Λειτουργικά Συστήματα

Ενότητα 4β: Αδιέξοδα

Αθηνά Βακάλη
Τμήμα Πληροφορικής

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

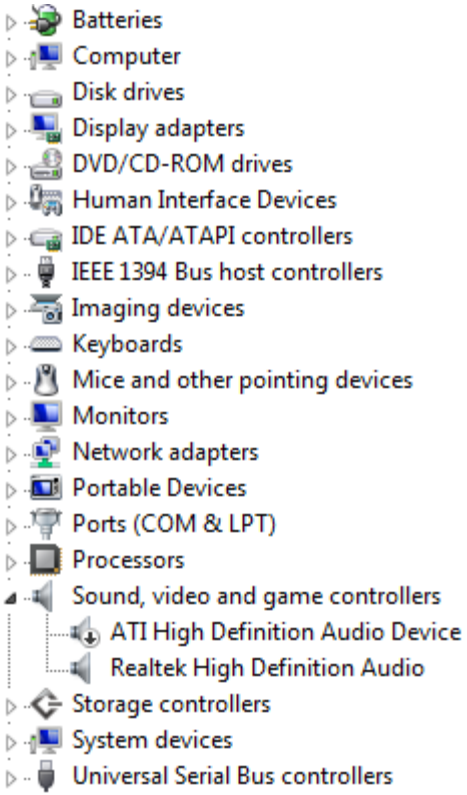
- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

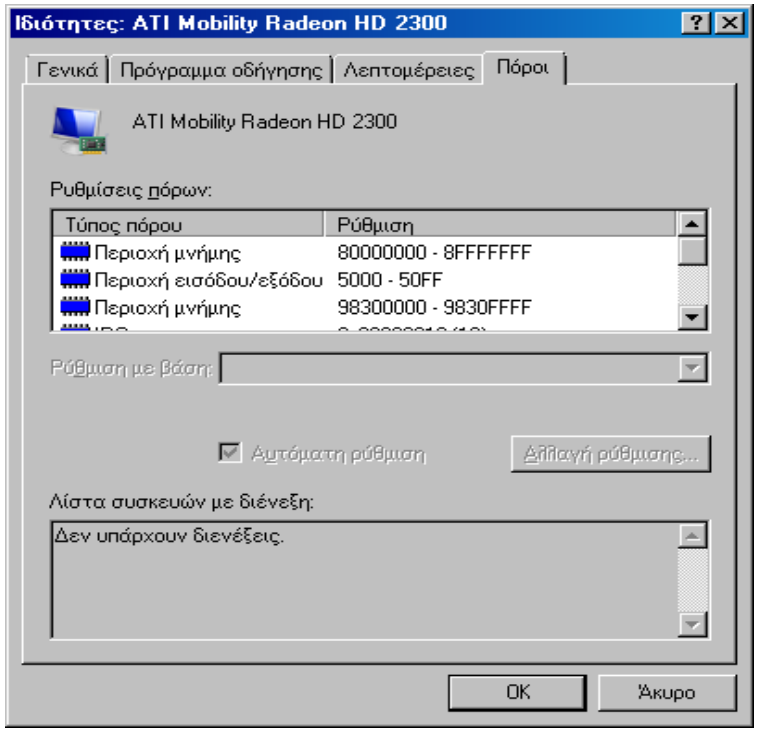
Πόροι

Πόρους ονομάζουμε τα αντικείμενα του λειτουργικού συστήματος τα οποία μπορούν να εκχωρηθούν σε μια διεργασία για να τα χρησιμοποιήσει.

- Είναι οτιδήποτε μπορεί να χρησιμοποιηθεί από μια μόνο διεργασία οποιαδήποτε χρονική στιγμή.
- Μπορεί να είναι μια συσκευή υλικού (για παράδειγμα ένας CD-ROM drive) ή κάποια πληροφορία (για παράδειγμα ένα αρχείο στο δίσκο).
- Ένας υπολογιστής έχει συνήθως πολλούς πόρους.
- Αν ένας πόρος απαιτείται από παραπάνω από μία διεργασίες μπορεί να συμβεί αδιέξοδο.

Πόροι στα Λειτουργικά Συστήματα Windows





Πόροι στα Λειτουργικά Συστήματα FreeBSD

- vr0: Quirks:0x0
- miibus1:<MII bus> on vr0
- ukphy0: <Generic IEEE 802.3u media interface> on miibus1
- ukphy0: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX,auto
- vr0: Ethernet address:00:50:ba:24:7d:14
- orm0: <ISA Option ROMs> at iomem 0xc0000-0xcbfff,0xcc000-0xd07ff,0xd1000-0xd17ff on isa0
- atkbdc0: <Keyboard controller (i8042)> at port 0x60,0x64 on isa0
- atkbd0: <AT Keyboard> irq 1 on atkbdc0
- kbd0 at atkbd0
- atkbd0: [GIANT-LOCKED]
- sc0: <System console> at flags 0x100 on isa0
- sc0:VGA <16 virtual consoles, flags=0x300>
- sio0 at port 0x3f8-0x3ff irq 4 flags 0x10 on isa0

Αδιέξοδο (Deadlock)

Ορισμός 1

Η μόνιμη / επ' αόριστον αναμονή ενός συνόλου διεργασιών που είτε συναγωνίζονται για πόρους του συστήματος είτε επικοινωνούν μεταξύ τους.

Ορισμός 2

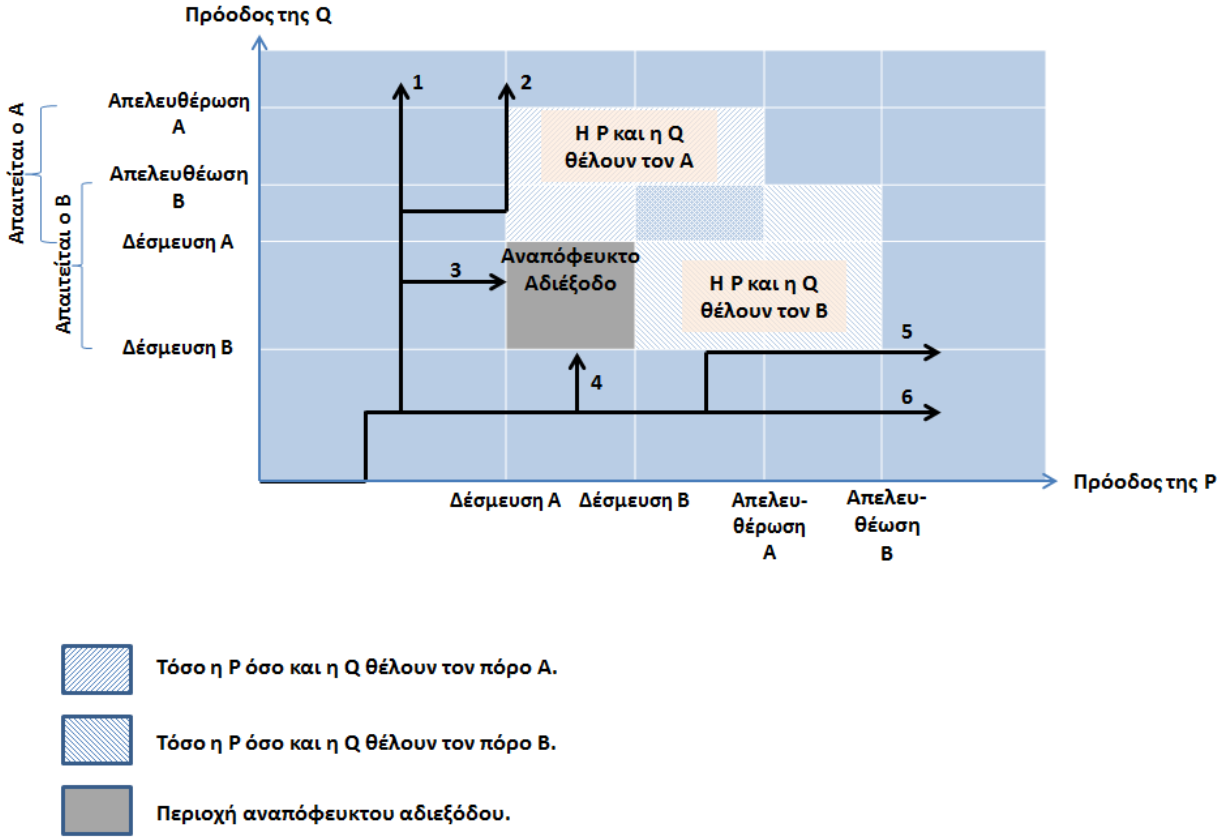
Ένα σύνολο από διεργασίες που δημιουργούν μια κυκλική αλυσίδα όπου κάθε process στη αλυσίδα δε μπορεί να προχωρήσει και περιμένει για κάποιο γεγονός που μπορεί να προκληθεί μόνο από κάποιο άλλο μέλος της αλυσίδας.

- Σε ένα σύστημα πολυπρογραμματισμού η συνολική απαίτηση πόρων από όλες τις συνεξελισσόμενες ενεργές διεργασίες υπερβαίνει κατά πολύ το συνολικό ποσό των διαθέσιμων πόρων.
- Όλα τα αδιέξοδα εμπεριέχουν τις συγκρουόμενες ανάγκες για πόρους από δύο ή περισσότερες διεργασίες.
- Βασικός σκοπός είναι η σχεδίαση συστημάτων όπου το αδιέξοδο δε θα μπορεί να συμβεί.
- Γενικά δεν υπάρχει αποτελεσματική λύση.

Πόροι και Διεργασίες
Παράδειγμα

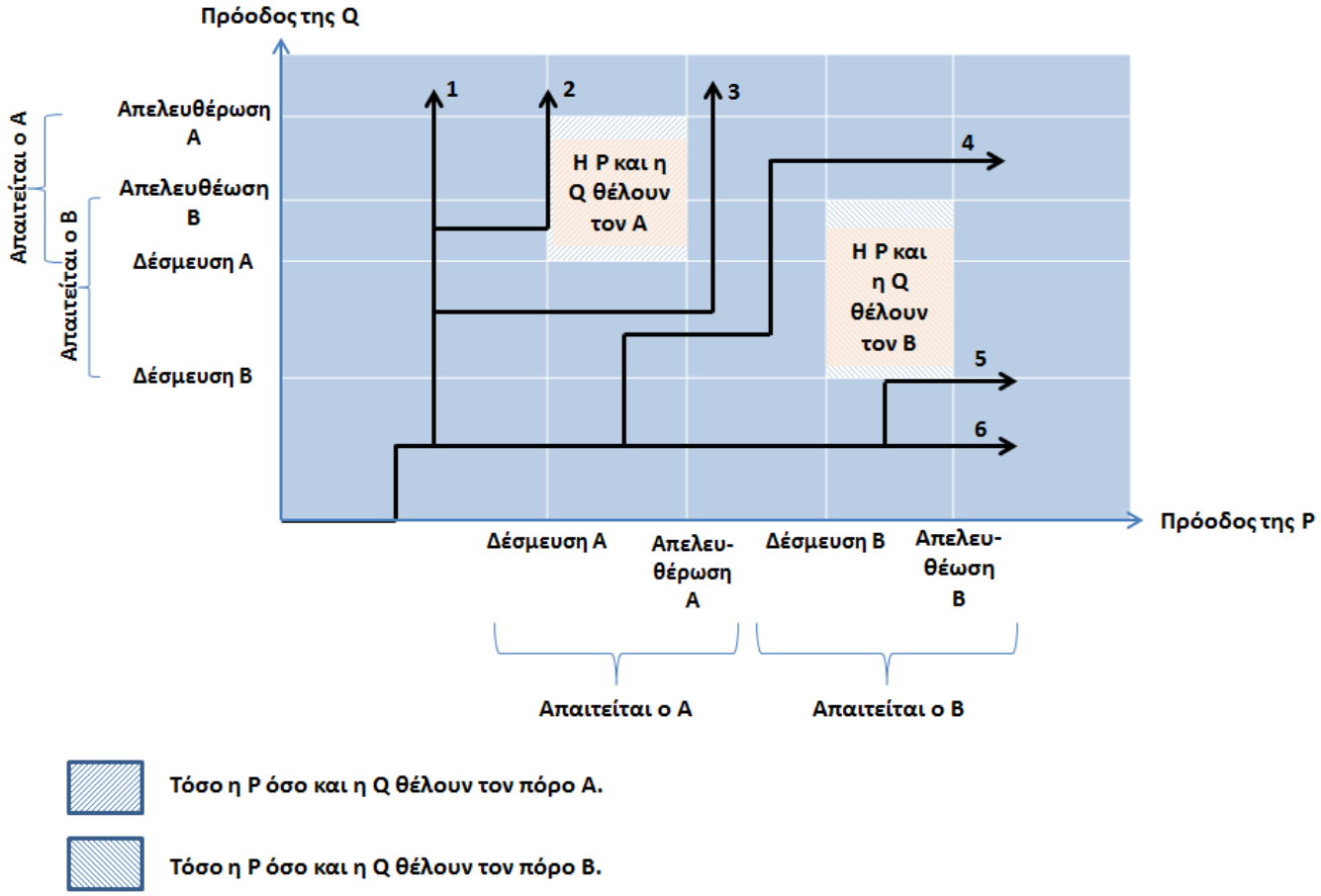
Παράδειγμα συνεργασίας πόρων και διεργασιών

Αδιέξοδο
Παράδειγμα



Παράδειγμα αδιεξόδου

Παράδειγμα χωρίς αδιέξοδο



Παράδειγμα έλλεψης αδιεξόδου

Κατηγορίες Πόρων

- **Προεκχωρούμενοι πόροι** (Preemptable Resources)

  Μπορούν να απομακρυνθούν από μια διεργασία χωρίς παρενέργειες, πχ η μνήμη.

