ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

Λειτουργικά Συστήματα

Αδιέξοδα / Deadlocks

Σκοποί ενότητας

- Κατανόηση της έννοιας των πόρων και του προβλήματος των αδιεξόδων κατά τη χρήση των πόρων
- Εξοικείωση με τις βασικές τεχνικές διαχείρισης αδιεξόδων: αγνόηση, εντοπισμός και ανάκαμψη, αποφυγή, αποτροπή
- Κατανόηση των πρακτικών προβλημάτων στη διαχείριση αδιεξόδων

Περιεχόμενα ενότητας

- Πόροι
- Αδιέξοδα
- Αντιμετώπιση αδιεξόδων
- Εντοπισμός και ανάκαμψη
- Αποφυγή αδιεξόδων
- Αποτροπή αδιεξόδων
- Άλλα θέματα

Πόροι / Resources

Resources (Πόροι)

- Πόρος (resource)
  - Γενικός όρος για οντότητες που δεσμεύουμε
  - Πόροι υλικού (εκτυπωτής) ή πληροφοριών (βάση)
  - Μπορεί να έχουμε πολλούς πόρους ενός τύπου
- Shareable resources: Ορισμένοι πόροι του ΛΣ δεν είναι κοινόχρηστοι
  - Παράδειγμα: εκτυπωτής
- Resource reservation and Deadlocks
  - Η **δέσμευση πόρων** *μπορεί* να οδηγήσει σε **αδιέξοδο** (όχι απαραίτητα!)
  - Το *process A* δεσμεύει τον *σαρωτή (scanner)* και ζητά τον *εκτυπωτή*
  - Το *process B* δεσμεύει τον *εκτυπωτή* και ζητά τον *σαρωτή*
- Τα αδιέξοδα είναι πιο γενικό πρόβλημα
  - Με πολλές διεργασίες (κυκλική αναμονή)
  - Με βάσεις δεδομένων (κλειδώματα εγγραφών)

Resource preemption (προεκτόπιση)

- Preemptive resources (προεκτοπίσιμοι πόροι)
  - Αποσπώνται εύκολα
  - Παράδειγμα: μνήμη
    - Η διεργασία Α δεσμεύει όλη τη μνήμη και τον εκτυπωτή
    - Η Β δεσμεύει όλη τη μνήμη αλλά όχι τον εκτυπωτή
    - Δεν έχουμε αδιέξοδο: παίρνουμε τη μνήμη της Β
- Non-preemptive (μη προεκτοπίσιμοι)
  - Δεν αποσπώνται εύκολα
  - Παράδειγμα: εκτυπωτής
- Τα αδιέξοδα αφορούν μη προεκτοπίσιμους πόρους

Allocating resources

- Ο τρόπος δέσμευσης εξαρτάται από το ΛΣ
  - Δέσμευση πόρου με κλήση συστήματος request
  - Δέσμευση πόρου με άνοιγμα αρχείου συσκευής

- Τι γίνεται όταν ένας πόρος δεν είναι διαθέσιμος;
  - Είτε η διεργασία μπλοκάρει και περιμένει
  - Είτε επιστρέφει σφάλμα και ξαναπροσπαθεί

Απόκτηση πόρων [1/2]

```
typedef int semaphore;
semaphore resource_1;

void process_A(void) {
    down(&resource_1);
    use_resource_1();
    up(&resource_1);
}
```

(α)

```
typedef int semaphore;
semaphore resource_1;
semaphore resource_2;

void process_A(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources();
    up(&resource_2);
    up(&resource_1);
}
```

(β)

- Δέσμευση επιπέδου χρήστη (π.χ., για βάσεις)
  - Δυαδική σημαφόρος με αρχική τιμή 1 (α)
  - Εναλλακτικά, μεταβλητή τύπου mutex
- Λειτουργεί και με δύο ή περισσότερους πόρους (β)
  - Αποδέσμευση αντίστροφα από τη δέσμευση

Απόκτηση πόρων [2/2]

```
typedef int semaphore;
semaphore resource_1;
semaphore resource_2;

void process_A(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources();
    up(&resource_2);
    up(&resource_1);
}

void process_B(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources();
    up(&resource_2);
    up(&resource_1);
}
```

(α)

```
typedef int semaphore;
semaphore resource_1;
semaphore resource_2;

void process_A(void) {
    down(&resource_1);
    down(&resource_2);
    use_both_resources();
    up(&resource_2);
    up(&resource_1);
}

void process_B(void) {
    down(&resource_2);
    down(&resource_1);
    use_both_resources();
    up(&resource_1);
    up(&resource_2);
}
```

(β)

- Πρέπει όλοι να συμφωνούν στη σειρά (α)
- Αλλιώς, έχουμε κίνδυνο αδιεξόδου (β)

Αδιέξοδα / Deadlocks

Deadlocks

Ορισμός αδιεξόδου [1/2]

Τυπικός ορισμός αδιεξόδου

- Έστω ένα σύνολο μπλοκαρισμένων διεργασιών
- Όλες οι διεργασίες περιμένουν κάποιο συμβάν
- Τα συμβάντα αυτά παράγονται μόνο εσωτερικά
  - Αφού όλες περιμένουν κάποια άλλη του συνόλου…
  - ..καμία δεν μπορεί να προχωρήσει!
- Με ορισμένες υποθέσεις
  - Υποθέτουμε ένα νήμα ανά διεργασία
  - Υποθέτουμε ότι δεν αφυπνίζονται από διακοπές (interrupts)

Ορισμός αδιεξόδου [2/2]

Αδιέξοδο πόρων

- Σύνολο μπλοκαρισμένων διεργασιών
- Περιμένουν να απελευθερωθεί κάποιος πόρος
- Οι πόροι κατέχονται από αυτές τις διεργασίες
- Ανεξάρτητα από το πλήθος και είδος των πόρων
- Η πιο συνηθισμένη μορφή αδιεξόδου
  - Όχι όμως και η μοναδική

Συνθήκες του Coffman

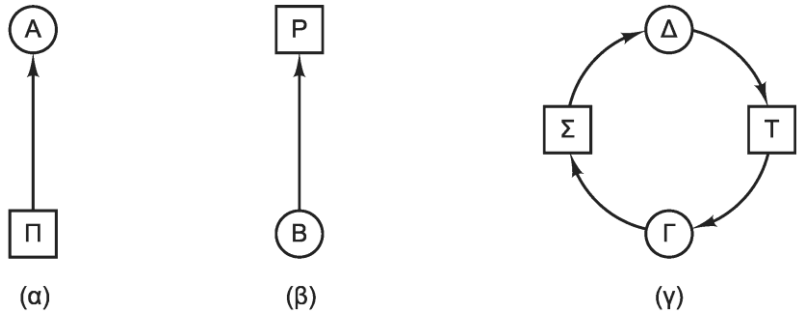
Συνθήκες του Coffman: Πρέπει να ισχύουν **όλες** για αδιέξοδο πόρων

