Sistemi Operativi

Compito 25-01-2022

Esercizio 1 (10 punti)

Un sistema operativo adotta la politica di scheduling dei thread a code multiple *senza prelazione* tra le code. Sono presenti due code una a bassa priorità con scheduling FCFS ed una ad alta priorità con scheduling round robin con quanto q=2ms. Inoltre se un thread ad alta priorità ha un CPU burst superiore al quanto viene portato nella coda a bassa priorità al termine del primo quanto usato completamente.

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

T_1 T_{arrivo}=4 pri=H CPU(1ms)/IO(3ms)/CPU(2ms)
T_2 T_{arrivo}=2 pri=H CPU(4ms)/IO(3ms)/CPU(2ms)
T_3 T_{arrivo}=0 pri=H CPU(1ms)/IO(3ms)/CPU(1ms)
T_4 T_{arrivo}=1 pri=H CPU(3ms)/IO(3ms)/CPU(1ms)

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. Inoltre indicare in ogni istante il numero di processi in attesa nelle code dei processi pronti. Dire quale problema ha questo tipo di scheduling.

Esercizio 2 (20 punti)

In un sistema vengono generati M valori interi (>=0) distinti ed inseriti in un contenitore, sono presenti N thread Worker che iterativamente prelevano un valore v (in una posizione casuale e aspetta se non ci sono valori) e lo posizionano in un array di dimensione M in posizione p = v mod M e se il posto è occupato in posizione p+1 mod M e così via fino a trovare un posto libero. Nel fare questo impedire le race conditions ma limitare al massimo il tempo in cui la struttura resta bloccata.

Quando tutti gli M valori sono stati posizionati un thread Collector stampa il contenuto dell'array quindi lo azzera e rimette i valori nel contenitore iniziale.

Dopo 3 volte che è stato ripetuta questa operazione il Collector termina.

Il programma principale inserisce i valori iniziali facendo in modo che V[i+1]=V[i]+random[1,99] e V[0]=random[0,99], avvia i thread, quindi aspetta la terminazione del *Collector*, termina gli *Worker* ed infine stampa per ogni thread il numero di volte in cui ha trovato nell'array il posto occupato.

Realizzare in java il sistema descritto usando i ***semafori*** per la sincronizzazione tra thread.