- **Μη προεκχωρούμενοι πόροι** (Nonpreemptable Resources)

  Προξενούν αποτυχία στη διεργασία όταν απομακρυνθούν, πχ μονάδα του cd-rom.

Παραδείγματα πόρων: εκτυπωτές, tape drives, μνήμη, CD - Recorders κλπ

Απαιτούμενα βήματα για τη χρήση ενός πόρου

- Αλληλουχία γεγονότων που απαιτούνται για τη χρήση ενός πόρου:
  - Απαίτηση (request).
  - Χρήση.
  - Απελευθέρωση πόρου.
- Αναμονή αν η απαίτηση δεν ικανοποιηθεί
  - Η αιτούμενη διεργασία αναστέλλεται.
  - Η αιτούμενη διεργασία αποτυγχάνει εμφανίζοντας μήνυμα λάθους.

Επαναχρησιμοποιήσιμοι Πόροι

- Χρησιμοποιούνται με ασφάλεια από μια διεργασία σε κάθε χρονική στιγμή και δεν εξαντλούνται από αυτή τη χρήση.
- Οι διεργασίες αποκτούν πόρους που θα απελευθερώσουν στη συνέχεια ώστε να χρησιμοποιηθούν από άλλες διεργασίες.
- Τέτοιοι πόροι είναι: επεξεργαστές, I/O κανάλια, κύρια και δευτερεύουσα μνήμη, αρχεία, βάσεις δεδομένων, και σημαφόροι.
- Το αδιέξοδο προκύπτει όταν μια διεργασία δεσμεύει έναν πόρο και απαιτεί έναν άλλο.

Παράδειγμα αδιεξόδου (1/2)

Παράδειγμα 1

p0 p1 q0 q1 p2 q2 ← **Αδιέξοδο**

Διεργασία P

Βήμα	Ενέργεια
p0	Request (D)
p1	Lock (D)
P2	Request (T)
p3	Lock (T)
p4	Perform function
p5	Unlock (D)
p6	Unlock (T)

Διεργασία Q

Βήμα	Ενέργεια
q0	Request (T)
q1	Lock (T)
q2	Request (D)
q3	Lock (D)
q4	Perform function
q5	Unlock (T)
q6	Unlock (D)

Παράδειγμα διεργασιών που ανταγωνίζονται για επαναχρησιμοποιούμενους πόρους

Παράδειγμα αδιεξόδου (2/2)

Παράδειγμα 2

Ο διαθέσιμος χώρος για κατανομή στην κύρια μνήμη είναι 200Kbytes, και πραγματοποιείται η παρακάτω σειρά αιτημάτων:

Το αδιέξοδο προκύπτει αν και οι δύο διεργασίες προχωρήσουν στο 2ο αίτημά τους.

Ζωντανό Αδιέξοδο (livelock)

- Στο livelock οι διεργασίες δεν περιμένουν κάτι (δεν είναι blocked).
- Οι διεργασίες αλλάζουν την κατάστασή τους ως απάντηση σε άλλες διεργασίες, αλλά χωρίς να κάνουν τίποτα χρήσιμο.
- Παράδειγμα (από την πραγματική ζωή)

  Σε ένα πεζοδρόμιο έρχονται αντιμέτωποι δύο άνθρωποι. Και οι δύο μετακινούνται προς μια κατεύθυνση για να περάσει ο άλλος. Επειδή και οι 2 μετακινούνται προς την ίδια κατεύθυνση δεν αλλάζει κάτι.

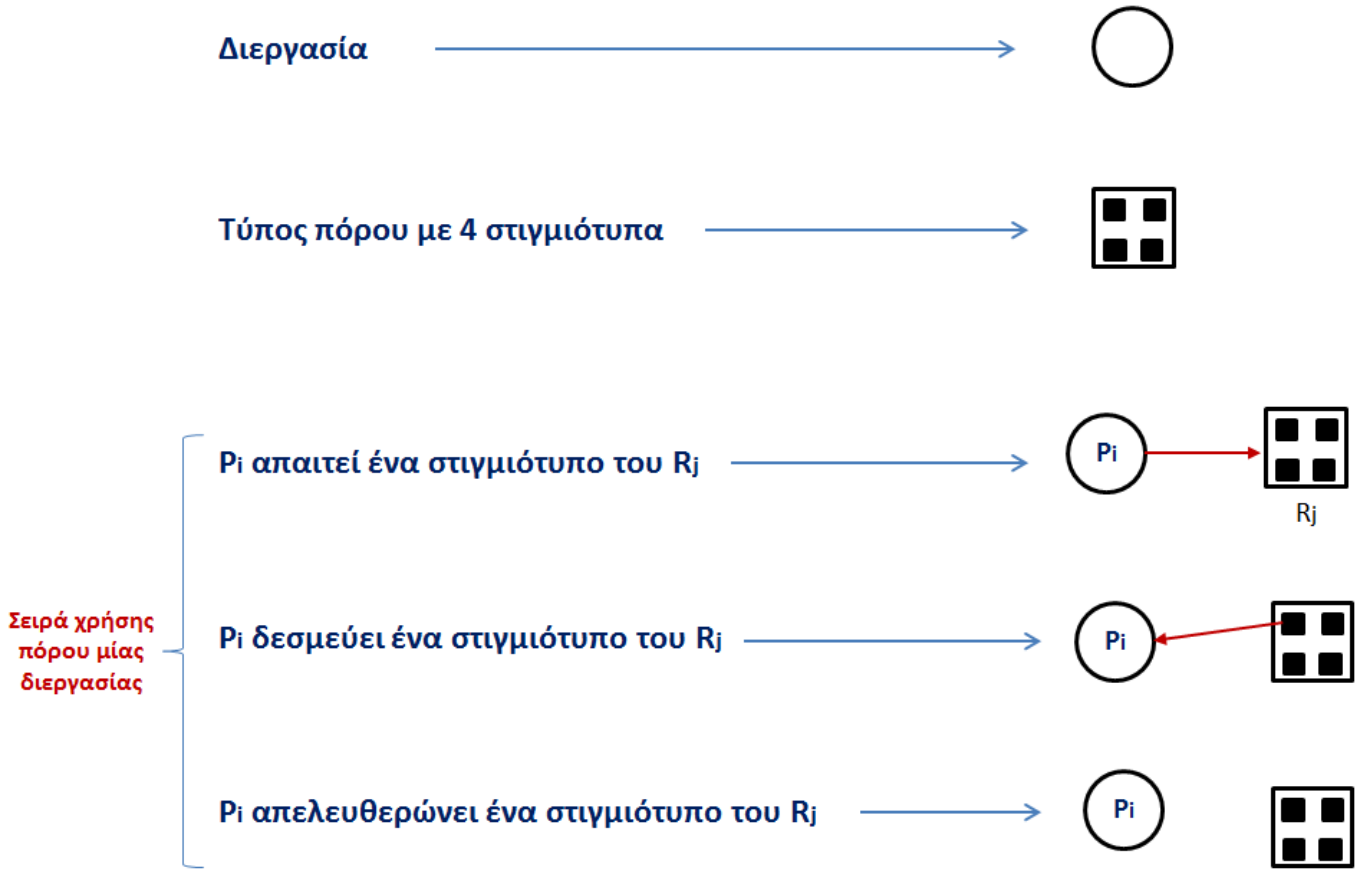
Μοντέλα Αδιεξόδων

Μοντελοποίηση του προβλήματος

- Δημιουργείται ένας κατευθυνόμενος γράφος.
- Κόμβοι του γράφου είναι οι διεργασίες και οι πόροι.
- Η ακμή P → R σημαίνει ότι η διεργασία P περιμένει για τον πόρο R.
- Η ακμή R → P σημαίνει ότι η διεργασία P κατέχει τον πόρο R.
- Στο σύστημα υπάρχει πιθανότητα deadlock **εάν και μόνο εάν** ο κατευθυνόμενος γράφος **περιέχει ένα κύκλο**.

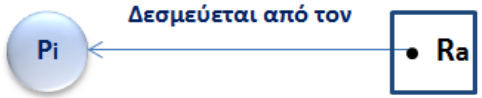
Το σύστημα μπορεί να χρησιμοποιεί ένα τέτοιο γράφο και ένα αλγόριθμο ανίχνευσης κύκλων για να ανιχνεύει deadlock.

Γράφοι εκχώρησης πόρων

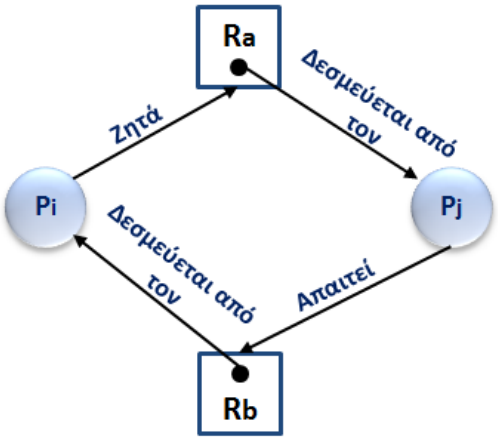


Γράφοι εκχώρησης πόρων Παραδείγματα (1/2)

