SISTEMI OPERATIVI IIN

prima prova in itinere - 03.05.2011

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):

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

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 a *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.
Applicando la suddetta politica di scheduling si determini:

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

Esercizio 2. (punti 18)

Scrivere il programma Java che manda in esecuzione i thread A e B secondo lo schema ABABAB.....
Lo schema di esecuzione è il seguente:

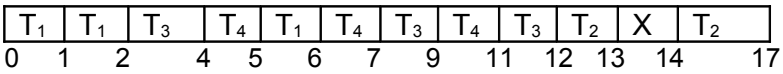
- Il metodo main istanzia un vettore di float di dimensione N, istanzia ed avvia i thread A e B;
- Il thread A riceve in ingresso un vettore di float e lo assegna con valori generati in modo random nell'intervallo [0,N);
- Il thread B attende la generazione del vettore da parte di A, riceve in ingresso un vettore di float, lo ordina in senso crescente e lo stampa. Per eseguire l'ordinamento chiama il metodo statico `sort(float[])` della classe *java.util.Arrays*;
- Il thread B esegue una nuova istanza di A e di se stesso.

Il coordinamento tra i thread può essere risolto utilizzando il metodo *join()* della classe *Thread*.

Soluzione

Esercizio 1.

Diagramma di Gantt:



Tempo di attesa medio:

$T_A = (t_A(T_1) + t_A(T_2) + t_A(T_3) + t_A(T_4))/4 = (0 + 12 + (2+1+2) + (4+1))/4 = 22/4 = 5.5$ ms

Tempo di risposta medio:

$T_R = (t_R(T_1) + t_R(T_2) + t_R(T_3) + t_R(T_4))/4 = (0 + 12 + 2 + 4)/4 = 18/4 = 4.5$ ms

Numero cambi di contesto: $N_{cs} = 10$

Esercizio 2.

```
package compito_20110503;

import java.util.Arrays;

public class Principale {
	// dimensione dell'array
	public static final int N = 10;

	public static void main( String[] args ) {
		float numbers[] = new float[N];
		// crea e avvia il thread A
		ThA A = new ThA( numbers );
		A.start();
		// crea e avvia il thread B
		ThB B = new ThB( A, numbers );
		B.start();
	}
}

public class ThA extends Thread {
	private float numbers[];

	public ThA( float numbers[] ) {
		this.numbers = numbers;
	}

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

public class ThB extends Thread {
	private float numbers[];
	private ThA A;

	public ThB( ThA A, float numbers[] ) {
		this.A = A;
		this.numbers = numbers;
```

```
	}

	public void run() {
		try {
			A.join();
			Arrays.sort( numbers );
			System.out.print( "array ordinato: " );
			for ( int i=0; i<numbers.length; i++ )
				System.out.print( " " + numbers[i] );
			// per rallentare l'esecuzione e debug
			Thread.sleep( 1000 );
			System.out.println();
			System.out.println( A.isAlive() + " " + this.isAlive() );
		}
		catch( InterruptedException ie ) {
			System.out.println( "thread A interrotto !" );
		}
		// nuova esecuzione di A e B
		A.run();
		this.run();
	}
}
```

Nota: chiamare il metodo `start()` su un thread già avviato causa l'eccezione `IllegalThreadStateException`