1. **Mutual Exclusion (Αμοιβαίος αποκλεισμός)**
   - Non-shareable resource
   - Κάθε πόρος είναι (α) εκχωρημένος σε μία διεργασία, ή (β) διαθέσιμος
2. **Hold and wait (Δέσμευση και αναμονή)**
   - Μια διεργασία με εκχωρημένους πόρους μπορεί να ζητάει κι άλλους
   - Δεν ζητάει υποχρεωτικά όλους τους πόρους μαζί
3. **Non-preemption (Μη προεκτόπιση)**
   - Δεν μπορούμε να αφαιρέσουμε εκχωρημένους πόρους
4. **Circular dependency (Κυκλική εξάρτηση)**
   - Κυκλική αλυσίδα διεργασιών που περιμένουν η μία την άλλη

Κάθε συνθήκη εκφράζει μία πολιτική συστήματος

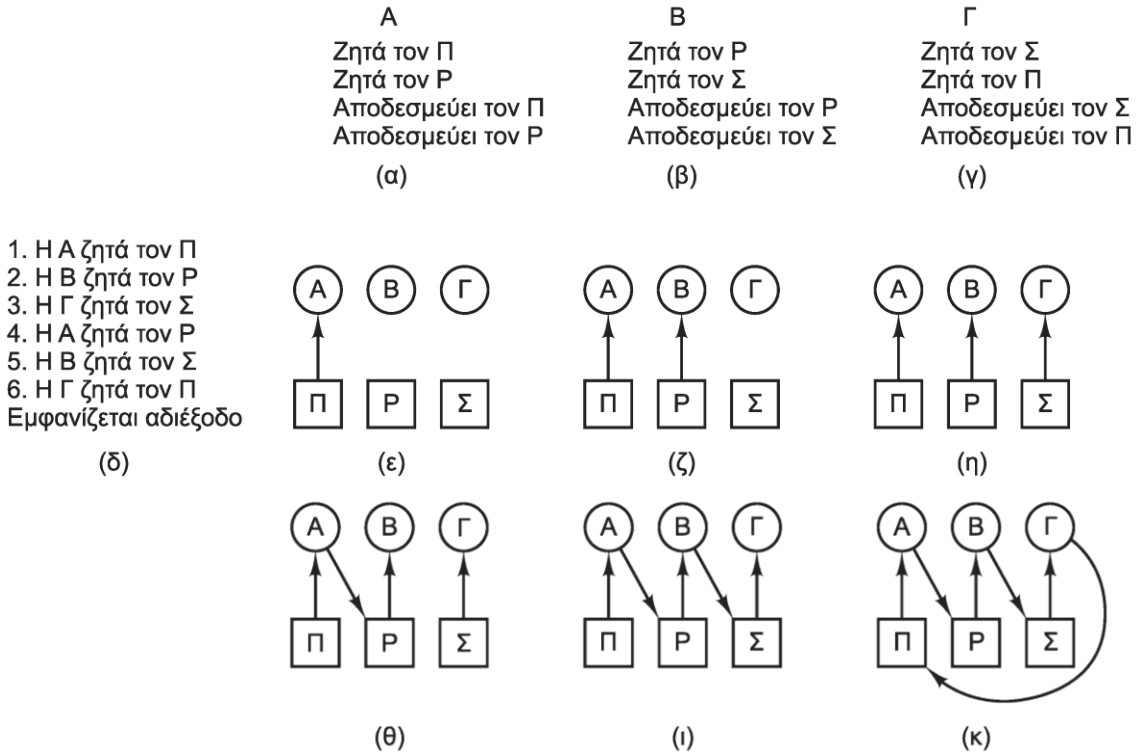
- Αν έστω και **μία** από αυτές δεν ισχύει, είναι **αδύνατον να έχουμε deadlock**

Μοντελοποίηση αδιεξόδων [1/3]



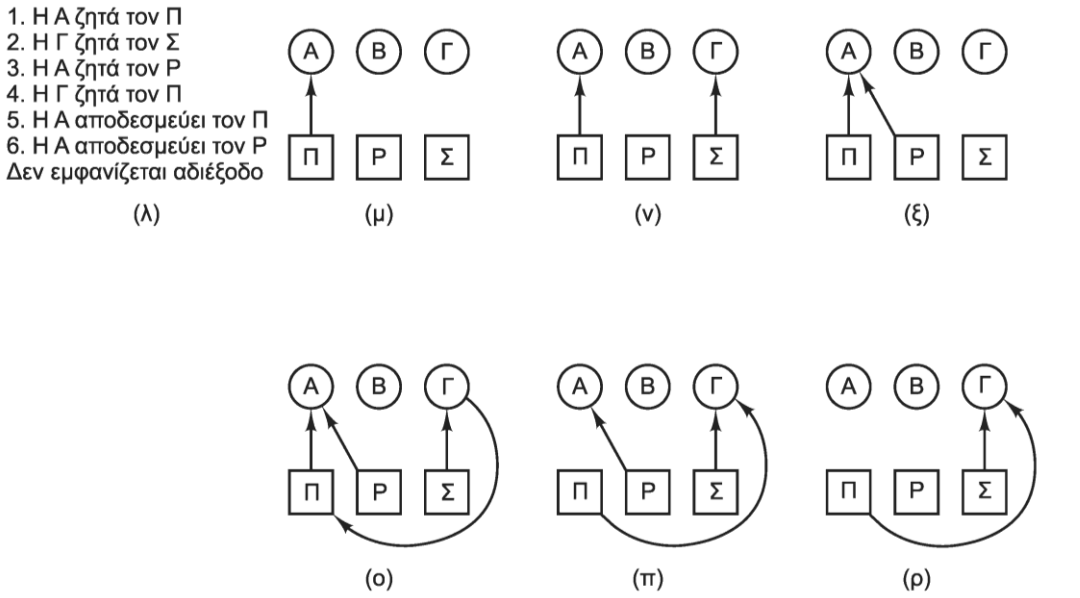
- Μοντελοποίηση αδιεξόδων με γράφους
  - Processes: κύκλοι
  - Resources: τετράγωνα
  - Resource → Process: κατοχή πόρου
  - Process → Resource: αίτηση για πόρο
- **Κύκλος** στον γράφο …σημαίνει **αδιέξοδο**!
  - Παράδειγμα:
    - Η διεργασία Δ κατέχει τον πόρο Σ και ζητάει τον πόρο Τ
    - Η διεργασία Γ κατέχει τον πόρο Τ και ζητάει τον πόρο Σ

Μοντελοποίηση αδιεξόδων [2/3]



- Παράδειγμα χρήσης γράφου διεργασιών και πόρων
  - Αν εκτελεστούν ακολουθιακά, δεν έχουμε πρόβλημα
  - Αν εκτελεστούν εκ περιτροπής όμως, έχουμε πρόβλημα

Μοντελοποίηση αδιεξόδων [3/3]



- Παράδειγμα χρήσης γράφου διεργασιών και πόρων
  - Το ΛΣ θα μπορούσε να εκτελέσει εκ περιτροπής τις Α και Γ
  - Η Β θα εκτελεστεί σε επόμενο στάδιο
  - Το αδιέξοδο λοιπόν εξαρτάται από την εκτέλεση!

