

Elementi di Architettura e Sistemi Operativi

Bioinformatica - Tiziano Villa

3 Settembre 2010

Nome e Cognome:

Matricola:

Posta elettronica:

problema	punti massimi	i tuoi punti
problema 1	9	
problema 2	6	
problema 3	7	
problema 4	8	
totale	30	

1. Rispondere in modo preciso ma conciso alle seguenti domande.

(a) Si descrivano le quattro condizioni necessarie perche' si abbia uno stallo.

Traccia di soluzione.

- i. Mutua esclusione (almeno una risorsa non condivisibile).
- ii. Possesso e attesa (un processo in possesso di almeno una risorsa e' in attesa di almeno un'altra risorsa detenuta da altri).
- iii. Impossibilita' di prelazione (una risorsa puo' essere rilasciata dal processo che la detiene solo volontariamente, non si puo' sottrargliela).
- iv. Attesa circolare (c'e' una catena chiusa di processi che attendono risorse detenute da altri processi).

(b) Si definisca il grafo di assegnazione delle risorse e se ne motivi l'uso.

Traccia di soluzione.

Si veda la Sez. 7.2.2 del Silberschatz.

Quale relazione esiste tra l'esistenza di cicli nel grafo delle risorse e la possibilità di uno stallo ?

Traccia di soluzione.

L'assenza di cicli nel grafo di assegnazione delle risorse implica l'assenza di situazioni di stallo. Ma la presenza di cicli non implica necessariamente uno stallo.

In particolare, se ciascun tipo di risorsa ha esattamente un'istanza, allora l'esistenza di un ciclo implica la presenza di uno stallo; se il ciclo riguarda solo un insieme di tipi di risorsa, ciascuno dei quali ha solo un'istanza, si è verificato uno stallo. Ogni processo che si trovi nel ciclo è in stallo. In questo caso l'esistenza di un ciclo nel grafo è condizione necessaria e sufficiente per l'esistenza di uno stallo.

Se ogni tipo di risorsa ha più istanze, un ciclo non implica necessariamente uno stallo. In questo caso l'esistenza di un ciclo nel grafo è una condizione necessaria ma non sufficiente per l'esistenza di uno stallo.

- (c) Per ciascuna delle quattro condizioni necessarie per uno stallo si discutano i modi di prevenirla.

Traccia di soluzione.

- i. Le risorse condivisibili non richiedono la mutua esclusione, ma non si può eliminare la mutua esclusione sulle risorse non condivisibili.
 - ii. Si può usare un protocollo
 - A. in cui ogni processo prima d'iniziare la propria esecuzione ottiene tutte le risorse che gli servano; oppure
 - B. in cui un processo richiede risorse solo se non ne possiede (se ne possiede, prima di richiederne altre deve rilasciare quelle che possiede).
- Si veda la Sez. 7.4.2.
- iii. S'introduce una qualche forma di prelazione nei casi in cui lo stato si può salvare e recuperare facilmente. Si veda la Sez. 7.4.3.
 - iv. Per impedire l'attesa circolare si può imporre un ordine totale all'insieme di tutti i tipi di risorse e un ordine crescente di numerazione alle risorse richieste da ciascun processo. Un protocollo è il seguente: un processo può richiedere inizialmente qualsiasi quantità di una certa risorsa, ad es. della risorsa R_i , dopo di che il processo può richiedere quantità della risorsa R_j solo se $R_j > R_i$ nell'ordine totale dato.

- (d) Si definisca quando uno stato si dice sicuro. Qual e' la relazione tra uno stato sicuro, uno insicuro e uno di stallo ?

Traccia di soluzione.

Uno stato si dice sicuro se c'e' un ordine in base al quale il sistema e' in grado di assegnare le risorse richieste da ciascun processo (eventualmente assegnandogli le risorse usate dai processi che vengono prima nell'ordine e che quindi si puo' supporre che abbiano finito).

