Sistemi Operativi

Compito 09-06-2023

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 FCFS. Quando il thread viene prelazionato viene rimesso in coda ai thread pronti.

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

T_1 T_{arrivo}=3 coda H CPU(3ms)/IO(4ms)/CPU(3ms)/IO(5ms)/CPU(1ms)

T_2 T_{arrivo}=2 coda H CPU(3ms)/IO(4ms)/CPU(3ms)/IO(5ms)/CPU(1ms)

T_3 T_{arrivo}=1 coda L CPU(2ms)/IO(5ms)/CPU(2ms)

T_4 T_{arrivo}=0 coda L CPU(4ms)/IO(3ms)/CPU(2ms)

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**.

Esercizio 2 (20 punti)

Si vuole realizzare in java il seguente sistema:

Sono presenti N thread *Generator* che iterativamente producono un messaggio contenente un valore ed attendono X ms, la richiesta viene messa in una coda limitata di lunghezza L.

I messaggi sono prelevati dalla coda da M thread *Worker* in modo che tutti gli worker estraggono lo stesso messaggio, che verrà tolto dalla coda solo quando tutti gli worker lo hanno acquisito. Ogni worker impiega un tempo variabile in [T, T+TT) per produrre il risultato. Tutti gli M risultati devono essere ricombinati in un array e messi in una coda illimitata di uscita.

Infine un thread *PrinterThread* iterativamente preleva da questa coda un array e lo stampa.

Per facilitare il debugging i *Generator* generano numeri in sequenza partendo da id*100+1 (id=0..N-1) *e gli Worker* producono come risultato il valore moltiplicato per l'id del worker (1..M).

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 *Generator* il numero di messaggi prodotti ed il totale dei messaggi generati, per ogni Worker il numero di richieste completate, il numero di messaggi rimasti nelle code ed il numero di array stampati da PrinterThread.

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