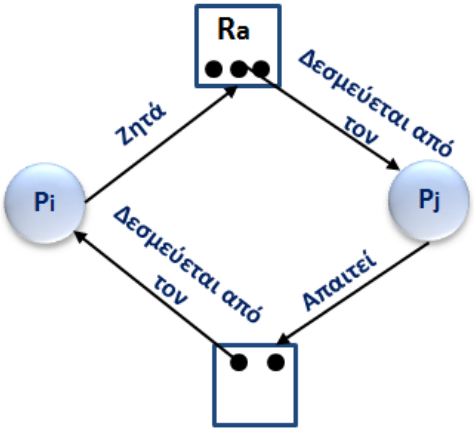
Ο πόρος ζητείται



Ο πόρος δεσμεύεται



Κυκλική αναμονή

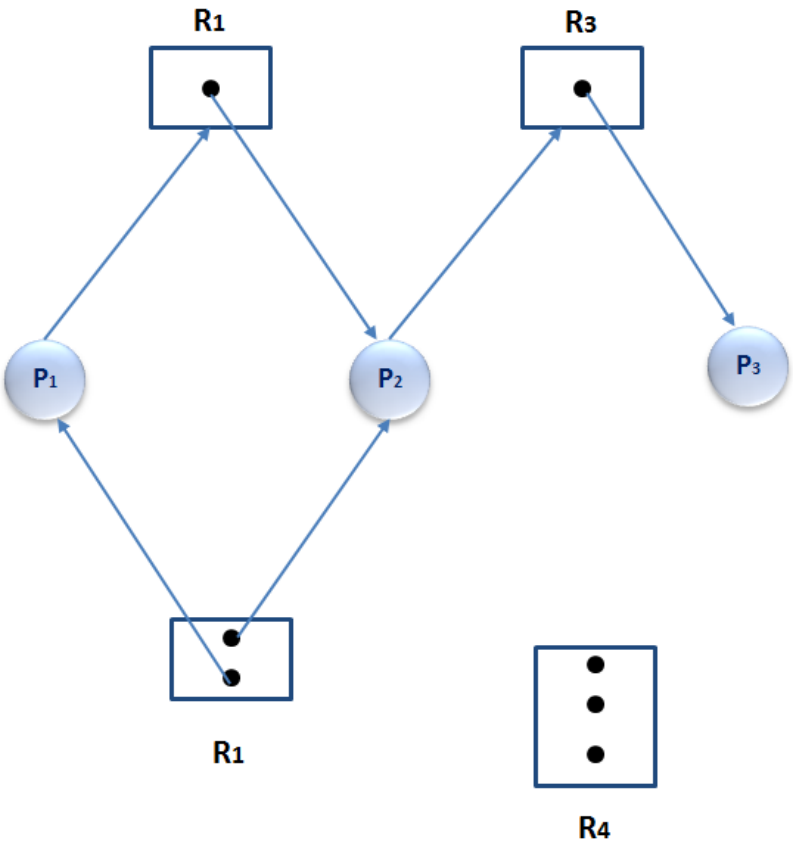


Χωρίς αδιέξοδο

Παραδείγματα γραφημάτων ανάθεσης πόρων

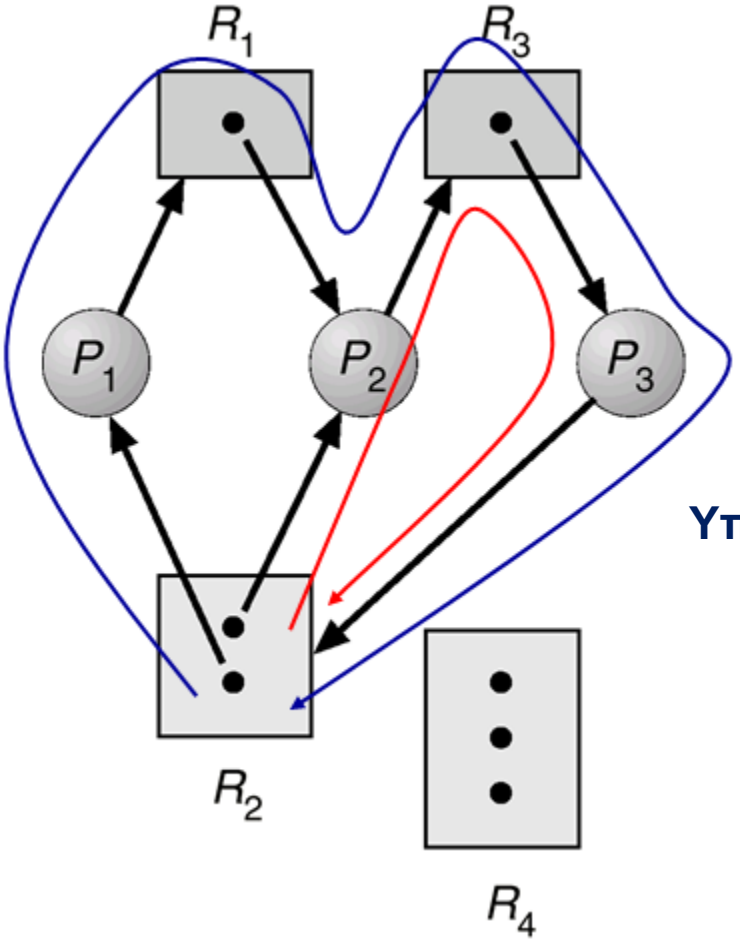
Γράφοι εκχώρησης πόρων Παραδείγματα (2/2)

Μπορεί να συμβεί αδιέξοδο;



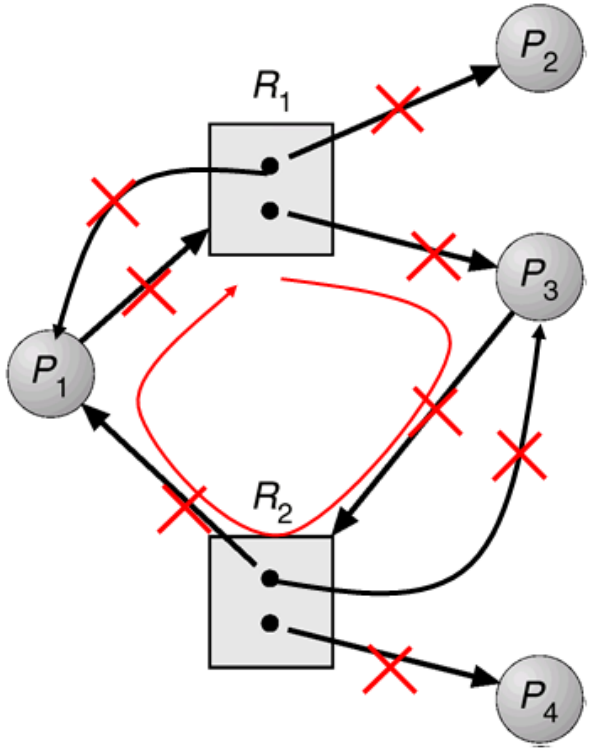
Όχι, γιατί δε δημιουργείται "κύκλος".

Γράφοι εκχώρησης πόρου με αδιέξοδο (Παράδειγμα)



Υπάρχουν δύο κύκλοι: Πιθανό αδιέξοδο!

Γράφοι εκχώρησης πόρου με κύκλο, χωρίς αδιέξοδο (Παράδειγμα)



- Αν ο γράφος δεν περιέχει κύκλους: **δεν υπάρχει αδιέξοδο**.
- Αν ο γράφος περιέχει ένα κύκλο:
  - Αν υπάρχει μόνο ένα στιγμιότυπο ανά τύπο πόρου, τότε **υπάρχει αδιέξοδο**.
  - Αν υπάρχουν αρκετά στιγμιότυπα ανά τύπο πόρου, τότε **υπάρχει πιθανότητα να συμβεί αδιέξοδο**.

Ασκήσεις (1/5)

Άσκηση 1

Να σχεδιαστεί ένας γράφος εκχώρησης πόρων για τα παρακάτω:

- Η διεργασία P1 απαιτεί τον πόρο R1
- Η διεργασία P2 απαιτεί τον πόρο R3
- Ο πόρος R1 εκχωρείται στη διεργασία P2
- Ο πόρος R2 εκχωρείται στη διεργασία P1
- Ο πόρος R3 εκχωρείται στη διεργασία P3

Υπάρχει αδιέξοδο;

Ασκήσεις (2/5)

Άσκηση 2

Ένα σύστημα αποτελείται από 4 διεργασίες, (p1, p2, p3, p4), και 3 τύπους από σειριακά επαναχρησιμοποιούμενους πόρους, (R1, R2, R3). Ο αριθμός των μονάδων των πόρων είναι E=<3,2,2>.

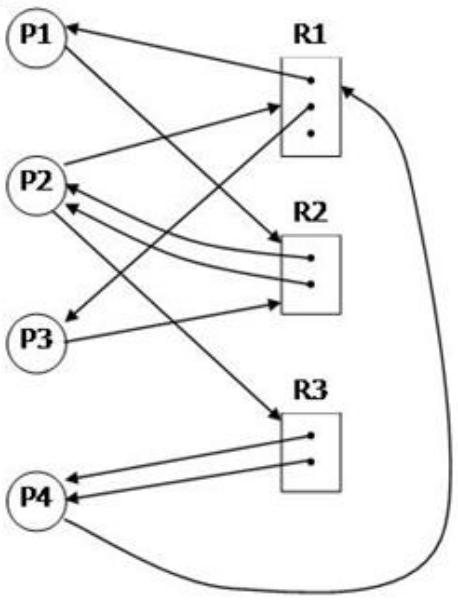
- Η διεργασία p1 κρατά 1 μονάδα του R1 και απαιτεί 1 μονάδα του R2.
- Η διεργασία p2 κρατά 2 μονάδες του R2 και απαιτεί 1 μονάδα καθενός από τους R1 και R3.
- Η διεργασία p3 κρατά 1 μονάδα του R1 και απαιτεί 1 μονάδα του R2.
- Η διεργασία p4 κρατά 2 μονάδες του R3 και απαιτεί 1 μονάδα του R1.

Να σχεδιάσετε το γράφημα εκχώρησης πόρων που αναπαριστά τη ν κατάσταση του συστήματος. Υπάρχουν διεργασίες που βρίσκονται σε αδιέξοδο στην κατάσταση αυτή;

Ασκήσεις (3/5)

Άσκηση 2 (Συνέχεια)

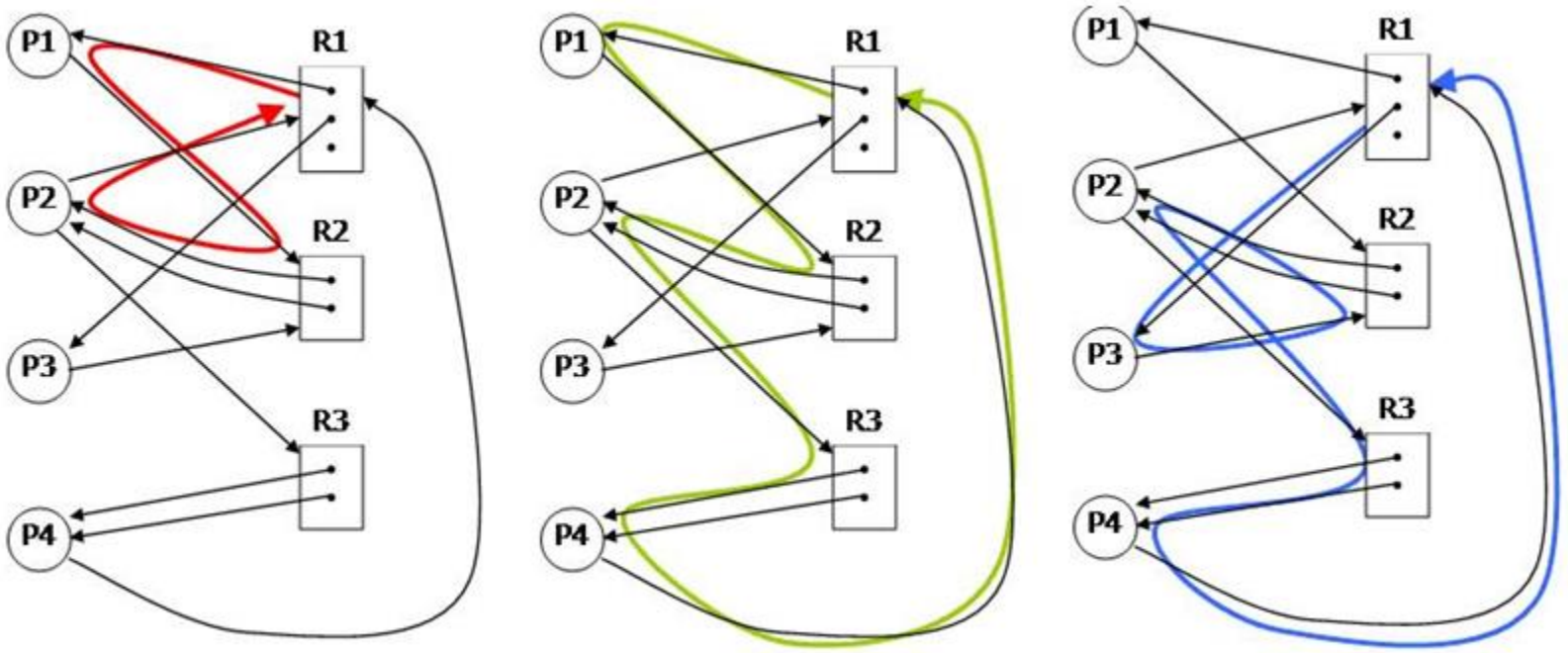
Το γράφημα εκχώρησης πόρων είναι ως εξής:



Γράφημα εκχώρησης πόρων

Ασκήσεις (4/5)

Άσκηση 2 (Συνέχεια)



Δημιουργούνται τουλάχιστον τρεις κύκλοι.

Ασκήσεις (5/5)

Άσκηση 3

Ένα σύστημα διαθέτει 6 όμοια tape drives και n σε πλήθος διεργασίες που ανταγωνίζονται για τη χρήση τους. Κάθε διεργασία μπορεί να απαιτήσει 2 tape drives. Για ποια τιμή του n το σύστημα είναι απαλλαγμένο από αδιέξοδα;

Άσκηση 3

Υποθέστε ένα σύστημα με P διεργασίες και R όμοιους επαναχρησιμοποιήσιμους πόρους. Αν κάθε διεργασία μπορεί να απαιτήσει κατά μέγιστο 2 μονάδες πόρων, να αποδείξετε ότι δεν μπορεί να υπάρξει αδιέξοδο μόνον όταν ισχύει η συνθήκη P<=R-1.