Αντιμετώπιση αδιεξόδων

Αγνόηση αδιεξόδων [1/2]

Τέσσερις γενικές στρατηγικές αντιμετώπισης

1. Αγνοούμε το πρόβλημα
   - Ελπίζουμε ότι δεν θα συμβούν αδιέξοδα
2. Εντοπίζουμε τα αδιέξοδα και ανακάμπτουμε
   - Απαιτεί ακύρωση ενεργειών και σπατάλη πόρων
3. Αποφεύγουμε (δυναμικά) τα αδιέξοδα
   - Απαιτεί προσεκτική κατανομή των πόρων
4. Αποτρέπουμε (δομικά) τα αδιέξοδα
   - Άρση <u>οποιασδήποτε</u> συνθήκης του Coffman

Αγνόηση αδιεξόδων [2/2]

Ο αλγόριθμος της στρουθοκαμήλου(!)

- Απλά …αγνοούμε το πρόβλημα των αδιεξόδων!
- Θεωρητικά, απαράδεκτη προσέγγιση
- Ρεαλιστικά, έχουμε έναν συμβιβασμό:
  - Πόσο συχνά συμβαίνουν αδιέξοδα;
  - Πόσο κοστίζει στο σύστημα η αντιμετώπιση;
- Πολλά συστήματα θέλουν τακτική επανεκκίνηση
  - Σφάλματα ΛΣ, αστοχίες υλικού, σφάλματα εφαρμογών
  - Δεν έχει νόημα να ασχολούμαστε ειδικά με τα αδιέξοδα

Detection and Recovery
(Εντοπισμός και ανάκαμψη)

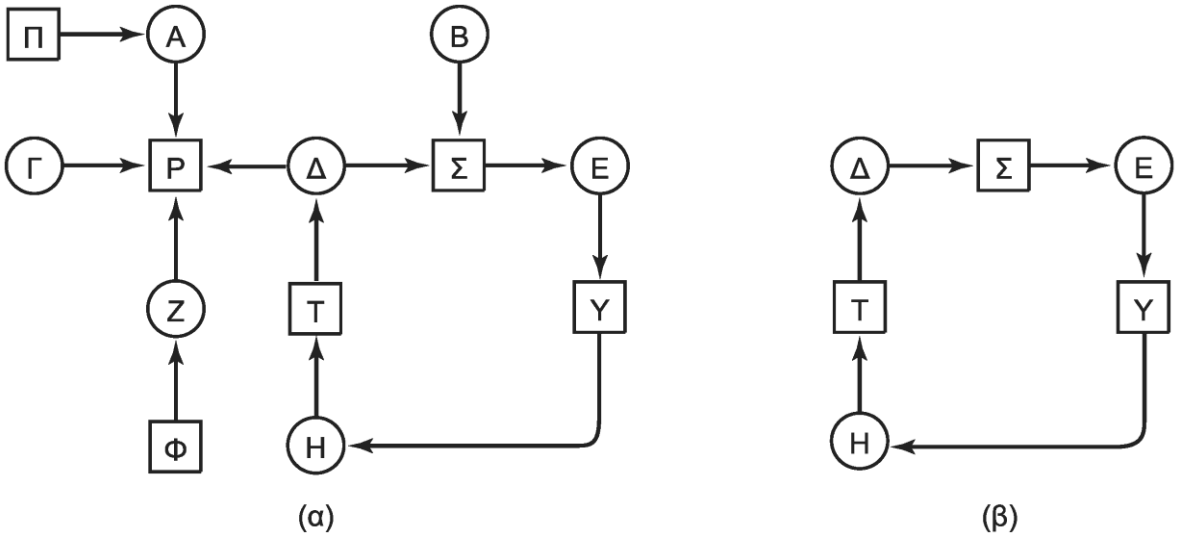
Detection and Recovery

- No prevention!
  - Let deadlocks happen
  - …but **detect** them
  - …and take action to **recover**!

- Δεν χρειάζεται αλλαγές
  - Οι εφαρμογές δεν έχουν περιορισμούς
  - Δεν αλλάζουμε τις κλήσεις συστήματος
  - Απλά προσθέτουμε λειτουργικότητα στο ΛΣ
    - Ελέγχει περιοδικά να εντοπίσει αδιέξοδα

- Τι λειτουργικότητα απαιτείται;
  - **Detection:** Μέθοδος εντοπισμού αδιεξόδων
  - **Recovery:** Μέθοδος ανάκαμψης από αδιέξοδα

Detection: One resource per type [1/2]



