Elementi di Sistemi Operativi

Bioinformatica - Tiziano Villa

23 Giugno 2008

Nome e Cognome:
Matricola:
Posta elettronica:

problema	punti massimi	i tuoi punti
problema 1	8	
problema 2	10	
problema 3	6	
problema 4	6	
totale	30	

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

   (a) Si descrivano succintamente i servizi di un sistema operativo relativamente all'interfaccia con l'utente e all'esecuzione di un programma.

   Traccia di soluzione.

   Si veda la Sez. 2.1 del testo di Silberschatz.

(b) In che modo si possono passare dei parametri al sistema operativo ?

Traccia di soluzione.

Tre modi di uso comune sono:

i. Passaggio dei parametri tramite registri.

ii. Passaggio in registri degl'indirizzi iniziali dei blocchi di parametri.

iii. Passaggio dei parametri tramite la pila ("stack") su cui il programma utente li scrive e il sistema operativo li legge.

Si veda alla fine della Sez. 2.3 del testo di Silberschatz.

(c) Quali sono le attivita' principali di un sistema operativo rispetto alla gestione del "file system" ?

Traccia di soluzione.

Si veda la Sez. 2.1 del testo di Silberschatz.

(d) Quali sono i modelli di comunicazione tra processi ? Quali sono i pro e contro di tali modelli ?

Traccia di soluzione.

I due modelli fondamentali sono lo scambio di messaggi oppure la memoria condivisa. Si veda la Sez. 2.4.5 del testo di Silberschatz.

2. Si consideri il seguente codice per gestire dei processi concorrenti di lettura e scrittura da una base di dati mediante il paradigma a monitor (con le primitive *wait*, *signal* e *broadcast* - quest'ultima risveglia tutti i processi sospesi).

   Siano definite le variabili di stato

```
int AR = 0; /* numero di lettori attivi */
int WR = 0; /* numero di lettori in attesa */
int AW = 0; /* numero di scrittori attivi */
int WW = 0; /* numero di scrittori in attesa */
okToRead = nil /* variabile di condizione (lettura) */
okToWrite = nil /* variabile di condizione (scrittura) */

Reader() {
  lock.Acquire();

  while ((AW + WW) > 0) {
    WR++;
    okToRead.wait(&lock);
    WR--;
  }
  AR++;
  lock.Release();

  /* Accedi in lettura alla base dati */
  AccessDatabase(ReadOnly);

  lock.Acquire();
  AR--;
  if (AR == 0 && WW > 0)
    okToWrite.signal();
  lock.Release
}

Writer() {
  lock.Acquire();

  while ((AW + AR) > 0) {
```

```
    WW++;
    okToWrite.wait(&lock);
    WW--;
  }
  AW++;
  lock.Release();

  /* Accedi in scrittura alla base dati */
  AccessDatabase(Write);

  lock.Acquire();
  AW--;
  if (WW > 0) {
    okToWrite.signal();
  } else if (WR > 0) {
    okToRead.broadcast();
  }
  lock.Release
}
```

Si commenti il codice precedente per spiegarne il funzionamento.

In particolare si risponda nello stesso ordine a tutte le seguenti domande:

(a) Lettori e scrittori hanno la medesima priorita' ?

Traccia di soluzione.

Gli scrittori sono privilegiati rispetto ai lettori poiche' i lettori non possono accedere alla base di dati se ci sono scrittori in attesa, secondo l'istruzione in `Reader`

```
while ((AW + WW) > 0) {
```

(b) Possono essere attivi contemporaneamente piu' processi nella base di dati e di che tipo ?

Traccia di soluzione.

Per la precedente istruzione possono essere attivi contemporaneamente piu' lettori. Tuttavia ci puo' essere un solo scrittore (senza altri lettori o scrittori), come si evince dall'istruzione in `Writer`

```
while ((AW + AR) > 0) {
```

che non rende attivi altri scrittori, se ce n'e' gia' uno attivo. Quindi uno scrittore non puo' accedere alla base di dati se vi e' attivo un lettore o un altro scrittore.

(c) Perche' in `Reader` c'e' l'istruzione

```
lock.Release();
```

? Se ne puo' fare a meno (eliminando cioe' la coppia `lock.Release` e `lock.Acquire` prima e dopo `AccessDatabase`) ?

Traccia di soluzione.

Per permettere a piu' lettori di entrare nella base di dati (altrimenti se ci fosse un lettore attivo nella base di dati, un secondo lettore non potrebbe entrare perche' il primo deterrebbe il lucchetto).

(d) Perche' in `Writer` c'e' l'istruzione

```
lock.Release();
```

? Se ne puo' fare a meno (eliminando cioe' la coppia `lock.Release` e `lock.Acquire` prima e dopo `AccessDatabase`) ?

Traccia di soluzione.

