MYY601 Λειτουργικά Συστήματα

Εαρινό 2024

Μάθημα 6

Ταυτοχρονισμός: Αδιέξοδο

Τμήμα Μηχανικών Η/Υ & Πληροφορικής
Πανεπιστήμιο Ιωαννίνων

1

Περίγραμμα

- Εισαγωγή
- Πρόληψη
- Αποφυγή
- Ανίχνευση
- Πρόβλημα Συνδαιτυμόνων Φιλοσόφων

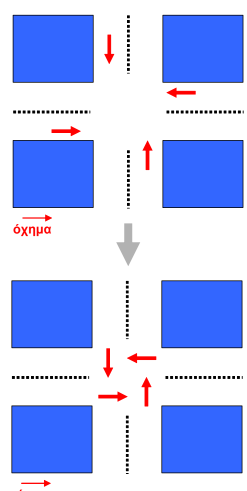
2

Ορισμός

- Αδιέξοδο
  - Μόνιμος αποκλεισμός συνόλου διεργασιών που είτε ανταγωνίζονται για πόρους ή επικοινωνούν μεταξύ τους
  - Κάθε διεργασία στο σύνολο περιμένει για ένα γεγονός που μόνο μια άλλη διεργασία του ίδιου συνόλου μπορεί να προκαλέσει

Παράδειγμα 1

- Τέσσερα οχήματα
  - Κινούνται προς την ίδια διαστάυρωση
  - Φτάνουν την ίδια στιγμή
  - Καθένας δίνει προτεραιότητα σε αυτόν που βρίσκεται δεξιά του (Κ.Ο.Κ.)
  - Τα οχήματα περιμένουν για πάντα
- Πόροι
  - 4 τεταρτημόρια διασταύρωσης
  - Κάθε όχημα έχει το ένα
  - Καθένα χρειάζεται ακόμη ένα



Παράδειγμα 2

- Διεργασία P
  - Κάνει αίτηση για πόρους A και B
- Διεργασία Q
  - Κάνει αίτηση για πόρους B και A
- Αδιέξοδο
  - Η P παίρνει το A και αιτείται το B
  - Η Q παίρνει το B και αιτείται το A
- Λύση:
  Αν θεωρήσουμε ότι η P γίνεται

```
get A
release A
get B
release B
```

Τότε δε μπορεί να συμβεί αδιέξοδο !

Process P	Process Q
get A	get B
…	…
get B	get A
…	…
release A	release B
…	…
release B	release A

Επαναχρησιμοποιήσιμος Πόρος

1. Έχει σταθερό αριθμό από μονάδες (π.χ. επεξεργαστής)
2. Μια μονάδα δε μπορεί να χρησιμοποιηθεί από κοινού
   - Είτε είναι διαθέσιμη Ή
   - Έχει εκχωρηθεί σε μία και μόνο μία διεργασία
3. Μια διεργασία μπορεί να αποδεσμεύσει μόνο μονάδες που της έχουν εκχωρηθεί πιο πριν

Παράδειγμα 1

- Δύο πόροι
  - Δίσκος D_1 και Δίσκος D_2
- Δύο διεργασίες
  - P αιτείται D_2 και D_1
  - Q αιτείται D_1 και D_2
- Αδιέξοδο αν
  - P πάρει μόνο το D_2
  - Q πάρει μόνο το D_1

Παράδειγμα 2

- Πόρος
  - Χώρος μνήμης 200 KB
- Δύο διεργασίες
  - P αιτείται 80+60 KB
  - Q αιτείται 70+80 KB
- Αδιέξοδο αν
  - P πάρει μόνο 80 KB
  - Q πάρει μόνο 70 KB

Καταναλώσιμος Πόρος

1. Το πλήθος των διαθέσιμων μονάδων
   - Ενδεχομένως απεριόριστο
   - Μεταβάλλεται από παραγωγούς/καταναλωτές
2. Ο παραγωγός αυξάνει το πλήθος μονάδων
   - Όταν δημιουργεί και αποδεσμεύει
3. Ο καταναλωτής μειώνει το πλήθος μονάδων
   - Όταν αιτείται και λαμβάνει
   - Δεν επιστρέφει πόρους που έχει λάβει