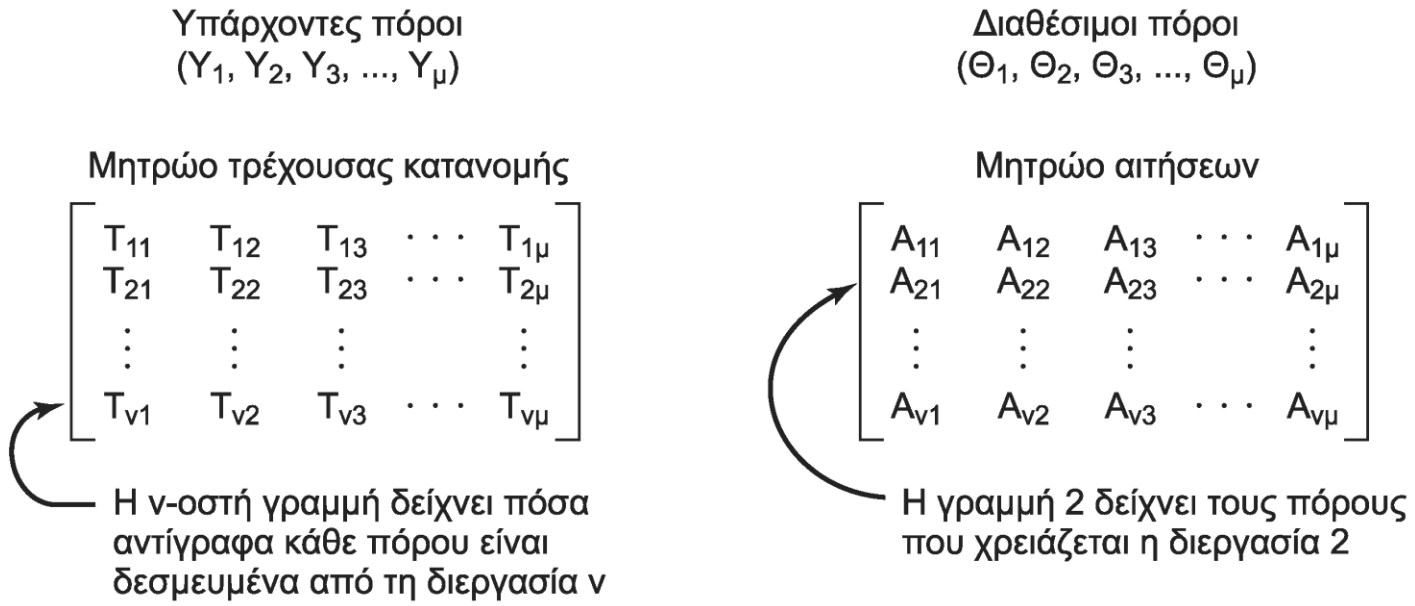
- Εντοπισμός αδιεξόδων με έναν πόρο ανά είδος
  - Παράδειγμα: 7 διεργασίες (Α-Η) και 6 πόροι (Π-Φ)
    - Η Α κατέχει τον Π και ζητάει τον Ρ, η Β ζητάει τον Σ
    - Η Γ ζητάει τον Ρ, η Δ κατέχει τον Τ και ζητάει τους Ρ και Σ
    - Η Ε κατέχει τον Σ και ζητάει τον Υ, η Ζ κατέχει τον Φ και ζητάει τον Ρ
    - Η Η κατέχει τον Υ και ζητάει τον Τ
    - Κατασκευάζουμε τον γράφο (α) και εντοπίζουμε τον κύκλο (β)

Detection: One resource per type [2/2]

- Για *κάθε* κόμβο Κ εκτελούμε τα επόμενα βήματα από τον Κ

1. Αρχικά Λ = κενή λίστα και όλα τα τόξα είναι ασημείωτα
2. Προσθέτουμε τον τρέχοντα κόμβο στο τέλος της Λ
3. Αν ο κόμβος εμφανίζεται δύο φορές στη Λ, η Λ περιέχει κύκλο
4. Αν δεν υπάρχουν ασημείωτα τόξα από τον κόμβο πάμε στο 7
5. Επιλέγουμε και σημειώνουμε ένα ασημείωτο εξερχόμενο τόξο
6. Χρησιμοποιούμε τον νέο κόμβο ως τρέχοντα και πάμε στο 2
7. Αν ο τρέχων κόμβος είναι ο αρχικός, τέλος χωρίς κύκλο
8. Αλλιώς, αφαιρούμε τον κόμβο από τη λίστα
9. Κάνουμε τον προηγούμενο κόμβο τρέχοντα και πάμε στο 2

Detection: Many res. per type [1/4]



- Εντοπισμός αδιεξόδων με πολλούς πόρους ανά είδος
  - Το **διάνυσμα υπαρχόντων Υ** δείχνει πόσοι **υπάρχουν**
  - Το **διάνυσμα διαθέσιμων Θ** δείχνει πόσοι είναι **διαθέσιμοι**
  - Το **μητρώο κατανομής Τ** δείχνει πού **εκχωρήθηκαν**
  - Το **μητρώο αιτήσεων Α** δείχνει **πόσους <u>ακόμα</u> πόρους** χρειάζεται η κάθε διεργασία, πέρα από όσους ήδη έχει

Detection: Many res. per type [2/4]

- Βασική συνθήκη πόρων
  - Κατανεμημένοι (εκχωρημένοι) + διαθέσιμοι = υπάρχοντες
  - Οι αιτήσεις δείχνουν πόσοι ακόμη χρειάζονται

$$\sum_{k=1}^{n} T_{kl} + \Theta_l = Y_l$$

- Υποθέσεις αλγόριθμου εντοπισμού αδιεξόδων
  - A<=B αν όλα τα στοιχεία του A <= των αντίστοιχων του B
  - Κάθε διεργασία διατηρεί τους πόρους της μέχρι τέλους
  - Αρχικά όλες οι διεργασίες είναι ασημείωτες
  - Όσες παραμείνουν ασημείωτες βρίσκονται σε αδιέξοδο

Detection: Many res. per type [3/4]

- Αλγόριθμος εντοπισμού αδιεξόδων
  1. Ψάξε μία ασημείωτη διεργασία k με A_k<=Θ
  2. Αν βρεθεί τέτοια, πρόσθεσε το T_k στο Θ
  3. Μετά σημείωσε τη διεργασία και πήγαινε στο 1
  4. Αν δεν υπάρχει, ο αλγόριθμος τερματίζεται

- Ψάχνουμε μια διεργασία που μπορεί να ολοκληρωθεί
  - Θεωρούμε ότι δεσμεύει όλους τους πόρους που θέλει
  - Μετά εκτελείται ως το τέλος και τους απελευθερώνει
  - Άρα ψάχνουμε μια ασφαλή ακολουθία εκτέλεσης

Detection: Many res. per type [4/4]

- Παράδειγμα αλγόριθμου εντοπισμού αδιεξόδων
  - Πρώτα μπορεί να εκτελεστεί η διεργασία 3
    - Υπόλοιπο πόρων (2,2,2,0)
  - Μετά μπορεί να εκτελεστεί η διεργασία 2
    - Υπόλοιπο πόρων (4,2,2,1)
  - Τέλος μπορεί να εκτελεστεί η διεργασία 1

Detection: When?

