Sistemi Operativi
19 settembre 2011
Compito

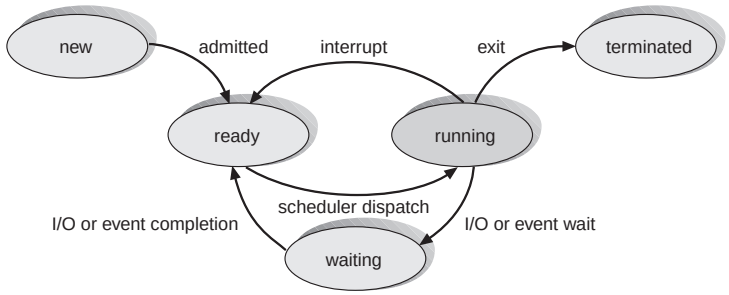
Si risponda ai seguenti quesiti, giustificando le risposte.

1. (a) Si tracci il diagramma degli stati di un processo nel caso di scheduling della CPU con prelazione e nel caso di scheduling senza prelazione.

   (b) Si diano esempi di algoritmi di scheduling con e senza prelazione e se ne discutano vantaggi e svantaggi.

   Risposta:

   (a) (3 punti) Diagramma degli stati con prelazione:

   Per ottenere il diagramma degli stati senza prelazione, è sufficiente togliere l'arco che provoca la transizione dallo stato running allo stato ready.

   (b) (3 punti) Un algoritmo di scheduling con prelazione è RR (Round Robin): un algoritmo specifico per sistemi time-sharing. Ogni processo riceve una piccola unità di tempo di CPU (il quanto): tipicamente 10-100 millisecondi. Dopo questo periodo, il processo viene prelazionato e rimesso nella coda ready. Se ci sono n processi in coda ready, e il quanto è q, allora ogni processo riceve 1/n del tempo di CPU in periodi di durata massima q. Nessun processo attende più di (n - 1)q tempo in coda.

   FCFS (First-Come, First-Served) è invece un esempio di algoritmo senza prelazione che schedula i processi nell'ordine di arrivo in coda ready. Essendo senza prelazione, è inadatto per sistemi time-sharing e può soffrire dell'effetto convoglio (i processi I/O-bound si accodano dietro un processo CPU-bound); un aspetto positivo è la sua equità, ovvero, l'assenza di pericolo di starvation.

2. Si consideri un sistema con scheduling SJF con prelazione (cioè SRTF), ove $\alpha = 0,6$ e $\tau_0 = 20$ 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 $20, 30$ msec per P_1 e $25, 20$ msec per P_2; mentre P_3 è andato in esecuzione una volta con CPU burst di 20 msec.
   Si determini:

   (a) Quale processo viene selezionato dallo scheduler all'istante 0?

   (b) All'istante 10 msec entra nella coda ready un nuovo processo P_4 con CPU burst previsto di 25 msec. Il processo selezionato precedentemente è ancora in esecuzione. Che cosa succede?

   (c) Che cosa succede quando il processo in esecuzione termina il suo burst?

   Risposta:

   (a) (3 punti) Per P_1 abbiamo $\tau_1 = 0,6 \cdot 20 + 0,4 \cdot 20 = 20$, $\tau_2 = 0,6 \cdot 30 + 0,4 \cdot 20 = 18 + 8 = 26$, per P_2 abbiamo $\tau_1 = 0,6 \cdot 25 + 0,4 \cdot 20 = 15 + 8 = 23$, $\tau_2 = 0,6 \cdot 20 + 0,4 \cdot 23 = 12 + 9,2 = 21,2$ ed infine per P_3 abbiamo $\tau_1 = 0,6 \cdot 20 + 0,4 \cdot 20 = 20$. Quindi all'istante 0 SRTF sceglie P_3.

   (b) (3 punti) All'istante 10 msec quando entra nella coda ready un nuovo processo P_4 con CPU burst previsto di 25 msec, continua P_3 perché gli mancano meno di 25 ms ($20 - 10 = 10$ ms) che è il burst previsto per P_4.

   (c) (3 punti) Quando P_3 termina, va in esecuzione P_2, dato che ha il burst previsto più piccolo.

3. (a) Si descriva la tecnica di gestione della memoria della paginazione.

   (b) Si consideri un sistema con tabella delle pagine in memoria fisica, il cui tempo di accesso è di 100ns. Il sistema è dotato di TLB, che può contenere 8 entry della page table e ha tempo di accesso di 10ns. In media l'85% delle entry richieste si trova nella TLB e solo il 2% dei

riferimenti genera un page fault. Il tempo medio richiesto per il rimpiazzamento di una pagina è di 2ms. Si calcoli il tempo effettivo di accesso alla memoria.

Risposta:

(a) (2 punti) La paginazione è una tecnica di allocazione della memoria non contigua che prevede:

- la suddivisione della memoria fisica in *frame* (blocchi di dimensione fissa, una potenza di 2, tra 512 e 8192 byte);
- la suddivisione della memoria logica in *pagine* (della stessa dimensione dei frame).

Quindi, per eseguire un programma di n pagine, servono n frame liberi in cui caricare il programma. È compito del sistema operativo tenere traccia dei frame liberi. Non esiste frammentazione esterna e la frammentazione interna è ridotta all'ultima pagina allocata al processo.

Per quanto riguarda la traduzione degli indirizzi logici in indirizzi fisici, ogni indirizzo generato dalla CPU si suddivide in un numero di pagina p e uno spiazzamento (offset) d all'interno della pagina. Il numero di pagina viene utilizzato come indice nella *page table* del processo correntemente in esecuzione: ogni entry della page table contiene l'indirizzo di base del frame fisico f che contiene la pagina in questione. Tale indirizzo di base f viene combinato (giustapposto come prefisso) con lo spiazzamento d per definire l'indirizzo fisico da inviare all'unità di memoria.