- Παραδείγματα
  - Διακοπές, σήματα, μηνύματα
- Παράδειγμα
  Διεργασίες P και Q σε κώδικα με σφάλματα
  - Λαμβάνουν μήνυμα πριν στείλουν
  - Μένουν σε αποκλεισμό για πάντα

Παράδειγμα

Process P
```
...
receive (Q)
...
send(Q, M1)
```

Process Q
```
...
receive (P)
...
send(P, M2)
```


7

Αναγκαίες Συνθήκες για Αδιέξοδο

1. Αμοιβαίος αποκλεισμός (mutual exclusion)
   - Μόνο μία διεργασία χρησιμοποιεί τον πόρο κάθε φορά
   - Π.χ. χρειάζεται για συνέπεια δεδομένων
2. Κατοχή και αναμονή (hold and wait)
   - Η διεργασία κατέχει πόρους ενώ περιμένει και άλλους
3. Μη εκτόπιση (no preemption)
   - Ο πόρος δεν αφαιρείται από διεργασία που τον κατέχει
   - Π.χ. χρειάζεται για διασφάλιση ακεραιότητας δεδομένων

- Οι τρεις συνθήκες
  - Αναγκαίες αλλά όχι ικανές για ύπαρξη αδιεξόδου

8

Συνθήκη Αδιεξόδου για Επάρκεια

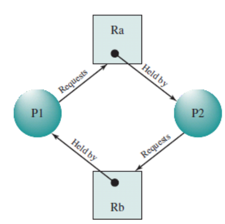
4. Κυκλική αναμονή (circular wait)
   - Κλειστή αλυσίδα από διεργασίες
   - Κάθε διεργασία έχει τουλάχιστο έναν πόρο τον οποίο αιτείται η επόμενη διεργασία
   - Είναι ενδεχόμενο αποτέλεσμα των συνθηκών 1-3
   - Μπορεί να συμβεί ως αποτέλεσμα συγκεκριμένης ακολουθίας αίτησης/αποδέσμευσης πόρων

- Οι τέσσερις συνθήκες
  - Αναγκαίες και ικανές για ύπαρξη αδιεξόδου

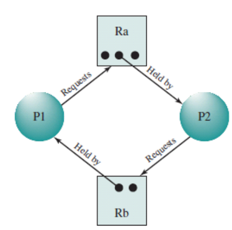
9

Μοντελοποίηση

- Δύο είδη κόμβων
  - *Κύκλος* συμβολίζει διεργασία
  - *Τετράγωνο* συμβολίζει πόρο
  - *Τελεία* συμβολίζει μονάδα πόρου
- Τόξο από διεργασία σε πόρο
  - Η διεργασία αιτείται μονάδα πόρου
- Τόξο από πόρο σε διεργασία
  - Μονάδα πόρου που ζητήθηκε, δόθηκε και κατέχεται
- Υπάρχει αδιέξοδο
  - Αν σχηματίζεται κύκλος και
  - Δεν υπάρχουν διαθέσιμοι πόροι για να διακόψουν το κύκλο



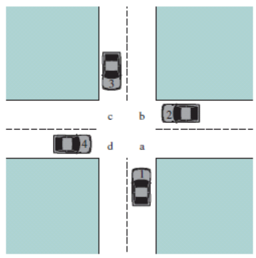
Κύκλος με αδιέξοδο

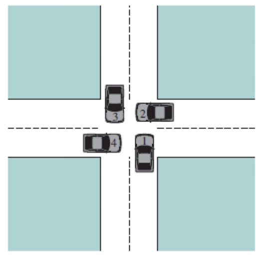


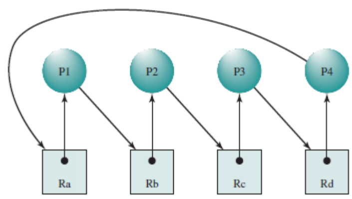
Κύκλος χωρίς αδιέξοδο

10

Επιπλέον Παράδειγμα με Αδιέξοδο







Πρόληψη

- Έμμεση μέθοδος
  - Πρόληψη μίας από τις συνθήκες 1-3
- Άμεση μέθοδος
  - Πρόληψη συνθήκης 4