Πότε ελέγχουμε για αδιέξοδα;

- Είτε όποτε ζητείται κάποιος πόρος
  - Μεγάλη επιβάρυνση
- Είτε περιοδικά
- Είτε όταν η απόδοση πέσει πολύ
  - Μπορεί να οφείλεται σε αδιέξοδο

Recovery by Preemtpion [1/3]

Ανάκαμψη μέσω προεκτόπισης (preemption)

- Παράδειγμα: αδιέξοδο που εμπλέκει τον εκτυπωτή
  - Διακόπτουμε τη διεργασία που έχει τον εκτυπωτή
  - Μαζεύουμε την έξοδό της και παραχωρούμε τον εκτυπωτή
  - Όταν τελειώσει, ενεργοποιούμε ξανά την αρχική διεργασία
- Γενικά αυτό δεν είναι πάντα εφικτό
  - Παράδειγμα: εγγραφέας DVD-R που γράφει ένα DVD-R
  - Αν διακοπεί, το DVD-R πάει για πέταμα

Recovery by Rollback [2/3]

Ανάκαμψη μέσω ανασκευής (rollback)

- Περιοδική δημιουργία σημείων ελέγχου
  - Λέγονται και checkpoints
  - Εγγραφή κατάστασης διεργασίας σε αρχείο
  - Περιλαμβάνει εικόνα μνήμης και πόρους
- Εντοπίζουμε τη διεργασία που εμπλέκεται
- Εντοπίζουμε τον πόρο που έχει πρόβλημα
- Επανεκκίνηση από προηγούμενο σημείο ελέγχου
  - Πρέπει να μην έχει δεσμευθεί ακόμα ο πόρος

Recovery by Killing Processes [3/3]

Ανάκαμψη μέσω εξάλειψης διεργασιών (killing processes)

- Σκοτώνουμε μία διεργασία του κύκλου
  - Ελπίζουμε ότι ο κύκλος θα σπάσει
- Σκοτώνουμε μία διεργασία με κατάλληλους πόρους
  - Αυτούς που ζητούνται στον κύκλο
- Ο στόχος είναι να σπάσει ο κύκλος αναμονής
- Προτιμάμε διεργασίες που τερματίζονται εύκολα
  - Κατάλληλη: απλό εκτελέσιμο
  - Ακατάλληλη: βάση δεδομένων

Deadlock Avoidance
(Αποφυγή αδιεξόδων)

Γιατί αποφυγή αδιεξόδων;

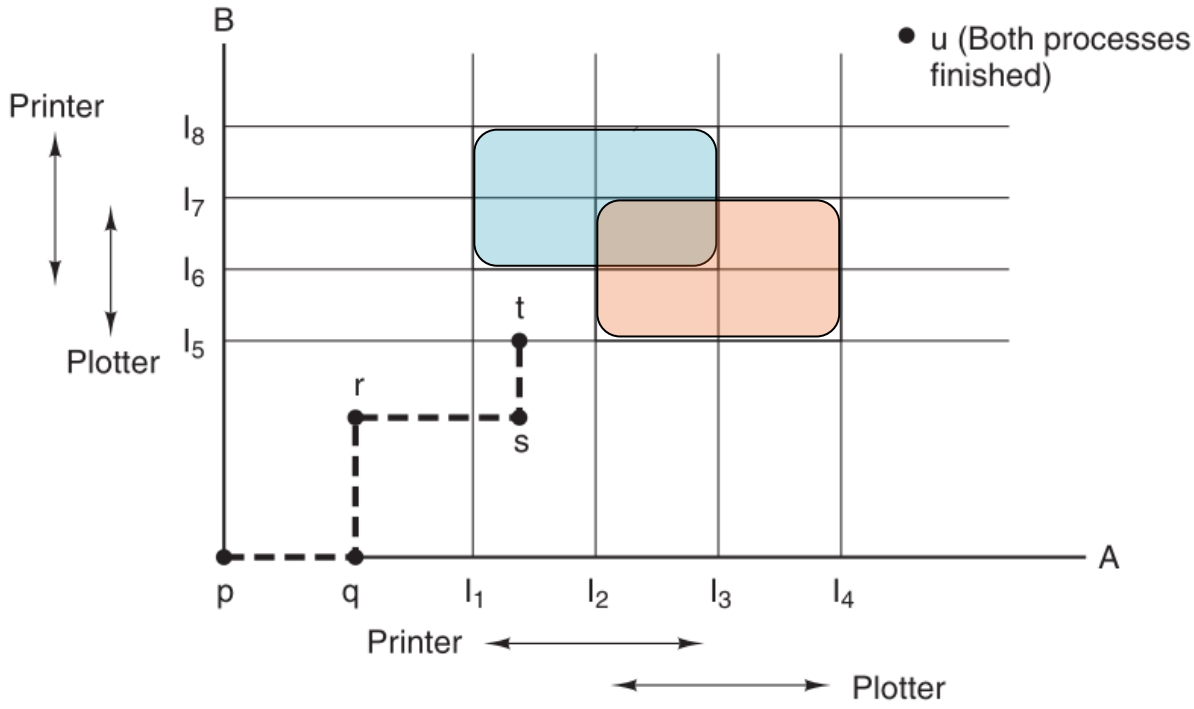
- Ο εντοπισμός (detection) μπορεί να αργήσει
  - Μέχρι τότε, το αδιέξοδο σπαταλάει πόρους
- Η ανάκαμψη (recovery) έχει μεγάλο κόστος
  - Γενικά χάνεται κάποια δουλειά
    - Είτε από την αρχή μιας διεργασίας
    - Είτε από ένα σημείο ανάκαμψης (rollback)
- Μπορούμε να αποφύγουμε τα αδιέξοδα;

Resource trajectories [1/2]

Τροχιές πόρων

- Έστω ότι έχουμε δύο διεργασίες (Α και Β) και δύο πόρους
- Κάθε διεργασία χρειάζεται τους πόρους για κάποιο διάστημα
- Άξονες
  - άξονας Χ: εξέλιξη της διεργασίας Α στον χρόνο
  - άξονας Υ: εξέλιξη της διεργασίας Β στον χρόνο
- Η τεθλασμένη δείχνει την εξέλιξη των διεργασιών
- Κίνηση μόνο προς τα πάνω ή τα δεξιά

Resource trajectories [2/2]



- Δεν μπορούμε να μπούμε στις σκιασμένες περιοχές
  - Αν το σύστημα μπει στην (I_1,I_5)-(I_2,I_6) έχουμε πρόβλημα
  - Από το t θα πρέπει να κινηθούμε δεξιά για αποφυγή
  - Άρα στο t δεν πρέπει να δώσουμε τον πόρο στην B

Ασφαλείς καταστάσεις [2/2]

	Κατέχει	Μέγιστο
A	3	9
B	2	4
Γ	2	7

Ελεύθεροι: 3
(α)

	Κατέχει	Μέγιστο
A	3	9
B	4	4
Γ	2	7

Ελεύθεροι: 1
(β)

	Κατέχει	Μέγιστο
A	3	9
B	0	–
Γ	2	7

Ελεύθεροι: 5
(γ)

	Κατέχει	Μέγιστο
A	3	9
B	0	–
Γ	7	7

Ελεύθεροι: 0
(δ)

