Sistemi Operativi

Prima prova in itinere 16-05-2016 - **Compito A**

Esercizio 1 (12 punti)

Un sistema operativo adotta la politica di scheduling dei thread a code multiple con *prelazione* tra le code. Sono presenti due code, una a priorità 1 (alta priorità) con scheduling round-robin con quanto q=1ms e una coda priorità 2 (bassa priorità) con scheduling FCFS. Inoltre quando un thread nella coda FCFS viene prelazionato questo rimane in testa alla coda ready.

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

T_1 T_{arrivo}=2 pri=1 CPU(1ms)/IO(2ms)/CPU(2ms)
T_2 T_{arrivo}=3 pri=1 CPU(3ms)/IO(3ms)/CPU(2ms)
T_3 T_{arrivo}=0 pri=2 CPU(3ms)/IO(5ms)/CPU(3ms)
T_4 T_{arrivo}=1 pri=2 CPU(4ms)/IO(2ms)/CPU(1ms)

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 e
5. il numero di cambi di contesto

Esercizio 2 (18 punti)

Data la seguente classe:

```
abstract class ArrayWorkerThread extends Thread {
        protected int[] result;
        protected ArrayWorkerThread(int n) {
                result=new int[n];
        }
        public int[] getResult() { return result; }
}
```

Definire la classe *ArrayRndGeneratorThread* derivata da *ArrayWorkerThread* che dato intero N inizializza su thread parallelo il vettore *result* di dimensione N con numeri random tra 1 e 100 (compresi).

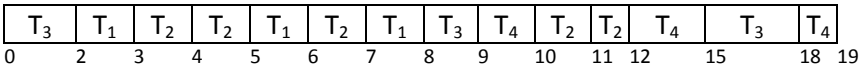
Definire la classe *ArraySumThread* derivata da *ArrayWorkerThread* che dati due *ArrayWorkerThread* attende (su thread separato) che entrambi siano terminati e calcola nel vettore *result* la somma dei risultati ottenuti da questi due thread.

Nel programma principale definito nella classe *Compito1* chiedere in input il valore N>0 quindi creare sei *ArrayRndGeneratorThread* (di dimensione N) quindi usare la classe *ArraySumThread* per sommare gli array due a due cercando di parallelizzare al massimo le operazioni. Il programma principale deve attendere che il thread che calcola il risulato finale termini e stampi il risultato.

SOLUZIONE Compito A

Esercizio 1

Diagramma Gantt



Tempo attesa medio T_a = (1+1+7+11)/4 = 20/4 = 5ms

Tempo di ritorno medio T_r = (6+9+18+18)/4 = 51/4 = 12,75ms

Tempo di risposta medio T_{rs} = (0+0+0+8)/4 = 8/4 = 2ms

Numero cambi contesto N_{cs} = 13

Esercizio 2

```
import java.util.Scanner;

abstract class ArrayWorkerThread extends Thread {
        protected int[] result;
        protected ArrayWorkerThread(int n) {
                result=new int[n];
        }
        public int[] getResult() { return result; }
}

class ArrayRndGenerator extends ArrayWorkerThread {
   public ArrayRndGenerator(int n) {
      super(n);
   }

   public void run() {
      String s="";
      for(int i=0; i<result.length; i++) {
         result[i]=1+(int)(Math.random()*100);
         s+=result[i]+" ";
      }
      System.out.println(getName()+": "+s);
   }
}

class ArraySumThread extends ArrayWorkerThread {
   ArrayWorkerThread w1,w2;
   public ArraySumThread(ArrayWorkerThread a1, ArrayWorkerThread a2) {
      super(a1.getResult().length);
      w1=a1;
      w2=a2;
   }

   public void run() {
      try {
         w1.join();
         w2.join();
         int[] r1=w1.getResult();
         int[] r2=w2.getResult();
         for(int i=0; i<result.length; i++)
```

```
            result[i]=r1[i]+r2[i];
        } catch(InterruptedException e) {
        }
    }
}

public class Compito1 {
    public static void main(String[] args) {
        int N;
        Scanner s=new Scanner(System.in);
        do {
            System.out.print("N (N>0) ? ");
            N=s.nextInt();
        }  while(N<=0);
        ArrayRndGenerator[] g= new ArrayRndGenerator[6];
        for(int i=0; i<6; i++) {
            g[i]=new ArrayRndGenerator(N);
            g[i].start();
        }
        ArraySumThread s1=new ArraySumThread(g[0], g[1]);
        s1.start();
        ArraySumThread s2=new ArraySumThread(g[2], g[3]);
        s2.start();
        ArraySumThread s3=new ArraySumThread(g[4], g[5]);
        s3.start();
        ArraySumThread s4=new ArraySumThread(s1, s2);
        s4.start();
        ArraySumThread s5=new ArraySumThread(s4, s3);
        s5.start();
        try {
            s5.join();
            int[] r=s5.getResult();
            for(int i=0; i<N; i++)
                System.out.println(r[i]);
        } catch(InterruptedException e) {
        }
    }
}
```