Anche se ci puo' un essere un solo scrittore per volta nella base di dati, rilasciando il lucchetto si permette ad altri scrittori e lettori di registrarsi come processi in attesa.

(e) Perche' in `Writer` c'e' l'istruzione

```
okToRead.broadcast();
```

invece che

```
okToRead.signal();
```

?

Traccia di soluzione.

Per risvegliare tutti i lettori in attesa, quando non ci sono scrittori in attesa.

(f) Puo' succedere che alcuni processi non divengano mai attivi ("starvation") ?

Traccia di soluzione.

Puo' succedere che l'arrivo continuo di scrittori impedisca ai lettori di accedere alla base di dati, poiche' un lettore non puo' accedere alla base di dati se c'e' uno scrittore in attesa.

(g) Si puo' eliminare in `Reader` la condizione `if (AR == 0 && WW > 0)`, scrivendo

```
lock.Acquire();
AR--;
okToWrite.signal();
lock.Release
```

invece di

```
lock.Acquire();
AR--;
if (AR == 0 && WW > 0)
  okToWrite.signal();
lock.Release
```

?

Traccia di soluzione.

Si, il codice e' ancora corretto, poiche' uno scrittore risvegliato da

```
okToWrite.signal();
```

ma senza che valga la condizione

```
if (AR == 0 && WW > 0)
```

(vedi (*)) se ne accorgera' quando ricalcolera' il predicato del ciclo `while` iniziale che controlla il diritto a diventare attivo

```
while ((AW + AR) > 0) {
```

e ritornera' in attesa. Percio' si sprecano dei cicli di calcolo, ma il codice e' ancora corretto.

(*) Corrisponde al caso in cui ci siano altri lettori attivi `AR > 0`; si noti che se non ci sono scrittori in attesa, `WW = 0`, il comando

```
okToWrite.signal();
```

non ha effetto.

(h) Si puo' ulteriormente cambiare il codice precedente in `Reader` da

```
lock.Acquire();
AR--;
okToWrite.signal();
lock.Release
```

a

```
lock.Acquire();
AR--;
okToWrite.broadcast();
lock.Release
```

?

Traccia di soluzione.

Si, saranno risvegliati tutti gli scrittori in attesa, ma uno solo di essi riuscira' ad entrare nella base di dati e gli altri torneranno in attesa (ciascun processo ricalcolera' il predicato del ciclo `while` iniziale che deve risultare falso affinche' il processo possa diventare attivo). Ancora una volta il codice sara' corretto, ma meno efficiente di quello originale.

(i) Si possono rimpiazzare le due variabili di condizione

```
okToRead = nil /* variabile di cond. (lettura) */
okToWrite = nil /* variabile di cond. (scrittura) */
```

con una sola

```
okToContinue = nil /* variabile di cond. (l/s) */
```

?

Traccia di soluzione.

Si, lettori e scrittori dormiranno nella coda della stessa variabile di condizione. Bisogna usare `broadcast()` invece di `signal()`. Ogni processo risvegliato verifichera' la condizione del ciclo `while` iniziale e solo uno diventira' attivo seconde le precedenze discusse (scrittori prima di lettori). Soluzione ancora funzionante, ma meno efficiente di quella originale.

(j) Si simuli il codice originale con i seguenti eventi, indicando i valori assunti dalle variabili di stato AR, WR, AW, WW, inizialmente a 0.

i. Arriva il lettore $R1$.
Traccia di soluzione. $R1$ diventa attivo nella base di dati.

$AR = 1, WR = 0, AW = 0, WW = 0$.

ii. Arriva il lettore $R2$.
Traccia di soluzione. $R2$ diventa attivo nella base di dati.
$AR = 2, WR = 0, AW = 0, WW = 0$.

iii. Arriva lo scrittore $W1$.
Traccia di soluzione. Poiche' ci sono dei lettori nella base di dati, $W1$ si mette a dormire nella coda degli scrittori in attesa `okToWrite`.
$AR = 2, WR = 0, AW = 0, WW = 1$.

iv. Arriva il lettore $R3$.
Traccia di soluzione. Poiche' ci sono degli scrittori in attesa, questo nuovo lettore si mette a dormire nella coda dei lettori in attesa `okToRead`.
$AR = 2, WR = 1, AW = 0, WW = 1$.

v. Il lettore $R2$ finisce.
Traccia di soluzione.
$AR = 1, WR = 1, AW = 0, WW = 1$.

vi. Il lettore $R1$ finisce.
Traccia di soluzione. Il lettore $R1$ quando finisce risveglia lo scrittore $W1$ che diventa attivo.
$AR = 0, WR = 1, AW = 1, WW = 0$.

vii. Lo scrittore $W1$ finisce.
Traccia di soluzione. Lo scrittore $W1$ quando finisce risveglia il lettore $R3$ che diventa attivo.
$AR = 1, WR = 0, AW = 0, WW = 0$.

viii. Il lettore $R3$ finisce.
Traccia di soluzione.
$AR = 0, WR = 0, AW = 0, WW = 0$.

