

Elementi di Architettura e Sistemi Operativi

Bioinformatica - Tiziano Villa

26 Febbraio 2015

Nome e Cognome:

Matricola:

Posta elettronica:

problema	punti massimi	i tuoi punti
problema 1	10	
problema 2	5	
problema 3	5	
problema 4	10	
totale	30	

1. (a) Si descriva brevemente che cos'è un semaforo e si mostri lo pseudo-codice della definizione classica delle operazioni P e V .

Traccia di soluzione.

Un semaforo è una variabile intera cui si può accedere, escludendo l'inizializzazione, solo tramite due operazioni atomiche predefinite: P (*Wait*) and V (*Signal*).

Le definizioni classiche di *Wait* e *Signal* in pseudo-codice sono le seguenti:

```
Wait (S)  {  
    while (S <= 0)  
        ;  
    S--;  
}
```

```
Signal (S) {  
    S++;  
}
```

- (b) Siete incaricati di scrivere un programma che usa semafori e variabili condivise per appaiare ragazzi e ragazze in una gara di ballo, cioè formare coppie costituite esattamente da un ragazzo e da una ragazza che balleranno insieme per tutta la gara. Ogni ragazzo e ogni ragazza saranno rappresentati da un processo. Quando il ragazzo o la ragazza arriveranno, il loro processo chiamerà una di due procedure: *ragazzo* o *ragazza* a seconda di chi si tratti. Ogni procedura ha un singolo argomento: *nome*, che è il nome del processo. La procedura di un/a ragazzo/a deve aspettare fino a che arriva la procedura di una/un ragazza/o, e poi si devono scambiare i nomi per appaiarsi; alla fine la procedura restituisce il nome della persona con cui è avvenuto l'appaiamento. Ragazzi e ragazze possono arrivare in qualsivoglia ordine e molti processi possono chiamare le procedure *ragazzo* o *ragazza* simultaneamente. Non ci sono preferenze negli appaiamenti, tranne appunto il fatto che ogni coppia è composta da un ragazzo e da una ragazza, ciascuno dei quali ottiene il nome dell'altro a scopo identificativo.

Uno studente in apprendistato da voi propone la seguente soluzione con 4 semafori: *mutex_ragazzo* per ammettere un solo ragazzo per volta, *mutex_ragazza* per ammettere una sola ragazza per volta, *ragazzo_da_appaiare* semaforo di sincronizzazione per segnalare che c'è un ragazzo che attende una ragazza per scambiarsi i nomi, *ragazza_da_appaiare* semaforo di sincronizzazione per segnalare che c'è una ragazza che attende un ragazzo per scambiarsi i nomi. Inoltre la soluzione usa due variabili condivise: *nome_ragazzo* e *nome_ragazza* per memorizzare i nomi da scambiarsi di un ragazzo o ragazza.

```
SEM *mutex_ragazzo, *mutex_ragazza,  
    *ragazzo_da_appaiare, *ragazza_da_appaiare;  
int nome_ragazzo, nome_ragazza;
```

```
init  
    mutex_ragazzo = seminit(1);  
    mutex_ragazza = seminit(1);  
    ragazzo_da_appaiare = seminit(0);  
    ragazza_da_appaiare = seminit(0);  
endinit;
```

```

ragazzo(nome)
int nome;
{
    int ragazza_appaiata;

    P(mutex_ragazzo);
    nome_ragazzo = nome;
    V(ragazzo_da_appaiare);
    P(ragazza_da_appaiare);
    ragazza_appaiata = nome_ragazza;
    V(mutex_ragazzo);
    return(ragazza_appaiata);
}

```

```

ragazza(nome)
int nome;
{
    int ragazzo_appaiato;

    P(mutex_ragazza);
    nome_ragazza = nome;
    V(ragazza_da_appaiare);
    P(ragazzo_da_appaiare);
    ragazzo_appaiato = nome_ragazzo;
    V(mutex_ragazza);
    return(ragazzo_appaiato);
}

```

Si analizzi la soluzione proposta discutendone il funzionamento. E' possibile che il sistema si blocchi ?

Traccia di soluzione.

Il sistema non si puo' bloccare: si noti che in ogni sincronizzazione c'e' un'operazione V prima di P (ad es., quando arriva un ragazzo egli segnala di essere disponibile con $V(\text{ragazzo_da_appaiare})$ e chiede con $P(\text{ragazza_da_appaiare})$ se c'e' una ragazza disponibile aspettando in caso contrario).

