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

Esonero - 27 Maggio 2016 - Turno 1 - Traccia 1

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 server multi-thread con comunicazione su socket.
Un server espone N-1 risorse identificate da un numero compreso tra 1 e N-1. I client possono connettersi al server tramite socket e richiedere l'utilizzo di una risorsa inviando un numero tra 1 e N-1. Il server mantiene un array *shared_counters* con N contatori, dove il contatore i-esimo tiene traccia del numero di richieste ricevute per la risorsa i-esima.

Ogni client viene gestito con un thread (su cui non è previsto fare join) che esegue la funzione *connection_handler()*. Questo thread riceve comandi dal client ed invoca la funzione *process_resource()*, che incrementa il contatore di *shared_counters* relativo alla risorsa richiesta. L'accesso al contatore i-esimo di *shared_counters* è protetto dal semaforo i-esimo in *semaphores*. Es: se il client invia al server "1", viene incrementato il contatore in posizione 1. Qualsiasi altro comando viene considerato come "0", ed in tal caso è il contatore in posizione 0 ad essere incrementato. L'unica eccezione è il comando "QUIT", che fa terminare il thread.

Obiettivi:
1. Gestione semafori
    - Inizializzazione semafori
    - Gestione sezione critica nella funzione process_resource()
2. Gestione multi-thread
    - Creazione thread di gestione connessione client
    - Rilascio risorse allocate
3. Gestione scambio messaggi su socket
    - Invio/ricezione messaggi su socket con gestione interruzioni
    - Chiusura descrittori

Esercizio 2 - Realizzazione di un processo con figli e nipoti.
Un processo lancia CHILD_COUNT processi figlio e poi ne attende il termine. Ogni processo figlio
- tiene traccia della propria identità (indice dell'iterazione del padre in cui è stato creato)
- crea a sua volta GRANDCHILD_COUNT processi figli (processi nipoti)
    - l'identità di un processo nipote è la seguente coppia
        - identità del processo figlio che lo ha creato
        - indice dell'iterazione del processo figlio durante la quale è stato creato
    - ogni processo nipote fa una sleep di un secondo e poi termina
- ne attende il termine

Obiettivi:
1. Gestione processi: creazione e attesa terminazione processi figli/nipoti
2. Distinzione flussi di esecuzione processo padre/processi figli/processi nipoti
    - Implementare questa distinzione sfruttando l'identità dei processi figli/nipoti

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**.