Sistemi Operativi

Compito 03-07-2024

Esercizio 1 (10 punti)

Un sistema operativo adotta la politica di scheduling dei thread a coda multipla e con prelazione tra le code. Sono presenti due code, una ad alta priorità (H) con scheduling round-robin e quanto q=2ms e una a bassa priorità (L) con scheduling round-robin e quanto q=3ms. I thread nella coda L che hanno un CPU burst inferiore al quanto passano nella coda H mentre quelli nella coda H che usano tutto il quanto vengono portati nella coda L. In caso di prelazione il thread in esecuzione viene rimesso in fondo alla coda e schedulato normalmente.

Il sistema deve schedulare i seguenti thread con tempi di arrivo, coda e uso CPU/IO indicati:

T_1 T_{arrivo}=2 coda H CPU(1ms)/IO(7ms)/CPU(3ms)/IO(5ms)/CPU(1ms)

T_2 T_{arrivo}=1 coda L CPU(1ms)/IO(4ms)/CPU(1ms)/IO(5ms)/CPU(1ms)/IO(3ms)/CPU(1ms)

T_3 T_{arrivo}=3 coda L CPU(4ms)/IO(4ms)/CPU(4ms)

T_4 T_{arrivo}=0 coda L CPU(4ms)/IO(5ms)/CPU(4ms)

Si determini: il **diagramma di Gantt**, il **tempo di attesa** medio, il **tempo di ritorno** medio, il **tempo di risposta** medio e il numero di **cambi di contesto**.
Nel diagramma di Gantt indicare in quale coda avviene l'esecuzione dei thread ed indicare quando si ha prelazione tra le code.

Esercizio 2 (20 punti)

Si vuole realizzare in java il seguente sistema:

Sono presenti N *GeneratorThread* che iterativamente producono un valore impiegando un tempo variabile in [TG, TG+DG), il valore viene inserito in oggetto condiviso GeneratedArray che tiene gli N valori dei generatori (aspetta se il valore precedente non è stato ancora prelevato).

Sono presenti M *ProcessorThread* dove ognuno iterativamente preleva dal *GeneratedArray* l'array con i dati degli N generator (aspetta se ancora non sono stati generati tutti) ed un numero progressivo che identifica l'estrazione (una volta estratti i dati i generatori potranno inserire il nuovo dato). L'elaborazione dell'array impiega un tempo variabile in [TP, TP+DP). Il risultato viene messo con gli altri dati e inserito in una coda illimitata. Infine sono presenti 3 *PostProcessorThread* che devono iterativamente ognuno estrarre atomicamente dalla coda tre risultati con progressivo in sequenza (quindi verranno estratti prima i risultati 1,2,3 poi i risultati 4,5,6 ecc. il thread deve aspettare se uno dei risultati attesi non è stato ancora inserito). Ed infine deve stampare i tre risultati acquisiti riportando per ognuno: id thread che stampa, progressivo, array, risultato.

Per facilitare il debugging i *GeneratorThread* generano numeri in sequenza partendo da id+1 (id=0..N-1) *e i ProcessorThread* producono come risultato la somma degli N valori.

Il programma principale fa partire i thread quindi dopo 10 secondi vengono fermati tutti i thread. Alla fine il programma principale deve stampare per ogni *GeneratorThread* il numero di messaggi prodotti ed il totale dei messaggi generati, per ogni *ProcessorThread e PostProcessorThread* il numero di operazioni completate ed il relativo totale ed infine il numero di messaggi rimasti nella coda.

Realizzare il sistema usando i **metodi sincronizzati** per la sincronizzazione dei thread.