Αναγκαίες συνθήκες αδιεξόδου

Το αδιέξοδο συμβαίνει όταν ισχύουν όλα τα παρακάτω (Συνθήκες Αδιεξόδου):

- **Αμοιβαίος αποκλεισμός** (mutual exclusion):

  Οι εκχωρούμενοι πόροι είναι στην αποκλειστική κυριότητα της διεργασίας. Κάθε πόρος εκχωρείται σε μία διεργασία ή είναι διαθέσιμος.

- **Κατοχή και αναμονή** (hold & wait):

  Η διεργασία μπορεί να δεσμεύει έναν πόρο ενώ περιμένει έναν άλλο

- **Μη προεκχώρηση** (No preemption):

  Κανένας πόρος δεν μπορεί να αποσπασθεί δια της βίας από τη διεργασία που την κατέχει

- **Κυκλική αναμονή** (circular wait):

  ΄Υπαρξη μιας κλειστής αλυσίδας διεργασιών 2 ή περισσοτέρων διεργασιών. Κάθε μία αναμένει ένα πόρο που κατέχεται από το επόμενο μέλος της αλυσίδας.

Χειρισμών Αδιεξόδων

- **Αποτροπή** (Prevent): Εφαρμογή πολιτικής που αποτρέπει μια από τις 4 συνθήκες
- **Αποφυγή** (Avoid): Δυναμικές επιλογές με βάση την τρέχουσα κατάσταση ανάθεσης πόρων
- **Εντοπισμός** (Detect): Εντοπισμός του αδιεξόδου και ενέργειες για την ανάκαμψη του συστήματος
- **Χειροκίνητη μεσολάβηση**: Ο χειριστής κάνει επανεκκίνηση του συστήματος, αν φαίνεται υπερβολικά αργό.

Αποτροπή
Αμοιβαίος Αποκλεισμός

Μόνο μια διεργασία τη φορά μπορεί να χρησιμοποιεί ένα πόρο:

- **Μη διαμοιραζόμενοι πόροι:** εκτυπωτής - δε μπορεί να διαμοιράζεται ταυτόχρονα σε πολλές διεργασίες.
- **Διαμοιραζόμενοι πόροι:** αρχεία για ανάγνωση - επιτρέπεται η πολλαπλή πρόσβαση σε ένα αρχείο για ανάγνωση μόνο.

Αποτροπή
Κατοχή και Αναμονή (Hold-and-wait)

Μια διεργασία μπορεί να κατέχει πόρους καθώς αναμένει την εκχώρηση κάποιων άλλων πόρων.

- Περιπτώσεις
  - Μια διεργασία απαιτεί όλους τους πόρους πριν ξεκινήσει (άρα δεν θα περιμένει ποτέ στη συνέχεια).
  - Μια διεργασία απαιτεί όλους τους πόρους όταν δεν διαθέτει κανέναν.
- Προβλήματα
  - Δεν είναι γνωστές οι απαιτήσεις πόρων κατά την έναρξη της διεργασίας.
  - Παρατεταμένη στέρηση.
  - Δεσμεύονται πόροι που θα μπορούσαν να χρησιμοποιηθούν από άλλες διεργασίες.

Αποτροπή
Μη προεκχώρηση (No preemption) (1/2)

- Αν μια διεργασία που δεσμεύει συγκεκριμένους πόρους αρνείται μια επιπλέον απαίτηση τότε η διεργασία πρέπει να αποδεσμεύσει τους αρχικούς πόρους.

- Αν μια διεργασία απαιτήσει έναν πόρο που δεσμεύεται από κάποια άλλη διεργασία, το Λειτουργικό Σύστημα θα μπορεί να προεκχωρήσει την 2η διεργασία και να απαιτήσει να απελευθερώσει τους πόρους που κατέχει.

Αποτροπή
Μη προεκχώρηση (No preemption) (2/2)

- Εφαρμόζεται σε πόρους η κατάσταση των οποίων μπορεί να αποθηκευθεί και αργότερα να γίνει επαναφορά, όπως: Καταχωρητές της CPU και χώρος μνήμης.
- Δεν είναι μια βιώσιμη επιλογή για την πλειονότητα των πόρων.

Παράδειγμα

Η διακοπή μιας εγγραφής CD δημιουργεί ένα σημαντικό κόστος:

- Προφανής λύση.
- Ο οδηγός συσκευής του CD-R απαγορεύει μια δεύτερη λειτουργία open().

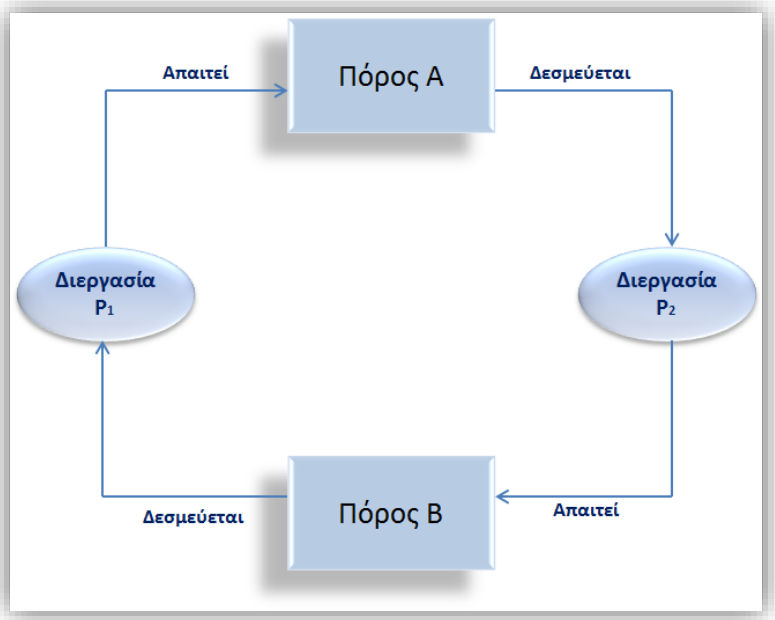
Αποτροπή
Κυκλική Αναμονή (1/2)

- Πρέπει να υπάρχει μια κυκλική αλυσίδα με δύο ή περισσότερες διεργασίες.
- Κάθε μια διεργασία περιμένει έναν πόρο που κατέχεται από άλλη διεργασία της αλυσίδας.
- Προλαμβάνεται με τον ορισμό μια γραμμικής διάταξης των πόρων.

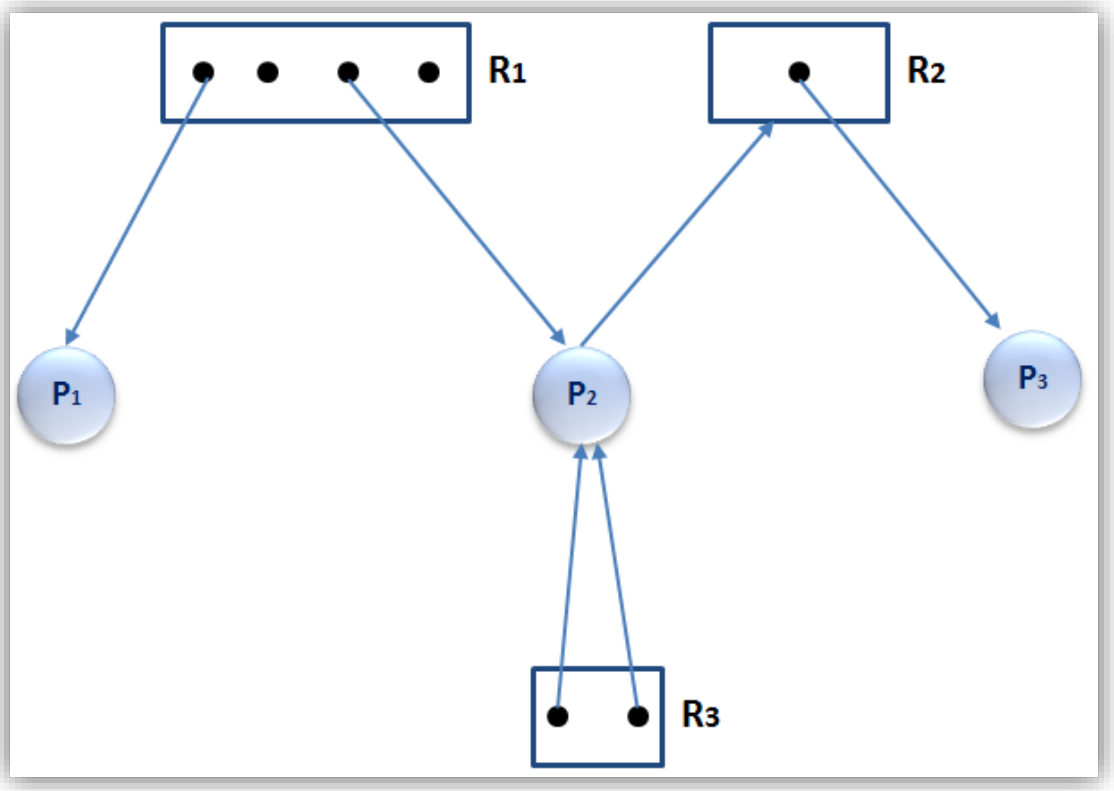
Αποτροπή
Κυκλική Αναμονή (2/2)

Παράδειγμα

Έστω ότι ο πόρος Ri προηγείται στη σειρά του Rj όπου i<j. Θεωρούμε ότι δύο διεργασίες Α και Β βρίσκονται σε κατάσταση αδιεξόδου επειδή η Α απέκτησε την Ri και ζήτησε την Rj και η Β κατέχει την Rj και ζητάει την Ri. Αυτό είναι αδύνατο καθώς θα σήμαινε ότι i<j και j<i.