	Κατέχει	Μέγιστο
A	3	9
B	0	–
Γ	0	–

Ελεύθεροι: 7
(ε)

- Ασφαλείς και ανασφαλείς καταστάσεις
  - Σχεδόν ίδια κατάσταση με τον αλγόριθμο εντοπισμού
    - Διανύσματα υπαρχόντων Υ και διαθέσιμων Θ
    - Μητρώα κατανομής Τ και αιτήσεων Α (εδώ είναι το μέγιστο)
  - Μία κατάσταση είναι ασφαλής (safe) εάν
    - Δεν είμαστε σε αδιέξοδο
    - Υπάρχει κάποια σειρά ολοκλήρωσης των διεργασιών
  - Παράδειγμα με έναν πόρο και τρεις διεργασίες
    - Αρχικά εκτελείται η Β και μετά η Γ μέχρι τέλους

Ασφαλείς καταστάσεις [2/2]

	Κατέχει	Μέγιστο
A	3	9
B	2	4
Γ	2	7

Ελεύθεροι: 3
(α)

	Κατέχει	Μέγιστο
A	4	9
B	2	4
Γ	2	7

Ελεύθεροι: 2
(β)

	Κατέχει	Μέγιστο
A	4	9
B	4	4
Γ	2	7

Ελεύθεροι: 0
(γ)

	Κατέχει	Μέγιστο
A	4	9
B	–	–
Γ	2	7

Ελεύθεροι: 4
(δ)

- Παράδειγμα με έναν πόρο και τρεις διεργασίες
  - Έστω ότι η Α ζητάει ένα ακόμη αντίγραφο του πόρου
  - Αρχικά εκτελείται η Β μέχρι τέλους
  - Μετά όμως δεν μπορεί να εκτελεστεί ούτε η Α ούτε η Γ
  - Άρα η κατάσταση δεν είναι ασφαλής
- Προσοχή: ανασφαλής κατάσταση <> αδιέξοδο
  - Οι διεργασίες δεν ζητάνε απαραίτητα όλους τους πόρους
  - Απλά δεν είναι εγγυημένο ότι δεν θα έχουμε αδιέξοδο

Αλγόριθμος τραπεζίτη (1 από 3)

	Μέγιστο	Κατέχει
Α	0	6
Β	0	5
Γ	0	4
Δ	0	7

Ελεύθεροι: 10

(α)

	Μέγιστο	Κατέχει
Α	1	6
Β	1	5
Γ	2	4
Δ	4	7

Ελεύθεροι: 2

(β)

	Μέγιστο	Κατέχει
Α	1	6
Β	2	5
Γ	2	4
Δ	4	7

Ελεύθεροι: 1

(γ)

- Ο αλγόριθμος του τραπεζίτη για έναν πόρο
  - Επέκταση του αλγορίθμου εντοπισμού (Dijkstra)
    - Πριν δώσουμε πόρο, βλέπουμε αν πάμε σε ασφαλή κατάσταση
    - Αν όχι, τότε δεν παραχωρούμε τον πόρο
  - Παράδειγμα με έναν πόρο
    - (α) αρχική κατάσταση με μέγιστες απαιτήσεις
    - (β) ασφαλής κατάσταση (σειρά ολοκλήρωσης Γ, Δ, Β, Α)
    - (γ) ανασφαλής κατάσταση (δεν υπάρχει σειρά ολοκλήρωσης)

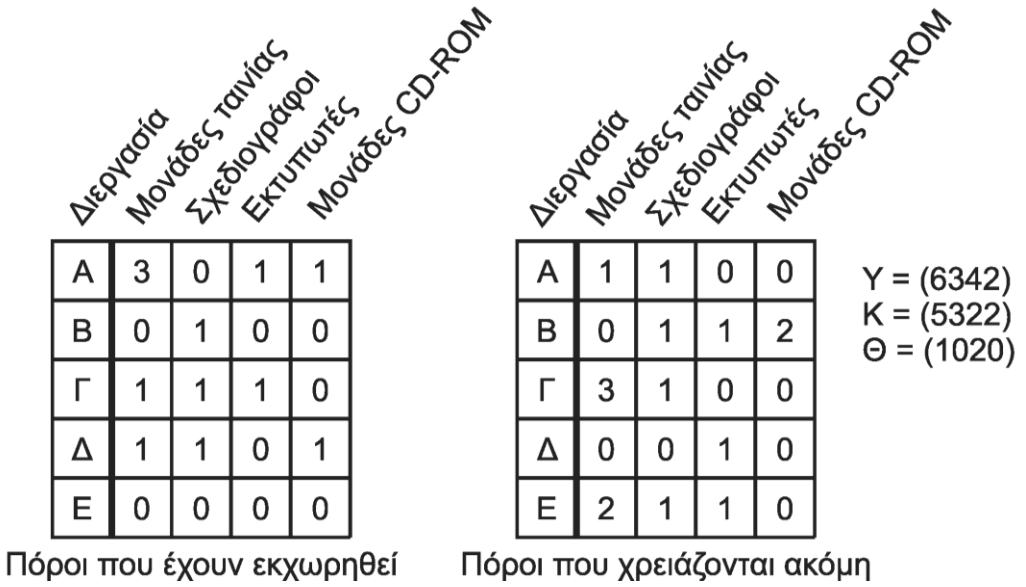
Αλγόριθμος τραπεζίτη (2 από 3)

Ο αλγόριθμος του τραπεζίτη για πολλούς πόρους

- Μητρώο εκχώρησης και πρόσθετων αιτήσεων
- Διανύσματα πόρων (Υ), εκχωρημένων (Κ), διαθεσίμων (Θ)

1. Βρες μία διεργασία R που μπορεί να ολοκληρωθεί
   - Η γραμμή αιτήσεων της R πρέπει να είναι μικρότερη ή ίση του Θ
   - Θεωρούμε ότι δεσμεύονται οι πόροι και μετά τερματίζει η R
2. Απελευθέρωσε τους πόρους προσθέτοντάς τους στο Θ
3. Επανέλαβε τα βήματα 1 και 2
   - Είτε τερματίζουν όλες οι διεργασίες
   - Είτε η κατάσταση ήταν ανασφαλής

Αλγόριθμος τραπεζίτη (3 από 3)



- Ο αλγόριθμος του τραπεζίτη για πολλούς πόρους
  - Στο παράδειγμα η κατάσταση είναι ασφαλής
    - Σειρά τερματισμού Δ, Α, Ε, …
  - Έστω ότι η Β ζητάει έναν εκτυπωτή
    - Πάλι σειρά τερματισμού Δ, Α, Ε, …
  - Έστω ότι η Ε ζητάει τον τελευταίο εκτυπωτή
    - Η κατάσταση είναι ανασφαλής

