Sistemi Operativi A

23 Gennaio 2007

Esercizio 1

Che differenza c'è tra un algoritmo di scheduling preemptive (con prelazione) e uno non preemptive (senza prelazione)? Illustrare basandosi sul diagramma degli stati dei processi. Dare, infine, un esempio di algoritmo preemptive con priorità statiche e di uno non preemptive con priorità dinamiche.

Esercizio 2

Si consideri un sistema con 2 Risorse di tipo R1 e 1 risorsa di tipo R2. Inizialmente, P1 richiede una risorsa di tipo R1 e P2 richiede una risorsa di tipo R2. Per terminare, il processo P1 necessita di una risorsa di tipo R2, mentre il processo P2 necessita di altre 2 risorse di tipo R1.

1. Discutere, utilizzando il grafo di assegnazione delle risorse, la possibilità di stallo nel sistema sopra descritto;

2. Mostrare come lo stallo viene evitato utilizzando una tecnica di allocazione gerarchica delle risorse;

3. Mostrare, infine, come lo stallo viene evitato tramite l'algoritmo del banchiere.

Esercizio 3

Cosa si intende per *attesa indefinita* (starvation)? Dare un esempio di attesa indefinita negli algoritmi di scheduling e un esempio nella sincronizzazione tra processi, spiegando brevemente eventuali soluzioni a tale problema.

Esercizio 4

Si consideri il seguente processo

```
process CopiaFile(i,j){
    fs.acquisisci(i,j)
    <copia dati dal file i al file j>
    fs.rilascia(i,j)
}

monitor fs {
    boolean lock[NUMFILE]=false;
    condition attendi;

    void acquisici(int i,int j) {        void rilascia(int i,int j) {




    }                                    }
```

in cui `num.acquisisci(i,j)` deve acquisire l'utilizzo dei file i e j in modo esclusivo, `num.rilascia(i,j)` deve rilasciare la mutua esclusione sui file i e j.

1. Completare il codice di tali funzioni (direttamente negli appositi spazi vuoti) spiegando, qui sotto, la soluzione proposta

2. Considerare l'esecuzione concorrente di CopiaFile(1,0) e CopiaFile(0,1) e valutare la possibilità di stallo di tali processi. Nella valutazione fare riferimento alle condizioni necessarie per lo stallo.

Esercizio 5

Si consideri un processo che fa rifermento a 5 pagine virtuali nel seguente ordine:

$$1, 2, 2, 3, 2, 3, 4, 2, 2, 3, 5, 1, 3, 5$$

Si consideri una memoria fisica (inizialmente vuota) di 3 frame e si mostri il funzionamento degli algoritmi seguenti:

1. FIFO (First In First Out),

   Spiegare brevemente:

	1	2	2	3	2	3	4	2	2	3	5	1	3	5
Frame 1														
Frame 2														
Frame 3														
Page Fault														

2. LRU (Least Recently Used),

   Spiegare brevemente:

	1	2	2	3	2	3	4	2	2	3	5	1	3	5
Frame 1														
Frame 2														
Frame 3														
Page Fault														

3. LFU (Least Frequently Used).

   Spiegare brevemente:

	1	2	2	3	2	3	4	2	2	3	5	1	3	5
Frame 1														
Frame 2														
Frame 3														
Page Fault														

Esercizio 6

Considerare tre processi $P1$, $P2$ e $P3$ il cui compito è di, rispettivamente, produrre un dato x, calcolare $y = f(x)$ e calcolare e stampare $z = g(y)$. Le tre variabili x, y, z sono condivise, pertanto i tre processi devono essere opportunamente sincronizzati in modo che $f(x)$ venga calcolata dopo che x è stata prodotta e che $g(y)$ venga calcolata dopo che $y = f(x)$ è stata calcolata. Aggiungere le necessarie sincronizzazioni al codice dei tre processi utilizzando i semafori. Spiegare il funzionamento della soluzione indicata discutendone la scalabilità nel caso di più istanze concorrenti di $P1, P2, P3$.

```
P1 {                        P2 {                        P3 {
  while(true) {               while(true) {               while(true) {


    <produci x>                 y = f(x);                   z = g(y);

                                                            print(z);


} }                         } }                         } }
```

Spiegazione:

Esercizio 7

Si consideri un sistema time-sharing con scheduling a priorità a code multiple: Le prime due code sono Round Robin con quanto $q = 1$ e $q = 2$, rispettivamente, la terza e ultima coda è SJF (Shortest Job First) (Questa coda verrà implementata con un'approssimazione del tempo di esecuzione futuro ma per l'esercizio considerare il tempo di esecuzione effettivo). I processi che utilizzano per intero il quanto vengono declassati alla coda inferiore. I processi che terminano un'attesa, invece, vengono inseriti nella prima coda RR.
Sono presenti 4 processi P1, P2, P3 e P4 nei seguenti stati: P1 è in esecuzione, proveniente dalla prima coda; P2 è pronto nella prima coda, P3 e P4 sono in attesa su operazione di I/O. I processi P1 e P2 devono utilizzare la CPU per 2 e 5 quanti di tempo prima di terminare. I processi P3 e P4 terminano l'operazione di I/O dopo 4 e 5 quanti di tempo, rispettivamente. Utilizzeranno la CPU per 2 e 3 quanti di tempo prima di terminare.

1. Illustrare come evolve lo stato del sistema utilizzando lo schema qui riportato.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1																				
P2																				
P3																				
P4																				

   Indicare con:

   - CPU il processo in esecuzione
   - R1,R2,R3 i processi pronti, rispettivamente, nelle code 1, 2 e 3.
   - I/O i processi in attesa

2. Descrivere, tramite un opportuno schema, le strutture dati utilizzate dall'algoritmo di scheduling.