K22: Λειτουργικά Συστήματα (Περιττοί Αριθμ. Μητρώου) - Χειμερινό ΄16

2ο Σετ Ασκήσεων
Λύσεις

- **Πρόβλημα 1:**

```
var choosing: shared array[0..n-1] of boolean;
   number: shared array[0..n-1] of integer;

repeat
    choosing[i] := true;
    number[i] := max(number[0],number[1],...,number[n-1]) + 1;
    choosing[i] := false;

    for j := 0 to 2 do begin
     while choosing[j] do (* nothing *);
        while number[j] <> 0 and (number[j], j) < (number[i],i) do
            (* nothing *);
    end;

    (* critical section *)
    number[i] := 0;
    (* remainder section *)
    until false;
```

Σύμφωνα με τον παραπάνω κώδικα, για κάθε διεργασία ελέγχεται το αν ¨κρατάει' κάποιο εισιτήριο. Στην περίπτωση που κάποια βρίσκεται στην διαδικασία έκδοσης του εισιτηρίου (choosing[j]), η εκτέλεση περιμένει μέχρι να τελειώσει η διαδικασία αυτή. Όταν η διεργασία έχει/λάβει τιμή εισιτηρίου, τότε ελέγχεται αν έχει την υψηλότερη προτεραιότητα και η εκτέλεση περιμένει μέχρι την ολοκλήρωσή της, αλλιώς προχωρά η εκτέλεση. Οι συνδυασμοί (εισιτήριο, αριθμός_/διεργασίας), δηλαδή (number[j], j) που θα παραχθούν, παρέχουν σε κάθε διεργασία προτεραιότητα ως εξής: Στην περίπτωση που διαφορετικά εισιτήρια υπάρχουν, το μικρότερο εισιτήριο δίνει μεγαλύτερη προτεραιότητα. Στην περίπτωση που τα εισιτήρια έχουν την ίδια τιμή, τότε μεγαλύτερη προτεραιότητα έχει η μικρότερη διεργασία. Σύμφωνα με το παραπάνω, εξασφαλίζεται ότι μόνο μία διεργασία θα βρίσκεται στην κρίσιμη περιοχή κάθε φορά (safety).

Κάθε διεργασία που ¨κρατάει' εισιτήριο, συγκρίνεται με τις υπόλοιπες που κρατάνε επίσης εισιτήριο ώστε να βρεθεί αυτή που έχει υψηλότερη προτεραιότητα από όλες τις άλλες. Αυτή θα εισέλθει στην κρίσιμη περιοχή. Για μια διεργασία k που λαμβάνει εισιτήριο και (num[k],k)<(num[i],i) θα ισχύει ότι η k δεν θα προχωρήσει μέχρι η διεργασία i να προχωρήσει πρώτα (k χαμηλότερης προτεραιότητας από την i). Επιπλέον, λόγω της ιδιότητας που περιγράφηκε παραπάνω και της διαδικασίας που καινούρια εισιτήρια δημιουργούνται, καμιά διεργασία δεν θα εκτελεί για πάντα κάποια από τις επαναλήψεις while του παραπάνω κώδικα (progress).

- **Πρόβλημα 2:**

```
wait(R) {
    while(R <= 0);
    R--;
}
```

Έστω ότι ο παραπάνω κώδικας δεν εκετελείται ατομικά και ισχύει R=1:

Τότε δύο threads, thread1 και thread2, μπορούν να εκτελέσουν την λειτουργία wait(R) ταυτόχρονα. Στην περίπτωση αυτή και τα δύο threads θα μεταβάλλουν την τιμή του R την ίδια στιγμή, καθώς και για τα δύο threads, η τιμή του R είναι 1. Επομένως, το thread1 θα μειώσει τη τιμή του R κατά ένα, όπως και το thread2 και θα εισέλθουν και τα δύο στην κρίσιμη περιοχή (mutual exclusion violation).

- **Πρόβλημα 3:**

```
integer readcount = 0
integer writecount = 0

semaphore mutex1 = 1
semaphore mutex2 = 1
semaphore mutex3 = 1
semaphore w = 1
semaphore r = 1
```

Readers:

```
wait(mutex3);
wait(r);
wait(mutex1);
readcount++;
if (readcount==1) wait(w);
signal(mutex1);
signal(r);
signal(mutex3);

doReading();

wait(mutex1);
readcount--;
if (readcount==0) signal(w);
signal(mutex1);
```

Writers:

```
wait(mutex2);
writecount++;
if (writecount==1) wait(r);
signal(mutex2);
wait(w);

doWriting();

signal(w);
wait(mutex2);
writecount--;
if (writecount==0) signal(r);
signal(mutex2);
```

- **Πρόβλημα 4:** (a) **FIFO**

J_0	[1, 18]
J_1	[19, 28]
J_2	[29, 52]
J_3	[53, 56]
J_4	[57, 65]

Waiting times
WT_0 =1
WT_1 = 19-3=16
WT_2 = 29-14=15
WT_3 = 53-19=34
WT_4 = 34

AWT = $\frac{1+16+15+34+34}{5} = 20$

Turnaround times
TT_0 = 18-0=18
TT_1 = 28-3=25
TT_2 = 52-14=38
TT_3 = 56-19=37
TT_4 = 65-23=42

ATT = $\frac{18+25+38+37+42}{5} = 32$

(b) **Optimal**

J_0	[1, 3]
J_1	[4, 13]
J_0	[14, 19]
J_3	[20, 23]
J_4	[24, 32]
J_0	[33, 43]
J_2	[44, 67]

Waiting times
WT_0 = 1+14-3+33-19=26
WT_1 = 1
WT_2 = 44-14=30
WT_3 = 1
WT_4 = 1

AWT = $\frac{26+3+30}{5}$ = 11,8

Turnaround times
WT_0 = 43
WT_1 = 13-3=10
WT_2 = 67-14=53
WT_3 = 23-19=4
WT_4 = 32-23=11

AWT = $\frac{43+10+53+4+11}{5}$ = 23,8

(c) **Round Robin**

J_0	[1, 7]
J_1	[8, 14]
J_0	[15, 21]
J_2	[22, 28]
J_1	[29, 32]
J_3	[33, 36]
J_0	[37, 42]
J_4	[43, 49]
J_2	[50, 56]
J_4	[57, 59]
J_2	[60, 71]

Waiting times
WT_0 = 1+(15-7)+(37-21)=25
WT_1 = (8-3)+(29-14)=20
WT_2 = (22-14)+(50-28)+(60-56)=34
WT_3 = 33-19=14
WT_4 = (43-23)+(57-49)=28

AWT = $\frac{25+20+34+14+28}{5}$ = 24.2

Turnaround times
TT_0 = 42-0=42
TT_1 = 32-3=29
TT_2 = 71-14=57
TT_3 = 36-19=17
TT_4 = 59-23=36

ATT = $\frac{42+29+57+17+36}{5}$ = 36.2