Sistemi Operativi

Compito 22-04-2025

Esercizio 1 (10 punti)

In un sistema sono presenti quattro processi P_1, P_2, P_3 ,P_4 che usano quattro tipi di risorsa R_1, R_2, R_3, R_4 di cui sono presenti 2 unità per ciascun tipo ed al massimo usano una risorsa per tipo. Il processo P_1 sta usando una risorsa R_2, una R_3 ed una R_4, il processo P_2 non sta usando risorse, il processo P_3 sta usando una risorsa R_3 ed una R_4 ed il processo P_4 sta usando una risorsa R_1. Usando l'**algoritmo del banchiere** dire se il sistema è in uno stato sicuro. Determinare tra le possibili richieste da parte di P_2 di **singoli tipi di risorsa** disponibili quelle che possono essere subito concesse e quali no, giustificare il risultato.

Esercizio 2 (20 punti)

Si vuole realizzare il seguente sistema:

Sono presenti N *ClientThread* dove ognuno iterativamente: genera K messaggi che inserisce in modo atomico in una coda condivisa, poi attende che tutti i K messaggi siano stati elaborati dagli *WorkerThread* e quindi stampa i valori inviati, quelli ricevuti ed il tempo totale impiegato. La coda condivisa è limitata a X messaggi, e all'inserimento se non c'è posto per inserire tutti i K messaggi il thread attende.

Sono presenti M *WorkerThread* che prelevano un messaggio dalla coda lo elaborano in un tempo variabile in [TW,TW+DW) e quindi inviano il risultato al *ClientThread* che lo ha inserito.

Il programma principale deve far partire i thread necessari quindi dopo 10 secondi deve indicare ai *ClientThread* di smettere di generare messaggi e poi terminare, quando tutti i *ClientThread* hanno finito, deve fermare gli *WorkerThread*. Infine deve stampare per ogni *ClientThread* il numero di operazioni effettuate ed il tempo medio impiegato e per ogni *WorkerThread* il numero di richieste soddisfatte.

Per facilitare il testing ogni *ClientThread* inserisce i valori 1..K nella coda ed il *WorkerThread* li restituisce moltiplicati per 2.

Realizzare in java il sistema descritto usando i ***metodi sincronizzati*** per la sincronizzazione tra thread (usare polling, attesa attiva o altri metodi di sincronizzazione sarà considerato errore)

Per prendere il tempo impiegato usare long System.currentTimeMillis()