

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

*Secondo appello - 15 Luglio 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 unidirezionale padre/figli via pipe con sincronizzazione

Il processo padre crea `CHILDREN_COUNT` processi figlio e condivide con loro una pipe unica dalla quale lui legge e nella quale i figli scrivono. Affinchè i figli scrivano nella pipe in mutua esclusione, viene impiegato un semaforo named binario. Tutti i figli provano a creare questo semaforo: uno solo ci riesce, gli altri lo aprono soltanto. All'inizio, il padre deve assicurare che il semaforo named non esista, così che i figli possano iniziarlo assumendone la non esistenza. Alla fine, il padre deve rimuovere il semaforo named per pulizia.

Una volta aperto il semaforo named, il figlio *i*-esimo deve scrivere sulla pipe `MSG_COUNT` messaggi, dove ogni messaggio è un array di interi di dimensione `MSG_SIZE` avente *i* come valore di ogni elemento. Una volta creati i figli, il padre deve leggere dalla pipe `CHILDREN_COUNT*MSG_COUNT` messaggi e verificarne l'integrità. Infine, il padre deve attendere il termine dei figli.

### Obiettivi

1. Gestione processi figlio: creazione/attesa terminazione processi figlio
2. Semafori named: creazione, apertura, chiusura, rimozione, uso per mutua esclusione
3. Comunicazione su pipe: invio e ricezione dati di lunghezza fissa

## Esercizio 2 - Ricerca multi-thread su array

Un processo crea `THREADS` thread che devono processare il contenuto di un array `array` in parallelo. Ogni thread, caratterizzato da un indice *i*, processa una porzione dell'array e salva il risultato del processamento nella cella *i*-esima di un secondo array `counters` composto da `THREADS` celle.

L'array da processare viene diviso in segmenti di `STEP` elementi. Il primo thread processa dalla cella 0 alla cella `STEP` (esclusa), il secondo inizia dalla cella `STEP` e così via. A riguardo si presti particolare attenzione al blocco di commento inserito nel codice.

Il main thread attende la terminazione degli altri thread, poi aggrega i risultati.

### Obiettivi principali

1. Gestione multi-thread: creazione/attesa terminazione thread
2. Gestione degli argomenti dei thread `thread_args_t`

## 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.**