Sistemi Operativi
4 settembre 2012
Compito

Si risponda ai seguenti quesiti, giustificando le risposte.

1. (a) Un sistema di prenotazione per una compagnia area che utilizza un database centralizzato risponde alle richieste degli utenti in maniera concorrente. In questa applicazione è preferibile usare thread o processi?

   (b) Si discuta lo speed-up dell'elaborazione della seguente applicazione in un sistema con una o più CPU: "Due matrici contengono m righe e n colonne ognuna, dove m, n sono numeri molto grandi. L'applicazione ottiene il risultato della somma delle due matrici creando m thread, ognuno dei quali effettua la somma di una riga delle matrici."

   Risposta:

   (a) (2 punti) Implementando il sistema di prenotazione con un singolo processo (a cui venga associato il database) costituito da più thread, è conveniente far gestire le singole prenotazioni a questi ultimi in modo da evitare di dover coinvolgere il kernel e da diminuire il sovraccarico dovuto all'alternanza delle richieste di accesso alla base di dati. Invece, utilizzando processi distinti per l'accesso al database e la gestione delle prenotazioni, si registrerebbe un maggior sovraccarico.

   (b) (2 punti) Se n è un numero grande, come effetto della soluzione descritta si ottiene un elevato parallelismo intrinseco tra i vari thread. Nel caso in cui però il sistema disponga di una sola CPU, non si ottiene nessuno speed-up dell'elaborazione dato che i thread svolgono esclusivamente attività di CPU (cioè sono CPU-bound). Avendo a disposizione più CPU invece si ottiene uno speed-up significativo dell'elaborazione.

2. Si discuta ciascuna delle seguenti affermazioni.

   (a) Un processo può subire una starvation all'entrata di una sezione critica se l'implementazione della sezione critica non soddisfa la condizione di progresso.

   (b) Attese attive indefinite si possono verificare in un SO che utilizza uno scheduling basato su priorità, ma non possono verificarsi in un SO che utilizza uno scheduling Round Robin.

   (c) In un sistema lettori-scrittori che favorisce gli scrittori, alcuni processi lettori che vogliono leggere i dati condivisi possono bloccarsi anche quando altri processi lettori stanno leggendo i dati condivisi.

   (d) Un deadlock non può verificarsi nel problema dei filosofi a cena se un filosofo può mangiare solo con una forchetta.

   Risposta:

   (a) (2 punti) Vero: per la definizione stessa di condizione di progresso (stabilisce che nessun processo in esecuzione fuori della sezione critica possa impedire l'accesso di altri processi alla sezione critica).

   (b) (2 punti) Falso: si possono verificare attese *attive* indefinite anche in un SO che utilizza uno scheduling Round Robin. Infatti, essendo l'attesa attiva, è il processo stesso che si pone in un loop (potenzialmente indefinito) in cui ad ogni iterazione controlla il verificarsi di una certa condizione.

   (c) (2 punti) Vero: infatti è sufficiente l'arrivo di un singolo processo scrittore per bloccare i processi lettori (anche se vi sono già dei processi lettori che stanno accedendo ai dati condivisi).

   (d) (2 punti) Vero: sicuramente almeno uno dei filosofi riuscirà in ogni instante ad acquisire la forchetta per mangiare.

3. Si descriva il funzionamento dell'istruzione TSL (Test and Set Lock) e si mostri come implementare una sezione critica utilizzando TSL.

   Risposta: (4 punti) L'istruzione TSL (Test-and-Set-Lock) 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).

L'istruzione assembler *TSL RX,LOCK* può essere utilizzata come segue per implementare le procedure di entrata/uscita da una regione critica:

```
enter_region:
    TSL REGISTER,LOCK
    CMP REGISTER,#0
    JNE enter_region
    RET

leave_region:
    MOVE LOCK,#0
    RET
```

4. In un SO, uguali richieste, ciascuna delle quali richiede 0.05 secondi di CPU, arrivano con la frequenza di 10 richieste al secondo. Il kernel usa una coda ready a dimensione fissa. Una nuova richiesta viene inserita nella coda ready se la coda non è già piena, altrimenti la richiesta viene scartata. Quale dovrebbe essere la dimensione della coda ready se fosse scartato meno dell'1% di richieste?

