Università degli Studi di Napoli Federico II

Corso di Laurea in Ingegneria Informaticaoel in Ingegneria Informatiatica
Esame di Sistemi Operativi

Prova pratica 18/03/2024
Durata della prova: 75 minuti

°°°

Lo studente completi il programma a corredo di questo documento, in base alle indicazioni qui riportate. La prova sarà valutata come segue:

- **A**: Prova svolta correttamente.
- **B**: Il programma non esegue correttamente, con errori minori di programmazione o di concorrenza.
- **C**: Il programma non esegue correttamente, con errori significativi (voto max: 22).
- **INSUFFICIENTE**: Il programma non compila o non esegue, con errori gravi di sincronizzazione.

°°°

Testo della prova

Si completi in linguaggio C o C++ un programma multi-processo e multi-thread che simuli un server grafico.

Un programma principale avvia **due eseguibili separati**: uno di generazione di coordinate a video (startRL) e uno che stampa le coordinate a video (startPL).

L'eseguibile **startRL** è un processo multi-threaded che implementa un problema produttori-consumatori basato su monitor con rate limiting e buffer circolare. I thread che lo compongono sono: i) 8 threads produttori, ii) 1 thread consumatore, iii) 1 thread di rate limiting. La funzione principale **main** dell'eseguibile alloca le risorse necessarie alla comunicazione tra threads, crea i theread passandogli le necessarie informazioni per la sincronizzazione, attende i loro completamento, e dealloca le risorse.

Un thread ***produttore*** esegue (funzione **create_coordinates**) un ciclo di 5 iterazioni, producendo una coppia di coordinate in ogni ciclo. Una coppia di coordinate è data da due interi: x è un parametro comunicatogli dalla funzione main, mentre y è generato randomicamente. La coppia di coordinate viene depositata nel monitor. Il monitor deve garantire che un massimo 4 coordinate al secondo possano essere depositate nel monitor. Se 4 coordinate sono state già generate nell'ultimo secondo, il thread produttore si blocca.

Il thread di ***rate limiting*** periodicamente (tramite sleep(1), funzione **rate_limiter_loop**) entra nel monitor per poter resettare il rate conteggiato di messaggi e sbloccare i produttori per permettere la produzione di altri 4 messaggi.

Il thread ***consumatore*** (funzione **send_values**) quando viene creato crea una coda di messaggi UNIX condivisa con l'eseguibile startPL, ed esegue un ciclo per 40 volte (8 thread produttori * 5 messaggi) in cui legge dal monitor appena un valore è disponibile, e lo manda sulla coda di messaggi. Al termine dei 40 cicli, rimuove la coda creata.

L'eseguibile **startPL** riceve in maniera bloccante le coordinate dalla coda e le stampa a video.