Sistemi Operativi
11 giugno 2012
Compito

Si risponda ai seguenti quesiti, giustificando le risposte.

1. Un sistema multiprogrammato utilizza un grado di multiprogrammazione molto grande.Viene proposto di raddoppiare il troughput del sistema migliorando/sostituendo le sue componenti hardware. Il risultato desiderato verrebbe raggiunto con qualcuna delle seguenti proposte?
   (a) Sostituire la CPU con una che abbia velocità doppia rispetto alla prima.
   (b) Raddoppiare la quantità di memoria.
   (c) Sostituire la CPU con una che abbia velocità doppia rispetto alla prima e raddoppiare la quantità di memoria.

   Risposta:
   (a) (2 punti) Sostituire la CPU con una che abbia velocità doppia rispetto alla prima non cambia il grado di multiprogrammazione. Quindi, ci potrebbero essere lunghi periodi di tempo in cui la CPU non esegue alcun processo e, di conseguenza, il throughput sarebbe limitato dalla disponibilità della memoria.
   (b) (2 punti) Raddoppiare la quantità di memoria, aumenta il grado di multiprogrammazione (dato che possono essere eseguiti più processi), ma il throughput sarebbe limitato dalla velocità della CPU e quindi potrebbe non raddoppiare.
   (c) (2 punti) Sostituire la CPU con una che abbia velocità doppia rispetto alla prima e raddoppiare la quantità di memoria è la soluzione migliore a patto di caricare in quest'ultima un insieme adeguato di processi.

2. Quale delle seguenti transizioni di stato per un processo può causare la transizione di stato *waiting* → *ready* per uno o più degli altri processi?
   (a) Un processo avvia un'operazione di I/O e passa nello stato *waiting*.
   (b) Un processo termina.
   (c) Un processo effettua la richiesta per una risorsa e passa nello stato *waiting*.
   (d) Un processo invia un messaggio.
   (e) Un processo effettua una transizione di stato *waiting* → *waiting swapped*.

   Risposta:
   (a) (1 punto) No, non influenza il passaggio da *waiting* a *ready* di un altro processo.
   (b) (1 punto) Sì se, per esempio, si tratta della terminazione di un processo figlio di cui il padre era in attesa o se un altro processo era in attesa di una risorsa detenuta dal processo terminato.
   (c) (1 punto) No, non influenza il passaggio da *waiting* a *ready* di un altro processo.
   (d) (1 punto) Sì: un altro processo poteva essere in attesa di ricevere un messaggio.
   (e) (1 punto) No, lo swapping su disco di un processo in stato *waiting* non può influenzare il passaggio da *waiting* a *ready* di un altro processo.

3. Classificare ognuna delle seguenti affermazioni come vera o falsa.
   (a) Un'applicazione può contenere una race condition solo se il sistema su cui è in esecuzione l'applicazione possiede più di una CPU.
   (b) Un processo può subire una starvation all'entrata di una sezione critica se l'implementazione della sezione critica non soddisfa la condizione di attesa limitata.
   (c) Un processo può subire una starvation all'entrata di una sezione critica se l'implementazione della sezione critica non soddisfa la condizione di progresso.

   Risposta:
   (a) (2 punti) Falsa: un'applicazione può essere costituita da più thread; quindi si possono verificare delle race condition indipendentemente dal numero di CPU.
   (b) (2 punti) Vera: può non entrare in un tempo limitato perché scavalcato da altri processi.
   (c) (2 punti) Vera: qualche processo che sta eseguendo al di fuori della sua sezione critica potrebbe impedirgli l'accesso.

4. Il nuovo stato di allocazione di un sistema dopo aver accettato una richiesta di risorsa non è uno stato di allocazione sicuro secondo l'algoritmo del banchiere.
   (a) Ciò implica che si verificherà sicuramente un deadlock?
   (b) Il sistema riesce ad effettuare una transizione in uno stato di allocazione sicuro? Se sì, dare un esempio che mostri tale transizione.

   Risposta:
   (a) (2 punti) No: potrebbe non verificarsi un deadlock nel caso in cui qualche processo rilasciasse un numero sufficiente di risorse.
   (b) (2 punti) Consideriamo due processi P_1 e P_2 nella seguente situazione:

	Richieste massime			*Risorse allocate*	
	R_1	R_2		R_1	R_2
P_1	3	3	P_1	0	1
P_2	3	2	P_2	1	1

   Risorse massime: $(3, 3)$
   Risorse disponibili: $(2, 1)$
   Lo stato corrente è sicuro, perché P_2 può procedere. Invece se P_1 chiede ed ottiene una risorsa (ad esempio R_2), allora il nuovo stato non sarà più sicuro. Tuttavia, se in seguito, P_1 rilascerà la risorsa in oggetto, allora si ritornerà in uno stato sicuro (in cui P_2 potrà chiedere le risorse e terminare).

