Ελληνική Δημοκρατία
Τεχνολογικό Εκπαιδευτικό
Ίδρυμα Ηπείρου

Λειτουργικά Συστήματα

Ενότητα 7 : **Αδιέξοδο$_{2/2}$**

Δημήτριος Λιαροκάπης

Τμήμα Μηχανικών Πληροφορικής Τ.Ε

Λειτουργικά Συστήματα

Ενότητα 7 : Αδιέξοδο$_{2/2}$

Δημήτριος Λιαροκάπης
Καθηγητής Εφαρμογών

Άρτα, 2015

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
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

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.

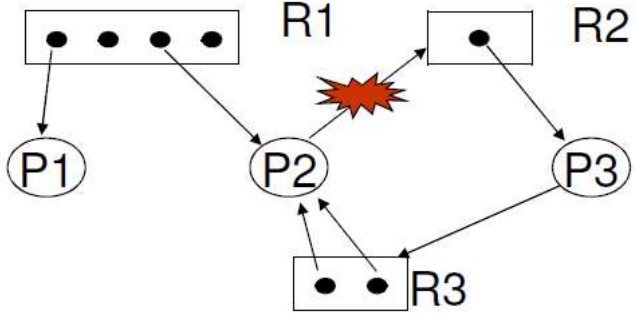
Χρηματοδότηση

- Το 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.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο ΤΕΙ Ηπείρου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.

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

Κυκλική αναμονή

- Ύπαρξη κυκλικής αλυσίδα με 2 ή περισσότερες διεργασίες όπου κάθε μια διεργασία περιμένει έναν πόρο που κατέχεται από άλλη διεργασία της αλυσίδας
- Λύση
  - Ορισμός μια γραμμικής αρίθμησης των πόρων
  - Οι αιτήσεις για πόρους από τις διεργασίες πρέπει να γίνονται με βάση τη σειρά αρίθμησης
  - Η ποικιλομορφία των πόρων και των χρήσεων τους κάνει δύσκολη την εφαρμογή στην πράξη



4.2. Αποφυγή αδιεξόδου

- Οι πόροι εκχωρούνται με τρόπο που εγγυάται ότι δεν θα βρεθεί ποτέ σημείο στο οποίο θα συμβεί αδιέξοδο
- Η αποφυγή του αδιεξόδου επιτρέπει τις τρεις πρώτες απαραίτητες συνθήκες αλλά κάνει διακριτικές επιλογές ώστε να εξασφαλιστεί ότι δεν 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

- 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

- 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

- Προσεγγίσεις αποφυγής αδιεξόδου
  - Μια διεργασία δεν ξεκινά αν οι απαιτήσεις της μπορούν να οδηγήσουν σε αδιέξοδο
  - 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

- 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

- 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 <P1, P2, …, Pn> είναι ασφαλής αν για κάθε Pi , οι πόροι που η Pi μπορεί ακόμη να απαιτήσει μπορούν να ικανοποιηθούν από τους τρέχοντες διαθέσιμους πόρους συν τους πόρους που δεσμεύονται από όλες τις διεργασίες Pj, με j < i.

Ασφαλής κατάσταση

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

Βασικά γεγονότα

- Αν το σύστημα είναι σε ασφαλή κατάσταση δεν υπάρχει αδιέξοδο.
- Αν το σύστημα δεν είναι σε ασφαλή κατάσταση υπάρχει πιθανότητα αδιεξόδου (το αδιέξοδο δεν θα συμβεί σίγουρα, απλώς είναι πιθανό)
- Αποφυγή αδιεξόδου : εξασφάλιση ότι το σύστημα δεν θα εισέλθει ποτέ σε μη ασφαλή κατάσταση

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
- Καμιά διεργασία δεν μπορεί να περιέλθει σε κατάσταση εξόδου (exit) ενώ δεσμεύει πόρους

Δομές δεδομένων του αλγορίθμου

- n = το πλήθος των διεργασιών,
- m = το πλήθος τύπων πόρων
- Available: Διάνυσμα μήκους m. Αν available [j] = k, υπάρχουν k στιγμιότυπα του τύπου πόρου Rj διαθέσιμα.
- Max: n x m πίνακας. Αν Max [i,j] = k, τότε η διεργασία Pi μπορεί να απαιτήσει κατά μέγιστο k στιγμιότυπα του τύπου πόρου Rj.
- Allocation: n x m πίνακας. Αν Allocation[i, j] = k τότε στη διεργασία Pi εκχωρούνται k στιγμιότυπα του πόρου Rj.
- Need: n x m πίνακας. Αν Need[i, j] = k, τότε η διεργασία Pi μπορεί να χρειαστεί k επιπλέον στιγμιότυπα του πόρου Rj για να ολοκληρωθεί.
- Need [i,j] = Max[i,j] – Allocation [i,j]

Αλγόριθμος

- Ψάξε στον πίνακα Need για μια γραμμή i όπου όλες οι απαιτήσεις είναι μικρότερες ή ίσες από τους διαθέσιμους πόρους, δλδ. Need[i,j] ≤ Available[j] ∀j
  - Αν δεν υπάρχει τέτοια γραμμή τότε τερμάτισε. Η κατάσταση είναι ανασφαλής.
- • Υπέθεσε ότι η διεργασία της γραμμής i ζητά όλους τους πόρους και τερματίζει.
  - Σημείωσε τη διεργασία ως ολοκληρωμένη.
  - Πρόσθεσε τους πόρους που κατείχε στους διαθέσιμους, δλδ. Available[j] = Available[j] + Allocation[i,j] ∀j
- • Επανέλαβε τα παραπάνω μέχρι να σημειωθούν όλες οι διεργασίες ως ολοκληρωμένες.
  - Σε αυτή την περίπτωση η κατάσταση είναι ασφαλής.