L'unico modo per il sistema di bloccarsi sarebbe che i semafori *mutex* non fossero mai rilasciati; se entrambi tali semafori fossero bloccati al valore 0, allora ci dovrebbe essere un ragazzo e una ragazza in ciascuna procedura. Ma poiche' nelle sincronizzazioni con le variabili *ragazzo_da_appaiare* e *ragazza_da_appaiare* c'e' un'operazione V prima di P , ne' il ragazzo ne' la ragazza si bloccheranno su tali semafori, per cio' entrambi arriveranno alle operazioni V finali che rilasceranno i semafori *mutex*.

Grazie al ruolo giocato dai semafori *mutex*, nel sistema ci possono essere contemporaneamente solo un ragazzo e una ragazza. All'arrivo la persona registra il proprio nome nella relativa variabile condivisa *nome_ragazzo* o *nome_ragazza*. Poi fa scattare il semaforo *ragazzo_da_appaiare* oppure *ragazza_da_appaiare* e aspetta l'altra o l'altro. Quando ci sono sia un ragazzo che una ragazza ognuno legge il nome dell'altro dalla variabile condivisa *nome_ragazzo* o *nome_ragazza*. Infine la coppia e' pronta per partecipare alla gara di ballo.

Si noti che il fatto che ci possa essere una circostanza in cui un/a ragazzo/a aspetta invano (poiche' non ci sono piu' ragazze/i) non costituisce un caso di stallo del sistema, poiche' non e' escluso che prima o poi arrivi la persona attesa per costituire una coppia, ma il sistema non e' inchiodato da richieste mutualmente esclusive.

- (c) La soluzione proposta riesce a garantire sempre che ogni ragazzo scambi il proprio nome con una e una sola ragazza e viceversa che ogni ragazza scambi il proprio nome con uno e un solo ragazzo, quale che sia l'algoritmo di schedulazione dei processi ? In altri termini, se un ragazzo (ragazza) ha dato il proprio nome a una ragazza (ragazzo) anche quest'ultima (ultimo) deve aver dato il proprio nome al primo (prima) e viceversa, e nessuno dei due deve aver dato il proprio nome a una terza persona, in modo che si possano ottenere coppie di ballo univoche.

Si argomenti la risposta. Se la risposta fosse negativa si mostri un controesempio e si proponga una soluzione (la variante piu' semplice di quella data).

Traccia di soluzione.

La specifica puo' essere violata, perche' la soluzione proposta non ha un meccanismo per imporre che una coppia di ballerini si aspettino per entrare simultaneamente nell'arena di ballo.

Si consideri il seguente scenario: sono arrivati il ragazzo *RO1* e la ragazza *RA1*, il processo del primo e' interrotto dallo schedatore dopo l'istruzione *V(ragazzo_da_appaiare)* e prima di *P(ragazza_da_appaiare)*, la ragazza invece quando e' il suo turno esegue tutta la sua procedura ed entra nella sala da ballo con il nome del ragazzo *RO1* come compagno di gara. Poi arriva un'altra ragazza *RA2* che scrive il suo nome nella variabile *nome_ragazza* e si ferma al semaforo *P(ragazzo_da_appaiare)*. Infine il processo del ragazzo *RO1* riprende e completa l'esecuzione con il nome della ragazza *RA2* come compagna di gara poiche' quando esegue l'istruzione *ragazza_appaiata = nome_ragazza* la variabile *nome_ragazza* contiene gia' il nome della ragazza *RA2* (si noti inoltre che il semaforo *ragazza_da_appaiare* adesso vale 2). A questo punto la ragazza *RA1* pensa di dover ballare con il ragazzo *RO1* che invece pensa di dover ballare con la ragazza *RA2* (si possono immaginare vari finali di scenario per la ragazza *RA2*), e i conti non tornano.

Per risolvere il problema bisogna introdurre altri due semafori per sincronizzare l'entrata in gara delle coppie di ballerini. All'uopo si introducano i seguenti due semafori: *ragazzo_pronto_gara* semaforo di sincronizzazione per segnalare che un ragazzo si e' appaiato con una ragazza ed e' quindi pronto per entrare in gara, *ragazza_pronta_gara* semaforo di sincronizzazione per segnalare che una ragazza si e' appaiata con un

ragazzo ed e' quindi pronta per entrare in gara.