Πρόληψη 1: αμοιβαίου αποκλεισμού

- Συνήθως δε μπορεί να αρθεί λόγω φυσικών περιορισμών
- Π.χ. χρήση επεξεργαστή, κατανάλωση μηνύματος

Πρόληψη 2: Κατοχής και Αναμονής

- Εφαρμογή
  - Η διεργασία αιτείται όλους τους πόρους μαζί
  - Μένει σε αποκλεισμό μέχρι να της δοθούν ταυτόχρονα όλοι οι πόροι που ζήτησε
- Αδυναμίες
  - Ένας πόρος μπορεί να μείνει αχρησιμοποίητος
  - Μια διεργασία περιμένει χωρίς να είναι απαραίτητο, αν και θα μπορούσε να συνεχίσει με λιγότερους πόρους
  - Μια διεργασία μπορεί να μη γνωρίζει όλους του πόρους που χρειάζεται από την αρχή
  - Δύσκολο να εφαρμοστεί σε αρθρωτά ή ιεραρχικά προγράμματα

Πρόληψη 3: Μη Εκτόπισης

- Θεωρούμε
  - Η διεργασία *P* κατέχει συγκεκριμένους πόρους
  - Η *P* δεν έχει πρόσβαση σε έναν επιπλέον πόρο *A*
- Λύση 1
  - Η διεργασία *P* αποδεσμεύει τους πόρους που κατέχει
  - Τους αιτείται ξανά αργότερα μαζί με τον πόρο *A*
- Λύση 2
  - Το ΛΣ εκτοπίζει τη διεργασία *Q* από τον πόρο *A*
  - Δίνει τον πόρο *A* στη διεργασία *P* που τον ζήτησε

Πρόληψη 4: Κυκλικής Αναμονής

- Ορίζουμε γραμμική διάταξη στους τύπους πόρων
  - Η διεργασία που κατέχει πόρους τύπου R αιτείται μόνο πόρους που βρίσκονται <u>ΜΕΤΑ</u> τον R στη διάταξη
- Γιατί δουλεύει
  - Έστω ο Ri προηγείται του Rj στη διάταξη αν i < j
  - Θεωρούμε ότι οι Α και Β σε αδιέξοδο
    - Η Α κατέχει τον Ri και ζήτησε τον Rj
    - Η Β κατέχει τον Rj και ζήτησε τον Ri
  - Λόγω της διάταξης έχουμε i < j και j < i
  - Αδύνατο !

Αποφυγή Αδιεξόδου

- Ορισμός
  - Σωστές επιλογές που εξασφαλίζουν αποφυγή αδιεξόδου
- Πλεονέκτημα
  - Επιτρέπει τις τρεις αναγκαίες συνθήκες
  - Περισσότερος ταυτοχρονισμός σε σχέση με την πρόληψη
- Περιορισμοί
  - Εκ των προτέρων γνώση των μέγιστων αιτήσεων σε πόρους από κάθε διεργασία
  - Ανεξαρτησία μεταξύ των εμπλεκόμενων διεργασιών
  - Σταθερό πλήθος πόρων που μπορούν να εκχωρηθούν
  - Καμία διεργασία δεν μπορεί να τερματιστεί ενώ κατέχει πόρο

Αποφυγή 1: Άρνηση Έναρξης Διεργασίας

- Θεωρούμε σύστημα με
  - n διεργασίες και m διαφορετικούς τύπους πόρων
- Ορισμοί
  - Συνολική ποσότητα πόρων στο σύστημα

