Κατανεμημένα Συστήματα
Παναγιώτα Φατούρου

2ο Σετ Ασκήσεων

Προθεσμία παράδοσης: 25/5 στο μάθημα

Άσκηση 1

α. Δίνεται η παραλλαγή του αλγορίθμου του Peterson Που φαίνεται στο Σχήμα 1. Είναι ο αλγόριθμος αυτός σωστή λύση του προβλήματος του αμοιβαίου αποκλεισμού για δύο διεργασίες; Τι θα συμβεί αν αλλάξουμε τη σειρά των γραμμών 4 και 5 του κώδικα των threads;

Thread 0	Thread 1
1. turn = 1	1. turn = 0
2. flag[0] = true	2. flag[1] = true
3. while (TRUE) do {	3. while (TRUE) {
4. if (flag[1]==FALSE) goto CS;	4. if (flag[0]==FALSE) goto CS;
5. if (turn == 0) goto CS;	5. if (turn = 1) goto CS;
6. }	6. }
7. CS: critical section	*7. CS: critical section*
8. flag[0] = false	8. flag[1] = false

Σχήμα 1: Παραλλαγή του Peterson

β. Παρουσιάστε έναν αλγόριθμο που να χρησιμοποιεί tournament tree για να επιλύσει το πρόβλημα του αμοιβαίου αποκλεισμού για n διεργασίες. Ο αλγόριθμος θα πρέπει να χρησιμοποιεί τον αλγόριθμο του Peterson για την επίλυση του προβλήματος του αμοιβαίου αποκλεισμού μεταξύ δυο διεργασιών. Περιγράψτε τις ιδιότητες του αλγορίθμου που προτείνατε και αναλύστε την πολυπλοκότητά του.

Άσκηση 2

α. Μπορούμε να αφαιρέσουμε την γραμμή “await y = 0” (γραμμή 9) του γρήγορου (fast) αλγορίθμου αμοιβαίου αποκλεισμού που περιγράφεται στο τέλος των διαφανειών αμοιβαίου αποκλεισμού;

β. Παρουσιάστε σενάριο εκτέλεσης του One Bit αλγορίθμου στο οποίο κάποια διεργασία υφίσταται παρατεταμένη στέρηση.

γ. Εξετάστε αν η παραλλαγή του γρήγορου αλγορίθμου (fast mutual exclusion) που περιγράφεται στο Σχήμα 2 παρουσιάζει κάποιο πρόβλημα;

Initially: y = 0 and b[i] = false for all i.

```
start:  b[i] = true;
        x = i;
        if (y != 0) {
                b[i] = false;
                await y == 0;
                goto start;
        }
        y = i;
        b[i] = false;
        if (x != i) {
                for j = 1 to n do await !(b[j]);
                if (y != i) {
                        await y == 0;
                        goto start;
                }
        }
        else y = i;

        <critical section>;

        y = 0;
```

Σχήμα 2: Παραλλαγή του Fast Mutual Exclusion Algorithm

Άσκηση 3

α. Παρουσιάστε σενάριο εκτέλεσης του Bakery αλγορίθμου στο οποίο ο καταχωρητής Number[i] κάποιας διεργασίας αποθηκεύει διαρκώς μεγαλύτερες τιμές.

β. Θα ήταν σωστός ο Black White αλγόριθμος αν κάθε διεργασία p_i χρησιμοποιούσε το id της ως Number[i];

γ. Επιχειρηματολογήστε για τη σημαντικότητα της σειράς με την οποία εκτελούνται οι δύο γραμμές του exit section του Black White αλγορίθμου. Τι θα συνέβαινε αν εκτελείτο πρώτα η γραμμή “Number[i] = 0” και στη συνέχεια η τρέχουσα πρώτη γραμμή του exit section (δηλαδή αν αλλάζαμε τη σειρά των γραμμών 11 και 12 του κώδικα);

δ. Εξηγήστε γιατί ο Black White αλγόριθμος οδηγεί σε αδιέξοδο (deadlock) αν η τρίτη συνθήκη του await της γραμμής 8 αφαιρεθεί.