

ΑΔΙΕΞΟΔΑ (Deadlocks)

Παράδειγμα 1

- Ένα σύστημα με
 - έναν εκτυπωτή και
 - ένα σαρωτή εγγράφων
- Δύο διεργασίες P1 και P2
- Η P1 δεσμεύει τον εκτυπωτή
- Η P2 δεσμεύει το σαρωτή
- Η P1 ζητά το σαρωτή και εμποδίζεται
- Η P2 ζητά τον εκτυπωτή και εμποδίζεται

→ **Θανάσιμο αγκάλιασμα (deadly embrace)**

ΑΔΙΕΞΟΔΑ

Παράδειγμα 2

Έστω A και B δύο δυαδικοί σηματοφορείς με αρχική τιμή 1

```
process P1;  
{  
    ...  
    wait (A);  
    wait (B);  
    ...  
}
```

```
process P2;  
{  
    ...  
    wait (B);  
    wait (A);  
    ...  
}
```

- Κάθε μία από τις διεργασίες ενός συνόλου περιμένει για ένα γεγονός ή συμβάν (event) που μπορεί να προκληθεί μόνο από μία άλλη διεργασία του συνόλου
- Οι διεργασίες ενός συνόλου εμποδίζουν η μία τη άλλη
- Γεγονός/Συμβάν = (συνήθως) απελευθέρωση πόρου

ΑΔΙΕΞΟΔΑ

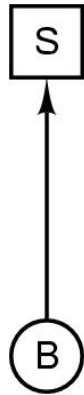
- Ένα σύνολο διεργασιών $P = \{P_1, P_2, \dots, P_n\}$
- Ένα σύνολο τύπων πόρων $R = \{R_1, R_2, \dots, R_m\}$
- Κάθε τύπος πόρων R_i έχει W_i ίδιες μονάδες
 - **Διαδοχικά επαναχρησιμοποιήσιμοι ή μόνιμοι πόροι (serially reusable ή permanent resources)**
 - W_i : σταθερό, πεπερασμένο
 - Διακρίνονται σε:
 - *Πόρους μιας χρήσης (single-use resources)*, π.χ. ΚΜΕ, συσκευές ε/ε
 - *Πόροι πολλαπλής χρήσης (multiple-use resources)*, π.χ. μνήμη, δίσκος
 - **Αναλώσιμοι ή προσωρινοί πόροι (consumable or temporary resources)**
 - W_i : μεταβάλλεται, όχι απαραίτητα πεπερασμένο
 - Μία διεργασία (παραγωγός) δημιουργεί μονάδες
 - Μία διεργασία (καταναλωτής) δεσμεύει μονάδες
 - Οι μονάδες που δεσμεύονται δεν επιστρέφονται
 - Υλισμικό: διακοπές ε/ε και χρονομέτρου
 - Λογισμικό: σηματοφορείς, μηνύματα
- Χρήση πόρου από μία εκτελούμενη διεργασία
 - (Απ)αίτηση (Request)
 - Χρήση (Use)
 - Αποδέσμευση (Release)

ΑΔΙΕΞΟΔΑ

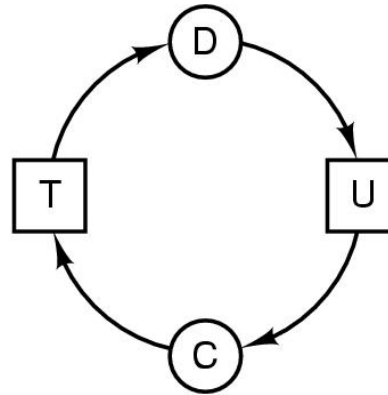
Γράφος πόρων (resource graph/Wait For Graph, WFG)



(a)



(b)



(c)

(a) ο πόρος R έχει εκχωρηθεί (δεσμευθεί) στη (από τη) διεργασία A

(b) Η διεργασία B ζητά τον πόρο S

(c) Αδιέξοδο (διεργασίες C και D, πόροι T και U)

- Καμία από τις διεργασίες δεν μπορεί
 - να συνεχίσει
 - να απελευθερώσει κάποιον πόρο
 - να αφυπνίσει κάποια άλλη διεργασία

