SISTEMI OPERATIVI con LABORATORIO

Progetto della parte di Laboratorio – a.a. 2022-23

"*Cena dei Filosofi*"

(le indicazioni per la consegna sono alla fine del documento)

Realizzare in linguaggio *C* ed in ambiente *Unix* (*Linux*, *macOS*, etc.) un'applicazione denominata "***Cena dei Filosofi"*** e strutturata nei seguenti processi:

- processo ***parent***
- più processi ***filosofi***

Tutti i processi sopra menzionati sono a sé stanti, cioè non sono *thread/sottoprocessi* l'uno dell'altro. L'applicazione è ispirata al ben noto problema della "*cena dei filosofi*" (si vedano le slide della parte teorica) ed è realizzata modellando i **filosofi come processi** e le **forchette come semafori**.
In breve, la *cena dei filosofi* è un problema in cui abbiamo N filosofi seduti ad un tavolo circolare e, tra un filosofo e l'altro, altrettante forchette. Ogni filosofo quindi ha ad entrambi i lati una sola forchetta. Il filosofo alterna momenti in cui pensa a momenti in cui mangia, ma per mangiare deve afferrare entrambe le forchette ai suoi lati ed è quindi chiaro che non tutti possono mangiare contemporaneamente e che sorgono problemi di concorrenza. Se ogni filosofo ha in mano una forchetta si ha una situazione di *stallo*.

Descrizione generale dell'applicazione

L'applicazione si avvia lanciando il ***parent*** che crea i processi filosofi con ***fork(2)***. Il numero di processi filosofi da creare è passato al ***parent*** come argomento da linea di comando. Più precisamente il ***parent*** deve ricevere da linea di comando i seguenti argomenti:
- numero di (processi) filosofi da creare
- *flag* che attiva/disattiva il rilevamento di *stallo*
- *flag* che abilita/disabilita la soluzione che evita *stallo* (tale soluzione è descritta più avanti)
- *flag* che abilita/disabilita il rilevamento di *starvation*

Di default ogni filosofo cerca di prendere prima la forchetta alla sua destra e poi quella alla sua sinistra. Per esempio, se numeriamo filosofi e forchette, il filosofo 1 prenderà prima la forchetta 1 e poi la 2, il filosofo 2 prenderà prima la forchetta 2 e poi la 3 e così via. L'ultimo filosofo prenderà prima l'ultima forchetta e poi la forchetta 1 (essendo il tavolo circolare).

Se attivato da linea di comando, l'applicazione deve rilevare lo stato di *stallo*, cioè la situazione in cui ogni filosofo ha in mano una forchetta ed è in attesa di afferrare anche l'altra per mangiare (cioè è in attesa del relativo semaforo). Rilevato *stallo* l'applicazione informa a video l'utente e termina.

Se attivato da linea di comando, l'applicazione attua la prima soluzione descritta a questa pagina e che riassumo di seguito: ogni filosofo si comporta come descritto sopra tranne l'ultimo che prende prima la forchetta 1 e poi l'ultima forchetta (inverte cioè l'ordine di presa delle forchette). Tale soluzione evita lo *stallo* ma non previene *starvation*.

Se attivato da linea di comando, l'applicazione rileva lo stato di *starvation*, cioè la situazione in cui almeno un filosofo non mangia da un certo periodo di tempo (per es.: 8 secondi può essere un tempo plausibile per diagnosticare *starvation*). Rilevata *starvation* l'applicazione informa a video l'utente e termina.

L'applicazione deve poter essere quindi avviata in diverse configurazioni, ed in particolare:

