Sistemi Operativi
09 aprile 2008
Compitino 1

Si risponda ai seguenti quesiti, giustificando le risposte.

1. (a) In quali situazioni può essere attivato lo scheduler della CPU?
   (b) Quando un algoritmo di scheduling è con prelazione? Quali sono vantaggi e svantaggi di uno scheduling con prelazione?

   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. Lo 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.

2. (a) Che cos'è un processo? Che cos'è un thread?
   (b) In un sistema basato su thread, quali attributi dei processi tradizionali sono assegnati ai processi e quali ai thread?

   Risposta:

   (a) Un processo è un programma in esecuzione nel sistema; quindi non consiste del solo codice, ma anche di tutte le risorse correlate: il program counter (PC), i registri, lo stack, lo stato di esecuzione, lo spazio di indirizzamento, le variabili globali, i file aperti, i timer in scadenza, i segnali e le routine di gestione dei segnali, le informazioni di accounting ecc.

   I thread sono unità di esecuzione all'interno di uno stesso processo; ogni thread è caratterizzato da un proprio PC, da un proprio insieme dei valori dei registri, da un proprio stack ed un proprio stato di esecuzione, mentre tutte le risorse rimanenti (spazio in memoria, file aperti ecc.) vengono condivise fra thread appartenenti allo stesso processo.

   (b) In un sistema basato su thread, gli attributi dei processi tradizionali assegnati ai processi sono i seguenti:

   - spazio di indirizzamento,
   - variabili globali,
   - file aperti,
   - processi figli,
   - timer in scadenza,
   - segnali e routine di gestione dei segnali,
   - informazioni di accounting.

   Gli attributi assegnati ai thread invece sono i seguenti:

   - il program counter (PC),
   - i registri,
   - lo stack,
   - lo stato di esecuzione.

3. Si consideri un sistema con scheduling SJF con prelazione (cioè SRTF), ove $\alpha = 0,5$ e $\tau_0 = 30$ msec. All'istante 0 il processore si libera e tre processi, P_1, P_2, P_3, sono in coda ready. Finora i processi P_1, P_2 sono andati in esecuzione due volte con CPU burst $30, 30$ msec per P_1 e $25, 40$ msec per P_2; mentre P_3 è andato in esecuzione una volta con CPU burst di 50 msec.

Sistemi Operativi
09 aprile 2008
Compitino 1

1. Quale processo viene selezionato dallo scheduler all'istante 0?
2. All'istante 10 msec entra nella coda ready un nuovo processo P_4 con CPU burst previsto di 15 msec. Il processo selezionato precedentemente è ancora in esecuzione. Che cosa succede?
3. Che cosa succede quando il processo in esecuzione termina il suo burst?

Risposta:

1. Data la situazione seguente:

	t_0	t_1
P_1	30	30
P_2	25	40
P_3	50	

abbiamo: $\tau_1^{P_1} = \frac{1}{2}\tau_0 + \frac{1}{2}t_0^{P_1} = 15 + 15 = 30\ ms$, $\tau_2^{P_1} = \frac{1}{2}\tau_1^{P_1} + \frac{1}{2}t_1^{P_1} = 15 + 15 = 30\ ms$, $\tau_1^{P_2} = \frac{1}{2}\tau_0 + \frac{1}{2}t_0^{P_2} = 15 + \frac{25}{2} = 27,5\ ms$, $\tau_2^{P_2} = \frac{1}{2}\cdot 27,5 + \frac{1}{2}\cdot 40 = 13,75 + 20 = 33,75\ ms$, $\tau_1^{P_3} = \frac{1}{2}\tau_0 + \frac{1}{2}t_0^{P_3} = 15 + 25 = 40\ ms$, quindi all'istante 0 va in esecuzione P_1 poiché $\tau_2^{P_1} < \tau_2^{P_2} < \tau_1^{P_3}$.
2. All'istante 10 ms arriva P_4 con $\tau_0^{P_4} = 15\ ms$: P_1 ha eseguito per 10 ms, quindi si prevede che debba eseguire ancora per 20 ms>15 ms. Pertanto P_1 viene prelazionato e parte P_4.
3. Quando P_4 termina il suo burst riparte P_1.

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

	C			Max		
	A	B	C	A	B	C
P_0	1	3	3	2	4	5
P_1	3	6	2	3	9	4
P_2	2	5	3	3	7	11

Available (A)		
A	B	C
1	x	x

   (a) Calcolare la matrice R delle richieste.
   (b) Determinare il minimo valore di x tale che il sistema si trovi in uno stato sicuro.
   (c) Dopo aver determinato il valore di x con le caratteristiche descritte al punto precedente, dire se la richiesta $(1, 2, 3)$ per il processo P_2 viene accettata oppure rifiutata dall'algoritmo del banchiere (spiegandone il motivo).

