Sistemi di Calcolo - Modulo 2 (A.A. 2015-2016)

Primo appello - 20 Giugno 2016

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 - Comunicazione bidirezionale via FIFO

Due processi denominati rispettivamente *master* e *slave* comunicano tramite FIFO per trasferire una struttura dati.Il master crea ed apre una coppia di FIFO per comunicare con lo slave, inizializza una struttura dati, ne calcola il checksum e lo stampa; la struttura dati creata dal master viene trasformata in un array di N bytes ed inviata allo slave preceduta da un header. Il master riceve un checksum dallo slave, lo confronta con quello calcolato precedentemente, ed infine termina.

Lo *slave* apre la coppia di FIFO e riceve l'header contenente la dimensione N dei dati da ricevere successivamente. Ricostruita la struttura a partire dall'array di N bytes ricevuto, esso calcola un checksum che viene stampato ed inviato al master prima di terminare.

Lo slave viene fornito in forma precompilata. Il file `master.c` andrà completato e compilato. Eseguire il master in un terminale e lo slave in un altro. Usare `make clean` per distruggere eventuali FIFO create e non rimosse da versioni incomplete del proprio codice.

Obiettivi

1. Ciclo di vita di una coppia di FIFO
    - Creazione e rimozione
    - Apertura e chiusura descrittori
2. Comunicazione su FIFO: invio e ricezione dati di lunghezza fissa

Esercizio 2 - Applicazione multi-processo multi-thread single-prod/single-cons

Un processo lancia in parallelo `CHILD_COUNT` processi figlio. Ognuno di essi lancia in parallelo un thread consumatore ed uno produttore che, rispettivamente, leggono `READ_COUNT` valori da e scrivono `WRITE_COUNT` valori in un buffer circolare secondo il *paradigma singolo produttore/singolo consumatore*. Ogni processo figlio attende la fine dei propri thread prima di terminare. Il processo padre attende la fine dei processi figlio prima di terminare.

Obiettivi principali

1. Gestione processi figlio: creazione/Attesa terminazione processi figlio
2. Gestione multi-thread: creazione thread, rilascio risorse allocate
3. Implementazione della semantica singolo produttore/singolo consumatore
    - Dichiarazione/Inizializzazione/Rilascio dei semafori necessari
    - Uso dei semafori nelle funzioni `enqueue()` e `dequeue()`

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**.