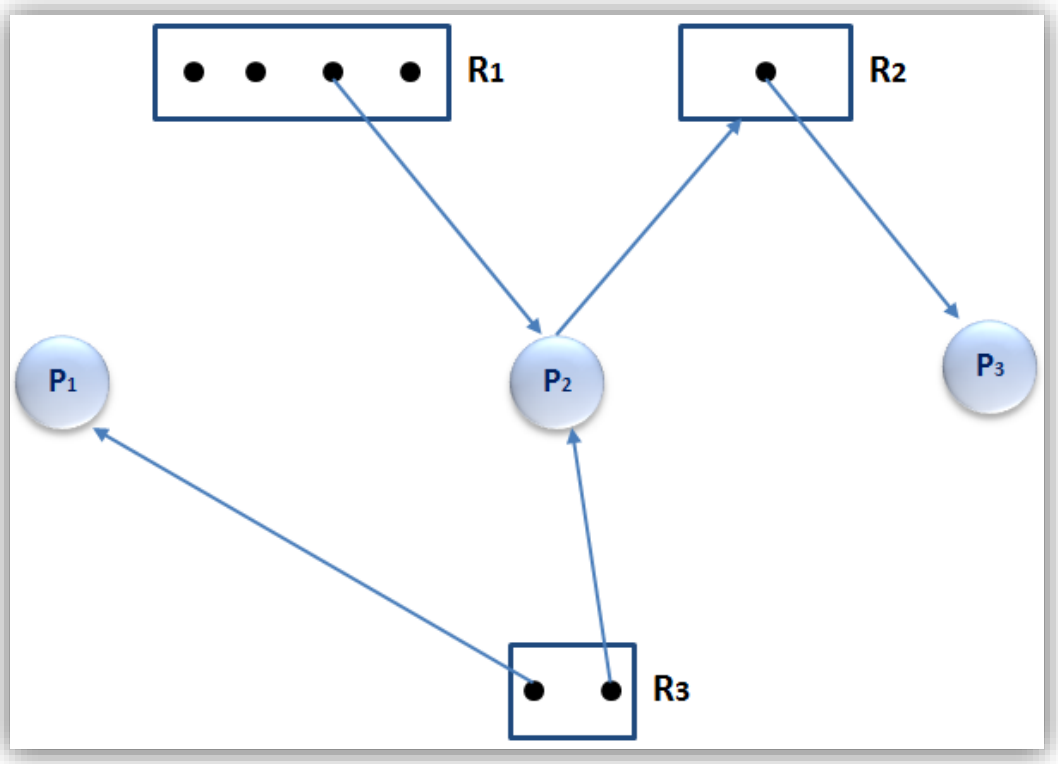


Κυκλική Αναμονή Παράδειγμα



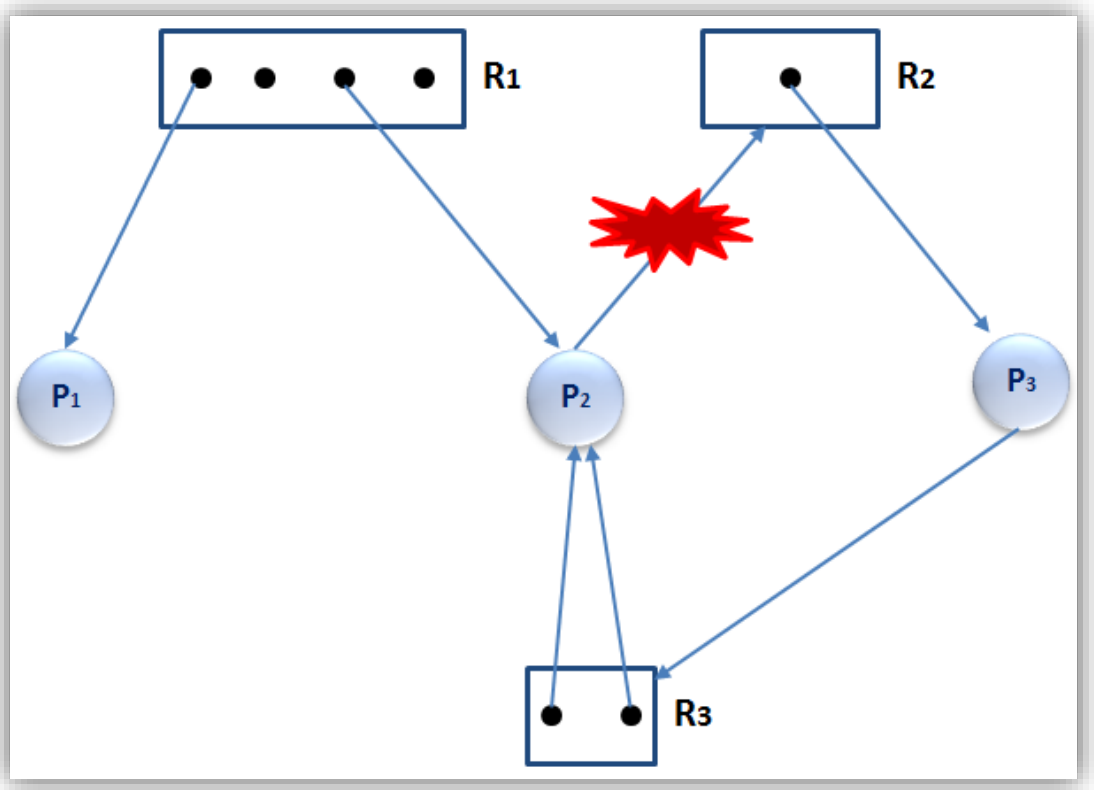
Παράδειγμα με κυκλική αναμονή

Χωρίς Κυκλική Αναμονή Παράδειγμα



Παράδειγμα χωρίς κυκλική αναμονή

Πρόληψη Κυκλικής Αναμονής Παράδειγμα



Παράδειγμα πρόληψης κυκλικής αναμονής

Αποφυγή Αδιεξόδου (1/4)

- Οι πόροι εκχωρούνται με τρόπο που εγγυάται ότι δε θα βρεθεί ποτέ σημείο στο οποίο θα συμβεί αδιέξοδο
- Η αποφυγή του αδιεξόδου επιτρέπει τις τρεις απαραίτητες συνθήκες, αλλά κάνει διακριτικές επιλογές ώστε να εξασφαλιστεί ότι δε θα προκύψει ποτέ αδιέξοδο. Έτσι επιτρέπεται μεγαλύτερος συγχρονισμός σε σχέση με την πρόληψη.
- Απαιτείται γνώση των μελλοντικών απαιτήσεων της διεργασίας.
- Προϋποθέτει ότι το σύστημα έχει κάποια πρόσθετη και εκ των προτέρων διαθέσιμη πληροφόρηση.

Αποφυγή Αδιεξόδου (2/4)

Λαμβάνεται δυναμικά μια απόφαση σχετικά με το αν η ικανοποίηση μιας απαίτησης, σε σχέση με την τρέχουσα εκχώρηση πόρων, θα οδηγήσει σε αδιέξοδο, που βασίζεται σε:

- Στο συνολικό ποσό των διαθέσιμων πόρων.
- Στους τρέχοντες διαθέσιμους πόρους.
- Στις απαιτήσεις πόρων εκ μέρους των διεργασιών.
- Στην πρόσφατη εκχώρηση πόρων στις διεργασίες.

Αποφυγή Αδιεξόδου (3/4)

- Το απλούστερο και πλέον χρήσιμο μοντέλο απαιτεί ότι κάθε διεργασία δηλώνει το μέγιστο πλήθος των πόρων κάθε τύπου που είναι πιθανόν να χρειαστεί.
- Ο αλγόριθμος αποφυγής αδιεξόδου εξετάζει δυναμικά την κατάσταση ανάθεσης πόρων για να εξασφαλίσει ότι δεν μπορεί να προκύψει κατάσταση κυκλικής αναμονής.
- Η κατάσταση ανάθεσης πόρων ορίζεται από το πλήθος των διαθέσιμων και εκχωρούμενων πόρων και από το μέγιστο πλήθος των απαιτήσεων εκ μέρους των διεργασιών.

Αποφυγή Αδιεξόδου (4/4)

Προσεγγίσεις αποφυγής αδιεξόδου

- **Process Initiation Denial**

  Μια διεργασία δε ξεκινά αν οι απαιτήσεις της μπορούν να οδηγήσουν σε αδιέξοδο, δηλαδή αν οι απαιτήσεις της διεργασίας και οι μέγιστες απαιτήσεις όλων των διεργασιών ξεπερνούν το συνολικό αριθμό των πόρων.

- **Resource Allocation Denial** (Αλγόριθμος του τραπεζίτη)

  Δεν ικανοποιείται μια αυξημένη απαίτηση για τη χρήση ενός πόρου από μια διεργασία αν αυτή η εκχώρηση του πόρου μπορεί να οδηγήσει σε αδιέξοδο.

Προβλήματα

- Μικρή χρήση πόρων.
- Μειωμένη ρυθμοαπόδοση (throughput) του συστήματος.

Αλγόριθμος του τραπεζίτη (Dijkstra 1956)

- Είναι γνωστός και ως άρνηση ανάθεσης πόρων
- Δεν επιτρέπει την ικανοποίηση αυξημένων απαιτήσεων για πόρους σε μια διεργασία αν αυτή η εκχώρηση πόρων μπορεί να οδηγήσει σε αδιέξοδο.
- **Κατάσταση του συστήματος**: είναι η τρέχουσα ανάθεση πόρων στις διεργασίες.
  - **Ασφαλής κατάσταση:** υπάρχει μια τουλάχιστον ακολουθία εκτέλεσης των διεργασιών που μπορεί να εκτελεστεί μέχρι το τέλος (δηλαδή δεν οδηγεί σε αδιέξοδο).
  - **Μη ασφαλής κατάσταση:** είναι η κατάσταση που δεν είναι ασφαλής

Ασφαλής Κατάσταση (1/2)

- Όταν μια διεργασία απαιτεί ένα διαθέσιμο πόρο, το σύστημα πρέπει να αποφασίσει αν η άμεση ανάθεση του πόρου θα το διατηρήσει σε ασφαλή κατάσταση.
- Το σύστημα είναι σε ασφαλή κατάσταση αν υπάρχει μια ασφαλής ακολουθία για όλες τις διεργασίες.
- Η ακολουθία $<P_1, P_2, \ldots, P_n>$ είναι ασφαλής αν για κάθε P_i, οι πόροι που η P_i μπορεί ακόμη να απαιτήσει μπορούν να ικανοποιηθούν από τους τρέχοντες διαθέσιμους πόρους συν τους πόρους που δεσμεύονται από όλες τις διεργασίες P_j, με $j < i$.

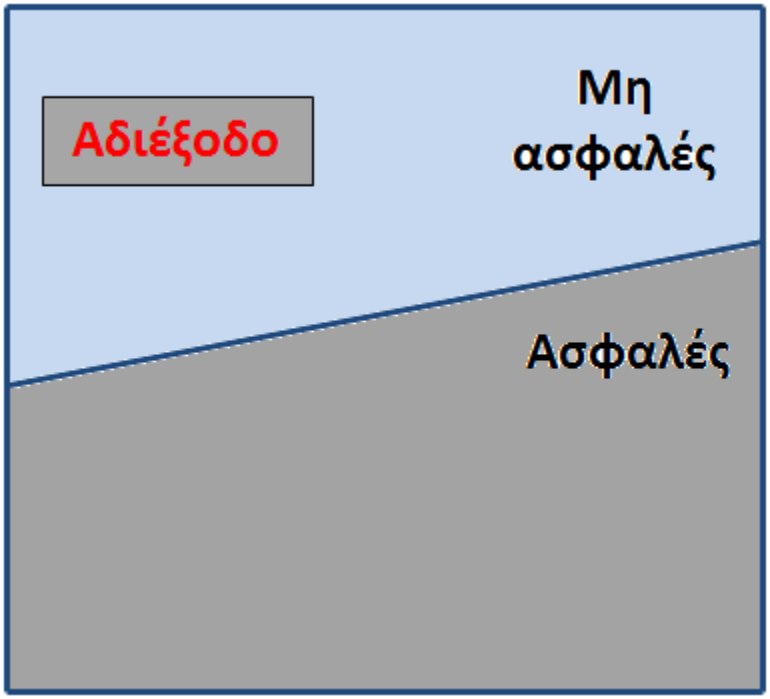
Ασφαλής Κατάσταση (2/2)

- Αν οι ανάγκες σε πόρους της P_i δεν είναι άμεσα διαθέσιμοι, τότε η P_i μπορεί να περιμένει μέχρις ότου να τελειώσουν όλες οι P_j.
- Όταν τελειώνει η P_j η P_i μπορεί να αποκτήσει τους πόρους που χρειάζεται, να εκτελεστεί, να επιστρέψει τους πόρους που της ανατέθηκαν και να τερματιστεί.
- Όταν τερματίσει η $P_{i,}$ η P_{i+1} μπορεί να αποκτήσει τους πόρους που χρειάζεται κ.ο.κ.

Βασικά Γεγονότα (1/2)

- Αν το σύστημα είναι σε ασφαλή κατάσταση δεν υπάρχει αδιέξοδο.
- Αν το σύστημα δεν είναι σε ασφαλή κατάσταση υπάρχει πιθανότητα αδιεξόδου.
- **Αποφυγή αδιεξόδου:** εξασφάλιση ότι το σύστημα δε θα εισέλθει ποτέ σε μη ασφαλή κατάσταση.

Βασικά Γεγονότα (2/2)



Ασφαλή κατάσταση, Μη ασφαλή, Αδιέξοδο

Παραδοχές

- Πολλαπλά στιγμιότυπα των πόρων.
- Κάθε διεργασία πρέπει εκ των προτέρων να διεκδικεί τη μέγιστη χρήση των πόρων.
- Όταν μια διεργασία απαιτεί έναν πόρο ίσως χρειαστεί να περιμένει.
- Όταν μια διεργασία λάβει όλους τους πόρους πρέπει να τους επιστρέψει σε πεπερασμένο χρονικό διάστημα.
- Η μέγιστη απαίτηση για πόρους πρέπει να δηλώνεται εκ των προτέρων.
- Οι σημαντικές διεργασίες πρέπει να είναι ανεξάρτητες και δεν υπόκεινται σε απαιτήσεις συγχρονισμού.
- Πρέπει να υπάρχει ένας σταθερός αριθμός πόρων προς ανάθεση.

Αλγόριθμος του Τραπεζίτη Δομές Δεδομένων (1/2)

- n = το πλήθος των διεργασιών.
- m = το πλήθος τύπων πόρων.
- **Available:** Διάνυσμα μήκους m.

  Αν available [j] = k, υπάρχουν k στιγμιότυπα του τύπου πόρου R_j διαθέσιμα.

- **Claim:** n x m πίνακας.

  Αν Claim [i,j] = k, τότε η διεργασία P_i μπορεί να απαιτήσει κατά μέγιστο k στιγμιότυπα του τύπου πόρου R_j.

Αλγόριθμος του Τραπεζίτη Δομές Δεδομένων (2/2)

- **Allocation:** n x m πίνακας.

  Αν Allocation[i, j] = k τότε στη διεργασία P_i εκχωρούνται k στιγμιότυπα του πόρου R_j.

- **Need:** n x m πίνακας.

  Αν Need[i, j] = k, τότε η διεργασία P_i μπορεί να χρειαστεί k επιπλέον στιγμιότυπα του πόρου R_j για να ολοκληρωθεί.