Risposta:

(a) Matrice delle richieste $R = Max - C$:

	R	
A	B	C
1	1	2
0	3	2
1	2	8

(b) Il minimo valore di x tale da rendere lo stato sicuro è 3. Infatti, se $x = 0$, non esiste nessuna riga di R minore od uguale a $A = (1, 0, 0)$. Lo stesso dicasi per $x = 1$. Se $x = 2$, allora $A = (1, 2, 2)$ e $R_0 \leq A$; quindi possiamo assegnare le risorse richieste a P_0 ed eseguirlo. Alla fine otteniamo il nuovo valore di $A = (1, 2, 2) + C_0 = (1, 2, 2) + (1, 3, 3) = (2, 5, 5)$. Siccome $R_1 \leq A$, assegniamo le risorse a P_1 e lo mandiamo in esecuzione. Alla fine il nuovo valore di A è $(2, 5, 5) + C_1 = (2, 5, 5) + (3, 6, 2) = (5, 11, 7)$. Dato che R_2 non è minore od uguale al nuovo valore di A, non riusciamo a completare la sequenza di esecuzione sicura. Pertanto il valore $x = 2$ non rende sicuro il sistema.

Con $x = 3$ invece abbiamo la sequenza sicura $\langle P_0, P_1, P_2 \rangle$. Infatti $A = (1, 3, 3)$ e $R_0 \leq A$; quindi possiamo assegnare le risorse richieste a P_0 ed eseguirlo. Alla fine otteniamo il nuovo valore di

$A = (1,3,3) + C_0 = (1,3,3) + (1,3,3) = (2,6,6)$. Siccome $R_1 \leq A$, assegniamo le risorse a P_1 e lo mandiamo in esecuzione. Alla fine il nuovo valore di A è $(2,6,6) + C_1 = (2,6,6) + (3,6,2) = (5,12,8)$. Si conclude con l'esecuzione di P_2 dato che $R_2 \leq A$, ottenendo il valore finale di A, ovvero, $(5,12,8) + C_2 = (5,12,8) + (2,5,3) = (7,17,11)$.

(c) Se la richiesta $(1,2,3)$ per il processo P_2 viene accettata, allora il nuovo valore di A diventa $(1,3,3) - (1,2,3) = (0,1,0)$. Le matrici C e R diventano le seguenti:

	C	
A	B	C
1	3	3
3	6	2
3	7	6

	R	
A	B	C
1	1	2
0	3	2
0	0	5

Non essendoci quindi nessuna riga di R minore od uguale al valore di A, non è possibile identificare una sequenza di esecuzione sicura. La richiesta va quindi rifiutata, dato che porta il sistema in uno stato non sicuro.

5. Si spieghi brevemente la differenza fra *starvation* e *deadlock*.

   Risposta: Nonostante il problema della starvation sia strettamente correlato a quello del deadlock, si tratta di due cose differenti. Infatti, con il termine starvation, si indica un'assenza di progresso per cui un programma in esecuzione non riesce ad ottenere una risorsa od un servizio (a causa ad esempio della politica di allocazione della risorsa/servizio in questione adottata nel sistema), nonostante non venga mai bloccato. Invece con il termine deadlock si intende una situazione in cui un insieme di processi è bloccato in quanto ogni processo è in attesa di un evento che soltanto un altro processo dell'insieme può provocare. Dato che tutti i processi sono in attesa, nessuno di essi potrà mai produrre uno degli eventi che sbloccherebbero un altro elemento dell'insieme.

6. Si elenchino le condizioni per una buona soluzione del problema della sezione critica.

   Risposta: Le condizioni per una buona soluzione del problema della sezione critica sono:

   - Mutua esclusione: se il processo P_i sta eseguendo la sua sezione critica, allora nessun altro processo può eseguire la propria sezione critica.
   - Progresso: nessun processo in esecuzione fuori dalla sua sezione critica può bloccare processi che desiderano entrare nella propria sezione critica.
   - Attesa limitata: se un processo P ha richiesto di entrare nella propria sezione critica, allora il numero di volte che si concede agli altri processi di accedere alla propria sezione critica prima del processo P deve essere limitato.

   Di solito si suppone che ogni processo venga eseguito ad una velocità non nulla, mentre non si suppone niente sulla velocità relativa dei processi (e quindi sul numero e tipo di CPU).

7. Cos'è un'istruzione TSL (Test and Set Lock) e a cosa serve?

   Risposta: Per istruzione TSL si intende Test-and-Set-Lock, ovvero, un'istruzione che controlla e modifica il contenuto di una parola atomicamente. Lo pseudo codice relativo è il seguente:

```
function Test-and-Set (var target: boolean): boolean;
  begin
    Test-and-Set := target;
    target := true;
  end;
```

Questi due passi devono essere implementati in modo atomico in assembler (ovvero, il bus della memoria deve essere bloccato mentre vengono eseguite le due operazioni in modo da impedire l'accesso ad altri processi).

Dato che l'atomicità delle istruzioni TSL mette al riparo dalle race condition, questo tipo di istruzione risulta utile per implementare le zone di entrata e uscita delle sezioni critiche (anche se con attesa attiva).

Il punteggio attribuito ai quesiti è il seguente: 2, 2, 2, 2, 3, 3, 3, 2, 2, 2, 3, 3, 3. Totale 32 punti.