ΑΔΙΕΞΟΔΑ

Συνθήκες Coffman

Αναγκαίες συνθήκες για να συμβεί αδιέξοδο

- **Αμοιβαίος αποκλεισμός (mutual exclusion)**
Οι πόροι δεν είναι (κατά)μεριζόμενοι
Ένας πόρος μπορεί σε κάποια χρονική στιγμή να αποκτηθεί από μια διεργασία
Ένας πόρος είτε είναι διαθέσιμος είτε είναι δεσμευμένος
- **Μη διακοπτόμενη χρήση πόρων (non-preemption)**
Ένας πόρος δεν μπορεί να αποσπαστεί από τη διεργασία που τον κατέχει
Μόνο η διεργασία αυτή μπορεί να τον ελευθερώσει
- **Μερική καταχώριση ή δέσμευση και αναμονή (partial allocation or Hold and wait)**
Όλες οι διεργασίες μπορούν να ζητήσουν πόρους
Οι διεργασίες που έχουν αποκτήσει πόρους μπορούν να ζητήσουν νέους πόρους
Οι διεργασίες αποκτούν/ζητούν τους πόρους σταδιακά
- **Κυκλική αναμονή (circular waiting)**
Οι διεργασίες περιμένουν η μία τη άλλη να ελευθερώσει τους πόρους που χρειάζονται

ΑΔΙΕΞΟΔΑ

ΑΝΤΙΜΕΤΩΠΙΣΗ ΑΔΙΕΞΟΔΩΝ

- **Έγκριση αδιεξόδων (deadlock allowance)**
 - Αγνοούμε το πρόβλημα
- **Ανακάλυψη και επανόρθωση αδιεξόδων (deadlock detection and recovery)**
 - Το σύστημα επιτρέπεται να φθάσει σε αδιέξοδο
 - Αλγόριθμος ανακάλυψης
 - Μηχανισμός επανόρθωσης
 - Χρήση γράφου πόρων
- **Αποφυγή αδιεξόδων (deadlock avoidance)**
 - Για να παραχωρηθεί ένας πόρος ένας αλγόριθμος ελέγχει αν η παραχώρηση αυτή μπορεί να οδηγήσει σε αδιέξοδο
 - **Μόνιμοι πόροι**
 - Ελεγχόμενη καταχώριση (controlled allocation)
 - **Αναλώσιμοι πόροι**
 - Ιεραρχικά συστήματα (Hierarchical systems)
- **Πρόληψη αδιεξόδων (deadlock prevention)**

ΔΕΝ ισχύει μία από τις τέσσερις συνθήκες του Coffman

ΑΔΙΕΞΟΔΑ

Έγκριση αδιεξόδων (deadlock allowance)

- Αγνοούμε το πρόβλημα
- Ο χρήστης/χειριστής τα διαπιστώνει και τα επανορθώνει
- Διαγραφή διεργασιών
- Επανεκκίνηση του συστήματος (reboot)
- Συμβιβασμός ανάμεσα
 - στη συχνότητα εμφάνισης και
 - στο κόστος αντιμετώπισης
- Λογική επιλογή αν
 - τα αδιέξοδα συμβαίνουν σπάνια
 - το κόστος αντιμετώπισης τους δεν είναι μεγάλο
- UNIX και WINDOWS ακολουθούν την προσέγγιση αυτή
- Συμβιβασμός ανάμεσα
 - στην ευκολία υλοποίησης και
 - στην ορθότητα του Λ.Σ.

ΑΔΙΕΞΟΔΑ

Ανακάλυψη και επανόρθωση αδιεξόδων (deadlock detection and recovery)

Πόροι

- Διακοπτόμενοι (preemptive), π.χ. μνήμη
- Μη διακοπτόμενοι (non-preemptive), π.χ. εκτυπωτής

Μια μονάδα κάθε τύπου πόρου

- Χρήση του γράφου πόρων
- Κύκλος = αδιέξοδο
 - Αλγόριθμος ανακάλυψης κύκλου σε γράφο n κόμβων
 - n = αριθμός διεργασιών
 - Πολυπλοκότητα $O(n^2)$
 - Περιοδική εκτέλεση

Πολλές μονάδες κάθε τύπου πόρου

- Χρήση του γράφου πόρων
- Κύκλος υποδεικνύει *πιθανό* αδιέξοδο

ΑΔΙΕΞΟΔΑ

Ανακάλυψη και επανόρθωση αδιεξόδων (deadlock detection and recovery)

Επανόρθωση (Recovery)

