Sistemi Operativi
4 febbraio 2013
Compitino I B

IMPORTANTE:

- **Si risponda ai seguenti quesiti, giustificando le risposte.**
- **Gli studenti del corso di laurea in Informatica risolvano gli esercizi 1–6.**
- **Gli studenti del corso di laurea in TWM, che hanno in piano di studi Sistemi Operativi come corso da 6 crediti, risolvano gli esercizi 1–5.**
- **Gli studenti del corso di laurea in TWM, che hanno in piano di studi Sistemi Operativi come corso da 9 crediti, risolvano gli esercizi 1–6.**

1. (a) Può un processo a priorità bassa influire sull'esecuzione di un processo a più alta priorità? Se sì, si diano alcuni esempi.
   (b) Il problema descritto sopra può verificarsi con thread di livello utente?
   (c) Può verificarsi se l'algoritmo di scheduling è round robin?

   Risposta:

   (a) Sì, il fenomeno è noto come *inversione di priorità*. Si avvera quanto, per esempio, un processo a bassa priorità detiene una risorsa indispensabile ad un processo ad alta priorità pronto ad eseguire. Di conseguenza si può creare una situazione in cui un processo ad alta priorità deve attendere (passando in stato waiting) un processo a bassa priorità per proseguire e quest'ultimo non può andare in esecuzione in quanto esiste un terzo processo a priorità intermedia fra i due che ottiene tutto il tempo della CPU.

   (b) Sì.

   (c) Nello scheduling round robin puro non ci sono priorità. Quindi, prima o poi il processo che detiene la risorsa terminerà la sua esecuzione rilasciandola e consentendo la continuazione del processo che l'attendeva.

2. (a) In quali situazioni può essere attivato lo scheduling della CPU?
   (b) Quando un algoritmo di scheduling è preemptive? Quali sono i vantaggi e gli svantaggi di un algoritmo preemptive?
   (c) Gli scheduler Round Robin mantengono una lista di tutti i processi pronti, in cui ogni processo compare una sola volta. Che cosa potrebbe accadere se un processo apparisse due volte nella lista? Ci sono situazioni in cui questo potrebbe essere permesso?

   Risposta:

   (a) Le situazioni in cui può essere attivato lo scheduler della CPU sono le seguenti:

   1. quando un processo passa da running a waiting,
   2. quando un processo passa running a ready,
   3. quando un processo passa da waiting a ready.
   4. quando un processo termina.

   Nei casi due e tre si parla di prelazione.

   (b) Un algoritmo di scheduling è con prelazione quando un processo può essere costretto a rilasciare la CPU. In genere ciò avviene lasciandolo in esecuzione per un certo quantitativo di tempo determinato dall'arrivo di un interrupt dell'orologio di sistema. Scaduto quindi il tempo di esecuzione concesso, la CPU viene assegnata ad un altro processo. Il vantaggio di uno scheduling con prelazione è che nessun processo potrà mai impossessarsi indefinitamente della CPU. Inoltre il tempo di CPU viene ripartito in modo più equo fra i vari processi. Uno svantaggio è che, se il criterio di scelta del nuovo processo da eseguire è basato sul livello di priorità, si corre il rischio che processi a bassa priorità vengono ritardati in modo indefinito. Altri svantaggi sono relativi alla condivisione dei dati fra processi; infatti, se un processo sta manipolando dei dati utilizzati anche da altri processi e viene prelazionato c'è il rischio che questi rimangano in uno stato inconsistente e generino così degli errori. Per evitare tutto ciò è necessario un attento uso di primitive come mutex e semafori per garantire un accesso esclusivo e corretto alle risorse condivise.

(c) Inserendo due occorrenze dello stesso processo, quest'ultimo, nel caso in cui ad esempio le due occorrenze siano consecutive, si vedrebbe assegnare un tempo di CPU doppio rispetto agli altri processi. Infatti, scaduto il primo quanto di tempo, la prima occorrenza verrebbe spostata in fondo alla coda, ma la seconda conferirebbe un ulteriore quanto di tempo al processo in questione. In generale quindi il processo beneficerebbe di un tempo di CPU doppio nel corso di ogni singola scansione della coda dei processi pronti. Un vantaggio derivante dall'utilizzo di questo sistema è quello di poter assegnare quanti di tempo diversi a seconda delle necessità di ogni singolo processo. Uno svantaggio potrebbe essere il rallentamento di alcuni processi a causa dell'inserimento "non accorto" di occorrenze "doppie" nella coda. Un ulteriore aspetto negativo è costituito dal fatto che nel caso di occorrenze consecutive dello stesso processo ci sarebbe comunque uno spreco di tempo per l'esecuzione delle system call dovute agli interrupt del timer di sistema per gestire l'assegnazione dei quanti di tempo (tali chiamate di sistema potrebbero essere evitate in quanto il processo che va in esecuzione rimane sempre lo stesso).

3. Si consideri un sistema con scheduling a priorità con tre code, A, B, C, di priorità crescente, con prelazione tra code. Le code A e B sono round robin con quanto di 10 e 15 msec, rispettivamente; la coda C è FCFS. Se un processo nella coda A o B consuma il suo quanto di tempo, viene spostato in fondo alla coda B o C, rispettivamente.

   Nelle code A, B, C entrano i seguenti processi:

   - P_1 arriva nella coda A all'istante 0 e ha CPU burst di 30 msec;
   - P_2 arriva nella coda C all'istante 5 e ha CPU burst di 20 msec;
   - P_3 arriva nella coda B all'istante 15 e ha CPU burst di 15 msec;
   - P_4 arriva nella coda A all'istante 20 e ha CPU burst di 15 msec.

