

Προσπέλαση κοινών πόρων

Πρωτόκολλα ελέγχου αμοιβαίου αποκλεισμού

Μ.Στεφανιδάκης

Κοινοί πόροι

- Κοινοί (διαμοιραζόμενοι) πόροι
 - με μία ή περισσότερες μονάδες (units)
- Αποκλειστική προσπέλαση
 - Αμοιβαίος αποκλεισμός (mutual exclusion)
 - κρίσιμες περιοχές
 - μη προεκτοπισμός!
 - Μηχανισμός lock - unlock
 - απόκτηση - απελευθέρωση πόρου
 - μέσω δομών όπως semaphores - blocking

```
funcA( ) {  
    ...  
    lock(R);  
    <use R>  
    unlock(R);  
    ...  
}
```

Προσπέλαση πολλαπλών πόρων

- Εμπιερχόμενες κρίσιμες περιοχές
 - nested critical regions
 - διάταξη LIFO
- Χρήση πολλαπλών πόρων
 - Πιθανότητα deadlock

```
funcB( ) {  
    ...  
    lock(R1);  
    lock(R2);  
    <use R1, R2>  
    unlock(R2);  
    unlock(R1);  
    ...  
}
```

Deadlock

- Πιθανότητα εμφάνισης
 - ταυτόχρονη χρήση πολλαπλών πόρων
 - χρήση κρίσιμων περιοχών που εμπιέρονται η μία μέσα στην άλλη (nested)
 - προεκτοπιστικό σύστημα
- Παράδειγμα:
 - όταν δύο διεργασίες προσπαθούν να αποκτήσουν η μία τον πόρο που κατέχει η άλλη

Deadlock

- Παράδειγμα

```
taskA( ) {  
    ...  
    lock(R1); ✓  
    lock(R2); ✗  
    <use R1, R2>  
    unlock(R2);  
    unlock(R1);  
    ...  
}
```

```
taskB( ) {  
    ...  
    lock(R2); ✓  
    lock(R1); ✗  
    <use R1, R2>  
    unlock(R1);  
    unlock(R2);  
    ...  
}
```

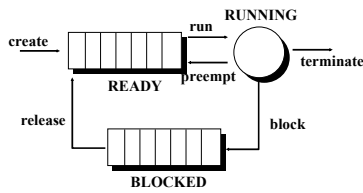
Αποφυγή deadlock

- Απόκτηση συνόλου πόρων πριν τη συνέχιση
- Απόκτηση των πόρων στην ίδια σειρά
- Ειδικά πρωτόκολλα αποφυγής deadlock

```
taskA( ) {  
    ...  
    lock(R1); ✓  
    lock(R2); ✓  
    <use R1, R2>  
    unlock(R2);  
    unlock(R1);  
    ...  
}
```

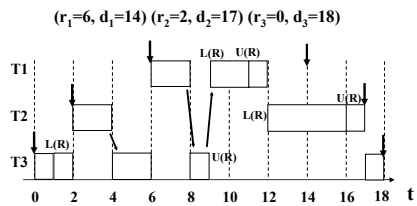
```
taskB( ) {  
    ...  
    lock(R2); ✗  
    lock(R1);  
    <use R1, R2>  
    unlock(R2);  
    unlock(R1);  
    ...  
}
```

Προσπέλαση πόρων και ουρές διεργασιών



- Αίτηση απόκτησης αποκλειστικής χρήσης πόρου
 - Γίνεται δεκτή (grant)
 - Δεν γίνεται δεκτή \Rightarrow η διεργασία απενεργοποιείται (block)
- Απελευθέρωση πόρου (release)

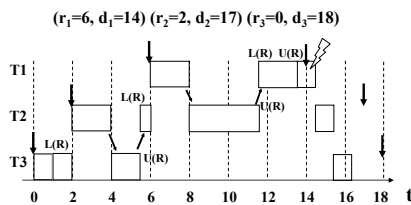
Σύγκρουση προσπέλασης πόρων



- Χρονοδρομολόγηση EDF
 - Χρόνοι εκτέλεσης: (5, 7, 6)
 - Χρόνοι σε κρίσιμη περιοχή: (2, 4, 4)
- Ένας κοινός πόρος (R)

Χρονικές ανωμαλίες

αύξηση συνολικού χρόνου ολοκλήρωσης με μείωση επιμέρους χρόνων εκτέλεσης!

