

Elementi di Architettura e Sistemi Operativi

Bioinformatica - Tiziano Villa

24 Settembre 2014

Nome e Cognome:

Matricola:

Posta elettronica:

problema	punti massimi	i tuoi punti
problema 1	10	
problema 2	6	
problema 3	4	
problema 4	10	
totale	30	

1. (a) Si descriva che cos'è un monitor.

(b) Si descrivano le condizioni necessarie per lo stallo di processi e come prevenirle.

- (c) Un sistema informatico ha 5 unita' nastro a disposizione per gli utenti. Un programma di analisi dati ricavati da esperimenti biologici richiede l'uso di due unita' nastro: una per i dati grezzi e una per i dati elaborati. L'amministratore del sistema ha scritto il seguente monitor per controllare l'accesso alle unita' nastro.

```
#define N 5

monitor GestoreNastri {

    int n_nastri, liberi[N];
    condition serve_nastro;

    entry AllocaNastro()
    {
        int i;

        while (n_nastri <= 0)
            wait(serve_nastro);
        for (i=0; liberi[i] == 0; i++)
            ;
        liberi[i] = 0;
        n_nastri--;
        return(i);
    }

    entry RilasciaNastro(i)
    {
        int i;
        {
            n_nastri++;
            liberi[i] = 1;
            wakeup(serve_nastro);
        }
    }

    /* Inizializzazione */
    int i;
```

```

    n_nastri = N;
    for (i=0; i < N; i++)
        liberi[i] = 1;

}

```

Nel codice precedente la parola chiave *entry* indica procedure che acquisiscono il lucchetto del monitor "*monitor lock*" quando all'inizio e lo rilasciano all'uscita.

La porzione rilevante del programma di analisi dei dati e'

```

NastroDatiGrezzi = AllocaNastro();
NastroUscita = AllocaNastro();

/* Codice che utilizza i due nastri */

RilasciaNastro(NastroDatiGrezzi);
RilasciaNastro(NastroUscita);

```

Si spieghi il funzionamento del codice del monitor.

.

- (d) Succede spesso che piu' utenti cerchino simultaneamente di lanciare il programma di analisi dati e quindi acquisire i nastri necessari. A volte l'amministratore del sistema rileva che una collezione di esecuzioni del programma di analisi dati si blocca, cioe' entra in stallo e nessuna esecuzione riesce a completare. Che cosa provoca tale blocco ?

Traccia di soluzione.

Se cinque utenti ottengono ciascuno un'unita' nastro, nessun processo puo' ottenere la seconda unita' e percio' si entra in una condizione di stallo (ogni processo ha bisogno di due unita' della risorsa nastro, ma ne ha una sola che non rilascia in attesa della seconda).

- (e) Si proponga un modo di prevenire la situazione di stallo (si supponga che non si possa restringere il numero degli utenti che vogliono eseguire simultaneamente il programma di analisi dei dati). Si mostrino i cambiamenti nel codice precedente per prevenire la situazione di stallo secondo la vostra proposta.

Traccia di soluzione.

Invece di assegnare un'unita' nastro per volta, se ne assegnino due per volta (visto che ad ogni utente ne servono esattamente due); se non ci sono due unita' nastro disponibili l'utente sara' messo in coda ad aspettarne la disponibilita'.

```
#define N 5

monitor GestoreNastri {
%
    int n_nastri, liberi[N];
    condition serve_nastro;

    entry AllocaNastri(nastrol,nastro2)
        int *nastrol,*nastro2;
    {
        int i, nastri_assegnati;

        while (n_nastri <= 1)
            wait(serve_nastro);
        nastri_assegnati = 0;
        for (i=0; nastri_assegnati < 2; i++)
        {
            if (liberi[i])
            {
                liberi[i] = 0;
                nastri_assegnati++;
                n_nastri--;
                if (nastri_assegnati == 1)
                    *nastrol = i;
                else *nastro2 = i;
            }
        }
    }
}
```



```

    }
}

entry RilasciaNastri(nastro1,nastro2)
    int nastro1,nastro2;
{
    n_nastri +=2;
    liberi[nastro1] = 1;
    liberi[nastro2] = 1;
    wakeup(serve_nastro);
}

/* Inizializzazione */
int i;

n_nastri = N;
for (i=0; i < N; i++)
    liberi[i] = 1;

}

AllocaNastri(&NastroDatiGrezzi,&NastroUscita);

/* Codice che utilizza i due nastri */

RilasciaNastri(NastroDatiGrezzi,NastroUscita);

```

2. (a) Con riferimento alla memoria virtuale, si descriva brevemente l'algoritmo di sostituzione delle pagine usate meno recentemente LRU (Least Recently Used).