   Si determini:

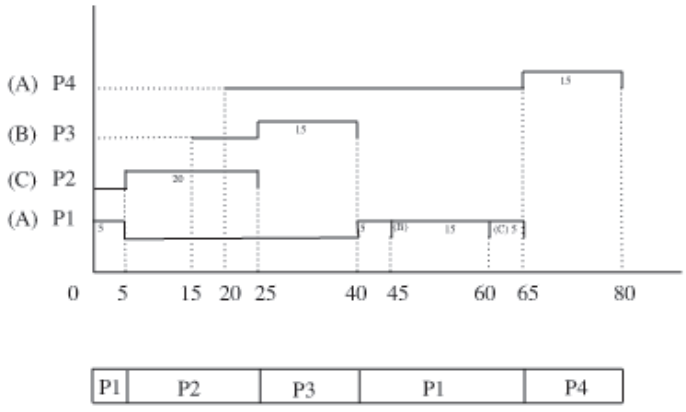
   (a) il diagramma di GANTT relativo all'esecuzione dei quattro processi;

   (b) il tempo di attesa medio;

   (c) il tempo di reazione medio.

   Risposta:

   (a) 1. Il diagramma di Gantt è il seguente:



   2. Tempo di attesa medio = $\frac{45+0+10+45}{4} = \frac{100}{4} = 25$ ms.
   3. Tempo di reazione medio = $\frac{0+0+10+45}{4} = \frac{55}{4} = 13,75$ ms.

4. Che cosa significa "corsa critica"? Si diano esempi in cui si può verificare.

   Risposta: Per "corsa critica" si intende la situazione in cui più processi accedono concorrentemente agli stessi dati ed il risultato finale dipende dall'ordine di interleaving dei processi. Si tratta di eventi frequenti nei sistemi operativi multitasking, sia per dati in user space sia per le strutture del kernel. Un tipico esempio è l'accesso concorrente al contenuto di un file su disco, di un database ecc.

5. Completare il seguente codice che implementa una versione del problema dei lettori e scrittori favorevole ai lettori.

```
semaphore mutex = 1;
semaphore db = 1;
int rc = 0;

void reader(void)
{
   while (TRUE) {
      ............;
      rc = rc+1;
      if (rc == 1) ................. ;
      ..................;
      read_data_base();
      ..................;
      rc = rc-1;
      if (rc == 0) ............;
      ..................;
      use_data_read();
   }
}

void writer(void)
{
   while (TRUE) {
      think_up_data;
      ............;
      write_data_base;
      ............
   }
}
```

Risposta:

```
semaphore mutex = 1;
semaphore db = 1;
int rc = 0;

void reader(void)
{
   while (TRUE) {
      down(&mutex);
      rc = rc+1;
      if (rc == 1) down(&db) ;
      up(&mutex);
      read_data_base();
      down(&mutex);
      rc = rc-1;
      if (rc == 0) up(&db);
      up(&mutex);
      use_data_read();
   }
}

void writer(void)
{
```

```
   while (TRUE) {
      think_up_data;
      down(&db);
      write_data_base;
      up(&db);
   }
}
```

6. Si consideri la seguente situazione, dove P_0, P_1, P_2, P_3, P_4 sono cinque processi in esecuzione, C è la matrice delle risorse correntemente allocate, Max è la matrice del numero massimo di risorse da assegnare ad ogni processo e A è il vettore delle risorse disponibili:

	C				Max			
	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2
P_1	1	0	0	0	1	7	5	0
P_2	1	3	5	4	2	3	5	6
P_3	0	6	3	2	0	6	5	2
P_4	0	0	1	4	0	6	5	6

Available (A)

A	B	C	D
1	5	2	0

(a) Calcolare la matrice R delle richieste.

(b) Il sistema è in uno stato sicuro?

(c) La richiesta (0 4 2 0) da parte del processo P_1 può essere soddisfatta immediatamente?

Risposta:

(a) $R = MAX - C$:

R			
A	B	C	D
0	0	0	0
0	7	5	0
1	0	0	2
0	0	2	0
0	6	4	2

(b) Il sistema è in uno stato sicuro in quanto esiste la sequenza di esecuzione sicura P_0 (nuovo valore di $A = (1\ 5\ 3\ 2)$), P_2 (nuovo valore di $A = (2\ 8\ 8\ 6)$), P_1 (nuovo valore di $A = (3\ 8\ 8\ 6)$), P_3 (nuovo valore di $A = (3\ 14\ 11\ 8)$) e P_4 (nuovo valore di $A = (3\ 14\ 12\ 12)$).

(c) Se la richiesta (0 4 2 0) da parte del processo P_1 viene soddisfatta immediatamente, si hanno i seguenti nuovi valori per le matrici C, R e A:

C				R			
A	B	C	D	A	B	C	D
0	0	1	2	0	0	0	0
1	4	2	0	0	3	3	0
1	3	5	4	1	0	0	2
0	6	3	2	0	0	2	0
0	0	1	4	0	6	4	2

$$A = (1\ 1\ 0\ 0)$$

La richiesta può essere accettata in quanto esiste la sequenza di esecuzione sicura P_0 (nuovo valore di $A = (1\ 1\ 1\ 2)$), P_2 (nuovo valore di $A = (2\ 4\ 6\ 6)$), P_1 (nuovo valore di $A = (3\ 4\ 6\ 6)$), P_3 (nuovo valore di $A = (3\ 10\ 9\ 8)$) e P_4 (nuovo valore di $A = (3\ 10\ 10\ 12)$).