Εντοπισμός ή αποφυγή;

- Μήπως είναι το ίδιο τελικά;
  - Αφού χρησιμοποιείται ο ίδιος αλγόριθμος;
    - Αν και δεν είναι ο μοναδικός
- Η διαφορά είναι <u>πότε</u> εκτελείται
  - Στον εντοπισμό και ανάκαμψη εκτελείται ενίοτε
    - Το αδιέξοδο μπορεί να έχει ήδη συμβεί
  - Στην αποφυγή εκτελείται και σε κάθε δέσμευση
    - Εγγυάται ότι δεν θα φτάσουμε ποτέ σε αδιέξοδο

Detection Prevention
(Αποτροπή αδιεξόδων)

Γιατί πρόληψη αδιεξόδων

Πόσο ρεαλιστική είναι η αποφυγή αδιεξόδων;

- Λίγο καλύτερη από τον εντοπισμό
  - Ελέγχουμε τους πόρους μόνο όταν ζητούνται
  - Δεν περιμένουμε να συμβεί αδιέξοδο
- Οι υποθέσεις όμως δεν είναι ρεαλιστικές
  - Οι απαιτήσεις των διεργασιών δεν είναι γνωστές!
  - Νέες διεργασίες μπορεί να έρχονται στο σύστημα

Προσβολή των συνθηκών

Προσβολή των συνθηκών του Coffman

- Φροντίζουμε τα αδιέξοδα να είναι αδύνατα
  - Αρκεί να μην ισχύει <u>ποτέ</u> μία συνθήκη του Coffman
  - Πρέπει να ισχύουν <u>όλες</u> για να έχουμε αδιέξοδα
- Πιο πρακτική λύση σε σχέση με την αποφυγή
- Απαιτεί επιβολή περιορισμών στο σύστημα

Συνθήκη αμοιβαίου αποκλεισμού

Προσβολή συνθήκης αμοιβαίου αποκλεισμού

- Ορισμένοι πόροι απαιτούν αμοιβαίο αποκλεισμό
  - Παράδειγμα: ο εκτυπωτής
- Κρύβουμε τον εκτυπωτή πίσω από μια διεργασία (printer spool)
  - Μόνο η διεργασία αυτή μπορεί να τυπώσει
  - Οι άλλες διεργασίες της στέλνουν δεδομένα για εκτύπωση
  - Η ίδια η διεργασία δεσμεύει μόνο τον εκτυπωτή

Συνθήκη δέσμευσης και αναμονής

Προσβολή συνθήκης δέσμευσης και αναμονής

- Γιατί να μην δεσμεύονται οι πόροι από την αρχή;
  - Είτε η διεργασία θα λάβει όλους τους πόρους
  - Είτε δεν θα είναι διαθέσιμοι και θα περιμένει
- Πώς μπορούμε να ξέρουμε τι θα χρειαστούμε;
  - Το ίδιο πρόβλημα με την αποφυγή αδιεξόδων
- Κακή χρήση των πόρων
  - Έστω ότι ξέρουμε τις μέγιστες απαιτήσεις της διεργασίας
  - Δεν χρειάζονται όλοι οι πόροι ταυτόχρονα
  - Δεσμεύουμε πόρους που δεν χρειαζόμαστε

Συνθήκη μη προεκτόπισης

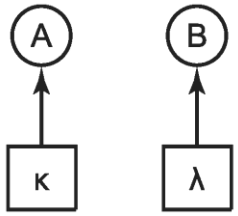
Προσβολή συνθήκης μη προεκτόπισης

- Γενικά μη εφικτό σε ορισμένες συσκευές
  - Παράδειγμα: εκτυπωτής (σπαταλάμε χαρτί)
  - Παράδειγμα: εγγραφέας DVD (χάνουμε το DVD)
- Χρήση διαχειριστών πόρων
  - Διαχειριστής εκτυπώσεων
  - Αποκρύπτει την ακολουθιακή χρήση των πόρων
- Ίδια προσέγγιση με τον αμοιβαίο αποκλεισμό

Συνθήκη κυκλικής αναμονής (1 από 2)

1. Εικονοθέτης
2. Σαρωτής
3. Σχεδιογράφος
4. Μονάδα ταινίας
5. Μονάδα CD-ROM

(α)



(β)

- Αρίθμηση όλων των πόρων του συστήματος
- Οι διεργασίες ζητούν πόρους μόνο με αύξουσα σειρά
  - Παράδειγμα: αν έχεις τον σαρωτή, μπορείς να ζητήσεις ταινία
  - Όχι όμως με την ανάποδη σειρά!
- Ουσιαστικά απαγορεύουμε τους κύκλους
  - Κάθε αλυσίδα πόρων περιλαμβάνει πόρους σε αύξουσα σειρά
  - Άρα δεν μπορεί να υπάρξει κύκλος!
- Περιορίζουμε τον τρόπο λειτουργίας των διεργασιών

Συνθήκη κυκλικής αναμονής (2 από 2)

Προσβολή συνθήκης κυκλικής αναμονής

- Παραλλαγή: δεν ζητάς «μικρότερους» πόρους
  - Πρώτα απελευθέρωση «μεγαλύτερων» πόρων
  - Ζητάμε τον σαρωτή όταν αφήσουμε την ταινία
  - Πάλι οι αλυσίδες δεν μπορούν να γίνουν κύκλοι

- Ποια είναι η σωστή αρίθμηση των πόρων;
  - Δεν είναι καθόλου προφανές!
  - Κάθε πρόγραμμα προτιμάει διαφορετική σειρά
  - Με πολλούς πόρους είναι δύσκολο να βρεθεί σειρά

Σύγκριση

Συνθήκη	Προσέγγιση
Αμοιβαίος αποκλεισμός	Παροχέτευση στα πάντα
Δέσμευση και αναμονή	Να ζητούνται όλοι οι πόροι από την αρχή
Μη προεκτόπιση	Αφαίρεση των πόρων
Κυκλική αναμονή	Αριθμητική διάταξη των πόρων

Τέσσερις συνθήκες του Coffman

- Μόνο δύο προσεγγίσεις λειτουργούν καλά
  - Διαχειριστές αμοιβαία αποκλειόμενων πόρων
  - Διατεταγμένη εκχώρηση πόρων
- Αλλιώς έχουμε σπατάλη πόρων
  - Αφαίρεση ή πλήρης δέσμευση των πόρων

Άλλα θέματα

Two-Phase Locking (2PL) [1/2]

