UNIVERSITÀ DEGLI STUDI DI BERGAMO

DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE E METODI MATEMATICI

Corso di Laurea in Ingegneria Informatica

Esame di **Sistemi Operativi**

Modulo di Informatica II e del C.I. di Reti di Calcolatori e Sistemi Operativi

Appello del 4 Luglio 2011

1. Spiegare le differenze tra *paginazione* e *segmentazione* come metodi di allocazione della memoria. **[max 5 pt]**
2. Considerato l'ambito della schedulazione Real-Time:
   a. Descrivere l'algoritmo di schedulazione *Hard Real-Time a frequenza monotona*. **[max 7 p.t.]**
   b. Si consideri il seguente insieme di processi (t è il tempo di picco, p il periodo e d la scadenza):

Processo	t	p=d	% uso CPU
P_1	30	100	0.30
P_2	10	40	0.25
P_3	20	50	0.4

   e si esegua l'algoritmo di cui al punto a., effettuando il *controllo di ammissione* e disegnando il *diagramma di Gantt* fino a 210 ms. L'esempio dato è schedulabile? Motivare la risposta. **[max 8 p.t.]**

3. *Quesito per gli studenti del* ***C.I. di Reti di calcolatori e Sistemi operativi:*** **[max 10 pt]**
   Nell'ambito dei metodi di sincronizzazione dei processi, si illustri, con un pseudolinguaggio di programmazione, una soluzione al *problema dei lettori-scrittori* che faccia uso dei *semafori*.

4. *Quesito riservato agli studenti di* ***Informatica II (21013+23014)*****:** **[max 10 p.t.]**
   Si illustri, con il linguaggio Java, una soluzione al problema di prelevare lattine di coca-cola da una macchinetta e di rifornirla nel caso in cui rimanga vuota. Si utilizzi a tale scopo i *lock* e le *variabili condizione* del package java.util.concurrent come meccanismo di sincronizzazione.
   ATTENZIONE! Non è richiesto definire le classi per i thread utenti e rifornitore. Definire, piuttosto, la classe CokeMachine contenente le lattine di coca-cola (oggetti condivisi) ed i metodi: remove(…), eseguito dal generico utente per prelevare una lattina dalla macchinetta; deposit(…), eseguito dal fornitore del servizio per caricare la macchinetta bel caso in cui rimane vuota. Si assume che inizialmente la macchinetta è piena, e che un utente (a scelta: il primo a trovare la macchinetta vuota o l'utente che preleva l'ultima lattina) può segnalare al fornitore che la macchinetta è vuota. Lo scheletro di tale classe è riportato di seguito come piccolo aiuto.

```
import java.util.concurrent.locks.*;
class CokeMachine{
        static final int N = 50; //Capacità della macchinetta
        int count ; //Numero effettivo di lattine presenti nella macchinetta
        final Lock lock = new ReentrantLock(); //Variabile di lock per la mutua esclusione
        …. //Condition variable
        …. <COMPLETARE>….
        ….
}
```

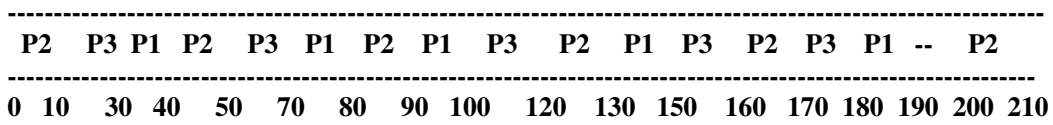
SOLUZIONE

1. Vedi Cap. 8 (sezioni 8.4 e 8.6) del libro adottato.

2.a Vedi Cap.19 (sez 19.5) del libro adottato.

2.b Il campione di processi è ammesso alla schedulazione perché la somma delle percentuali t/p è 0,3+0,25+0,4=0,95 ovvero 95% < 100% . Inoltre, in base alla priorità statica (l'inverso del periodo), i processi vengono eseguiti nell'ordine: P2, P3, P1. P1 si ripresenta/scade (p=d) agli istanti 100, 200, 300, ecc..; P2 si ripresenta/scade (p=d) a 40, 80, 120, 160, 200, ecc..; e P3 si ripresenta/scade (p=d) a 50, 100, 150, 200, ecc..

In base al diagramma di *Gantt* di seguito riportato, il campione di processi è schedulabile con tale algoritmo. Ogni processo viene, infatti, portato a termine (a volte anche al limite!) prima di ogni scadenza.



P1 interrotto all'istante 40 da P2 che si ripresenta, restano 20 ms di P1
P1 nuovamente interrotto all'istante 80 da P2 che si ripresenta, restano 10 ms di P1
La prima esecuzione di P1 si completa appena in tempo a 100, e si ripresenta a 100
P1 viene interrotto a 150 da P3 che si ripresenta, restano 10 ms di P1
P3 viene interrotto a 160 da P2 che si ripresenta, restano 10 ms di P3
La quarta esecuzione di P3 si completa a180 (scadrebbe a 200)
La seconda esecuzione di P1 si completa a 190, seguono 10 istanti di inattività e poi si ricomincia a 200 (dove si ripresentano tutti) con P2.

3. Vedi Cap. 6 del libro adottato.

4. Omessa.