Il paio d'istruzioni P/V aggiunte alla fine del codice sincronizza le coppie di ballerini: in questo caso non permette alla ragazza $RA1$ di completare la sua procedura prima che il suo compagno di ballo $RO1$ abbia recuperato il ritardo e preso il nome di $RA1$ come compagna di ballo, e quindi sia ora pronto a completare a sua volta la procedura.

```
SEM *mutex_ragazzo, *mutex_ragazza,  
    *ragazzo_da_appaiare, *ragazza_da_appaiare;  
    *ragazzo_pronto_gara, *ragazza_pronta_gara;  
int nome_ragazzo, nome_ragazza;
```

```
init  
    mutex_ragazzo = seminit(1);  
    mutex_ragazza = seminit(1);  
    ragazzo_da_appaiare = seminit(0);  
    ragazza_da_appaiare = seminit(0);  
    ragazzo_pronto_gara = seminit(0);  
    ragazza_pronta_gara = seminit(0);  
endinit;
```

```
ragazzo(nome)  
int nome;  
{  
    int ragazza_appaiata;  
  
    P(mutex_ragazzo);  
    nome_ragazzo = nome;  
    V(ragazzo_da_appaiare);  
    P(ragazza_da_appaiare);  
    ragazza_appaiata = nome_ragazza;  
    V(ragazzo_pronto_gara);  
    P(ragazza_pronta_gara);  
    V(mutex_ragazzo)  
    return(ragazza_appaiata);  
}
```

```
ragazza(nome)
int nome;
{
    int ragazzo_appaiato;

    P(mutex_ragazza);
    nome_ragazza = nome;
    V(ragazza_da_appaiare);
    P(ragazzo_da_appaiare);
    ragazzo_appaiato = nome_ragazzo;
    V(ragazza_pronta_gara);
    P(ragazzo_pronto_gara);
    V(mutex_ragazza)
    return(ragazzo_appaiato);
}
```


2. Si consideri un sistema con un indirizzo logico di 32 cifre binarie e una dimensione di pagina di 4KB. Il sistema ammette 512 MB di memoria fisica.

Quanti elementi ci sono in una tavola delle pagine convenzionale a un solo livello ?

Quanti elementi ci sono in una tavola delle pagine invertita ?

Si argomentino le risposte.

Traccia di soluzione.

Il numero degli elementi in una tavola delle pagine convenzionale a un solo livello e' uguale al numero di pagine logiche.

Il numero di pagine logiche in una tavola delle pagine convenzionale a un solo livello = spazio d'indirizzamento logico / dimensione di una pagina logica = $2^{32} / 2^{12} = 2^{20}$.

Il numero degli elementi in una tavola delle pagine invertita e' uguale al numero di pagine fisiche.

Il numero di pagine fisiche = spazio d'indirizzamento fisico / dimensione di una pagina fisica = $2^{29} / 2^{12} = 2^{17}$ (spazio d'indirizzamento fisico = 512 MB = 2^{29} , dimensione pagina fisica = dimensione pagina logica).

3. Si consideri il seguente programma scritto nel linguaggio macchina LC-3.

```

                .ORIG
                LD      R2, ZERO
                LD      R0, M0
                LD      R1, M1
LOOP           BRz     DONE
                ADD     R2, R2, R0
                ADD     R1, R1, -1
                BR      LOOP
DONE           ST      R2, RESULT
                HALT
RESULT        .FILL    x0000
ZERO          .FILL    x0000
M0            .FILL    x0004
M1            .FILL    x0803
                END
```

Si spieghi il funzionamento di tale programma. S'indichi quale valore (in esadecimale, in binario, in decimale) sarà contenuto nella locazione di memoria RESULT alla fine della sua esecuzione.

Traccia di soluzione

Il programma moltiplica i valori agli indirizzi M0 e M1, e memorizza il risultato all'indirizzo RESULT:

$$mem[RESULT] = mem[M0] * mem[M1]$$

$mem[RESULT]$ alla fine conterrà $x200C_{16} = 0010\ 0000\ 0000\ 1100_2 = 8204_{10}$ (da $x0004 * x0803 = x200C$, $4_{10} * 2051_{10} = 8204_{10}$).

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 (cioe', 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.

Si puo' rappresentare lo stato come $X_1(t - 1)X_1(t - 2)$; quando arriva un ingresso $X_1(t)$ si ha una transizione dallo stato $X_1(t - 1)X_1(t - 2)$ allo stato $X_1(t)X_1(t - 1)$, che produce $Z(t) = X_1(t - 1)$ se $X_2(t) = 0$ e $Z(t) = X_1(t - 2)$ se $X_2(t) = 1$.

X1	X2	SP	SF	Z
0	0	00	00	0
0	1	00	00	0
1	1	00	10	0
1	0	00	10	0
0	0	01	00	0
0	1	01	00	1
1	1	01	10	1
1	0	01	10	0
0	0	11	01	1
0	1	11	01	1
1	1	11	11	1
1	0	11	11	1
0	0	10	01	1
0	1	10	01	0
1	1	10	11	0
1	0	10	11	1

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

- (c) Si scriva la tavola delle transizioni minimizzata con gli stati futuri e le uscite, si assegnino codici binari agli stati, e si riscriva la tavola delle transizioni codificata.

- (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. Si mostri il procedimento di minimizzazione logica.

- (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.