Sistemi Operativi

Prima prova in itinere 03-05-2019 - **Compito A**

Esercizio 1 (12 punti)

Un sistema operativo adotta la politica di scheduling dei thread a code multiple con *time slice*. Sono presenti due code, la coda A con scheduling FCFS e la coda B con scheduling round-robin con quanto q=3ms. Lo scheduling tra le due code è a time slice e viene assegnato alla coda A un tempo di 4ms e alla coda B un tempo di 9ms. Inoltre se una coda si svuota (tutti i thread sono in attesa o terminati) quando ancora non è teminato il tempo riservatole lo scheduling passa all'altra coda. Quando a un thread viene fatta prelazione da l'altra coda il thread rimane in testa alla coda ready.

Il sistema deve schedulare i seguenti thread arrivati in sequenza T_1, T_2, T_3, T_4,(ai tempi indicati) nelle code indicate e con uso CPU/IO indicati:

T_1	T_{arr}=0	coda=A	CPU(3ms)/IO(6ms)/CPU(2ms)
T_2	T_{arr}=1	coda=A	CPU(3ms)/IO(4ms)/CPU(2ms)
T_3	T_{arr}=2	coda=B	CPU(5ms)/IO(2ms)/CPU(2ms)
T_4	T_{arr}=1	coda=B	CPU(5ms)/IO(4ms)/CPU(3ms)

Inoltre si assuma che per prima venga schedulata la coda A poi la coda B, etc.

Si determini:

1. il **diagramma di Gantt**,
2. il **tempo di attesa** medio,
3. il **tempo di ritorno** medio,
4. il **tempo di risposta** medio,
5. il numero di cambi di contesto

Esercizio 2 (18 punti)

In un sistema sono presenti N segnali, per ogni segnale ogni secondo deve essere eseguito un thread *SignalAcqThread* che acquisisce M valori del segnale e li inserisce in un array di interi e termina. L'acquisizione dei valori di un segnale avviene tramite un oggetto che implementa l'interfaccia

```
interface SignalValues {
      int[] getValues(int signal);
}
```

Un thread *AvgThread* (uno per ogni segnale) prende gli ultimi 3 array generati (o quelli disponibili) da un segnale e ne calcola la media e la stampa, questa attività viene ripetuta ad ogni nuova acquisizione del segnale.

La acquisizione dei valori di un segnale deve essere controllata da un thread *SignalController* che controlla l'acquisizione dei valori e l'attivazione del thread *AvgThread*.

Definire la classe *RandomSignal* che genera i segnali in modo casuale con valori interi in [*s*A,s*A+2A*] dove *s* è il numero del segnale (s=0..N-1) e A>0. Questa classe (che implementa l'interfaccia `SignalValues`) viene utilizzata dai thread per acquisire i valori dei segnali.

Il programma principale deve chiedere in input i valori di N>0, M>0 e A>0, attivare i thread *SignalController* e dopo 30 secondi il programma deve terminare questi thread (usando il metodo *interrupt*()).

Realizzare in Java il sistema descritto usando il metodo *join* per la sincronizzazione tra thread.

Sistemi Operativi

Prima prova in itinere 03-05-2019 - **Compito B**

Esercizio 1 (12 punti)

Un sistema operativo adotta la politica di scheduling dei thread a code multiple con *time slice*. Sono presenti due code, la coda A con scheduling FCFS e la coda B con scheduling round-robin con quanto q=2ms. Lo scheduling tra le due code è con time slice, viene assegnato alla coda A un tempo di 6ms e alla coda B un tempo di 10ms. Inoltre se una coda si svuota (tutti i thread sono in attesa o terminati) quando ancora non è teminato il tempo riservatole lo scheduling passa all'altra coda. Inoltre quando a un thread viene fatta prelazione da l'altra coda il thread rimane in testa alla coda ready.

Il sistema deve schedulare i seguenti thread arrivati al tempo 0 ed in sequenza T_1, T_2, T_3, T_4, nelle code indicate e con uso CPU/IO indicati:

T_1	coda=A	CPU(4ms)/IO(3ms)/CPU(2ms)
T_2	coda=A	CPU(1ms)/IO(3ms)/CPU(3ms)
T_3	coda=B	CPU(5ms)/IO(3ms)/CPU(2ms)
T_4	coda=B	CPU(3ms)/IO(3ms)/CPU(3ms)

Inoltre si assuma che per prima venga schedulata la coda A poi la coda B, etc.
Si determini:

1. il **diagramma di Gantt**,
2. il **tempo di attesa** medio,
3. il **tempo di ritorno** medio,
4. il **tempo di risposta** medio,
5. il numero di cambi di contesto.

Esercizio 2 (18 punti)

In un sistema sono presenti N segnali, per ogni segnale ogni mezzo secondo deve essere eseguito un thread *SignalAcquisitionThread* che acquisisce M valori del segnale e li inserisce in un array di float e termina. L'acquisizione dei valori di un segnale avviene tramite un oggetto che implementa l'interfaccia

```
interface SignalValues {
      float[] getValues(int signal);
}
```

Un thread *Collector* prende gli ultimi array generati dagli N segnali e ne calcola il vettore con la media e di questo vettore calcola il valore massimo e lo stampa, questa attività viene ripetuta ad ogni nuova acquisizione dei segnali.

La acquisizione dei valori dei segnali deve essere controllata da un thread *SignalsController* che controlla l'acquisizione dei valori degli N segnali e l'attivazione del thread *Collector*.

Definire la classe *RandomSignal* che genera i segnali in modo casuale $v_i = v_{i-1} + x$ con x random in [-A,A) e $v_{-1} = 0$. Attenzione che ogni segnale dipende dal valore precedente anche tra acquisizioni diverse. Questa classe (che implementa l'interfaccia `SignalValues`) viene utilizzata dai thread per acquisire i valori dei segnali.

Il programma principale deve chiedere in input i valori di N>0, M>0 e A>0, attivare il thread *SignalsController* e dopo 30 secondi il programma deve terminare questo thread (usando il metodo *interrupt*()).

Realizzare in Java il sistema descritto usando il metodo *join* per la sincronizzazione tra thread.