  Need [i,j] = Claim[i,j] – Allocation [i,j]

Καθορισμούς μιας ασφαλούς κατάστασης (Παράδειγμα) (1/3)

- Υπάρχουν 3 τύποι πόρων με πλήθος:

  R(1) = 9, R(2) = 3, R(3) = 6

- Είναι η παρακάτω κατάσταση ασφαλής;

Πίνακας απαιτήσεων

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Πίνακας ανάθεσης

R1	R2	R3
1	0	0
6	1	2
2	1	1
0	0	2

Συνολικά

R1	R2	R3
9	3	6

Διαθέσιμα

0	**1**	**1**

Καθορισμούς μιας ασφαλούς κατάστασης (Παράδειγμα) (2/3)

Πίνακας απαιτήσεων

	R1	R2	R3
P1	3	2	2
P2	0	0	0
P3	3	1	4
P4	4	2	2

Πίνακας ανάθεσης

R1	R2	R3
1	0	0
0	0	0
2	1	1
0	0	2

Συνολικά

R1	R2	R3
9	3	6

Διαθέσιμα

6	**2**	**3**

Πίνακας απαιτήσεων

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	3	1	4
P4	4	2	2

Πίνακας ανάθεσης

R1	R2	R3
0	0	0
0	0	0
2	1	1
0	0	2

Συνολικά

R1	R2	R3
9	3	6

Διαθέσιμα

7	**2**	**3**

Καθορισμούς μιας ασφαλούς κατάστασης (Παράδειγμα) (3/3)

Πίνακας απαιτήσεων

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	4	2	2

Πίνακας ανάθεσης

R1	R2	R3
0	0	0
0	0	0
0	0	0
0	0	2

Συνολικά

R1	R2	R3
9	3	6

Διαθέσιμα

9	**3**	**4**

Πίνακας απαιτήσεων

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	0	0	0

Πίνακας ανάθεσης

R1	R2	R3
0	0	0
0	0	0
0	0	0
0	0	0

Συνολικά

R1	R2	R3
9	3	6

Διαθέσιμα

9	**3**	**6**

Η κατάσταση είναι ασφαλή: P2 -> P1 -> P3 -> P4.

Παράδειγμα (1/13)

Μεγαλύτερη αξίωση (Maximum claim)

Διεργασία	R_0	R_1	R_2	R_3
p_0	3	2	1	4
p_1	0	2	5	2
p_2	5	1	0	5
p_3	1	5	3	0
p_4	3	0	3	3

Δεσμευμένοι πόροι (Allocated resources)

Διεργασία	R_0	R_1	R_2	R_3
p_0	2	0	1	1
p_1	0	1	2	1
p_2	4	0	0	3
p_3	0	2	1	0
p_4	1	0	3	0

Συνολικά: <8, 5, 9, 7>

Παράδειγμα (2/13)

Μεγαλύτερη αξίωση (Maximum claim)

Διεργασία	R0	R1	R2	R3
p_0	3	2	1	4
p_1	0	2	5	2
p_2	5	1	0	5
p_3	1	5	3	0
p_4	3	0	3	3

Δεσμευμένα = <7, 3, 7, 5>

Δεσμευμένοι πόροι (Allocated resources)

Διεργασία	R0	R1	R2	R3
p_0	2	0	1	1
p_1	0	1	2	1
p_2	4	0	0	3
p_3	0	2	1	0
p_4	1	0	3	0
Σύνολο	**7**	**3**	**7**	**5**

Συνολικά: <8, 5, 9, 7>

Παράδειγμα (3/13)

Μεγαλύτερη αξίωση (Maximum claim)

Διεργασία	R0	R1	R2	R3
p0	3	2	1	4
p1	0	2	5	2
p2	5	1	0	5
p3	1	5	3	0
p4	3	0	3	3

Δεσμευμένα = <7, 3, 7, 5>

Διαθέσιμες μονάδες: Συνολικά - Δεσμευμένες

Διαθέσιμα = <8-7, 5-3, 9-7, 7-5> = <1, 2, 2, 2>

Δεσμευμένοι πόροι (Allocated resources)

Διεργασία	R0	R1	R2	R3
p0	2	0	1	1
p1	0	1	2	1
p2	4	0	0	3
p3	0	2	1	0
p4	1	0	3	0
Σύνολο	**7**	**3**	**7**	**5**

Συνολικά: <8, 5, 9, 7>

Παράδειγμα (4/13)

Maximum claim (Maximum claim)

Διεργασία	R0	R1	R2	R3
p0	3	2	1	4
p1	0	2	5	2
p2	5	1	0	5
p3	1	5	3	0
p4	3	0	3	3

Δεσμευμένα = <7, 3, 7, 5>

Διαθέσιμα = <8-7, 5-3, 9-7, 7-5>
= <1, 2, 2, 2>

Can p0's maxc be met?

maxc[0,0]-alloc'[0,0] = 3-2 = 1<=1 = avail[0]

maxc[0,1]-alloc'[0,1] = 2-0 = 2<=2 = avail[1]

maxc[0,2]-alloc'[0,2] = 1-1 = 0<=2 = avail[2]

maxc[0,3]-alloc'[0,3] = 4-1 = 3<=2 = avail[3]

Process p0 Fails on avail[3].

Δεσμευμένοι πόροι (Allocated resources)

Διεργασία	R0	R1	R2	R3
p0	2	0	1	1
p1	0	1	2	1
p2	4	0	0	3
p3	0	2	1	0
p4	1	0	3	0
Σύνολο	**7**	**3**	**7**	**5**

Συνολικά: <8, 5, 9, 7>

Παράδειγμα (5/13)

Maximum claim (Maximum claim)

Διεργασία	R0	R1	R2	R3
p0	3	2	1	4
p1	0	2	5	2
p2	5	1	0	5
p3	1	5	3	0
p4	3	0	3	3

Δεσμευμένα = <7, 3, 7, 5>

Διαθέσιμα = <8-7, 5-3, 9-7, 7-5>
= <1, 2, 2, 2>

Can p1's maxc be met?

maxc[1,0]-alloc'[1,0] = 0-0 = 0<=1 = avail[0]

maxc[1,1]-alloc'[1,1] = 2-1 = 1<=2 = avail[1]

maxc[1,2]-alloc'[1,2] = 5-2 = 3<=2 = avail[2]

maxc[1,3]-alloc'[1,3] = 2-1 = 1<=2 = avail[3]

Process p1 Fails on avail[2].

Δεσμευμένοι πόροι (Allocated resources)

Διεργασία	R0	R1	R2	R3
p0	2	0	1	1
p1	0	1	2	1
p2	4	0	0	3
p3	0	2	1	0
p4	1	0	3	0
Σύνολο	**7**	**3**	**7**	**5**

Συνολικά: <8, 5, 9, 7>

Παράδειγμα (6/13)

Maximum claim (Maximum claim)

Διεργασία	R0	R1	R2	R3
p0	3	2	1	4
p1	0	2	5	2
p2	5	1	0	5
p3	1	5	3	0
p4	3	0	3	3

Δεσμευμένα = <7, 3, 7, 5>

Διαθέσιμα = <8-7, 5-3, 9-7, 7-5>
= <1, 2, 2, 2>

Can p2's maxc be met?

maxc[2,0]-alloc'[2,0] = 5-4 = 1<=1 = avail[0]

maxc[2,1]-alloc'[2,1] = 1-0 = 1<=2 = avail[1]

maxc[2,2]-alloc'[2,2] = 0-0 = 0<=2 = avail[2]

maxc[2,3]-alloc'[2,3] = 5-3 = 2<=2 = avail[3]

Δεσμευμένοι πόροι (Allocated resources)

Διεργασία	R0	R1	R2	R3
p0	2	0	1	1
p1	0	1	2	1
p2	4	0	0	3
p3	0	2	1	0
p4	1	0	3	0
Σύνολο	**7**	**3**	**7**	**5**

Συνολικά: <8, 5, 9, 7>

Παράδειγμα (7/13)

Maximum claim (Maximum claim)

Διεργασία	R0	R1	R2	R3
p0	3	2	1	4
p1	0	2	5	2
p2	5	1	0	5
p3	1	5	3	0
p4	3	0	3	3

Δεσμευμένα = <7, 3, 7, 5>

Διαθέσιμα = <8-7, 5-3, 9-7, 7-5>
= <1, 2, 2, 2>

Can p2's maxc be met?

maxc[2,0]-alloc'[2,0] = 5-4 = 1<=1 = avail[0]

maxc[2,1]-alloc'[2,1] = 1-0 = 1<=2 = avail[1]

maxc[2,2]-alloc'[2,2] = 0-0 = 0<=2 = avail[2]

maxc[2,3]-alloc'[2,3] = 5-3 = 2<=2 = avail[3]

Yes. Redo avail/Update alloc'.

avail[0] = avail[0]+alloc'[2,0] = 1+4 = 5

avail[1] = avail[1]+alloc'[2,1] = 2+0 = 2

avail[2] = avail[2]+alloc'[2,2] = 2+0 = 2

avail[3] = avail[3]+alloc'[2,3] = 2+3 = 5

Δεσμευμένοι πόροι (Allocated resources)

Διεργασία	R0	R1	R2	R3
p0	2	0	1	1
p1	0	1	2	1
p2	4	0	0	3
p3	0	2	1	0
p4	1	0	3	0
Σύνολο	**7**	**3**	**7**	**5**

Συνολικά: <8, 5, 9, 7>

Παράδειγμα (8/13)

Maximum claim (Maximum claim)

Διεργασία	R0	R1	R2	R3
p0	3	2	1	4
p1	0	2	5	2
p2	5	1	0	5
p3	1	5	3	0
p4	3	0	3	3

Δεσμευμένα = <3, 3, 7, 2>

Διαθέσιμα = <8-3, 5-3, 9-7, 7-2>
= <5, 2, 2, 5>

Can anyone's maxc be met?

maxc[4,0]-alloc'[4,0] = 3-1 = 2□5 = avail[0]

maxc[4,1]-alloc'[4,1] = 0-0 = 0□2 = avail[1]

maxc[4,2]-alloc'[4,2] = 3-3 = 0□2 = avail[2]

maxc[4,3]-alloc'[4,3] = 3-0 = 3□5 = avail[3]

Δεσμευμένοι πόροι (Allocated resources)

Διεργασία	R0	R1	R2	R3
p0	2	0	1	1
p1	0	1	2	1
p2	0	0	0	0
p3	0	2	1	0
p4	1	0	3	0
Σύνολο	**3**	**3**	**7**	**2**

Συνολικά: <8, 5, 9, 7>

Παράδειγμα (9/13)

Maximum claim (Maximum claim)

Διεργασία	R0	R1	R2	R3
p_0	3	2	1	4
p_1	0	2	5	2
p_2	5	1	0	5
p_3	1	5	3	0
p_4	3	0	3	3

Δεσμευμένα = <3, 3, 7, 2>

Διαθέσιμα = <8-3, 5-3, 9-7, 7-2>
= <5, 2, 2, 5>

Can anyone's maxc be met?

maxc[4,0]-alloc'[4,0] = 3-1 = 2□5 = avail[0]
maxc[4,1]-alloc'[4,1] = 0-0 = 0□2 = avail[1]
maxc[4,2]-alloc'[4,2] = 3-3 = 0□2 = avail[2]
maxc[4,3]-alloc'[4,3] = 3-0 = 3□5 = avail[3]

P_4 can exercise max claim

avail[0] = avail[0]+alloc'[4,0] = 5+1 = 6
avail[1] = avail[1]+alloc'[4,1] = 2+0 = 2
avail[2] = avail[2]+alloc'[4,2] = 2+3 = 5
avail[3] = avail[3]+alloc'[4,3] = 5+0 = 5

Δεσμευμένοι πόροι (Allocated resources)

Διεργασία	R0	R1	R2	R3
p_0	2	0	1	1
p_1	0	1	2	1
p_2	0	0	0	0
p_3	0	2	1	0
p_4	1	0	3	0
Σύνολο	**3**	**3**	**7**	**2**

Συνολικά: <8, 5, 9, 7>

Παράδειγμα (10/13)

Maximum claim
(Maximum claim)

Διεργασία	R0	R1	R2	R3
p0	3	2	1	4
p1	0	2	5	2
p2	5	1	0	5
p3	1	5	3	0
p4	3	0	3	3

Δεσμευμένα = <2, 3, 4, 2>

Διαθέσιμα = <8-2, 5-3, 9-4, 7-2>
= <6, 2, 5, 5>

Can anyone's maxc be met?

Ναι. Όλοι μπορούν.

Επέλεξε p0.