(b) (3 punti) Assumendo che quando avviene il rimpiazzamento della pagina l'entry corrispondente venga messa in memoria, abbiamo che $EAT = \underbrace{0,85 \cdot (10+100)}_{\text{entry in TLB}} + \underbrace{(0,98-0,85) \cdot (10+200)}_{\text{entry non in TLB, ma senza p.f.}} + \underbrace{0,02 \cdot (10+100+2 \cdot 10^6+10+200)}_{\text{entry non in TLB e p.f.}} = 93,5+27,3+0,02 \cdot 2.000.320 = 120,8+40.006,4\ ns \cong 40,1\ ms$

4. Un sistema che usa l'algoritmo del Banchiere contiene unità di risorse n_1 e n_2 di classe R_1 e R_2, rispettivamente, e 3 processi, P_1, P_2, P_3. Le risorse non allocate sono $(1,1)$. Si osservano i seguenti fatti:

   1. Se il processo P_1 fa la richiesta $(1,0)$ seguita da $(0,1)$, allora la prima richiesta viene soddisfatta ma non la seconda.
   2. Se invece il processo P_1 fa solo la richiesta $(0,1)$, allora questa viene soddisfatta.

   Si trovi un possibile insieme di valori per l'allocazione corrente di risorse e le richieste massime di risorse da parte dei processi, compatibili con i fatti sopraelencati.

   Risposta: (3 punti) Come insieme di valori possibile per l'allocazione corrente di risorse assumiamo $n_1 = 8$ e $n_2 = 11$, ovvero, $E = (8,11)$. Le matrici C ed R siano definite come segue:

	C		R	
	R_1	R_2	R_1	R_2
P_1	2	3	5	5
P_2	3	5	4	5
P_3	2	2	2	3

   A questo punto:

   - se la richiesta di P_1 (1,0) è accettata, allora una sequenza di esecuzione sicura è P_3, P_2, P_1;
   - se anche la richiesta di P_1 (0,1) fosse accettata, allora si entrerebbe in uno stato non sicuro;
   - se P_1 fa solo la richiesta (0,1), allora una sequenza di esecuzione sicura è P_2, P_1, P_3.

5. Si illustrino brevemente le due soluzioni più diffuse per la gestione dello spazio libero su disco, evidenziandone pregi e svantaggi.

   Risposta: (3 punti) Una soluzione consiste nell'utilizzare una bitmap in cui ogni singolo bit rappresenta un blocco del disco: se il blocco è allocato ad un file il bit è 0, se il blocco è libero il bit è 1. Il vantaggio di questa codifica è essenzialmente l'efficienza nel trovare il primo blocco libero dato che la maggior parte delle CPU più diffuse mettono a disposizione delle istruzioni macchina che forniscono

l'offset del primo bit a 1 in una parola. In questo modo il numero del primo blocco libero può essere calcolato come segue:

(numero di bit in una parola) × (numero delle parole con tutti i bit 0) + offset del primo bit a 1

Lo svantaggio è che la bitmap deve essere tenuta in memoria per un utilizzo veloce e quindi può portare ad uno spreco di quest'ultima se il disco è di dimensioni ragguardevoli.

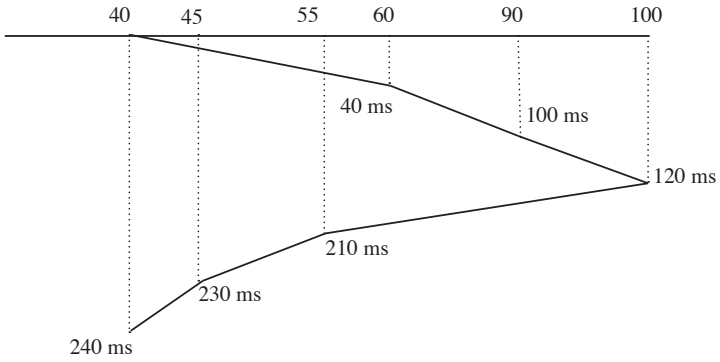
Una soluzione che non richiede un impiego considerevole di memoria è mantenere una lista concatenata dei blocchi del disco liberi (in cui ogni blocco contiene un puntatore al blocco libero successivo). Così facendo è sufficiente mantenere in memoria il puntatore al primo blocco della lista per essere in grado di reperire un blocco libero all'occorrenza. Lo svantaggio è che se si rende necessario attraversare la lista occorre leggere ogni singolo blocco, degradando le prestazioni del sistema in modo considerevole.

6. Si consideri un disco con un intervallo di tracce da 0 a 100, gestito con politica SCAN. Inizialmente la testina è posizionata sul cilindro 40; lo spostamento ad una traccia adiacente richiede 2 ms. Al driver di tale disco arrivano richieste per i cilindri 90, 45, 40, 60, 55, rispettivamente agli istanti 0 ms, 20 ms, 30 ms, 40 ms, 80 ms. Si trascuri il tempo di latenza.

   1. In quale ordine vengono servite le richieste?
   2. Il tempo di attesa di una richiesta è il tempo che intercorre dal momento in cui è sottoposta al driver a quando viene effettivamente servita. Qual è il tempo di attesa medio per le cinque richieste in oggetto?

   Risposta:

   1. (3 punti) Assumendo una direzione di movimento ascendente all'istante 0, le richieste vengono servite nell'ordine 60, 90, 55, 45, 40:



   2. (2 punti) Il tempo di attesa medio per le cinque richieste in oggetto è $\frac{(40-40)+(100-0)+(210-80)+(230-20)+(240-30)}{5} = \frac{0+100+130+210+210}{5} = \frac{650}{5} = 130\ ms.$