

Elementi di Architettura e Sistemi Operativi

Bioinformatica - Tiziano Villa

25 Febbraio 2013

Nome e Cognome:

Matricola:

Posta elettronica:

problema	punti massimi	i tuoi punti
problema 1	6	
problema 2	7	
problema 3	7	
problema 4	10	
totale	30	

1. Si consideri il seguente programma che crea processi invocando la chiamata di sistema `fork`.

```
#include <stdio.h>
#include <unistd.h>

int main()
{
    /* crea mediante fork un processo figlio */
    fork();

    /* crea mediante fork un altro processo figlio */
    fork();

    /* crea mediante fork un altro processo figlio */
    fork();

    return 0;
}
```

(a) Si spieghi la semantica della chiamata di sistema `fork`.

Si analizzi il codice precedente spiegandone il funzionamento.

Traccia di soluzione.

La chiamata di sistema `fork()` crea un nuovo processo figlio che avrà una copia dello spazio degli indirizzi del processo genitore. Entrambi i processi genitore e figlio continuano l'esecuzione dall'istruzione successiva alla chiamata di sistema `fork()`, con una differenza: la chiamata di sistema `fork()` restituisce il valore 0 nel processo figlio, ma restituisce l'identificatore del processo figlio (il PID diverso da 0) nel processo genitore.

(b) Considerando anche il processo padre iniziale, quanti processi sono creati dal programma precedente ?

Si argomenti la risposta e si mostri con uno schema grafico l'albero della creazione di nuovi processi.

Traccia di soluzione.

I processi sono 8 includendo quello iniziale. Il processo1 padre crea successivamente i tre processi: processo2, processo3, processo4. Il processo2 crea successivamente i due processi: processo5 e processo6. Il processo5 crea successivamente il processo7. Il processo3 crea successivamente il processo8.

2. Si consideri il seguente scenario di lavanderia automatica. Un cliente in arrivo depone delle monete in una di due postazioni di servizio e richiede una macchina per lavare. Le due postazioni sono collegate a un calcolatore centrale che assegna automaticamente una macchina disponibile e produce un cartellino con il numero della macchina assegnata. Il cliente utilizza la macchina indicata, che alla fine del ciclo di lavaggio informa il calcolatore centrale di essere di nuovo disponibile. Il calcolatore centrale gestisce un vettore `disponibile[NMACCHINE]`, dove la posizione i e' $\neq 0$ se la macchina i -esima e' disponibile. La costante `NMACCHINE` indica il numero di macchine totali nella lavanderia. C'e' un semaforo `nlibere` che indica quante macchine sono disponibili. Il codice per assegnare e rilasciare le macchine segue. Il vettore `disponibile` e' inizializzato con tutti uno, la variabile `nlibere` e' inizializzata a `NMACCHINE`.

```
int allocazione() {
    int i;
    P(nlibere);
    for (i = 0; i < NMACCHINE; i++) {
        if (disponibile[i] != 0) {
            disponibile[i] = 0;
            return i;
        }
    }
}

rilascio(int macchina) {
    disponibile[macchina] = 1;
    V(nlibere);
}
```

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

- (b) E' riportato che occasionalmente, quando due persone richiedono una macchina nello stesso momento, si ritrovano assegnata la medesima lavatrice. Vi si chiede di analizzare il codice e spiegare quando cio' possa succedere.

Traccia di soluzione.

Il problema insorge perche' la manipolazione del vettore `disponibile` non e' protetta da un meccanismo di mutua esclusione. Puo' succedere che il processo di un cliente *A* esegua fino a prima dell'assegnamento `disponibile[i] = 0` e poi sia interrotto; e che poi sia attivato il processo di un cliente *B* che riesca ad eseguire tutta la procedura di allocazione ricevendo la macchina *i*; infine il processo del cliente *A* riprende e completa l'esecuzione ricevendo anch'esso la macchina *i*.

In altri termini, il semaforo `nlibere` garantisce che ci siano abbastanza macchine disponibili per ogni processo che raggiunge l'assegnamento `disponibile[i] = 0`, ma non garantisce che un processo abbia accesso esclusivo alle variabili di stato.

- (c) Si modifichi il codice precedente aggiungendo due istruzioni per eliminare l'inconveniente riportato.

Traccia di soluzione.

La soluzione e' di aggiungere un semaforo di mutua esclusione `Mutex` inizializzato a 1.

```
int allocazione() {
    int i;
    P(nlibere);
    P(Mutex);
    for (i = 0; i < NMACCHINE; i++) {
        if (disponibile[i] != 0) {
            disponibile[i] = 0;
            V(Mutex);
            return i;
        }
    }
}

rilascio(int macchina) {
```

```
P (Mutex) ;  
disponibile[macchina] = 1;  
V (Mutex) ;  
V (nlibere) ;  
}
```

Si noti che non e' necessario introdurre la mutua esclusione nella procedura `rilascio`, ma e' una buona pratica proteggere con la mutua esclusione tutte le manipolazioni delle variabili di stato.