Δεσμευμένοι πόροι
(Allocated resources)

Διεργασία	R0	R1	R2	R3
p0	2	0	1	1
p1	0	1	2	1
p2	0	0	0	0
p3	0	2	1	0
p4	0	0	0	0
Σύνολο	**2**	**3**	**4**	**2**

Συνολικά: <8, 5, 9, 7>

Παράδειγμα (11/13)

Maximum claim (Maximum claim)

Διεργασία	R0	R1	R2	R3
p0	3	2	1	4
p1	0	2	5	2
p2	5	1	0	5
p3	1	5	3	0
p4	3	0	3	3

Δεσμευμένα = <0, 3, 3, 1>

Διαθέσιμα = <8-0, 5-3, 9-3, 7-1>
= <8, 2, 6, 6>

Can anyone's maxc be met?

Ναι. Όλοι μπορούν.

Επέλεξε p1.

Δεσμευμένοι πόροι (Allocated resources)

Διεργασία	R0	R1	R2	R3
p0	0	0	0	0
p1	0	1	2	1
p2	0	0	0	0
p3	0	2	1	0
p4	0	0	0	0
Σύνολο	**0**	**3**	**3**	**1**

Συνολικά: <8, 5, 9, 7>

Παράδειγμα (12/13)

Maximum claim (Maximum claim)

Διεργασία	R0	R1	R2	R3
p0	3	2	1	4
p1	0	2	5	2
p2	5	1	0	5
p3	1	5	3	0
p4	3	0	3	3

Δεσμευμένα = <0, 2, 1, 0>

Διαθέσιμα = <8-0, 5-2, 9-1, 7-0>
= <8, 3, 8, 7>

Can anyone's maxc be met?

Ναι. Επέλεξε p3.

Δεσμευμένοι πόροι (Allocated resources)

Διεργασία	R0	R1	R2	R3
p0	0	0	0	0
p1	0	0	0	0
p2	0	0	0	0
p3	0	2	1	0
p4	0	0	0	0
Σύνολο	**0**	**2**	**1**	**0**

Συνολικά: <8, 5, 9, 7>

Παράδειγμα (13/13) Περίληψη

Maximum claim (Maximum claim)

Διεργασία	R0	R1	R2	R3
p0	3	2	1	4
p1	0	2	5	2
p2	5	1	0	5
p3	1	5	3	0
p4	3	0	3	3

- Alloc' Now Zeroed.
- Ασφαλή κατάσταση με ολοκλήρωση όλων των διεργασιών.
- Halt Algorithm με επιτυχία.
- Σειρά διεργασιών των:
  - **p2, p4, p0, p1, p3.**
  - Ασφαλή εκτέλεση.
- All Maximum Claims Satisfied in Some Order.
- Δεν υπάρχει αδιέξοδο ή μη-ασφαλή κατάσταση.

Δεσμευμένοι πόροι (Allocated resources)

Διεργασία	R0	R1	R2	R3
p0	0	0	0	0
p1	0	0	0	0
p2	0	0	0	0
p3	0	0	0	0
p4	0	0	0	0
Σύνολο	**0**	**0**	**0**	**0**

Ανίχνευση Αδιεξόδου (1/2)

- **Ο αλγόριθμος του τραπεζίτη είναι απαισιόδοξος**

  Πάντοτε υποθέτει ότι μια διεργασία δεν θα απελευθερώσει τους πόρους που κατέχει μέχρι να αποκτήσει όλους τους πόρους που χρειάζεται.

- **Συνέπειες**
  - Μικρό ποσό παραλληλίας.
  - Πολύπλοκοι έλεγχοι για κάθε απαίτηση εκχώρησης πόρου (πολυπλοκότητα $O(n^2)$).

- Οι **στρατηγικές ανίχνευσης αδιεξόδου** δεν οριοθετούν την πρόσβαση σε πόρους και δεν περιορίζουν τις ενέργειες των διεργασιών.

- Οι απαιτούμενοι πόροι εκχωρούνται στις διεργασίες, όποτε αυτό είναι δυνατό.

Ανίχνευση Αδιεξόδου (2/2)

- Περιοδικά το Λειτουργικό Σύστημα εφαρμόζει έναν αλγόριθμο ανίχνευσης αδιεξόδου που δίνει τη δυνατότητα να εντοπισθεί η συνθήκη της κυκλικής αναμονής.
- Ο αλγόριθμος μπορεί να εκτελέιται με βάση:
  - Πόσο συχνά συνηθίζεται να συμβαίνει αδιέξοδο.
  - Σε κάθε αίτηση πόρου.
- Αν ο αλγόριθμος καλείται αυθαίρετα δεν εξασφαλίζεται η ανίχνευση της διεργασίας που προκαλεί το αδιέξοδο

Στρατηγικές όταν ανιχνευθεί αδιέξοδο

- Διακοπή όλων των διεργασιών που περιήλθαν σε αδιέξοδο

  Η πιο κοινή τακτική στα Λειτουργικά Συστήματα.

- Δημιουργία αντιγράφου ασφαλείας κάθε διεργασίας που βρίσκεται σε αδιέξοδο, σε κάποιο προηγούμενο σημείο ελέγχου και επανεκκίνηση όλων των διεργασιών.
  - Το αρχικό αδιέξοδο μπορεί να ξανασυμβεί.
- Διαδοχική διακοπή όλων των διεργασιών που βρίσκονται σε αδιέξοδο ώστε να μην υπάρχει πλέον αδιέξοδο.
- Διαδοχική προεκχώρηση πόρων έως ότου δε θα υπάρχει αδιέξοδο.

Κριτήρια επιλογής των διεργασιών που βρίσκονται σε αδιέξοδο

Επιλέγεται η διεργασία που έχει:

- Καταναλώσει το μικρότερο ποσό χρόνου επεξεργασίας μέχρι την τρέχουσα στιγμή.
- Παράγει το μικρότερο πλήθος γραμμών εξόδου ποσό μέχρι την τρέχουσα στιγμή.
- Το μεγαλύτερο εκτιμώμενο χρόνο.
- Το μικρότερο πλήθος πόρων που της έχουν εκχωρηθεί.
- Τη μικρότερη προτεραιότητα.

Συνδυασμένη προσέγγιση ανίχνευσης αδιεξόδου

- Ο συνδυασμός των τριών προσεγγίσεων:
  - πρόληψη,
  - αποφυγή,
  - ανίχνευση,

  επιτρέπει τη χρήση της βέλτιστης προσέγγισης για κάθε πόρο του συστήματος.

- Μια κατά το δυνατόν βέλτιστη προσέγγιση περιλαμβάνει:
  - Διαμοίραση των πόρων σε ιεραρχικά διατεταγμένες κλάσεις.
  - Χρήση της στρατηγικής γραμμικής διάταξης για την αποτροπή της κυκλικής αναμονής.
  - Χρήση των πλέον κατάλληλων τεχνικών για τη διαχείριση αδιεξόδων μέσα σε κάθε κλάση.

Συνδυασμένη Προσέγγιση (Παράδειγμα) (1/2)

- **Χώρος εναλλαγής:** Τμήματα μνήμης στο δίσκο που χρησιμοποιούνται κατά την εναλλαγή (swapping) των διεργασιών.

  Αποτροπή αδιεξόδου απαιτώντας ότι όλες οι απαιτήσεις πόρων θα ικανοποιηθούν την ίδια στιγμή, όπως στην περίπτωση της στρατηγικής για κατοχή και αναμονή. Η αποφυγή του αδιεξόδου θα μπορούσε επίσης να χρησιμοποιηθεί.

- **Πόροι διεργασίας:** Οι συσκευές που ανατίθενται.

  Η αποφυγή είναι εφαρμόσιμη καθώς είναι λογική απαίτηση οι διεργασίες να δηλώνουν εκ των προτέρων τους πόρους που θα χρειαστούν. Η πρόληψη με την έννοια της διάταξης των πόρων είναι επίσης εφικτή.

Συνδυασμένη Προσέγγιση (Παράδειγμα) (2/2)

- **Κύρια μνήμη**: Ανατίθεται στις διεργασίες με τη μορφή σελίδων (pages) ή τμημάτων (segments).

  Η πρόληψη μέσω της προ-εκχώρησης είναι η πλέον κατάλληλη τακτική. Όταν μια διεργασία προ-εκχωρείται εναλάσσεται στο δίσκο και ελευθερώνει τη μνήμη για να επιλύσει το αδιέξοδο.

- **Εσωτερικοί πόροι**: Για παράδειγμα κανάλια Εισόδου/Εξόδου.

  Η αποτροπή με την έννοια της διάταξης πόρων μπορεί να εφαρμοστεί.

Άσκηση (1/2)

Θεωρείστε ένα σύστημα με 4 διεργασίες (A, B, C, D) και 5 επαναχρησιμοποιήσιμους πόρους (R0, R1, R2, R3, R4). Η τρέχουσα ανάθεση πόρων (τρέχουσα κατάσταση) στις διεργασίες δίνεται από τον παρακάτω πίνακα:

Εκχώρηση	R0	R1	R2	R3	R4
A	1	0	2	1	1
B	2	0	1	1	0
C	1	1	0	1	0
D	1	1	1	1	0

Άσκηση (2/2)

Θεωρείστε επίσης ότι οι μέγιστες απαιτήσεις των διεργασιών δίνονται από τον παρακάτω πίνακα:

Μέγιστη απαίτηση	R0	R1	R2	R3	R4
A	1	1	2	1	2
B	2	2	2	1	0
C	2	1	3	1	0
D	1	1	2	2	1

Οι διαθέσιμοι προς εκχώρηση πόροι είναι: **0 0 x 1 1**

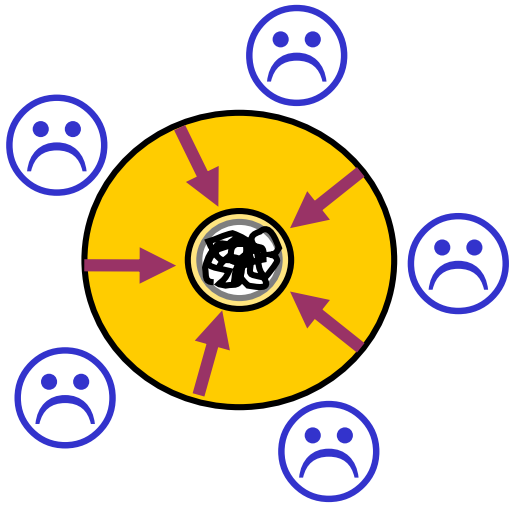
Να βρεθεί η μικρότερη τιμή **x** για την οποία το σύστημα θα οδηγηθεί σε ασφαλή κατάσταση.

Το πρόβλημα των συνδαιτημόνων φιλοσόφων

Πέντε φιλόσοφοι κάθονται γύρω από ένα κυκλικό τραπέζι. Κάθε φιλόσοφος καταναλώνει το χρόνο του διαδοχικά σκεπτόμενος και τρώγοντας. Στο κέντρο του τραπεζιού υπάρχει ένα μεγάλο πιάτο με spaghetti. Κάθε φιλόσοφος χρειάζεται δύο πηρούνια (forks) για να φάει λίγο spaghetti.

Υπάρχει ένα πηρούνι ανάμεσα σε κάθε ζεύγος φιλοσόφων και όλοι συμφωνούν ότι θα χρησιμοποιούν μόνον τα πηρούνια που βρίσκονται δεξιά και αριστερά από τον καθένα.

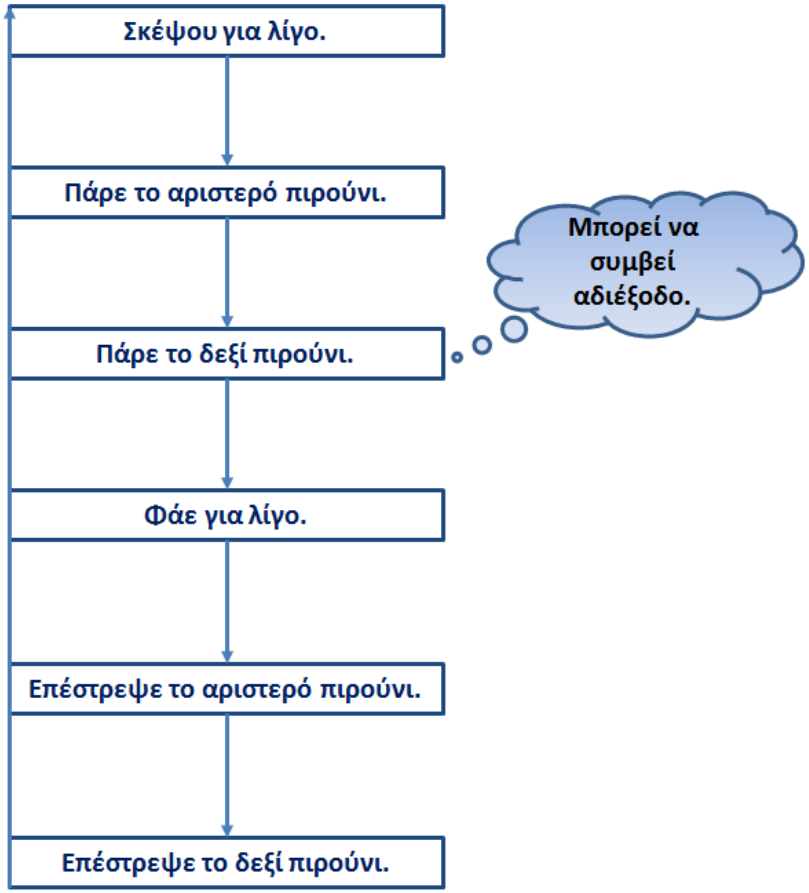
Κάθε φιλόσοφος είναι μια διεργασία και κάθε πηρούνι είναι ένας διαμοιραζόμενος πόρος με ενέργειες δέσμευσης και απελευθέρωσης. Αν ένας φιλόσοφος πεινάσει, πρέπει πρώτα να πάρει τα πηρούνια δεξιά και αριστερά του δεξιό για να μπορέσει να ξεκινήσει να τρώει.