Se non esiste tale ordine lo stato del sistema si dice insicuro. Uno stato di stallo e' uno stato insicuro, ma non tutti gli stati insicuri sono di stallo. Uno stato insicuro potrebbe condurre a uno stallo, perche' il sistema operativo non puo' impedire ai processi di richiedere risorse in modo da causare uno stallo.

Si veda la Fig. 7.5 nella Sez. 7.5.1.

- (e) Se un sistema e' in uno stato insicuro, tale sistema deve finire inevitabilmente in uno stato di stallo ? In altri termini, e' possibile che i processi del sistema completino la loro esecuzione senza che il sistema entri in uno stato di stallo ?

Traccia di soluzione.

E' possibile che in un sistema in uno stato insicuro tutti i processi completino senza uno stallo; si veda la domanda successiva per la discussione di un esempio che presenta tale situazione. Quindi uno stato insicuro non conduce necessariamente a uno stallo, ma d'altro canto non possiamo garantire che non si arrivi a uno stallo.

(f) Si consideri un sistema con 12 risorse e 3 processi P_0, P_1, P_2 con la seguente distribuzione di risorse:

P_0 ha bisogno al massimo di 10 risorse ed ora detiene 5 risorse;

P_1 ha bisogno al massimo di 4 risorse ed ora detiene 2 risorse;

P_2 ha bisogno al massimo di 9 risorse ed ora detiene 3 risorse.

In che stato si trova il sistema ? E' inevitabile che il sistema finisca in stallo ? E' possibile che il sistema non finisca in stallo ?

Traccia di soluzione.

In questo momento nel sistema ci sono 2 risorse libere. Il sistema e' in un stato insicuro perche' pur se P_1 potrebbe acquisire le due risorse libere e poi terminare, liberando 4 risorse, non potremmo ancora garantire che i processi P_0 e P_2 terminino. Tuttavia sarebbe possibile che un processo rilasciasse risorse prima di richiederne altre. Ad esempio, il processo P_2 potrebbe rilasciare una risorsa, portando a 5 il numero delle risorse libere. Questo permetterebbe al processo P_0 di completare, portando a 10 il numero delle risorse libere, permettendo infine al processo P_2 di completare.

2. Si consideri il seguente codice per risolvere il problema dei lettori e scrittori mediante semafori (*OKToRead*, *OKToWrite*, *Mutex*) e variabili di stato (*AR* numero lettori attivi, *WR* numero lettori in attesa, *AW* numero scrittori attivi, *WW* numero scrittori in attesa).

```
Semaphore OKToRead = 0; OKToWrite = 0; Mutex = 1;  
AR = WR = AW = WW = 0;
```

```
Lettore {  
    P (Mutex) ;  
    if (AW == 0) {  
        V (OKToRead) ;  
        AR = AR + 1 ;  
    } else WR = WR + 1 ;  
    V (Mutex) ;  
    P (OKToRead) ;  
  
    leggere i dati ;  
  
    P (Mutex) ;  
    AR = AR - 1 ;  
    if (AR == 0 && WW > 0) {  
        V (OKToWrite) ;  
        AW = AW + 1 ;  
        WW = WW - 1 ;  
    }  
    V (Mutex) ;  
}
```



```

Scrittore {
    P(Mutex);
    if ((AW + AR + WW) == 0) {
        V(OKToWrite);
        AW = AW + 1;
    } else WW = WW + 1;
    V(Mutex);
    P(OKToWrite);

    scrivere i dati;

    P(Mutex);
    AW = AW - 1;
    while (WR > 0) {
        V(OKToRead);
        AR = AR + 1;
        WR = WR - 1;
    }
    if (AR == 0 && WW > 0) {
        V(OKToWrite);
        AW = AW + 1;
        WW = WW - 1;
    }
    V(Mutex);
}

```

(a) Si analizzi il codice precedente spiegandone il funzionamento.

Traccia di soluzione.

- Solo un processo alla volta manipola le variabili di stato con il semaforo *Mutex*.
- Più lettori possono accedere al codice contemporaneamente, ma gli scrittori devono avere accesso esclusivo.
- I lettori possono procedere solo se non ci sono scrittori attivi. Si noti che nel codice del lettore c'è il controllo

$$if (AW == 0) \{$$

per cui un lettore non aspetta quando non ci sono scrittori attivi e ci sono scrittori in attesa.