1. **senza che nessuna delle funzionalità attivabili da linea di comando sia richiesta dall'utente**. L'applicazione non attiva la soluzione per evitare lo *stallo* né rileva *stallo* o *starvation*. Dovrebbe/potrebbe insorgere *stallo* e l'applicazione non procede (cioè non ci sono ulteriori messaggi a video da parte dei filosofi). L'utente può interrompere l'applicazione con *Ctrl+c*
2. **attivando solo il rilevamento dello *stallo***. L'applicazione non attiva la soluzione per evitare *stallo* né rileva *starvation*. Dovrebbe/potrebbe insorgere *stallo*, l'applicazione lo rileva e tutti i processi terminano correttamente
3. **attivando la soluzione per evitare *stallo* ma non il rilevamento di *starvation***. Indipendentemente se il rilevamento di *stallo* sia attivato o meno, esso non si verifica e l'applicazione procede all'infinito; l'utente può interromperla con *Ctrl+c*
4. **attivando la soluzione per evitare *stallo* e il rilevamento di *starvation***. *Stallo* non si verifica e dovrebbe/potrebbe verificarsi *starvation*; se si verifica, l'applicazione la rileva e tutti i processi chiudono correttamente

Per favorire l'insorgere di stallo il filosofo non esegue l'atto del pensare come invece indicato dal tradizionale modello teorico.

È lasciata liberta allo studente di realizzare come crede il rilevamento di *stallo* e *starvation*. Per tale scopo, e se ritenuto opportuno, possono essere implementati ulteriori processi o threads "di servizio".

Gestione di SIGINT:

L'applicazione deve rilevare anche la pressione del tasto *Ctrl+c* da parte dell'utente, che comporta l'invio del *signal* ***SIGINT***. La ricezione di tale *signal* dovrà essere gestita con un *handler*. Quando SIGINT è ricevuto l'applicazione termina (vedi il punto "***Terminazione dell'applicazione***")

Terminazione dell'applicazione:

L'applicazione termina quando:
- *SIGINT* è ricevuto
- si rileva *stallo*
- si rileva *starvation*

Quando l'applicazione termina deve farlo nel modo più "pulito" possibile, e cioè:
- tutti i processi dell'applicazione devono essere informati che devono terminare
- prima di chiudere, tutti i processi devono liberare il più possibile eventuali risorse allocate: chiudere file descriptor aperti, liberare aree di memorie allocate, etc.
- tutti i processi devono terminare con ***return*** od ***exit(3)*** e non essere interrotti dal sistema operativo "brutalmente"
- ciò presuppone anche che il processo ***parent*** sia l'ultimo a terminare altrimenti i processi filosofi ancora attivi sarebbero terminati (in quanto suoi ***child***) bruscamente

Tutti i processi, all'atto di chiudere, notificano a video all'utente che stanno terminando.

È lasciata liberta allo studente di informare i vari processi della necessità di chiudere come crede, utilizzando comunque uno degli strumenti di *IPC* (*inter-process communication*) illustrati durante il Laboratorio

Dettaglio dei filosofi

Ogni (processo) filosofo esegue all'infinito le seguenti operazioni:
- presa della prima forchetta (secondo quanto indicato sopra nella *Descrizione generale dell'applicazione*)
- presa della seconda forchetta
- mangiare. L'atto di mangiare è simulato dall'attesa (passiva) per un certo tempo
- rilascio delle due forchette

Ogni (processo) filosofo informa a video l'utente quando:
- inizia ad attendere la presa di una forchetta indicando se di destra o di sinistra
- mangia

Per favorire l'insorgere di stallo il filosofo non esegue l'atto del pensare come invece indicato dal tradizionale modello teorico.

==========

Lo studente dovrà fornire anche uno script *Bash* che compila l'applicazione da zero e avvia il **parent**.

Sarà considerata nota di merito l'inserimento di commenti nel codice che ne facilitino la comprensione, così come l'inserimento di commenti atti a motivare le scelte di progetto dell'applicazione fatte.

Il docente valuterà il progetto consegnato a terminale, non in ambiente grafico a finestre.
Il progetto per funzionare non deve richiedere l'esecuzione dall'utente *root*.
L'applicazione deve poter essere avviata e deve funzionare da un unico terminale a carattere.
Evitare attese attive (cicli continui che impegnino intensivamente la CPU).

Indicazioni per la consegna:
Il progetto deve essere spedito via email al Docente all'indirizzo, oppure messo a disposizione su un qualsiasi *cloud* e reso scaricabile tramite *link* da inviare sempre al suddetto indirizzo.

La consegna deve avvenire tassativamente almeno una settimana prima della data dell'esame orale. Il formato può essere tar, tgz o zip.