3. Un sistema operativo offre la memoria virtuale paginata, utilizzando un processore con una durata di ciclo di 1 microsecondo. L'accesso a una pagina in memoria diversa da quella corrente richiede un altro microsecondo. Le pagine hanno 1000 parole e lo strumento di paginazione e' un cilindro che ruota a 3000 giri al minuto e trasferisce un milione di parole al secondo.

Dal sistema si ottengono le seguenti misurazioni statistiche:

- L'1% di tutte le istruzioni eseguite hanno avuto accesso a una pagina diversa da quella corrente.
- L'80% delle istruzioni che hanno avuto accesso a una pagina diversa da quella corrente hanno avuto accesso a una pagina gia' in memoria.
- Quando e' stata richiesta una nuova pagina, la pagina da rimpiazzarsi era stata modificata nel 50% dei casi.

Calcolare (in microsecondi = $\mu sec.$) il tempo effettivo di esecuzione delle istruzioni di questo sistema, assumendo che esso stia eseguendo un solo processo e che il processore sia inattivo durante i trasferimenti del cilindro.

Traccia.

Prima si calcoli il tempo di accesso al cilindro. Si noti che il cilindro si puo considerare come un tipo particolare di disco rigido per cui non si debba considerare il tempo di posizionamento della testina, cioe' per il cilindro si deve tener conto solo del tempo di rotazione e del tempo di trasferimento.

Dopo si calcoli il tempo per le istruzioni che accedono alla pagina corrente, quello per le pagine non correnti ma in memoria, quello per le pagine che devono essere prelevate dal cilindro, e infine quello per le pagine che devono essere riscritte sul cilindro (in quanto modificate in scrittura e quindi che devono essere di nuovo salvate sul cilindro). Da ultimo si sommino tutti i contributi.

. Traccia di soluzione.

Si noti che un accesso al tamburo richiede un tempo di rotazione e un tempo di trasferimento:

- Tempo di rotazione (si sceglie di calcolare il tempo di una semirotazione):
3000 rotazioni al minuto = $(3000/60)=50$ rotazioni al secondo =
100 semirotazioni al secondo
una semirotazione richiede $1/100$ sec., cioè'
 $(1/100) (10^{-6}/10^{-6})_{sec.} = (1/10^{-4}) 10^{-6}_{sec.} = 1/10^{-4} \mu sec. = 10^4 \mu sec.$
- Tempo di trasferimento:
si trasferiscono 10^6 parole al secondo = 10^3 pagine al secondo (1 pagina
= 10^3 parole)
il trasferimento di una pagina richiede $1/1000$ sec., cioè'
 $(1/1000) (10^{-6}/10^{-6})_{sec.} = (1/10^{-3}) 10^{-6}_{sec.} = 1/10^{-3} \mu sec. = 10^3 \mu sec.$
- Tempo di rotazione + tempo di trasferimento:
 $10^4 \mu sec. + 10^3 \mu sec.$

Tempo medio effettivo di esecuzione di un'istruzione:

- Il 99% delle istruzioni accedono alla pagina corrente e perciò richiedono solo un microsecondo (ciclo del processore):

$$0,99 \times 1 \mu sec. = 0,99 \mu sec.$$

- L'80% delle istruzioni rimanenti accedono a un'altra pagina in memoria al costo addizionale di un altro microsecondo:

$$(80\% \times 0,01 =) 0,008 \times (1 + 1) \mu sec. = 0,008 \times 2 \mu sec. = 0,016 \mu sec.$$

- Il restante 20% delle istruzioni rimanenti richiedono un accesso al tamburo per prelevare una pagina:

$$(20\% \times 0,01 =) 0,002 \times (10^4 \mu sec. + 10^3 \mu sec.) = 22 \mu sec.$$

- La metà del precedente restante 20% delle istruzioni rimanenti richiede una scrittura sul tamburo della pagina rimpiazzata (in quanto modificata):

$$((20\% \times 0,01)/2 =) 0,001 \times (10^4 \mu sec. + 10^3 \mu sec.) = 11 \mu sec.$$

- Sommando tutti i contributi si ha:

$$0,99 \mu sec. + 0,016 \mu sec. + 22 \mu sec. + 11 \mu sec. \approx 34 \mu sec.$$

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

- Ci sono un segnale binario d'ingresso X e un segnale binario d'uscita Z .
- L'uscita z vale 1 al tempo t se e solo se il valore assunto da x al tempo t e' identico a quello assunto da x a due unita' di tempo precedenti: $x(t) = x(t - 2)$. In tutte le altre condizioni z vale 0.

(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.

Nota. Si supponga che il circuito sia stato portato nello stato iniziale con la sequenza d'ingressi 000, cioe' che quale che sia lo stato al momento dell'accensione al circuito sia applicata la sequenza d'ingressi 000 per portarlo nello stato iniziale. Ne consegue in particolare che nello stato iniziale il circuito produrra' 1 sotto l'ingresso 0 (corrispondente alla sequenza d'ingressi piu' recente 000) e produrra' 0 sotto l'ingresso 1 (corrispondente alla sequenza d'ingressi piu' recente 001).

Traccia di soluzione.

Si veda il libro di testo Sezione 6.8, Esempio 1).

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

- (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.

- (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.