    Διάνυσμα 1xm: *Σύνολο Πόρων* $\boldsymbol{R} = (R_1, \ldots, R_m)$
  - Ποσότητα πόρου που δεν έχει εκχωρηθεί σε διεργασία

    Διάνυσμα 1xm: *Διαθεσιμότητα* $\boldsymbol{V} = (V_1, \ldots, V_m)$
  - Μέγιστη αίτηση κάθε διεργασίας για κάθε πόρο

    Πίνακας nxm: *Μέγιστη Αίτηση* $\boldsymbol{C} = [C_{ij}],\ i = 1,\ldots,n,\ j = 1,\ldots,m$
  - Τρέχουσα εκχώρηση πόρων στις διεργασίες

    Πίνακας nxm: *Εκχώρηση* $\boldsymbol{A} = [A_{ij}],\ i = 1,\ldots,n,\ j = 1,\ldots,m$

Πολιτική

- Οι πόροι είτε είναι διαθέσιμοι ή έχουν εκχωρηθεί

$$R_i = V_i + \sum_{k=1}^{n} A_{ki}, \quad i = 1,\ldots,m$$

- Καμιά διεργασία δε μπορεί να ζητήσει περισσότερο από τη συνολική ποσότητα πόρου

$$C_{ki} \le R_i, \quad k = 1,\ldots,n,\ i = 1,\ldots,m$$

- Καμιά διεργασία δεν παίρνει περισσότερα από όσο ζήτησε αρχικά

$$A_{ki} \le C_{ki}, \quad k = 1,\ldots,n,\ i = 1,\ldots,m$$

- Ξεκινά μια νέα διεργασία P_{n+1} μόνο αν δεν οδηγεί σε αδιέξοδο

$$R_i \ge C_{(n+1)i} + \sum_{k=1}^{n} C_{ki}, \quad i = 1,\ldots,m$$

Αποφυγή 2: Άρνηση Εκχώρησης Πόρου

- Θεωρούμε σύστημα με
  - n διεργασίες και m διαφορετικούς τύπους πόρων
  - Κάθε στιγμή μια διεργασία μπορεί να κατέχει ≥ 0 πόρους
- Κατάσταση
  - Τρέχουσα εκχώρηση πόρων στις διεργασίες

  Διανύσματα 1xm

  Σύνολο Πόρων ***R****= (Rj), Διαθεσιμότητα* ***V****= (Vj),* j = 1,…,m

  Πίνακες nxm

  Μέγιστη Αίτηση ***C****= [Cij], Εκχώρηση* ***A****= [Aij],* i =1,…,n, j=1,…,m
- Ασφαλής κατάσταση
  - Υπάρχει τουλάχιστο μια ακολουθία εκχώρησης μέγιστων αιτήσεων χωρίς αδιέξοδο
- Επισφαλής κατάσταση
  - Δεν υπάρχει ακολουθία εκχώρησης πόρων χωρίς αδιέξοδο

Αλγόριθμος του Τραπεζίτη

- Όταν μια διεργασία κάνει αίτηση για σύνολο πόρων
  - Θεωρούμε ότι η αίτηση ικανοποιείται
  - Ενημερώνουμε την κατάσταση του συστήματος
  - Αποφασίζουμε αν η κατάσταση που προκύπτει είναι ασφαλής
- Αν η νέα κατάσταση είναι ασφαλής
  - Ικανοποιούμε την αίτηση
  - Κάνουμε μόνιμη την αλλαγή κατάστασης στο σύστημα
- Αν η νέα κατάσταση είναι επισφαλής
  - Βάζουμε τη διεργασία σε αποκλεισμό μέχρι να γίνει ασφαλής η ικανοποίηση της αίτησης
  - Επιστροφή στην προηγούμενη κατάσταση πριν την τελευταία ενημέρωση