- Gli scrittori possono procedere solo se non ci sono lettori attivi o scrittori attivi o in attesa.

(b) In caso di conflitto tra lettori e scrittori chi ha la priorit  ? Perche' ?

Traccia di soluzione.

I lettori hanno la priorit . Agli scrittori puo' essere negata la risorsa indefinitamente.

La priorit  dei lettori e' stabilita dalle seguenti condizioni:

i. Nel codice del lettore c'e' il controllo

$$if (AW == 0) \{$$

che abilita la lettura, per cui un lettore non aspetta quando non ci sono scrittori attivi e ci sono scrittori in attesa.

ii. Nel codice dello scrittore c'e' il controllo

$$while (WR > 0) \{$$

che abilita la lettura per prima se ci sono lettori in attesa, e solo dopo a ciclo *while* concluso c'e' il controllo

$$if (AR == 0 \&\& WW > 0) \{$$

che abilita la scrittura se ci sono scrittori in attesa.

Questa e' la differenza tra la presente versione del problema dei lettori e scrittori ed altre versioni proposte altrove. Altre piccole differenze nel codice sono una conseguenza di questa diversita' di specifica.

(c) Puo' essere $OKToRead > 1$? Puo' essere $OKToWrite > 1$? Perche' ?

Traccia di soluzione.

Si alla prima domanda. No alla seconda domanda.

3. Siano dati 5 processi con durata dell'esecuzione e tempo d'arrivo espressi dalla seguente tabella:

Processo	Durata	Arrivo
P1	2	0
P2	6	1
P3	1	4
P4	4	7
P5	3	8

- (a) Si disegni lo schema GANTT (come nel libro di testo) che illustri l'esecuzione di questi processi con i seguenti algoritmi di schedulazione: FCFS (First Come First Serve), SRTF (Shortest-Remaining-Time-First, cioè Shortest-Job-First con prelazione), RR (Round-Robin) con quanto di tempo = 1.

Traccia di soluzione.

FCFS:

```
P1 P1 P2 P2 P2 P2 P2 P2 P3 P4 P4 P4 P4 P5 P5 P5
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

SRTF:

```
P1 P1 P2 P2 P3 P2 P2 P2 P2 P5 P5 P5 P4 P4 P4 P4
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

