Sistemi Operativi

PROVA IN ITINERE – 14 novembre 2011

Esercizio 1. (punti 18)

Si definisca una classe Java che dato un vettore di numeri float di dimensione $N=2^k$, calcoli in modo ricorsivo il prodotto degli elementi del vettore. A tal fine si divida ciascun problema di dimensione L in due problemi di dimensione L/2. La soluzione di ciascun sottoproblema deve avvenire in un flusso di esecuzione separato (thread).

Si scriva un programma Java che utilizzando la classe di cui sopra calcoli il prodotto degli elementi di un vettore. Si ipotizzi che la dimensione N del vettore sia letta da linea di comando e che gli elementi del vettore siano assegnati con valori casuali nell'intervallo [0,500].

Per coordinare l'attività dei thread utilizzare il metodo `join()` della classe Thread.

Esercizio 2. (punti 12)

Un insieme di thread è caratterizzato dai tempi di CPU burst e IO burst riportati di seguito:

Processi di *sistema*:	q = 1	T_1	CPU(4) / IO(3) / CPU(1)
Processi *interattivi*:	q = 2	T_2	CPU(2) / IO(5) / CPU(3)
		T_3	CPU(3) / IO(2) / CPU(2)
Processi *background*:	q = 3	T_4	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 (per processi di *sistema*, *interattivi* e *background)*.

I thread sono schedulati in accordo ad un algoritmo a *code multiple* che serve ciclicamente le code in modo FIFO nell'ordine *sistema/interattivi/background* (lo scheduler passa a servire i processi della coda successiva solo se non esistono processi pronti nella coda attuale). I processi all'interno di ogni coda sono schedulati secondo l'algoritmo *round-robin* con un quanto di tempo che dipende dalla coda.

Applicando la suddetta politica di scheduling si determini:

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

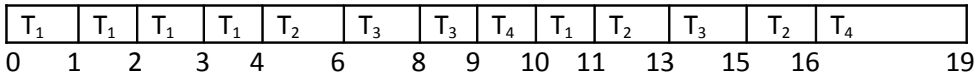
Soluzione

Esercizio 1. (punti 18)

```
package compito_20111114;

public class Muliplier extends Thread
{
	private int[] v;
	private int length;
	private int start;
	private int product = 1;

	public Muliplier ( int[] v, int length, int start ) {
		this.v = v;
		this.length = length;
		this.start = start;
	}

	public void run() {
		if ( length > 1 ) {
			System.out.println( "start = " + start + "  length = " +
length );
			Multiplier mul1 = new Multiplier( v, length/2, start );
			Multiplier mul2 = new Multiplier( v, length/2, start +
length/2 );
			mul1.start();
			mul2.start();
			try {
				mul1.join();
				mul2.join();
			}
			catch ( InterruptedException ie ) {}
			product = mul1.getProduct() * mul2.getProduct();
		}
		else
			product = v[ start ];
	}

	public int getProduct() {
		return product;
	}

	public static void main( String[] args ) {
		if ( args.length < 1 ) {
			System.out.println( "usage: java Multiplier <vector_size" );
			System.exit( 1 );
		}
		int[] vv = new int[ Integer.parseInt(args[0]) ];
		for ( int ii = 0; ii < vv.length; ii++ ) {
			vv[ ii ] = (int)(Math.round(Math.random()*500));
			System.out.print( vv[ ii ] + " " );
		}
		System.out.println();
		Multiplier mul = new Multiplier( vv, vv.length, 0 );
		mul.start();
		try {
			mul.join();
		}
		catch ( InterruptedException ie ) {}
		System.out.println( "product: " + mul.getProduct() );
	}
}
```

<u>Esercizio 2</u>. (punti 12)

Diagramma di Gantt:



Tempo di attesa medio:

$T_A = (t_A(T_1) + t_A(T_2) + t_A(T_3) + t_A(T_4))/4 = (3 + (4+2) + (6+2) + (9+5))/4 = 31/4 = 7.75$ ms

Tempo di ritorno medio:

$T_R = (t_R(T_1) + t_R(T_2) + t_R(T_3) + t_R(T_4))/4 = (11 + 16 + 15 + 19)/4 = 61/4 = 15.25$ ms

Numero cambi di contesto: $N_{cs} = 12$