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 24 Gennaio 2011

1. Spiegare la differenza tra processo I/O-bound e processo CPU-bound. In quale ambito nella progettazione di un sistema operativo bisogna prestare attenzione a questa classificazione? Perché? **[max 4 pt]**
2. Nell'ambito delle strategie di allocazione dei file per l'implementazione del file system, si illustri la *tecnica di allocazione indicizzata* ed in particolare lo *schema combinato* usato nel file system UFS di Unix. **[max 8 p.t.]**
3. Dato il sistema descritto dalla seguente rappresentazione insiemistica delle assegnazioni di risorsa, dove P denota l'insieme dei processi, R l'insieme delle risorse R_i^j di indice i e molteplicità j, $\mathcal{E}(\mathsf{P})$ l'insieme delle richieste di accesso a risorse in R emesse da processi in P e *attualmente pendenti*, e $\mathcal{E}(\mathsf{R})$ l'insieme delle *assegnazioni* attuali di risorse in R a processi in P: **[max 9 p.t.]**

$$\mathcal{P} = \{P_1; P_2; P_3; P_4; P_5; P_6\} \quad (1)$$
$$\mathcal{R} = \{R_1^2; R_2^1; R_3^2; R_4^1\} \quad (2)$$
$$\mathcal{E}(\mathcal{P}) = \{P_2 \rightarrow R_2; P_3 \rightarrow R_4; P_4 \rightarrow R_1; P_5 \rightarrow R_4; P_6 \rightarrow R_3\} \quad (3)$$
$$\mathcal{E}(\mathcal{R}) = \{R_1 \rightarrow (P_1, P_2); R_2 \rightarrow P_3; R_3 \rightarrow (P_4, P_5); R_4 \rightarrow P_6\} \quad (4)$$

   - Si analizzi il grafo di allocazione delle risorse, determinando se il sistema si trovi attualmente in situazione di stallo o meno.
   - Successivamente, si studi l'evoluzione dello stato del sistema ove il processo P_1 richiedesse accesso alla risorsa R_2 a partire dalla situazione data.

4. *Quesito per gli studenti del **C.I. di Reti di calcolatori e Sistemi operativi:*** **[max 9 pt]**
   Nell'ambito dei metodi di sincronizzazione dei processi:
   a. Si descriva l'implementazione dei semafori con *attesa attiva* (*busy waiting*).
   b. Si illustri, con un pseudolinguaggio di programmazione, un frammento di codice che fà un uso scorretto di un semaforo mutex portando ad una situazione di *starvation*.

5. *Quesito riservato agli studenti di **Informatica II (21013+23014)**:* **[max 9 p.t.]**

Il problema del *Single Lane Bridge* Si consideri il problema di concorrenza rappresentato in Figura 1. Un ponte (*bridge*) lungo un fiume è talmente stretto da permettere una sola carreggiata di transito da usare in modo *bidirezionale* dalle autovetture provenienti da entrambe le direzioni (auto rosse da sinistra e auto blu da destra). Uno scontro tra due autovetture (violazione della proprietà di *safety*) avviene se due autovetture (vedi Figura 1) provenienti da direzioni opposte (una rossa ed una blu) entrano nel ponte nello stesso tempo.

Figura 1

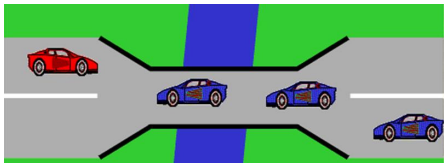


Figura 2

La Figura 2 suggerisce una soluzione al problema del transito attraverso una turnazione tra macchine blu e rosse. Si fornisca tale soluzione in Java sfruttando il meccanismo di sincronizzazione diretta (i metodi *wait()/notify()*). A tale scopo, si definisca una classe `SafeBridge` che implementi l'interfaccia `Bridge` di seguito riportata contenente i metodi eseguiti dai thread (le auto rosse e blu) per l'ingresso (*enter*) e l'uscita (*exit*) dal ponte.

```
interface Bridge {
   abstract void redEnter() throws InterruptedException;
   abstract void redExit();
   abstract void blueEnter() throws InterruptedException;
   abstract void blueExit();
}
```

[Suggerimento: Definire nella classe `SafeBridge` attributi per mantenere il conteggio delle auto blu e rosse in transito sul ponte. Non occorre definire le classi dei thread.*]*

