caso di vittoria. Ad ogni giornata i processi dei singoli scontri devono comunicare al processo "giornata" - che previa eventuale elaborazione parziale dei dati si occuperà di passare i dati IN BLOCCO, cioè dell'intera giornata, al processo principale che aggiornerà le informazioni - il risultato. Al termine del torneo si stila una classifica basata sul punteggio: a pari punti si deve tener conto del punteggio ottenuto negli scontri diretti e ad ulteriore parità della "differenza punti" (vinti, persi). In caso di ulteriore parità si procede per sorteggio. I primi 8 in classifica devono poi affrontare un percorso di "quarti di finale", "semifinali" e "finalissima" con scontri diretti: il processo principale deve generare 1 processo "quarti" che poi ne genera a sua volta 4 analogamente a quanto visto sopra per le singole giornate, così successivamente per le 2 semifinali e infine per la finalissima.

> Riassumendo, essendo P il processo principale la generazione che si deve avere qualcosa come:
>
> P -> G1 -> S1,1, S1,2, ... (gli scontri si svolgono in "parallelo" in ogni giornata)
> P -> G2 -> S2,1, S2,2, …
> P -> …
>
> P -> Quarti -> Q1, Q2, Q3, Q4 (gli scontri si volgono in "parallelo")
> P -> Semifinali -> S1, S2 (gli scontri si volgono in "parallelo")
> P -> Finale -> F
>
> I singoli scontri devono comunicare al processo padre il risultato il quale raccoglie tutti quelli dei figli e li passa in blocco al processo principale.

I processi che simulano gli scontri devono attribuire casualmente per ogni lancio uno dei 3 elementi possibili (Carta, Forbici o Sasso) ad ogni concorrente. Si può inserire un ritardo fittizio - ad esempio un secondo o un valore parametrizzabile ad ogni lancio - se utile a rendere più comprensibile l'eventuale feedback del programma.

Il programma DEVE fornire un feedback (output schermo/log/entrambi) comprensibile sull'evoluzione dei processi mostrando in maniera chiara lo svolgimento delle azioni (dove ci sono processi "in parallelo" l'output non deve essere confuso, quindi si deve adottare una strategia che consenta di comprendere

cosa succede specificando ad esempio i soggetti coinvolti e/o con un riassunto al termine dei vari scontri/giornate).

Preferibilmente parametrizzare (consentendo ad esempio di passare delle opzioni all'esecuzione) gli elementi "variabili" come il numero di concorrenti, eventuali tempi di "sospensione" per agevolare l'output, altro… (ad esempio i "nomi" dei concorrenti potrebbero provenire da un file di testo esterno).

Si devono prediligere - ovviamente quando è possibile - le comunicazioni dirette tra processi, in particolare le "pipe anonime", utilizzando solo se si ritiene necessario e non evitabile i segnali e a seguire le "named pipe (fifo)" o le code (message queues). Non si possono usare file esterni "gestiti a mano" per la comunicazione tra processi, né segmenti di memoria condivisa.

NOTA GENERALE: quando si parla di "processi" nel testo si intende che DEVE esistere almeno un processo con le caratteristiche principali indicate, ma è possibile - se ritenuto necessario o preferibile - creare ULTERIORI processi a supporto.

Implementazione
Creare una cartella principale con denominazione tipo "*LabSO2018-2019EXTRA__matricola1__matricola2__....*" (con i numeri di matricola dei componenti del gruppo) e all'interno inserire:

- un file README con i nomi completi e i numeri di matricola dei componenti del gruppo oltre ad una sintesi del progetto realizzato con le eventuali peculiarità
- una sottocartella "project" con dentro:
    - un makefile "Makefile" con <u>almeno</u> tre ricette:
        - "help" (default) che mostra brevi info testuali
        - "build" che compila il progetto, la compilazione DEVE avvenire con il tool "`gcc`" e flag "`-std=gnu90`" e nessun altro
        - "clean" che rimuove eventuali file temporanei e riporta tutto allo stato iniziale
    - una sotto-sottocartella "src" con tutti i sorgenti COMMENTATI DIFFUSAMENTE

Il progetto DEVE partire se lanciato da "terminale bash" su s.o. Ubuntu 18.x (specificare comunque se funziona anche su altri ambienti, come Ubuntu 14.x o Ubuntu 16.x, o altri testati) entrando nella sottocartella "project" e digitando "make build" e poi eseguendo i file opportuni. Si può interagire unicamente attraverso il "terminale bash", ma è possibile aprirne più di uno (ad esempio per mostrare interazioni combinate).
Si terrà conto anche di eventuali avvisi/errori o artifici sia in fase di compilazione che di esecuzione, della rispondenza ai requisiti, della soluzione adottata in generale e anche della resa d'utilizzo (ad esempio se ci sono dei feedback chiari a terminale).
Si deve usare il linguaggio C "gnu90" e si possono adottare solo le librerie "standard".

Rispettare tempi e indicazioni ulteriori relative al progetto.