Παράδειγμα Ασφαλούς Κατάστασης

1. Αρχική κατάσταση

Μέγιστη Αίτηση

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

Εκχώρηση

	R1	R2	R3
P1	1	0	0
P2	6	1	2
P3	2	1	1
P4	0	0	2

Σύνολο Πόρων

R1	R2	R3
9	3	6

Διαθεσιμότητα

R1	R2	R3
0	1	1

2. Η P2 τερματίζει

Μέγιστη Αίτηση

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

Εκχώρηση

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

Συνολικοί Πόροι

R1	R2	R3
9	3	6

Διαθεσιμότητα

R1	R2	R3
6	2	3

Παράδειγμα Ασφαλούς Κατάστασης (συνέχεια)

3. Η P1 τερματίζει

Μέγιστη Αίτηση

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

Εκχώρηση

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

Σύνολο Πόρων

R1	R2	R3
9	3	6

Διαθεσιμότητα

R1	R2	R3
7	2	3

4. Η P3 τερματίζει

Μέγιστη Αίτηση

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

Εκχώρηση

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

Σύνολο Πόρων

R1	R2	R3
9	3	6

Διαθεσιμότητα

R1	R2	R3
9	3	4

Παράδειγμα Επισφαλούς Κατάστασης

3. Αρχική κατάσταση

Μέγιστη Αίτηση

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

Εκχώρηση

	R1	R2	R3
P1	1	0	0
P2	5	1	1
P3	2	1	1
P4	0	0	2

Σύνολο Πόρων

R1	R2	R3
9	3	6

Διαθεσιμότητα

R1	R2	R3
1	1	2

Η P1 δίνεται 1 μονάδα από κάθε πόρο R1 και R3.
Στη συνέχεια όλες οι διεργασίες χρειάζονται ≥ 1 μονάδες του R1

- Επισφαλής κατάσταση

Μέγιστη Αίτηση

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

Εκχώρηση

	R1	R2	R3
P1	2	0	1
P2	5	1	1
P3	2	1	1
P4	0	0	2

Σύνολο Πόρων

R1	R2	R3
9	3	6

Διαθεσιμότητα

R1	R2	R3
0	1	1

Ανίχνευση Αδιεξόδου

- Χαρακτηριστικά
  - Δεν περιορίζει την πρόσβαση στους πόρους
  - Δεν περιορίζει τις ενέργειες των διεργασιών
  - Ικανοποιεί τις αιτήσεις των διεργασιών όταν είναι εφικτό
- Το λειτουργικό σύστημα περιοδικά
  - Τρέχει έναν αλγόριθμο για ανίχνευση αδιεξόδου
  - Ανιχνεύει κυκλική αναμονή
  - Κάνει κατάλληλες ενέργειες για αποκατάσταση

Ορισμοί

- Θεωρούμε σύστημα με
  - *n* διεργασίες και *m* διαφορετικούς τύπους διεργασιών
- Ορισμοί
  - Συνολική ποσότητα κάθε πόρου στο σύστημα

  Διάνυσμα 1xm: *Σύνολο Πόρων* $\boldsymbol{R}= (R_j),\ j = 1,\dots,m$
  - Ποσότητα κάθε πόρου j που είναι διαθέσιμη

  Διάνυσμα 1xm: *Διαθεσιμότητα* $\boldsymbol{V}= (V_j),\ j = 1,\dots,m$
  - Τρέχουσα εκχώρηση πόρου τύπου j στη διεργασία i

  Πίνακας nxm: *Εκχώρηση* $\boldsymbol{A}= [A_{ij}],\ i = 1,\dots,n,\ j = 1,\dots,m$
  - Τρέχουσα ποσότητα πόρου j που ζητείται από τη διεργασία i

  Πίνακας nxm: *Τρέχουσα Αίτηση* $\boldsymbol{Q}= [Q_{ij}],\ i = 1,\dots,n,\ j = 1,\dots,m$