RISULTATI PROVA SCRITTA

Belotti Michele 20
Ghalbi Mohsen 23
Servalli Denis 28
Terzi Fabio 29

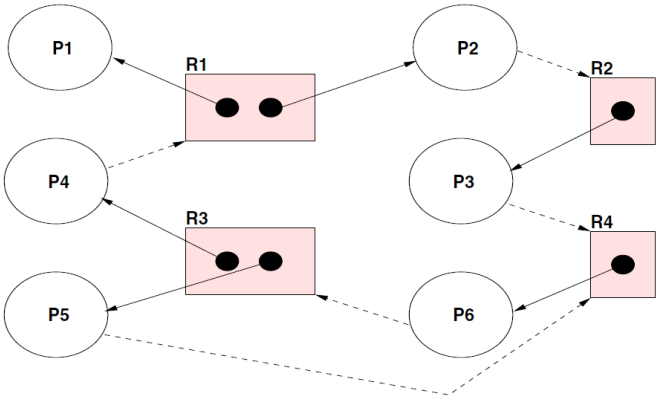
La prova orale è fissata per martedì 31 ore 10.00 nell'ufficio del Docente (Ed. B terzo piano).

SOLUZIONE

1. Un processo **I/O-bound** è un processo che spende più tempo facendo I/O che elaborazione (molti e brevi utilizzi di CPU); mentre, un processo **CPU-bound** spende più tempo facendo elaborazione che I/O (pochi e lunghi utilizzi di CPU). Questa classificazione viene tenuta in considerazione quando si progettano gli schedulatori di SO *multitasking* per la turnazione dei processi sulla CPU. L'obiettivo è garantire buone prestazioni del sistema e massimizzare lo sfruttamento della CPU con una combinazione bilanciata di processi CPU-bound e I/O-bound. Tale compito è lasciato allo *schedulatore a breve termine*, e anche allo *schedulatore a medio termine* (se presente) in quanto lo *swapping* può essere necessario per migliorare la distribuzione dei processi tra le due tipologie.

2. Omessa

3. Grafo di allocazione delle risorse per il sistema dato:



Si distinguono due percorsi chiusi (cicli):

percorso 1:

$$\{P_5 \rightarrow R_4 \rightarrow P_6 \rightarrow R_3 \rightarrow P_5\}$$

percorso 2:

$$\{P_2 \rightarrow R_2 \rightarrow P_3 \rightarrow R_4 \rightarrow P_6 \rightarrow R_3 \rightarrow P_4 \rightarrow R_1 \rightarrow P_2\}$$

In entrambi i percorsi è presente almeno una risorsa completamente impegnata ma a molteplicità > 1 e quindi con almeno un processo potenzialmente non implicato nel percorso chiuso. **In situazioni di questo genere non si può trarre alcuna conclusione a priori sullo**

stato di stallo del sistema, ma bisogna analizzare il problema specifico dato nei dettagli. Il percorso 1 condivide la risorsa R_3 a molteplicità > 1 con il percorso 2, il che comporta la stessa caratteristica per entrambi i percorsi: o sono entrambi bloccati o si potranno liberare successivamente. Lo stato di stallo del percorso 1 dipende dall'evoluzione della risorsa R3, cioè dalla possibilità che essa si liberi indipendentemente dai processi P5 e P6 implicati nel percorso. Lo stato della risorsa R3 però dipende anche dal percorso 2, e quindi dall'evoluzione della risorsa R1 (l'altra risorsa a molteplicità > 1). La risorsa R1 è in possesso del processo P1, che non è incluso nei due percorsi chiusi, e che quindi può procedere regolarmente e prima o poi rilasciare R1.

- **Quando P1 rilascia R1**, P4 potrà accedere alla risorsa R1, e quindi riprendere ad avanzare. Al proprio completamento, P4 rilascerà una istanza della risorsa R3, che potrà essere devoluta al processo P6, che potrà così a sua volta avanzare fino al proprio completamento. Infine verrà rilasciata anche la risorsa R4, che potrà essere assegnata al processo P5 o P3. La sequenza di completamento e di rilascio procede poi a catena, consentendo la ripresa di tutti gli altri processi in attesa. Si può pertanto concludere che **la situazione proposta non è di stallo**.
- **Qualora invece il processo P1 richiedesse la risorsa R2**, partendo dalla situazione iniziale, questo creerebbe un nuovo percorso chiuso che manterrebbe sospeso anche P1, così **determina**ndo **lo stallo completo del sistema.**

4.a Omessa
4.b Omessa
5. Omessa