RR:

```
P1 P2 P1 P2 P3 P2 P2 P4 P5 P2 P4 P5 P2 P4 P5 P4
0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

- (b) Per i tre algoritmi precedenti si calcoli il tempo di completamento di ciascun processo.

Traccia di soluzione.

Il tempo di completamento e' la differenza tra il tempo di terminazione e il tempo d'arrivo, cioe' il tempo trascorso da quando il processo arriva nella coda a quando termina. Nella letteratura in inglese e' designato anche come TRT (turnaround time), cioe' tempo trascorso.

Per FCFS i tempi di completamento sono:

P1 2,

P2 7,

P3 5,

P4 6,

P5 8

Per SRTF i tempi di completamento sono:

P1 2,

P2 8,

P3 1,

P4 9,

P5 4

Per RR i tempi di completamento sono:

P1 3,

P2 12,

P3 1,

P4 9,

P5 7

- (c) Per i tre algoritmi precedenti si calcoli il tempo di attesa di ciascun processo.

Traccia di soluzione.

Il tempo di attesa e' la differenza tra il tempo di completamento e la durata, cioe' e' il tempo trascorso in attesa nella coda senza eseguire.

Per FCFS i tempi di attesa sono:

P1 0,

P2 1,

P3 4,

P4 2,

P5 5

Per SRTF i tempi di attesa sono:

P1 0,

P2 2,

P3 0,

P4 5,

P5 1

Per RR i tempi di attesa sono:

P1 1,

P2 6,

P3 0,

P4 5,

P5 4

- (d) Per i tre algoritmi precedenti si calcoli il tempo di attesa medio di tutti i processi.

Traccia di soluzione.

Tempo di attesa medio per FCFS: $(0+1+4+2+5)/5 = 2,4$.

Tempo di attesa medio per SRTF: $(0+2+0+5+1)/5 = 1,6$.

Tempo di attesa medio per RR: $(1+6+0+5+4)/5 = 3,2$.

Si noti che SRTF ottiene il tempo di attesa medio minimo come dalla teoria.

4. L'interazione con le unita' d'ingresso/uscita puo' avvenire tramite interrogazione ciclica ("polling") o interruzioni ("interrupt"). Per minimizzare il lavoro del processore si e' inventata una modalita' di accesso diretto alla memoria (DMA), dove il controllore del dispositivo trasferisce i dati direttamente da o verso la memoria, senza coinvolgere il processore. Il meccanismo delle interruzioni e' usato dal dispositivo per comunicare con il processore al termine dell'operazione o per segnalare errori.

Si vuole studiare la convenienza di usare lo schema DMA per gestire l'interazione con il disco rigido.

Si supponga che l'operazione d'inizializzazione del trasferimento in modo DMA richieda 1000 cicli d'orologio per il processore, e che la gestione della interruzione al termine della fase di DMA richieda 500 cicli del processore. Inoltre si supponga che il processore lavori con una frequenza di 500 MHz.

Nell'ipotesi che il disco trasferisca dati alla velocita' di 4 MB/s, che la dimensione media di un trasferimento dal disco sia pari a 8 KB, e che il disco rigido trasferisca dati per il 100% del tempo, si determini la frazione di tempo del processore consumata dalla modalita' d'interazione DMA.

Che cosa si puo' concludere sulla convenienza del meccanismo DMA per interagire con il disco rigido ?

Traccia di soluzione.

Il numero di trasferimenti al secondo di blocchi di 8 KB e' uguale a 500 (4 MB/secondo / 8 KB/trasferimento = 500 trasferimenti/secondo).

I cicli di orologio richiesti al secondo si ottengono dal prodotto dei trasferimenti al secondo per i cicli utilizzati per ogni trasferimento

$500 \text{ trasferimenti/secondo} \times (1000 + 500) \text{ cicli/trasferimento} = 750 \times 10^3 \text{ cicli al secondo.}$

Nell'ipotesi che il disco trasferisca in continuazione, la frazione di cicli del processore utilizzati dal modo DMA e' data da

$$750 \times 10^3 / 500 \times 10^6 = 1.5 \times 10^{-3} = 0,15\%.$$

Si conclude che sarebbe conveniente usare la modalita' DMA per interagire con il disco. Si noti che ci si e' messi nel caso peggiore in cui il disco trasferisca dati per tutto il tempo, per cui la percentuale precedente in pratica sara' ancora piu' bassa. Si e' trascurato il problema della contesa sulla memoria tra processore e disco (in parte risolvibile dalla presenza di memorie "cache").

5. (Problema per chi non abbia sostenuto l'esame di Elementi di Architettura).

Si progetti un circuito sequenziale che funziona come contatore a 3 cifre binarie secondo la sequenza seguente: $000 \rightarrow 010 \rightarrow 011 \rightarrow 101 \rightarrow 110 \rightarrow 000$.

(a) Si disegni il grafo delle transizioni di una macchina a stati finiti che corrisponde alla specifica. S'indichi lo stato iniziale.

Si minimizzi il numero degli stati della macchina proposta.

- (b) Si scriva la tavola delle transizioni con gli stati futuri e le uscite e la si codifichi.

- (c) Supponendo di usare bistabili di tipo D, si derivino le equazioni minimizzate di eccitazione degl'ingressi dei bistabili e le equazioni minimizzate delle uscite.

- (d) Si realizzi il circuito sequenziale corrispondente con bistabili di tipo D campionati sul fronte di salita, invertitori e porte NAND (a 2, 3, o 4 ingressi). Si etichettino con chiarezza i segnali.