Στρατηγική

- Βρίσκουμε διεργασία της οποίας
  - Οι αιτήσεις ικανοποιούνται με τους διαθέσιμους πόρους
- Υποθέτουμε ότι
  - Οι αιτούμενοι πόροι παραχωρούνται στην πιο πάνω διεργασία
  - Η διεργασία τρέχει μέχρι να τερματίσει
  - Η διεργασία αποδεσμεύει όλους τους πόρους
- Επαναλαμβάνουμε την αναζήτηση για άλλη διεργασία
  - Αν η αναζήτηση χωρίς αποτέλεσμα, όσες διεργασίες δε βρέθηκαν βρίσκονται σε αδιέξοδο
- Σημείωση
  - Ο αλγόριθμος αποφασίζει αν υπάρχει αδιέξοδο
  - Ο αλγόριθμος δεν εξασφαλίζει πρόληψη αδιεξόδου

Αλγόριθμος

0. Αρχικά καμιά διεργασία δεν είναι σημαδεμένη
1. Σημαδεύουμε κάθε διεργασία που έχει γραμμή μηδενικών στον ***A***
2. Αρχικοποιούμε το διάνυσμα ***W*** = ***V*** (δηλαδή $W_j=A_j$, $j=1,\ldots,m$)
3. Βρίσκουμε δείκτη i έτσι ώστε
   - Η διεργασία i να μην είναι σημαδεμένη και
   - $Q_{ik} \le W_k$, για $1 \le k \le m$ (δηλαδή η i-στή γραμμή του ***Q*** ≤ ***W***)
4. Αν βρεθεί τέτοια γραμμή
   - Σημαδεύουμε τη διεργασία i
   - Θέτουμε $W_k = W_k + A_{ik}$, $1 \le k \le m$ (δηλαδή ***W*** **=** ***W*** **+** *i-στή γραμμή* ***A***)
   - Επιστρέφουμε στο βήμα 3
5. Αν δεν υπάρχει τέτοια γραμμή, τερματίζει ο αλγόριθμος
   - Όσες διεργασίες δε σημαδεύτηκαν βρίσκονται σε αδιέξοδο

Παράδειγμα

0. Αρχικά όλες οι διεργασίες μη σημαδεμένες
1. Σημαδεύουμε την P4, επειδή δεν κατέχει καθόλου πόρους
2. Θέτουμε ***W*** = ***V*** = *(0 0 0 0 1)*
3. Σημαδεύουμε την P3 επειδή $Q_{3j} \le W_j$, $j = 1,\ldots,m$.
   Θέτουμε ***W*** = ***W*** + *(0 0 0 1 0)* = *(0 0 0 1 1)*
4. Καμιά μη σημαδεμένη διεργασία δεν έχει **γραμμή Q ≤ W**
   Τερματίζουμε τον αλγόριθμο
5. Συμπέρασμα
   - Οι διεργασίες P1 και P2 παραμένουν μη σημαδεμένες
   - Αυτές οι διεργασίες είναι σε αδιέξοδο

Παράδειγμα (συνέχεια)

	R1	R2	R3	R4	R5
P1	**0**	**1**	**0**	**0**	**1**
P2	**0**	**0**	**1**	**0**	**1**
P3	**0**	**0**	**0**	**0**	**1**
P4	**1**	**0**	**1**	**0**	**1**

Τρέχουσα Αίτηση **Q**

	R1	R2	R3	R4	R5
P1	**1**	**0**	**1**	**1**	**0**
P2	**1**	**1**	**0**	**0**	**0**
P3	**0**	**0**	**0**	**1**	**0**
P4	**0**	**0**	**0**	**0**	**0**

Εκχώρηση **A**

R1	R2	R3	R4	R5
2	1	1	2	1

Σύνολο Πόρων **R**

R1	R2	R3	R4	R5
0	0	0	0	1

Διαθεσιμότητα **V**

R1	R2	R3	R4	R5
0	0	0	0	1

Προσωρινό διάνυσμα
W = V

R1	R2	R3	R4	R5
0	0	0	1	1

Προσωρινό διάνυσμα
W = W+A[row P3]