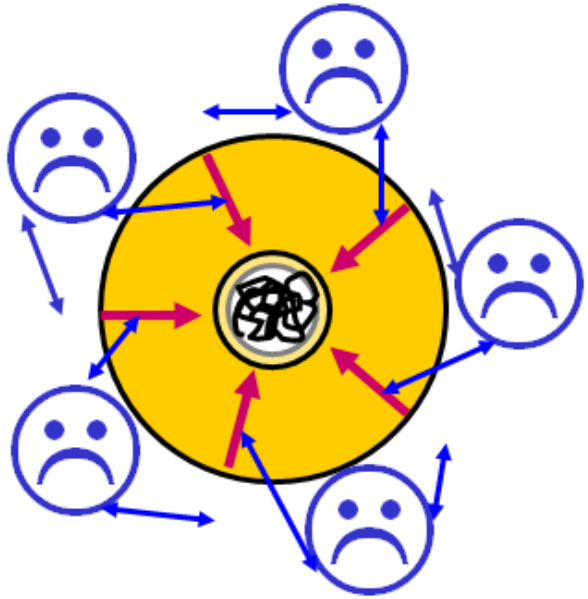


Σύνδεση του προβλήματος με τα αδιέξοδα στα Λειτουργικά Συστήματα

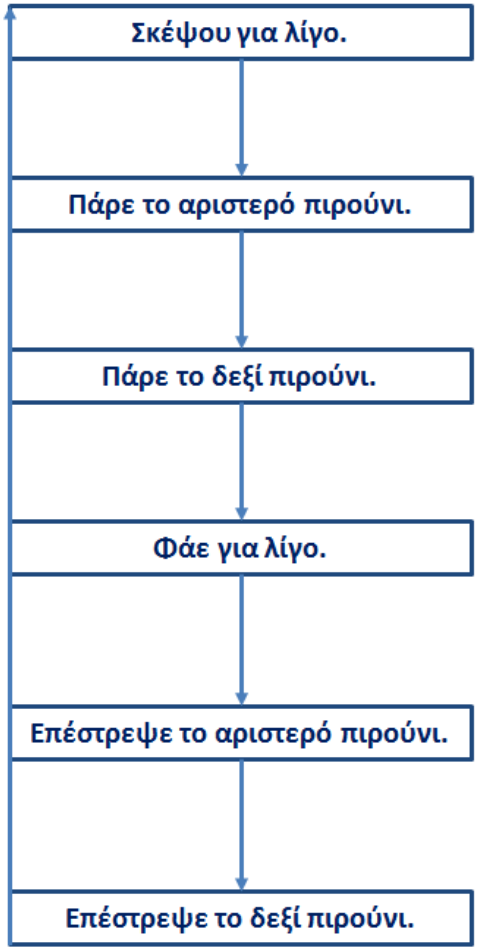
- Το πρόβλημα αυτό είναι ένα πρότυπο που χρησιμοποιείται για την αποτίμηση μεθόδων σχετικών με το συγχρονισμό ταυτόχρονων διεργασιών.
- Καταδεικνύει τη δυσκολία της εκχώρησης πόρων μεταξύ διεργασιών χωρίς αδιέξοδα και παρατεταμένες στερήσεις.
- Στόχος είναι η ανάπτυξη ενός πρωτοκόλλου απόκτησης των πηρουνιών που θα εξασφαλίζει:
  - Την απαλλαγή από αδιέξοδα.
  - Τη δικαιοσύνη: κανένας φιλόσοφος δεν πρέπει να υποφέρει από παρατεταμένη στέρηση.
  - Το μέγιστο δυνατό συγχρονισμό.

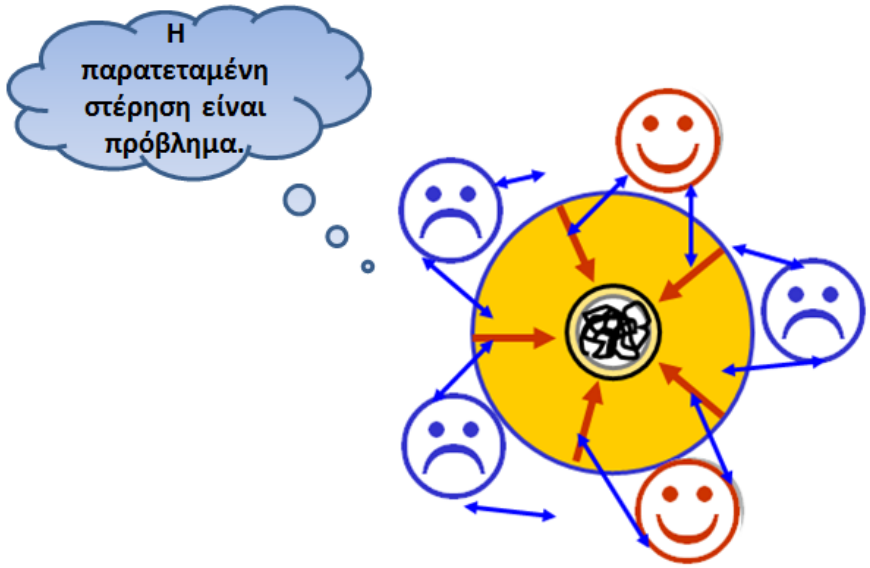
Το πρόβλημα των συνδαιτημόνων φιλοσόφων Λύση (1/2)





Το πρόβλημα των συνδαιτημόνων φιλοσόφων
Λύση (2/2)





Το πρόβλημα των συνδαιτημόνων φιλοσόφων Πρώτη Λύση – Προφανής (1/2)

```
/* program diningphilosophers – deadlock and starvation*/
semaphore fork[5]={1};
int i;
void philosopher (int i)
{
    while (true)
    {
            think();
            wait (fork[i]);
            wait (fork[(i+1) mod 5]);
            eat();
            signal(fork[(i+1) mod 5]);
            signal(fork[i]);
    }
}
void main()
{parbegin(philosopher(0),philosopher(1),philosopher(2),
                    philosopher(3),philosopher(4));}
```

Το πρόβλημα των συνδαιτημόνων φιλοσόφων Πρώτη Λύση – Προφανής (2/2)

- Προσθήκη ενός ακόμη πηρουνιού.
- Μέγιστος αριθμός 4 φιλοσόφων στο τραπέζι (κυκλική αναμονή).
- Να εκτελείται διαφορετική αλληλουχία ενεργειών για τους φιλοσόφους με άρτιο και περιττό αύξοντα αριθμό, δηλαδή δημιουργία δύο ομάδων φιλοσόφων. Η μία ομάδα (περιττός α/α) θα αποκτά πρώτα το δεξιό και μετά το αριστερό πηρούνι και η άλλη ομάδα (άρτιος α/α) πρώτα το αριστερό και μετά το δεξιό πηρούνι.
- Ένας φιλόσοφος επιτρέπεται να αποκτήσει τα πηρούνια μόνον όταν και τα δύο είναι διαθέσιμα διαφορετικά ελευθερώνει το πιρούνι που πιθανώς να κατέχει (κρίσιμο τμήμα) - (κατοχή και αναμονή).
- Σχεδιασμός του συστήματος έτσι ώστε ένας φιλόσοφος να “κλέψει” ένα πηρούνι που δεν είναι γειτονικό του.
- Αλγόριθμος Lehmann-Rabin (non deterministic).

Το πρόβλημα των συνδαιτημόνων φιλοσόφων Δεύτερη Λύση

```
/* program diningphilosophers – no deadlock – no starvation */
semaphore fork[5]={1};
semaphore room={4};
int i;
void philosopher (int i)
{
    while (true)
    {
            think();
            wait(room);
            wait (fork[i]);
            wait (fork[(i+1) mod 5]);
            eat();
            signal(fork[(i+1) mod 5]);
            signal(fork[i]);
            signal(room);
    }
}
void main()
{parbegin(philosopher(0),philosopher(1),philosopher(2),
                philosopher(3),philosopher(4));}
```

Πόροι
Προσεγγίσεις και Πολιτικές (1/2)

Προσέγγιση	Πολιτική ανάθεσης πόρων	Διαφορετικά σχήματα	Πλεονεκτήματα	Μειονεκτήματα
Αποτροπή	**Συντηρητική** υπο-παραχωρεί πόρους	Απαιτεί όλους τους πόρους με τη μία.	✓ Αποτελεσματική για διεργασίες που πραγματοποιούν μια δραστηριότητα. ✓ Δεν είναι απαραίτητη η προ-εκχώρηση.	✓ Μη αποδοτική. ✓ Καθυστερεί την έναρξη των διεργασιών. ✓ Μελλοντικές απαιτήσεις σε πόρους πρέπει να είναι γνωστές για τις διεργασίες.
		Προ-εκχώρηση.	Βολική στην περίπτωση πόρων που η κατάσταση τους μπορεί να αποθηκεύεται και εύκολα να ανακτάται.	Προ-εκχωρεί περισσότερο συχνά από ότι χρειάζεται.
		Διάταξη πόρων.	✓ Επιβάλλεται μέσω ελέγχων κατά τη μεταγλώττιση. ✓ Δεν απαιτεί υπολογισμούς καθώς το πρόβλημα επιλύεται κατά το σχεδιασμό του συστήματος.	Δεν επιτρέπει κλιμακούμενες αιτήσεις πόρων.

Πόροι
Προσεγγίσεις και Πολιτικές (2/2)

Προσέγγιση	Πολιτική ανάθεσης πόρων	Διαφορετικά σχήματα	Πλεονεκτήματα	Μειονεκτήματα
Αποφυγή	Ενδιάμεση λύση ανάμεσα στον εντοπισμό και την πρόληψη.	Επιδιώκει να βρει έστω ένα ασφαλές μονοπάτι.	Δεν απαιτείται προ-εκχώρηση.	✓ Μελλοντικές απαιτήσεις πόρων πρέπει να είναι γνωστές στο Λειτουργικό Σύστημα. ✓ Διεργασίες μπορούν να αναστέλλονται για μεγάλα διαστήματα.
Εντοπισμός	Πολύ φιλελεύθερος: οι απαιτούμενοι πόροι ανατίθενται όπου είναι εφικτό.	Καλείται περιοδικά για τον έλεγχο αδιεξόδων.	✓ Δεν καθυστερείται ποτέ η έναρξη διεργασιών. ✓ Διευκολύνει τον απευθείας χειρισμό.	Απώλειες λόγω προ-εκχώρησης.

Αναφορές

[1]. Stallings William, “Operating systems: Internal and Design Principles”, Fourth edition, Publishing as Prentice Hall, 2000.

[2]. A.S.Tanenbaum, "Σύγχρονα Λειτουργικά Συστήματα" Τόμος Α', Εκδόσεις, Παπασωτηρίου.

[3]. H.M. Deitel, "Operating Systems", 2nd edition, Addison-Wesley Publishing Company.

[3]. W. Stallings, Λειτουργικά Συστήματα Αρχές Σχεδίασης, 6η έκδοση, ΕΚΔΟΣΕΙΣ ΤΖΙΟΛΑ, 2009, Θεσσαλονίκη.

[4]. Silberschatz, Galvin, Gagne , “Λειτουργικά Συστήματα”, ΕΚΔΟΣΕΙΣ ΙΩΝ, 2007, Αθήνα.

Τέλος Ενότητας

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης