SISTEMI OPERATIVI IIN

prima prova in itinere - 12.04.2010

Nome: _____ Cognome: _____

Esercizio 1. (punti 12)

Un insieme di thread è caratterizzato dai tempi di CPU burst e IO burst e dalle priorità riportate di seguito (a valori minori corrispondono priorità maggiori):

$T_{1:}$ priorità = 1	q = 1	CPU(2) / IO(3) / CPU(3)
$T_{2:}$ priorità = 3	q = 3	CPU(1) / IO(1) / CPU(3)
$T_{3:}$ priorità = 2	q = 2	CPU(3) / IO(2) / CPU(3)
$T_{4:}$ priorità = 2	q = 2	CPU(1) / IO(3) / CPU(2)

I thread sono arrivati nell'ordine T_1, T_2, T_3, T_4 ed al tempo iniziale sono tutti presenti nelle rispettive code di ready (esiste una coda diversa per ogni livello di priorità).

I thread sono schedulati in accordo ad un algoritmo con *code multiple,* con prelazione tra le code. Ad ogni coda corrisponde una priorità ed ogni coda è gestita con un algoritmo di scheduling r*ound robin* con i quanti di tempo indicati.

Alle code è inoltre applicata la seguente politica di *aging*: la priorità di un thread che rilascia la cpu prima o in corrispondenza all'esaurimento del quanto di tempo è aumentata di 1 (il valore di priorità è decrementato). In ogni caso la priorità non può essere minore di 1.

Applicando la suddetta politica di scheduling si determinino:

- il diagramma di *Gantt*;
- il tempo di attesa medio dei thread;
- il numero di cambi di contesto.

Esercizio 2. (punti 18)

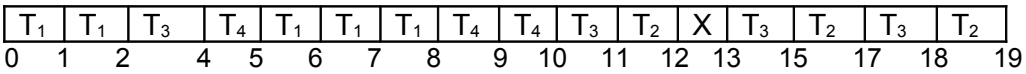
Scrivere il programma Java che opera nel modo seguente:

- Il metodo main istanzia ed avvia tre thread di tipo diverso, rispettivamente, A, B e C.
- Il thread A genera un vettore riga di dimensione N e lo assegna con valori float generati in modo random nell'intervallo [0,NA).
- Il thread B genera un vettore riga di dimensione N e lo assegna con valori float generati in modo random nell'intervallo [0,NB).
- Il thread C, all'interno di un ciclo infinito deve attendere il termine dei thread A e B, eseguire il prodotto tra i vettori generati da A e B, e stampare il risultato del prodotto. Il thread C deve inoltre indurre una nuova esecuzione dei thread A e B chiamando i rispettivi metodi run().

Soluzione

Esercizio 1.

Diagramma di Gantt:



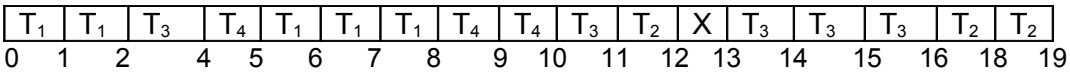
Tempo di attesa medio:

$T_A = (t_A(T_1) + t_A(T_2) + t_A(T_3) + t_A(T_4))/4 = ((0) + (11+2+1) + (2+6+2) + (4))/4 = 28/4 = 7$ ms

Numero cambi di contesto: $N_{cs} = 14$

Notare che all'istante 4, T_3 è prelazionato da T_4 e torna ready con ancora CPU(1). Il residuo però non è considerato come minore di q e quindi al ritorno in esecuzione (in 13) la priorità di T_3 non cambia.

Dato che su questo punto poteva nascere un'ambiguità nella interpretazione del testo sono state comunque ritenute valide soluzioni che attribuiscono a questa condizione un cambio di priorità del thread T_3. Con questa scelta il diagramma di Gantt cambia come segue:



Tempo di attesa medio: $T_A = ((0) + (11+3) + (2+6) + (4))/4 = 26/4 = 6.5$ ms
Numero cambi di contesto: $N_{cs} = 15$

Esercizio 2.

```
package compito_20100412;

public class Principale {
	// dimensione degli array di A e B
	public static final int N = 10;
	public static final int NA = 100;
	public static final int NB = 50;

	public static void main( String[] args ) {
		// crea e avvia il thread A
		ThA A = new ThA( Principale.N, Principale.NA );
		A.start();
		// crea e avvia il thread B
		ThA B = new ThA( Principale.N, Principale.NB );
		B.start();
		// crea e avvia il thread C
		ThC C = new ThC( A, B );
		C.start();
	}
}

public class ThA extends Thread {
	private int N;
	private int max;
	private float row[];
```

```
	public ThA( int n, int n1 ) {
		N = n;
		max = n1;
		row = new float[N];
	}

	public void run() {
		for ( int i = 0; i < N; i ++ )
			row[i] = (float)(Math.random()*max);
	}

	public float[] getRow() {
		return row;
	}
}

public class ThC extends Thread {
	private ThA A;
	private ThA B;

	public ThC( ThA A, ThA B ) {
		this.A = A;
		this.B = B;
	}

	public void run() {
		while ( true ) {
			try {
				A.join();
				B.join();
				float rowA[] = A.getRow();
				float rowB[] = B.getRow();
				float sum = 0;
				for ( int i = 0; i < rowA.length; i ++ ) {
					sum += rowA[i] * rowB[i];
				}
				System.out.println( "Prodotto = " + sum );

				// per rallentare l'esecuzione e debug
				Thread.sleep( 2000 );
				System.out.println( A.isAlive() + " " + B.isAlive() );

				A.run();
				B.run();
			}
			catch( InterruptedException ie ) {
			}
		}
	}
}
```