Αποκατάσταση από Αδιέξοδο

1. Διακοπή όλων των διεργασιών του αδιεξόδου
2. Επανεκκίνηση των διεργασιών από προηγούμενο ασφαλές σημείο
   - Μη νομοτελειακή εκτέλεση μπορεί να αποφύγει το αδιέξοδο
3. Διαδοχική διακοπή των διεργασιών του αδιεξόδου
   - Η διακοπή διεργασιών σταματά όταν πάψει το αδιέξοδο
   - Επιλογή της επόμενης διεργασίας με βάση κριτήριο κόστους
4. Διαδοχική απομάκρυνση πόρων από διεργασίες
   - Η απομάκρυνση σταματά όταν πάψει το αδιέξοδο
   - Επιλογή του επόμενου πόρου με βάση κριτήριο κόστους
   - Επιστροφή διεργασιών σε σημείο πριν τη λήψη του πόρου

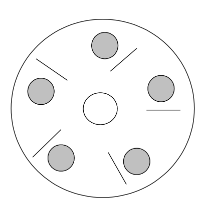
Ολοκληρωμένη Στρατηγική

- Συνδυασμός μεθόδων χειρισμού αδιεξόδου
  1. Ομαδοποίηση πόρων σε διαφορετικές κατηγορίες
  2. Γραμμική διάταξη κατηγοριών για πρόληψη κυκλικής αναμονής
  3. Χρήση του κατάλληλου αλγορίθμου σε κάθε κατηγορία

- Πιθανές κατηγορίες
  1. Όλος ο χώρος εναλλαγής (swap space) δίνεται εξαρχής (πρόληψη)
  2. Οι αιτήσεις των διεργασιών δηλώνονται εξαρχής (αποφυγή)
  3. Η εκχώρηση κύριας μνήμης επιδέχεται εκτόπιση (πρόληψη)
  4. Οι εσωτερικοί πόροι μπορούν να διαταχθούν (πρόληψη)

Πρόβλημα Συνδαιτυμόνων Φιλοσόφων

- Θεωρούμε
  - Στρογγυλό τραπέζι με πιατέλα στο κέντρο
  - Πέντε πιάτα ένα για κάθε φιλόσοφο
  - Πέντε ξυλάκια, ένα δίπλα σε κάθε πιάτο
- Θέλουμε να επινοήσουμε αλγόριθμο που εξασφαλίζει
  - Αμοιβαίο αποκλεισμό (δηλαδή κανένα ξυλάκι δε χρησιμοποιείται ταυτόχρονα από δύο φιλοσόφους)
  - Αποφυγή αδιεξόδου
  - Αποφυγή στέρησης



Πρώτη Λύση (Με Πιθανό Αδιέξοδο!)

```
/* program dining philosophers */
semaphore chopstick[5] = {1,1,1,1,1};
int i;
void philosopher (int i) {
      while (true) {
            think();
            wait(chopstick[i]);
            wait(chopstick[(i+1) mod 5]);
            eat();
            signal(chopstick[(i+1) mod 5]);
            signal(chopstick[i]);
      }
}
void main() {      parbegin (philosopher(0), philosopher(1),
                   philosopher(2), philosopher(3), philosopher(4)); }
```


Δεύτερη Λύση

```
/* program dining philosophers */
semaphore chopstick[5] = {1,1,1,1,1};
semaphore room = 4;
int i;
void philosopher (int i) {        /* αποφεύγει αδιέξοδο και στέρηση */
      while (true) {
            think();
            wait(room);  /* περιορίζει τέσσερις φιλοσόφους στο τραπέζι κάθε φορά */
            wait(chopstick[i]);
            wait(chopstick[(i+1) mod 5]);
            eat();
            signal(chopstick[(i+1) mod 5]);
            signal(chopstick[i]);
            signal(room);
      }
}
void main() {      parbegin (philosopher(0), philosopher(1),
                   philosopher(2), philosopher(3), philosopher(4)); }
```