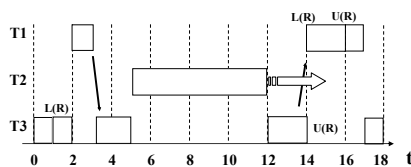


- Χρονοδρομολόγηση EDF
 - Χρόνοι εκτέλεσης: (5, 7, 6)
 - Χρόνοι σε κρίσιμη περιοχή: (2, 4, 5)

Προσπέλαση κοινών πόρων σε ΛΣΠΧ

- Αντιστροφή προτεραιότητας (priority inversion)
 - διεργασία υψηλής προτεραιότητας περιμένει διεργασία χαμηλής προτεραιότητας
 - όμως αναπόφευκτη!
- Μη προβλέψιμο σύστημα
 - χρόνος σε κρίσιμη περιοχή
 - χρονικές ανωμαλίες
 - μη φραγμένη διάρκεια αντιστροφής προτεραιότητας
 - deadlock
- Αναγκαίος ο ακριβής έλεγχος προσπέλασης
 - δεν αρκούν οι παραδοσιακές δομές ενός ΛΣ

Ανεξέλεγκτη αντιστροφή προτεραιότητων



- Η καθυστέρηση του T1 δεν εξαρτάται μόνο από τον χρόνο του T3 στην κρίσιμη περιοχή!
- T2 δεν προσπελάει κοινό πόρο!

Έλεγχος προσπέλασης πόρων

- Μέθοδος ελέγχου
 - Μη διακοπή σε κρίσιμες περιοχές
 - Priority Inheritance
 - Priority Ceiling
- Αποτέλεσμα
 - Εξάλειψη ανεξέλεγκτης αντιστροφής προτεραιότητων
 - Μείωση χρόνων αναμονής για τη χρήση πόρων
 - Αποφυγή deadlock
- Υλοποίηση
 - Για συστήματα με στατικές ή δυναμικές προτεραιότητες;
 - Απαιτείται γνώση εκ των προτέρων για τη χρήση των κοινών πόρων;

Μη διακοπή εκτέλεσης σε κρίσιμη περιοχή

- Εκτέλεση με την υψηλότερη προτεραιότητα
 - Μη προεκτοπισμός (πλήρης)
 - εμποδίζει εκτέλεση ακόμα κι όταν δεν υπάρχει σύγκρουση!
 - Αποφυγή deadlock
 - Αποφυγή ανεξέλεγκτης αντιστροφής προτεραιότητας
 - μόνο μία φορά
 - μέγιστη καθυστέρηση = max χρόνων εκτέλεσης σε κρίσιμες περιοχές

$$B_i = \max_{i \neq k, k \leq n} (C_k)$$

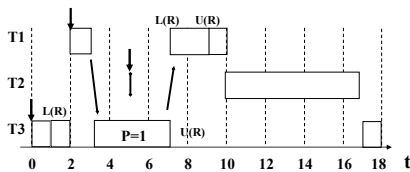
- fixed-priority: των διεργασιών χαμηλότερης προτεραιότητας
- EDF: των διεργασιών με αργότερο deadline

- απλό πρωτόκολλο
 - Δεν απαιτείται πληροφορία χρήσης πόρων
 - Για συστήματα με στατικές ή δυναμικές προτεραιότητες

Priority Inheritance Protocol (PIP)