5. In un sistema che usa paginazione, l'acceso al TLB richiede 150ns, mentre l'accesso alla memoria richiede 400ns. Quando si verifica un page fault, si perdono 8ms per caricare la pagina che si sta cercando in memoria. Se il page fault rate è il 2% e il TLB hit il 70%, indicare l'EAT ai dati.

   Risposta: (4 punti) Convertendo tutti i tempi in ms, abbiamo:

   - tempo di accesso al TLB: $150\ ns = 150 \cdot 10^{-6}\ ms$;
   - tempo di accesso alla memoria: $400\ ns = 400 \cdot 10^{-6}\ ms$;
   - tempo di gestione del page fault: $8\ ms$;
   - page fault rate: $2\% = 0,02$;
   - TLB hit: $70\% = 0,7$.

   Quindi:

$$\begin{aligned} EAT &= \text{TLB hit} \cdot (150 \cdot 10^{-6}) + (1 - \text{page fault rate}) \cdot (150 \cdot 10^{-6} + 400 \cdot 10^{-6}) + \\ &\quad \text{page fault rate} \cdot (150 \cdot 10^{-6} + 8 + 400 \cdot 10^{-6}) \\ &= 0,7 \cdot (150 \cdot 10^{-6}) + 0,28 \cdot (150 \cdot 10^{-6} + 400 \cdot 10^{-6}) + 0,02 \cdot (150 \cdot 10^{-6} + 8 + 400 \cdot 10^{-6}) \\ &= 0,7 \cdot (1,5 \cdot 10^{-4}) + 0,28 \cdot (1,5 \cdot 10^{-4} + 4 \cdot 10^{-4}) + 0,02 \cdot (1,5 \cdot 10^{-4} + 8 + 4 \cdot 10^{-4}) \\ &= 1,05 \cdot 10^{-4} + 1,54 \cdot 10^{-4} + 0,160011 \\ &= 0,16027\ ms \end{aligned}$$

6. Si descrivano i passi principali eseguiti dal driver delle interruzioni.

   Risposta: (2 punti) I passi eseguiti dal driver delle interruzioni sono i seguenti:

   - salvare i registri della CPU,
   - impostare un contesto per la procedura di servizio (inizializzare TLB, MMU, stack ecc.),
   - inviare un segnale di *acknowledge* al controllore degli interrupt (per avere interrupt annidati),
   - copiare la copia dei registri nel PCB,
   - eseguire la procedura di servizio che accede al dispositivo,
   - eventualmente, cambiare lo stato a un processo in attesa (e chiamare lo scheduler di breve termine),
   - organizzare un contesto (TLB, MMU ecc.) per il processo successivo,
   - caricare i registri del nuovo processo dal suo PCB,

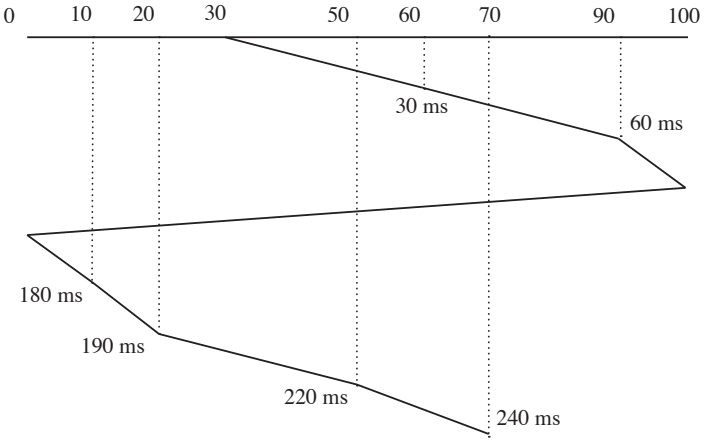
- continuare l'esecuzione del processo selezionato.

7. Si consideri un disco con un intervallo di tracce da 0 a 100, gestito con politica C-SCAN (SCAN circolare) con direzione di servizio verso tracce con numero crescente. Inizialmente la testina è posizionata sul cilindro 30; lo spostamento ad una traccia adiacente richiede 1 ms. Al driver di tale disco arrivano richieste per i cilindri 90, 20, 60, 70, 50, 10, rispettivamente agli istanti 0 ms, 10 ms, 20 ms, 80 ms, 90 ms, 150 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 sei richieste in oggetto?

Risposta:

1. (3 punti) Le richieste vengono servite nell'ordine 60, 90, 10, 20, 50, 70:



2. (1 punto) Il tempo di attesa medio per le sei richieste in oggetto è $\frac{(30-20)+(60-0)+(180-150)+(190-10)+(220-90)+(240-80)}{6} = \frac{10+60+30+180+130+160}{6} = \frac{570}{6} = 95\ ms.$