.

.

3. Siano dati 5 processi con la durata della sequenza di operazioni della CPU espressa in millesecondi.

Processo	Durata
P1	10
P2	1
P3	2
P4	1
P5	5

Si supponga che i processi siano arrivati nell'ordine $P1, P2, P3, P4, P5$ e che siano tutti presenti al tempo 0.

(a) i. Si disegni lo schema GANTT (come nel libro di testo) che illustri l'esecuzione di questi processi con l'algoritmo di schedulazione FCFS (First-Come,First-Served).
Traccia di soluzione.

P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P2	P3	P3	P4	P5	P5	P5	P5	P5
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18 19

ii. Si calcoli il tempo di completamento di ciascun processo.
Traccia di soluzione.
I tempi di completamento sono:
P1 10,
P2 11,
P3 13,
P4 14,
P5 19

iii. Si calcoli il tempo di attesa di ciascun processo.
Traccia di soluzione.
I tempi di attesa sono (tempo di completamento - durata):
P1 0,
P2 10,
P3 11,
P4 13,
P5 14

iv. Si calcoli il tempo di attesa medio tra tutti i processi.
Traccia di soluzione.

Tempo di attesa medio: (0+10+11+13+14)/5 = 9,6

(b) i. Si disegni lo schema GANTT (come nel libro di testo) che illustri l'esecuzione di questi processi con l'algoritmo di schedulazione SJF (Shortest Job First).
Traccia di soluzione.

```
P2 P4 P3 P3 P5 P5 P5 P5 P5 P1 P1 P1 P1 P1 P1 P1 P1 P1 P1
X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19
```

ii. Si calcoli il tempo di completamento di ciascun processo.
Traccia di soluzione.
I tempi di completamento sono:
P1 19,
P2 1,
P3 4,
P4 2,
P5 9

iii. Si calcoli il tempo di attesa di ciascun processo.
Traccia di soluzione.
I tempi di attesa sono (tempo di completamento - durata):
P1 9,
P2 0,
P3 2,
P4 1,
P5 4

iv. Si calcoli il tempo di attesa medio tra tutti i processi.
Traccia di soluzione.
Tempo di attesa medio: (9+0+2+1+4)/5 = 3,2

4. Si consideri un programma che genera la seguente successione di accessi a pagina:
   A A A A B B C A C B B D D A E B C E A E D E.

   Si assuma che il programma abbia a sua disposizione tre pagine fisiche (”frames”).

   (a) Quante assenze di pagina (”page faults”) si generano usando lo schema LRU (pagine usate meno frequentemente) per la sostituzione di pagina ?
   Traccia di soluzione.

```
accessi:
 A A A A B B C A C B B D D A E B C E A E D E

esecuzione algoritmo LRU:
pagina fisica 1:
 A A A A A A A A A A A D D D D B B B A A A A
 ^                     ^       ^     ^

pagina fisica 2:
 - - - - B B B B B B B B B B E E E E E E E E
         ^                   ^

pagina fisica 3:
 - - - - - - - C C C C C C C A A A C C C C D D
             ^               ^     ^       ^
```

   Ci sono 10 assenze di pagina (indicate dal segno ^).

(b) Quante assenze di pagina (”page faults”) si generano usando lo schema dell'Insieme di Lavoro (”working-set”) per la sostituzione di pagina ? Si supponga che la finestra dell'insieme di lavoro sia $\delta = 6$, incluso il riferimento corrente. Si supponga che le pagine siano eliminate dalla memoria quando lasciano l'insieme di lavoro. In questo caso al programma possono servire piu' di 3 pagine fisiche (cioe' non vale piu' la restrizione che abbia a disposizione solo 3 pagine). S'indichi la dimensione dell'insieme di lavoro in ogni momento.

Traccia di soluzione.

```
accessi:
 A A A A B B C A C B B D D A E B C E A E D E

esecuzione algoritmo Insieme di Lavoro:
pagina fisica 1:
 A A A A A A A A A A A A A A A A A A A A A A
 ^
pagina fisica 2:
 - - - - B B B B B B B B B B B B B B B B B -
         ^
pagina fisica 3:
 - - - - - - C C C C C C C C E E E E E E E E
             ^               ^
pagina fisica 4:
 - - - - - - - - - - - - D D D D D D D - - D D
                         ^                 ^
pagina fisica 5:
 - - - - - - - - - - - - - - - - - - C C C C C C
                                     ^
dimensione insieme di lavoro:
 1 1 1 1 2 2 3 3 3 3 3 4 4 4 4 4 5 5 4 4 5 4
```

Ci sono 7 assenze di pagina (indicate dal segno^).