- **Διαγραφή (killing) διεργασιών**
 - Όλων των διεργασιών που εμπλέκονται στο αδιέξοδο (στον κύκλο)
 - Σταδιακή διαγραφή διεργασιών
 - διαγραφή μιας τυχαίας διεργασίας στον κύκλο
 - επιλογή μιας διεργασίας του κύκλου με κριτήρια
 - απελευθέρωση πόρων ώστε κάποια άλλη/ες διεργασίες να συνεχίσει/ουν την εκτέλεσή της/τους
 - ελαχιστοποίηση χρήσης πόρων από τη διεργασία μέχρι τη διαγραφή της
- **Οπισθοδρόμηση (Rollback)**
 - Περιοδική αποθήκευση σε αρχεία (κατάσταση διεργασίας + πόροι) σε κάθε σημείο ελέγχου (*checkpoint*)
 - Αριθμός σημείων ελέγχου
 - Επιλογή διεργασίας που κατέχει κάποιον πόρο
 - Οπισθοδρόμησή της σ' ένα σημείο ελέγχου προτού αποκτήσει τον πόρο
 - Ο πόρος μπορεί τώρα να δοθεί σε άλλη διεργασία

ΑΔΙΕΞΟΔΑ

Ανακάλυψη και επανόρθωση αδιεξόδων

- Διακοπή χρήσης πόρων (Preemption)
 - Απόσπαση ενός πόρου από μια διεργασία και παραχώρηση του σε κάποια άλλη

ΠΡΟΒΛΗΜΑΤΑ

- Χρόνος
 - Ανακάλυψη αδιεξόδου
 - Επανόρθωση αδιεξόδου
- Οι πόροι που θα ζητηθούν δεν είναι γενικά γνωστοί
- Οι πόροι δε ζητούνται όλοι μαζί
- Πιθανότητα *υποσιτισμού* (*starvation*)

ΑΔΙΕΞΟΔΑ

Αποφυγή αδιεξόδων (deadlock avoidance)

- Ανακάλυψη αδιεξόδων: υποθέσαμε ότι μια διεργασία ζητά όλους τους πόρους μαζί
- ΑΛΛΑ οι πόροι μπορεί να ζητούνται σταδιακά

ΙΔΕΑ ΑΠΟΦΥΓΗΣ: ένας πόρος παραχωρείται μόνο αν η παραχώρησή του είναι ασφαλής (δεν πρόκειται να οδηγήσει σε αδιέξοδο)

ΑΣΦΑΛΗΣ ΚΑΤΑΣΤΑΣΗ

- Δεν υπάρχει αδιέξοδο
- Υπάρχει τρόπος να ικανοποιηθούν όλες οι εκκρεμείς απαιτήσεις χωρίς να οδηγηθούμε σε αδιέξοδο

ΑΔΙΕΞΟΔΑ

Αποφυγή αδιεξόδων

Πρόβλημα του τραπεζίτη (banker's problem) [Dijkstra]

- Τραπεζίτης = Λ.Σ.
- Πελάτες = Διεργασίες
- Χρήματα = Πόροι
- Δάνεια = Απαιτήσεις (η συνολική απαίτηση ικανοποιείται σταδιακά)
- Ασφαλής κατάσταση = ο τραπεζίτης μπορεί να εξυπηρετήσει τη *συνολική απαίτηση ενός τουλάχιστον πελάτη*

Αλγόριθμος τραπεζίτη (γενικά)

εξέτασε κάθε (απ)αίτηση όταν αυτή συμβαίνει;
εξέτασε αν η ικανοποίησή της οδηγεί σε ασφαλή κατάσταση;
αν ναι τότε ικανοποίησε την (απ)αίτηση
αλλιώς ανάβαλε την ικανοποίησή της;

ΑΔΙΕΞΟΔΑ

Αποφυγή αδιεξόδων

ΠΡΟΒΛΗΜΑΤΑ

- Αλγόριθμοι μέγιστης απαίτησης (maximum claim algorithms)
- Συντηρητικοί (conservative) αλγόριθμοι
- Σε μια συγκεκριμένη κατάσταση ελέγχουμε τη χειρότερη περίπτωση, δηλαδή ότι
 - Κάποια στιγμή μπορεί όλες οι διεργασίες να ζητήσουν ταυτόχρονα όλες τις μονάδες που χρειάζονται ακόμη
 - Καμιά μονάδα ενός δεσμευμένου πόρου δε θα έχει απελευθερωθεί μέχρι τότε
- Δέσμευση πόρων χωρίς λόγο
- Χρονοβόρα εκτέλεση
- Οι απαιτήσεις των διεργασιών δεν είναι προκαταβολικά γνωστές

ΑΔΙΕΞΟΔΑ

Πρόληψη αδιεξόδων (deadlock prevention)

Μία από τις τέσσερις συνθήκες Coffman ΔΕΝ ισχύει

1. ΟΧΙ Αμοιβαίος αποκλεισμός

Παράδειγμα : αρχεία ανάγνωσης μόνο (read only files)

