PROGETTI LabSO1 - AA 2019-2020

INDICAZIONI

- i progetti devono funzionare correttamente sul sistema operativo Ubuntu 18.04 (modello Docker) o su Raspberry (PI 4b) a seconda della variante di progetto scelta
- quando si parla di "processo" si intende che DEVE esistere un processo "autonomo" (con un suo pid distinto dagli altri) che però può essere affiancato da altri processi di "servizio" aggiuntivi se ritenuto necessario/utile costituendo di fatto un gruppo.
- la generazione dei processi (forking) deve parallelizzare le attività il più possibile limitando al massimo l'eventualità di creare processi "in sequenza"
- si devono evitare processi "zombie"
- le comunicazioni tra processi devono avvenire nel modo più "diretto" possibile favorendo quindi primariamente le "pipe anonime" (quando vi possa essere una gerarchia diretta) e poi nell'ordine - quindi cercando di usare un metodo di questo elenco solo se si ritiene non applicabile nessuno dei precedenti (perché impossibile per le specifiche) - segnali, fifo (named pipe), code, memoria condivisa e file-system.
- gli errori di sistema devono essere tutti gestiti correttamente e non devono verificarsi né avvisi/errori, né crash: in particolare si deve puntare a gestire l'impossibilità di creare nuovi processi (forking) o usare risorse in generale con un feedback all'utente senza interrompere però l'esecuzione (se possibile mantenendo lo stato fino all'eventualità di nuova disponibilità di risorse)
- il progetto deve essere completamente contenuto in un'unica macrocartella con modello di denominazione `LabSO1-AA_2019_2020--xxxxxx_xxxxxx_xxxxxx_xxxxxx` dove le parti `xxxxxx` rappresentano i numeri di matricola dei componenti del gruppo (tanti quanti necessari) che deve contenere almeno:
    - la sottocartella `src` con tutti i "sorgenti"
    - un file `README` con il riepilogo delle informazioni (mettendo nella prima riga il nome della macrocartella stessa, poi un'eventuale mail "di gruppo" collettiva se esistente e poi la lista dei componenti del gruppo con nome+cognome completo, numero di matricola e indirizzo mail di ciascuno. A seguire una brevissima descrizione delle scelte implementative e/o delle peculiarità/difficoltà/etc. del progetto, segnalando cosa eventualmente non si è riusciti a completare del tutto o in parte). Infine le indicazioni (rapide) su come utilizzare/testare il progetto.
    - un "`Makefile`" con almeno le regole (più altre che si ritengono utili/necessarie):
        - `help` (default): mostra un veloce "help" (ad esempio il README)
        - `build`: effettua la compilazione completa
        - `clean`: ripulisce eventuali file temporanei creati in fase di building
- il "make" deve funzionare nell'ambiente di riferimento (modello docker e/o hardware Raspberry a seconda della versione) e l'esecuzione deve avvenire senza avvisi o errori. Il prodotto finito deve trovarsi in una sottocartella "bin" (che può essere creata "al volo")
- la compilazione deve avvenire con il tool `gcc -std=gnu90` a cui eventualmente aggiungere i flag: `-o`, `-c`, `-S`, `-E`, `-I`, `-D`, `-pthread`. Si possono usare le librerie: `assert.h`, `ctype.h`, `errno.h`, `fcntl.h`, `float.h`, `limits.h`, `math.h`, `pthread.h`, `signal.h`, `stddef.h`, `stdio.h`, `stdlib.h`, `string.h`, `sys/ipc.h`, `sys/msg.h`, `sys/shm.h`, `sys/stat.h`, `sys/types.h`, `sys/wait.h`, `time.h`, `unistd.h` (oltre ad eventuali necessarie per l'interfacciamento hardware nel caso del Raspberry)