- Χρονοδρομολόγηση με (στατικές) προτεραιότητες
 - ονομαστική και τρέχουσα προτεραιότητα
- Μία διεργασία σε κρίσιμη περιοχή αποκτά τη μέγιστη προτεραιότητα από όλες τις διεργασίες που εμποδίζει.
 - μεταβατική ιδιότητα
 - αποφυγή ανεξέλεγκτης αντιστροφής προτεραιότητας
 - $\min(u, k)$ φορές
 - u = αριθμός πόρων που προσπελαύνει
 - k = αριθμός διεργασιών χαμηλότερης προτεραιότητας με τις οποίες συγκρούεται
 - αλλά όχι και deadlock
 - δεν ελαχιστοποιεί χρόνους blocking
- Δεν απαιτείται πληροφορία χρήσης κοινών πόρων
 - μόνο για τον υπολογισμό χρόνου blocking
 - αλγοριθμικά

Priority Inheritance Protocol

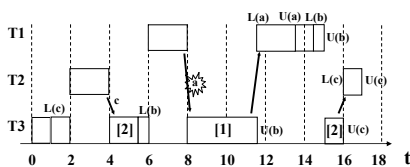


Priority Ceiling Protocol

- Βασική ιδέα:
 - μία διεργασία δεν επιτρέπεται να εισέλθει σε κρίσιμη περιοχή, εάν υπάρχουν άλλοι "κλειδωμένοι" πόροι που θα μπορούσαν να την μπλοκάρουν.
 - για συστήματα με στατικές προτεραιότητες
 - απαιτείται γνώση χρησιμοποιούμενων πόρων
 - κάθε πόρος χαρακτηρίζεται από ένα όριο (priority ceiling)
 - η υψηλότερη προτεραιότητα διεργασίας που μπορεί να τον χρησιμοποιήσει (υπολογισμός off-line)
 - προσπέλαση πόρου
 - μόνον όταν προτεραιότητα διεργασίας > μεγαλύτερο priority ceiling χρησιμοποιούμενων πόρων (εκτός ίδιας)
 - priority inheritance

Priority Ceiling Protocol (PCP)

$$C(a)=1, C(b)=1, C(c)=2$$



Priority Ceiling Protocol

- Αποτέλεσμα μεθόδου:
 - Blocking: μόνο μία φορά
 - μετά την είσοδο στην κρίσιμη περιοχή ⇒ όχι blocking
 - μείωση χρόνων blocking
 - το πολύ κατά τον χρόνο μίας κρίσιμης περιοχής

$$B_i = \max_{k,s} (c_{k,s}) \mid k > i, C(s) \leq i$$

k: διεργασίες (χαμηλότερης προτεραιότητας)
s: κοινοί πόροι (με ceiling ≥ από προτεραιότητα διεργασίας i)

- αποφυγή deadlock
 - όχι κυκλικό blocking

Stack Resource Policy (SRP)

- Παρόμοιο με PCP
 - preemption level (στατική παράμετρος)
 - συσχετισμός με προτεραιότητες
 - έλεγχος κατά την εκκίνηση διεργασίας
 - Δυνατή η χρήση δυναμικής χρονοδρομολόγησης
 - Αποφυγή ανεξέλεγκτης αντιστροφής προτεραιοτήτων
 - Αποφυγή deadlock
 - Καλύπτει πολλαπλούς πόρους
 - Επιτρέπει τη χρήση κοινού runtime stack

Σύνοψη

- Αλγόριθμοι (πρωτόκολλα) ελέγχου προσπέλασης κοινών πόρων
 - PIP
 - δεν απαιτεί πρόσθετη πληροφορία (ceilings κλπ.)
 - υποστηρίζεται σε πολλά εμπορικά ΛΣΠΧ
 - δεν αποφεύγει deadlock
 - δεν ελαχιστοποιεί χρόνους blocking
 - PCP & SRP
 - αποφυγή deadlock
 - μείωση χρόνου blocking
 - υπολογισμός παραμέτρων off-line
 - SRP: δυνατότητα χρήσης δυναμικής χρονοδρομολόγησης