Sistemi Operativi

PRIMA PROVA IN ITINERE – 13 maggio 2015

Esercizio 1. (punti 12)

Un insieme di thread è caratterizzato dai tempi di CPU burst e I/O burst, dai tempi di arrivo in coda di ready e dalle priorità riportate di seguito (a valori più piccoli corrispondono priorità maggiori):

Thread	**T_{arrivo}**	**Priorità**	**Sequenza CPU/IO**
T_1	0	3	CPU(4) / IO(2) / CPU(3)
T_2	1	2	CPU(2) / IO(3) / CPU(1)
T_3	2	1	CPU(1) / IO(1) / CPU(3)
T_4	3	2	CPU(3) / IO(3) / CPU(4)

I thread sono organizzati in un'unica coda di ready. La coda è gestita in accordo ad un algoritmo di scheduling con *priorità*, *prelazione* ed *aging*. La politica di aging opera nel modo seguente:

- La priorità dei thread è ricalcolata ogni 3ms. I thread che al momento del ricalcolo sono in coda di ready aumentano di uno la loro priorità (il valore diminuisce di 1). La priorità non può essere comunque inferiore a 0. Al ricalcolo, un thread che assume una priorità maggiore o uguale a quella del thread in esecuzione lo prelaziona;
- Un thread in esecuzione che torna in coda di ready, a seguito di I/O o perché prelazionato, riottiene la sua priorità iniziale;
- A parità di priorità tra processi ready, la CPU è assegnata al processo da più tempo in attesa in coda di ready.

Applicando la suddetta politica di scheduling si determinino:

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

Esercizio 2. (punti 18)

Scrivere il programma Java che opera nel modo seguente:

- All'interno del metodo `main` di una classe `Principale` è istanziato un array di `N` oggetti di tipo `java.util.ArrayList` (con `N` valore intero > 0 letto da tastiera);
- Gli elementi di ogni lista sono inizializzati con valori di tipo intero generati in modo random tra 1 e 100 e trasformati ad oggetti chiamando il costruttore `Integer(int value)` della classe `java.lang.Integer`;
- Un thread ti tipo `SearchMinimum` è creato per ogni lista dell'array. Ad ogni thread è passata ordinatamente una delle liste;
- Il thread cerca il valore massimo tra quelli della lista ad esso passata;
- Il metodo `main` verifica il massimo assoluto tra tutti i massimi delle liste calcolati dai thread e lo stampa.

NOTA

La classe `ArrayList` implementa una lista di oggetti generici. Metodi della classe utili per la soluzione:

```
public ArrayList(); // costruisce una lista con capacità iniziale di 10 elementi
public boolean add(Object obj); // aggiunge un elemento in coda alla lista
public Object get(int i); // ritorna l'elemento della lista in posizione i
public int size(); // ritorna il numero di elementi della lista
```

La classe `Integer` realizza un oggetto contenitore del tipo primitivo `int`. Metodi della classe utili per la soluzione:

```
public Integer(int value); // costruttore
public int intValue(); // ritorna il valore intero associate all'oggetto Integer
```