MANIPOLAZIONE LISTA PROCESSI

Realizzare un programma (processo "M") che rilevi 4 azioni di input (4 interruttori/pulsanti: A, B, C e D) che abbiano un corrispondente output (4 led) a indicarne lo stato (on/off). Ogni interruttore/pulsante+led deve essere gestito da un processo a sé (S_1, S_2, S_3 e S_4) che poi comunica lo stato iniziale e l'eventuale variazione al processo gestore il quale deve fornire un feedback visivo - cioè a video - della situazione). Inizialmente deve essere generato inoltre un ulteriore sottoprocesso "P_1" cui sia assegnato un "id" interno univoco (ad esempio una stringa casuale di lunghezza fissa o un progressivo numerico che non si ripete nel tempo). Gli interruttori/pulsanti A e B devono rispettivamente diminuire e aumentare il numero di sottoprocessi richiesti (da un minimo di 1 al massimo possibile) per cui ad esempio premendo B si deve generare un nuovo figlio P_2 da P_1 e con un'ulteriore pressione si deve avere P_3 da P_2 e così via. Premendo invece A si deve eliminare il processo P_1 e "spostare" tutti gli altri di una posizione (P_2 "diventa" P_1, P_3 "diventa" P_2 e così via: "diventa" significa che si deve ricreare opportunamente la gerarchia in modo che gli identificativi univoci si "spostino" lungo la catena). Deve essere sempre presente un "feedback visivo" della situazione in cui sia chiara la struttura dei processi con gli identificativi generati. L'interruttore/pulsante C deve invece consentire di selezionare uno dei processi della catena (il primo è selezionato di default): ad ogni attivazione la selezione si deve spostare su quello successivo fino all'ultimo e da qui ricominciando in maniera ciclica. All'attivazione dell'interruttore/pulsante D deve essere eliminato il processo selezionato facendo "spostare" i successivi indietro di una posizione: il nuovo selezionato diventa quello che occupa la posizione dell'eliminato. Deve rimanere sempre almeno un processo P. Le modifiche NON possono avvenire solo copiando l'identificativo ma eliminando i processi da cancellare/spostare e ricreandone di nuovi.
Deve essere realizzato poi un eseguibile a parte (processo "Q") che - interfacciandosi in qualche modo con l'altro principale - possa conoscere i processi P_i esistenti e comunicando direttamente con ciascuno di essi ne possa recuperare l'identificativo: dato "l'indice" della sequenza deve risalire all'identificativo, dato quest'ultimo deve invece restituirne la posizione.

Note:

- i processi P_i rappresentano sostanzialmente una lista in cui si può effettuare un'eliminazione in testa (con A), un inserimento in coda (con B), una selezione (con C) e un'eliminazione selettiva (con D)
- è sufficiente avere dei pulsanti: se si usano degli interruttori l'azione di attivazione corrisponde all'accensione cui deve seguire uno spegnimento manuale prima di un'eventuale ulteriore attivazione.

Variante “R”: RASPBERRY
Utilizzare un kit Raspberry PI 4b e un sistema hardware di input/output che utilizzi per i 4 interruttori/pulsanti e i 4 led componenti fisici. Gli interruttori/pulsanti devono interagire con i processi S_1, S_2, S_3 e S_4 che comunicano anche con i led corrispondenti.

Variante “U”: UBUNTU
Realizzare dei programmi aggiuntivi che emulino via software i 4 interruttori/pulsanti (ciascuno sarà rappresentato da un processo che possa comportarsi sia come interruttore che come pulsante in base a un parametro da usare all’avvio) e i 4 led (ciascuno sarà rappresentato da un processo con uno stato acceso/spento). Indicando con T_1, T_2, T_3 e T_4 i processi dei 4 interruttori/pulsanti e con L_1, L_2, L_3 e L_4 quelli per i LED si ha che (per i = 1, 2, 3, 4) S_i comunica con T_1 ed L_1 (e T_1 ed L_1 comunicano entrambi SOLO con S_i).

Riassumendo si ha (salvo processi “aggiuntivi” per necessità/comodità/scelta):

- M - main: gestione generale e input/output con utente. Può comunicare con tutti gli altri sottosistemi tranne T_i e L_i
- T_i - interruttore/pulsante: si possono impostare come interruttori o pulsanti in base a un parametro in fase di esecuzione e si possono poi manovrare. Comunicano solo con il corrispondente S_i (*) [solo variante “U”, sono oggetti fisici nella variante “R”]
- L_i - led: hanno uno stato acceso/spento verificabile. Comunicano solo con il corrispondente S_i (*) [solo variante “U”, sono oggetti fisici nella variante “R”]
- S_i - comandi: si interfacciano con i corrispondenti T_i ed L_i [variante “U”] o con le componenti hardware [variante “R”]
- P_i - elementi della lista: hanno un identificativo univoco
- Q - analizzatore: interfacciandosi con M può chiedere informazioni sui vari P_i tranne l’identificativo univoco che può richiedere solo direttamente a ciascuno di essi (evidentemente dopo averlo individuato in qualche modo)

(*) esempi di implementazione:
- Come eseguibili a sé (ognuno su un terminale): lo stato è visibile a video e la manovra può avvenire tramite la tastiera premendo un tasto opportuno o con dei comandi ad-hoc
- Come processi in background: deve essere possibile interrogarli/manovrarli opportunamente attraverso comandi specifici dedicati (con eseguibili appositi o direttamente tramite M) o via bash (con azioni verso i processi)