- Πολλές διεργασίες μπορούν αποκτήσουν ταυτόχρονη πρόσβαση σ' αυτά

ΑΛΛΑ γενικά

- Δεν είναι δυνατό όλοι οι πόροι να καταμερίζονται π.χ.
 - εκτυπωτής
 - αρχείο που ενημερώνεται
- Ένας διαχειριστής/παρακολουθητής ανά πόρο
 - Πιθανότητα αυτοί να εμποδίσουν ο ένας τον άλλον

2. Διακοπτόμενη χρήση πόρων

- Ένας πόρος αποσπάται από τη διεργασία που τον κατέχει
- Διακοπτόμενοι (preemptive) πόροι
 - ΚΜΕ: εύκολα
 - Μνήμη: σχετικά δύσκολα, πρόβλημα αλωνίσματος (thrashing)

ΑΛΛΑ

- Οι περισσότεροι πόροι είναι ΜΗ διακοπτόμενοι
 - π.χ. εκτυπωτής, αρχείο που ενημερώνεται
- Σπατάλη χρόνου (διακοπή - επαναφορά)

ΑΔΙΕΞΟΔΑ

Πρόληψη αδιεξόδων

Μία από τις τέσσερις συνθήκες Coffman ΔΕΝ ισχύει

3. ΟΧΙ Μερική καταχώριση πόρων

- 1) Μία ΜΟΝΟ διεργασία αποκτά πόρους → όχι ταυτόχρονες διεργασίες!
- 2) Προκαταβολική δέσμευση ΟΛΩΝ των πόρων που πιθανόν θα χρειαστεί μια διεργασία κατά την ενεργοποίησή της
 - αν οι πόροι είναι διαθέσιμοι η διεργασία τους αποκτά και συνεχίζει, αλλιώς περιμένει
 - μια διεργασία ΕΙΤΕ κατέχει πόρους ΕΙΤΕ περιμένει
 - π.χ. συσκευές ε/ε στο σύστημα IBM OS/360

ΑΛΛΑ

- Οι πόροι που θα ζητηθούν ΔΕΝ είναι γνωστοί προκαταβολικά
- Οι διεργασίες περιμένουν μέχρι να είναι διαθέσιμοι όλοι οι πόροι που πιθανώς θα χρησιμοποιήσουν
- χαμηλή χρήση πόρων (resource utilization)
- πιθανός υποσιτισμός

ΑΔΙΕΞΟΔΑ

Πρόληψη αδιεξόδων

Μία από τις τέσσερις συνθήκες Coffman ΔΕΝ ισχύει

4. ΟΧΙ Κυκλική αναμονή

Διατεταγμένη καταχώρηση πόρων (ordered resource allocation)

- Οι τύποι των πόρων διατάσσονται σε τάξεις $R_1, R_2, R_3, \dots, R_m$
- Μία διεργασία μπορεί να απαιτήσει πόρους μόνο σε αύξουσα σειρά της διάταξης
 - αν έχει δεσμεύσει πόρους τάξης R_i
 - μπορεί να απαιτήσει μόνο πόρους τάξεων $R_j > R_i$
- Όλες οι μονάδες ίδιου τύπου πόρου ζητούνται μαζί
 - Έστω π ο "μεγαλύτερος" δεσμευμένος πόρος και έστω δ η διεργασία η οποία τον δεσμεύει
 - Η διεργασία δεν μπορεί να ζητήσει κάποιο πόρο $\pi' \leq \pi$
 - Η διεργασία είτε θα τελειώσει και θα ελευθερώσει τον π , είτε θα δεσμεύσει κάποιο πόρο $\pi'' > \pi$
 - Η διεργασία (κάποτε) θα τελειώσει και θα ελευθερώσει τους πόρους που έχει δεσμεύσει
 - κ.ο.κ.

ΘΕΩΡΗΜΑ

Δεν μπορεί να συμβεί αδιέξοδο σ' ένα σύστημα που χρησιμοποιεί διατεταγμένη καταχώρηση πόρων (τυπική απόδειξη με επαγωγή στη σειρά διάταξης).

ΑΔΙΕΞΟΔΑ

ΣΥΜΠΕΡΑΣΜΑΤΑ

- **Αντιμετώπιση αδιεξόδων**
 - Έγκριση αδιεξόδων
 - Ανακάλυψη και επανόρθωση αδιεξόδων
 - Αποφυγή αδιεξόδων
 - Πρόληψη αδιεξόδων
- Προβλήματα σε όλες τις περιπτώσεις
- **UNIX, Windows**
 - Έγκριση αδιεξόδων
 - Πρόληψη αδιεξόδων
 - διακοπτόμενη χρήση ΚΜΕ και μνήμης