Κλείδωμα σε δύο φάσεις

- Χρησιμοποιείται σε βάσεις δεδομένων
- Φάση 1: κλειδώνουμε τις εγγραφές
  - Αν αποτύχουμε, ξεκλειδώνουμε τα πάντα
  - Προσπαθούμε ξανά από την αρχή
- Φάση 2: ενημερώνουμε τις εγγραφές
  - Μετά απελευθερώνουμε όλα τα κλειδώματα

Two-Phase Locking (2PL) [2/2]

Κλείδωμα σε δύο φάσεις

- Δέσμευση όλων των πόρων από την αρχή
  - Για κάθε ομάδα ενημερώσεων εγγραφών όμως
- Το κλείδωμα μπορεί να καθυστερήσει πολύ
- Πρέπει το πρόγραμμα να μπορεί να κάνει πίσω
  - Δεν πρέπει να γίνουν αλλαγές στην πρώτη φάση
  - Απαιτείται κατάλληλη σχεδίαση του προγράμματος

Αδιέξοδα επικοινωνίας (1 από 3)

Αδιέξοδα επικοινωνίας

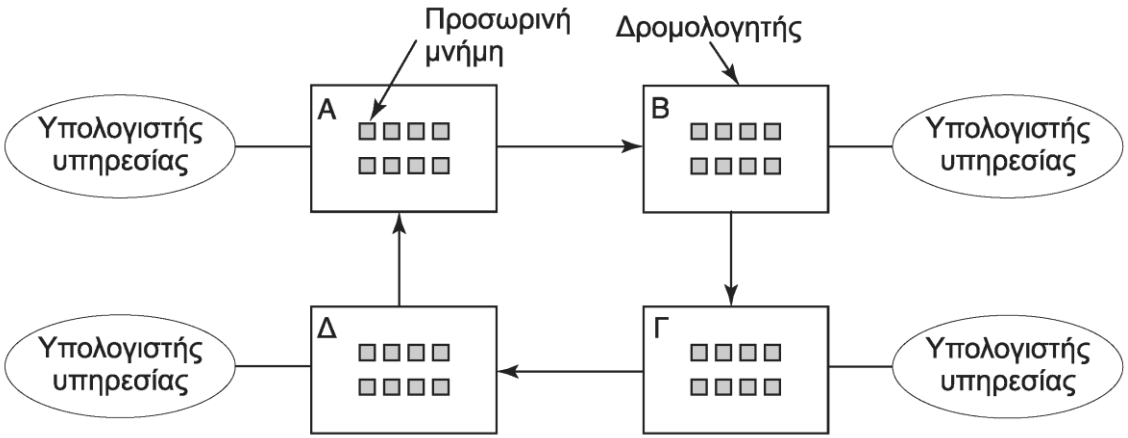
- Η διεργασία Α στέλνει ένα μήνυμα στη Β
- Η διεργασία Α μπλοκάρει περιμένοντας απάντηση
- Αν χαθεί η απάντηση;
  - Η Α θα μείνει μπλοκαρισμένη
  - Η Β θα περιμένει για ένα νέο μήνυμα από την Α
- Έχουμε αδιέξοδο σύμφωνα με τον συνήθη ορισμό
  - Όλες οι διεργασίες περιμένουν άλλες της ομάδας

Αδιέξοδα επικοινωνίας (2 από 3)

Αδιέξοδα επικοινωνίας

- Η διαχείριση αδιεξόδων πόρων δεν λειτουργεί
- Υπάρχει μια άλλη λύση: τα χρονόμετρα
  - Όταν στέλνεται ένα μήνυμα, ξεκινάει ένα χρονόμετρο
  - Αν λήξει χωρίς απάντηση, επαναλαμβάνουμε
  - Τι θα γίνει όμως αν το μήνυμα απλά άργησε;
  - Συνηθισμένο πρόβλημα στα πρωτόκολλα επικοινωνίας

Αδιέξοδα επικοινωνίας (3 από 3)



- Αδιέξοδο πόρων σε δίκτυα
  - Οι υπολογιστές υπηρεσίας ανήκουν στους χρήστες
  - Οι κόμβοι του δικτύου είναι δρομολογητές
  - Οι δρομολογητές έχουν πεπερασμένο χώρο αποθήκευσης
  - Στο παράδειγμα έχουν γεμίσει όλοι οι χώροι
  - Κανένας δρομολογητής δεν μπορεί να στείλει μήνυμα!

Livelock [1/2]

Ενεργό αδιέξοδο (livelock) σε κρίσιμες περιοχές

- Έστω ότι χρησιμοποιούμε busy waiting (αναμονή με απασχόληση)
  - Για αποφυγή της επιβάρυνσης από το μπλοκάρισμα
- Δύο διεργασίες προσπαθούν να αποκτήσουν πόρους
  - Με διαφορετική σειρά, οπότε αποκτούν από έναν πόρο
  - Αν δεν δεσμεύσουν τον δεύτερο, ξαναδοκιμάζουν
- Οι διεργασίες εκτελούνται χωρίς να κάνουν πρόοδο
- Η κατάσταση αυτή λέγεται ενεργό αδιέξοδο (livelock)
  - Δεν έχουμε μπλοκάρισμα, αλλά δεν έχουμε και πρόοδο

Livelock [2/2]

- Πολλοί τρόποι να συμβεί ενεργό αδιέξοδο
  - Συνήθως εξαντλείται κάποιος πεπερασμένος πόρος
- Παράδειγμα: πίνακας διεργασιών
  - Ο πίνακας είναι γεμάτος και τα fork() αποτυγχάνουν
- Παράδειγμα: πίνακας αρχείων
  - Δεν μπορούν να ανοίξουν αρχεία και τα open() αποτυγχάνουν
- Τα περισσότερα συστήματα αγνοούν το πρόβλημα
  - Είναι απίθανο να συμβεί με λογικό φόρτο
  - Αρκεί να έχουμε διαστασιολογήσει σωστά τους πίνακες
- Οι διεργασίες μπορούν να είναι προετοιμασμένες
  - Αν δεν δεσμευτεί ο πόρος n φορές, αποτυχία και τερματισμός

Starvation (Λιμοκτονία)

- Ποιος θα πάρει έναν πόρο που ζητούν πολλοί;
  - Κάθε σύστημα εφαρμόζει κάποια πολιτική
- Παράδειγμα: εκτυπωτής
  - Έστω ότι επιλέγουμε τη μικρότερη εκτύπωση
    - Έτσι οι μικρές εκτυπώσεις εξυπηρετούνται γρήγορα
    - Οι μεγάλες όμως μπορεί να μην εκτελεστούν ποτέ!
- Λιμοκτονία (starvation): αναβολή εξυπηρέτησης
  - Αντιμετωπίζεται π.χ. με FCFS ή Round Robin