Sistemi di Calcolo - Modulo 2 (A.A. 2016-2017)

Simulazione d'esame – 17 maggio 2017

Tempo a disposizione: 1h 30'.

Attenzione: assicurarsi di compilare il file **studente.txt** e che il codice prodotto non contenga **errori di compilazione**, pena una valutazione negativa dell'elaborato.

Esercizio 1 - Realizzazione di un processo multi-thread con paradigma prod/cons

Un processo lancia THREAD_COUNT thread, dei quali metà sono produttori e metà consumatori; si assume che THREAD_COUNT sia un numero pari. Ogni thread ha un identificativo (*idx*, che varia tra 0 e THREAD_COUNT-1) ed un ruolo (*role*, che può valere PROD_ROLE oppure CONS_ROLE). Una volta lanciati, il processo si mette in attesa della loro terminazione. Infine, il processo deve rilasciare le risorse che aveva allocato.

Ogni thread esegue ITERATION_COUNT iterazioni. I thread produttori invocano la funzione *enqueue()* ad ogni iterazione *i*, con un valore pari a *idx* * *i*. I thread consumatori invocano la funzione *dequeue()* ad ogni iterazione. Le funzioni *enqueue()* e *dequeue()* implementano rispettivamente la scrittura su e la lettura da un buffer circolare, gestito secondo la semantica *più produttori/più consumatori*. Al termine delle iterazioni, ogni thread deve rilasciare le risorse che aveva allocato.

Obiettivi

1. Implementazione della semantica più produttori/più consumatori
    - Dichiarazione/Inizializzazione/Rilascio dei semafori necessari
    - Uso dei semafori nelle funzioni enqueue() e dequeue()
2. Gestione multi-thread
    - Creazione thread
    - Rilascio risorse allocate

Esercizio 2 - Realizzazione di comunicazione bidirezionale via pipe tra due processi

Due processi padre-figlio condividono due pipe per realizzare un canale di comunicazione bidirezionale. Il *figlio* genera N numeri interi casuali e li invia al padre come stringhe di lunghezza variabile, usando '\n' come delimitatore di fine messaggio. Il *padre* stampa a video ogni messaggio ricevuto, raddoppia il valore numerico ricevuto e lo invia al *figlio*, sempre come stringa di lunghezza variabile con '\n' come delimitatore di fine messaggio. Il *figlio* stampa a video ogni messaggio ricevuto. Dopo aver gestito tutti i messaggi, i processi terminano, con il padre che attende la terminazione del figlio.

Obiettivi

1. Gestione processi figlio
    - Creazione/Attesa terminazione processo figlio
2. Comunicazione via pipe
    - Creazione pipe e chiusura descrittori
    - Lettura messaggi di lunghezza variabile (con condizione di terminazione)
    - Gestione degli errori di invio e ricezione messaggi, compreso il caso di messaggio troppo lungo
    - Invio messaggi

Altro

- **I commenti nel codice contengono molte informazioni utili per lo svolgimento della prova, si consiglia quindi di tenerli in debita considerazione**
- In caso di necessità, nella cartella `backup/` è presente una copia della traccia
- Il file `dispensa.pdf` contiene una copia della dispensa *Primitive C per UNIX System Programming* preparata dai tutor di questo corso
- Il file `raccomandazioni.pdf` contiene una serie di considerazioni sugli errori riscontrati più di frequente

Regole Esame

- Domande ammesse
  Le domande possono riguardare solo la specifica dell'esame e la struttura di alto livello del codice, nessuna domanda può riguardare singole istruzioni.
- Oggetti vietati
  I seguenti oggetti non devono essere presenti sulla scrivania, né tantomeno usati: smartphone, smartwatch, telefonini, tablet, portatili, dispositivi di archiviazione USB, copie cartacee della dispensa, astucci e qualsiasi forma di libri ed appunti. **Chi verrà sorpreso ad usare uno di questi oggetti verrà automaticamente espulso dall'esame.**
- Azioni vietate
  È assolutamente vietato comunicare in qualsiasi modo con gli altri studenti. **Chi verrà sorpreso a comunicare con gli altri studenti per la prima volta verrà richiamato, la seconda volta verrà invece automaticamente espulso dall'esame**.