   Risposta: (4 punti) Nelle ipotesi del problema arrivano 10 richieste in media ogni secondo e la velocità di servizio è tale che le 10 richieste vengono "smaltite" in mezzo secondo. Quindi il caso critico si ha allo scadere di un secondo, se fino a quel punto non è ancora arrivata nessuna richiesta. Infatti potrebbe capitare che arrivino tutte e 10 all'ultimo istante e si inizi a servire la prima di queste. Intanto, passando al secondo successivo, potrebbero arrivare allo stesso istante anche tutte le 10 richieste di pertinenza del nuovo secondo. In tal caso ci sarebbe una richiesta in corso di servizio e 19 in attesa. 19 risulta essere quindi la lunghezza minima della coda ready. Infatti con una posizione di meno, ovvero, con una coda ready con 18 posizioni, si andrebbe a perdere 1 richiesta su 19, ovvero, il $\frac{1}{19} = 0,053\%$ di richieste che è ben superiore alla soglia consentita (meno dello 0,01%).

5. Si discutano i vantaggi dell'algoritmo clock a due lancette per la gestione della memoria virtuale, rispetto all'algoritmo di clock a una lancetta.

   Risposta: (3 punti) Nella versione dell'algoritmo di clock con due lancette si può ottenere una maggiore efficienza. Infatti mentre la prima lancetta provvede ad azzerare il reference bit della pagina puntata, la seconda sceglie la pagina vittima. In particolare, se l'angolo fra le due lancette è molto piccolo (le lancette sono "vicine"), solo le pagine realmente usate rimarranno in memoria. Invece se l'angolo è di 359 gradi (quasi un angolo giro), l'algoritmo degenera in un CLOCK normale (ad una lancetta) dato che la seconda passerà a distanza di un giro dalla prima.

6. Si spieghi cos'è un'interruzione precisa (elencando le quattro condizioni).

   Risposta: (4 punti) Un'interruzione è precisa se:

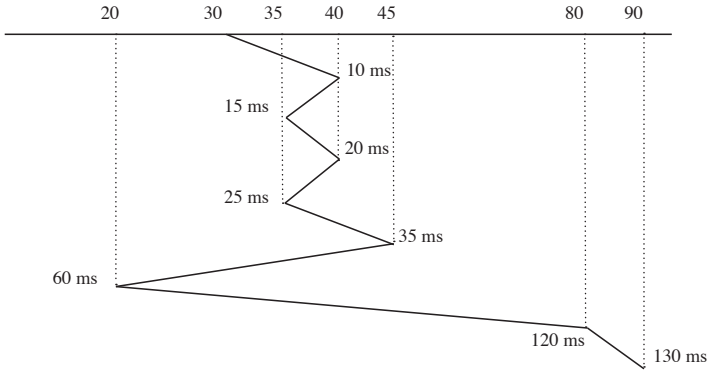
   - il program counter (PC) è salvato in un posto noto,
   - tutte le istruzioni precedenti a quella puntata dal PC sono state eseguite completamente,
   - nessuna istruzione successiva a quella puntata dal PC è stata eseguita,
   - lo stato dell'esecuzione dell'istruzione puntata dal PC è noto.

7. Si consideri un disco gestito con politica SSTF. 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, 40, 45, 80, rispettivamente agli istanti 0 ms, 10 ms, 15 ms, 25 ms, 40 ms. Si trascuri il tempo di latenza.
   (a) In quale ordine vengono servite le richieste?
   (b) 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:

Sistemi Operativi
4 settembre 2012
Compito

1. (3 punti) Le richieste vengono servite nell'ordine 40, 45, 20, 80, 90:



2. (1 punto) Il tempo di attesa medio per le cinque richieste in oggetto è $\frac{(20-15)+(35-25)+(60-10)+(120-40)+(130-0)}{5} = \frac{5+10+50+80+130}{5} = \frac{275}{5} = 55\ ms.$