(b) Si consideri la seguente successione di riferimenti a pagine di memoria

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

Quante eccezioni per pagina mancante si verificherebbero con l'algoritmo di sostituzione LRU, usando rispettivamente una memoria fisica con 1, 2, 3, 4, 5, 6, 7 pagine ? Si supponga che tutte le pagine fisiche siano inizialmente vuote, per cui la prima volta di ogni pagina costituirà un'eccezione per pagina mancante. Si mostrino i calcoli.

Traccia di soluzione.

Pagine	Eccezioni
1	20
2	18
3	15
4	10
5	8
6	7
7	7

Esempio con 3 pagine fisiche.

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	4	4	4	5	5	5	1	1	1	7	7	7	2	2	2	2	2
	2	2	2	2	2	2	6	6	6	6	3	3	3	3	3	3	3	3	3
		3	3	3	1	1	1	2	2	2	2	2	6	6	6	1	1	1	6
X	X	X	X		X	X	X	X	X		X	X	X		X	X			X

3. Si consideri un sistema di paginazione con la tabella delle pagine conservata in memoria.

- (a) Se un riferimento alla memoria necessita di 200 nanosecondi per essere servito, di quanto necessita un riferimento alla memoria paginata ?
- (b) Se si aggiunge una TLB (cache degl'indirizzi), e il 75% di tutti i riferimenti si trovano nella TLB, quale sara' il tempo medio effettivo di un riferimento a memoria (si trascuri il tempo di accesso alla TLB) ?

Si argomentino le risposte.

Traccia di soluzione.

- (a) $200 \text{ ns (accesso alla tavola degl'indirizzi)} + 200 \text{ ns (accesso a memoria)} = 400 \text{ ns.}$
- (b) $0,75 \times 200 \text{ ns (accesso diretto a memoria)} + 0,25 \times 400 \text{ ns (accesso a tavola e accesso a memoria)} = 150 \text{ ns} + 100 \text{ ns} = 250 \text{ ns.}$

4. Si progetti un circuito sequenziale che realizza la seguente specifica:

- Ci sono due segnali binari d'ingresso X_1X_2 e un segnale binario d'uscita Z .
- Se $X_2 = 0$, l'uscita Z nel generico istante t vale $Z(t) = X_1(t - 1)$, se $X_2 = 1$, l'uscita Z nel generico istante t vale $Z(t) = X_1(t - 2)$.

Sia 00 00 la sequenza di azzeramento, cioè alla partenza del circuito si danno gl'ingressi 00 00.

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

Traccia di soluzione.

Gli stati sono codificati naturalmente come $X_1(t - 1)X_1(t - 2)$, cioè con i due ultimi valori della variabile X_1 .

Dato lo stato presente $X_1(t - 1)X_1(t - 2)$ e l'ingresso $X_1(t)$, lo stato futuro è $X_1(t)X_1(t - 1)$.

Dato lo stato presente $X_1(t - 1)X_1(t - 2)$ e l'ingresso $X_2(t) = 0$, l'uscita è $X_1(t - 1)$; Dato lo stato presente $X_1(t - 1)X_1(t - 2)$ e l'ingresso $X_2(t) = 1$, l'uscita è $X_1(t - 2)$.

Segue la tavola delle transizioni della macchina:

00	s00	s00	0
01	s00	s00	0
10	s00	s10	0
11	s00	s10	0

00	s01	s00	0
01	s01	s00	1
10	s01	s10	0
11	s01	s10	1

00	s10	s01	1
01	s10	s01	0
10	s10	s11	1
11	s10	s11	0

00	s11	s01	1
----	-----	-----	---

01	s11	s01	1
10	s11	s11	1
11	s11	s11	1

- (b) Si minimizzi il numero degli stati della macchina proposta, applicando l'algoritmo di minimizzazione degli stati.

Traccia di soluzione.

Applicando l'algoritmo di minimizzazione degli stati, si deduce che la macchina precedente è già minimizzata.

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

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

Traccia di soluzione.

Siano D e Q rispettivamente l'ingresso (stato futuro) e l'uscita (stato presente) di un bistabile. Si ha: $D_1 = X_1$, $D_2 = Q_1$, $Z = \overline{X}_2 Q_1 + X_2 Q_2$.

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