Παράδειγμα

- Εφαρμογή του αλγορίθμου για καθορισμό μιας ασφαλούς κατάστασης
- Υπάρχουν 3 τύποι πόρων με πλήθος :
  - R(1) = 9, R(2) = 3, R(3) = 6
- Είναι η παρακάτω κατάσταση ασφαλής;

Max

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

Allocated

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

Total

R1	R2	R3
9	3	6

Available

R1	R2	R3
0	1	1

Need

R1	R2	R3
2	2	2
0	0	1
1	0	3
4	2	0

Παράδειγμα

	Max			Allocated			Need		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	0	0	0	0	0	0	0	0	0
P2	0	0	0	0	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0
P4	4	2	2	0	0	2	4	2	0

Total		
R1	R2	R3
9	3	6

Available

9	3	4

	Max			Allocated			Need		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	0	0	0	0	0	0	0	0	0
P2	0	0	0	0	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	0	0	0

Total		
R1	R2	R3
9	3	6

Available

9	3	6

Η κατάσταση είναι ασφαλής: P2∗P1∗P3∗P4

4.3. Ανίχνευση αδιεξόδου

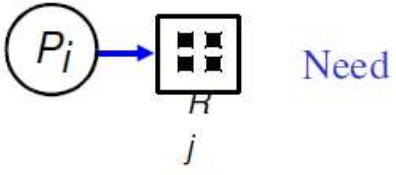
- Ο αλγόριθμος του τραπεζίτη είναι απαισιόδοξος : πάντοτε υποθέτει ότι μια διεργασία δεν θα απελευθερώσει τους πόρους που κατέχει μέχρι να αποκτήσει όλους τους πόρους που χρειάζεται.
  - Συνέπειες :
    - Μικρό ποσοστό παραλληλίας
    - Πολύπλοκοι έλεγχοι για κάθε απαίτηση εκχώρησης πόρου (πολυπλοκότητα 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.

ΛΣ και ανίχνευση αδιεξόδου

- Περιοδικά το ΛΣ εφαρμόζει έναν αλγόριθμο ανίχνευσης αδιεξόδου που δίνει τη δυνατότητα να εντοπισθεί η συνθήκη της κυκλικής αναμονής στο γράφο εκχώρησης πόρων.
- Το πότε και πόσο συχνά καλείται ο αλγόριθμος εξαρτάται από:
- Πόσο συχνά συνηθίζεται να συμβαίνει αδιέξοδο
- Το πλήθος των διεργασιών που πρέπει να επιστρέψουν σε προηγούμενη κατάσταση

Αλγόριθμος ανίχνευσης αδιεξόδου

- n = το πλήθος των διεργασιών,
- m = το πλήθος τύπων πόρων
- Available: Διάνυσμα μήκους m. Αν available [j] = k, υπάρχουν k στιγμιότυπα του τύπου πόρου Rj διαθέσιμα.
- Allocation: n x m πίνακας. Αν Allocation[i, j] = k τότε στη διεργασία Pi εκχωρούνται k στιγμιότυπα του πόρου Rj.
- Need: n x m πίνακας. Αν Need[i, j] = k, τότε η διεργασία Pi απαιτεί k επιπλέον στιγμιότυπα του πόρου Rj για να ολοκληρωθεί.



Αλγόριθμος

- Ψάξε στον πίνακα Need για μια γραμμή i όπου όλες οι απαιτήσεις είναι μικρότερες ή ίσες από τους διαθέσιμους πόρους, δλδ. Need[i,j] ≤ Available[j] ∀j
  - Αν δεν υπάρχει τέτοια γραμμή τότε τερμάτισε. Οι διεργασίες που δεν έχουν σημειωθεί βρίσκονται σε αδιέξοδο.
- Η διεργασία της γραμμής i αποκτά όλους τους πόρους που απαιτεί και τερματίζει.
  - Σημείωσε τη διεργασία ως ολοκληρωμένη.
  - Πρόσθεσε τους πόρους που κατείχε στους διαθέσιμους, δλδ.
- Available[j] = Available[j] + Allocation[i,j] ∀j
- Επανέλαβε τα παραπάνω μέχρι να σημειωθούν όλες οι διεργασίες ως ολοκληρωμένες.
  - Σε αυτή την περίπτωση δεν υπάρχει αδιέξοδο κατά την τρέχουσα χρονική στιγμή.

ΛΣ και ανίχνευση αδιεξόδου

- Στην πράξη, τα περισσότερα Λ.Σ. εθελοτυφλούν και αντιμετωπίζουν το πρόβλημα με συνδυασμό τεχνικών, όπως:
  - –Διακοπή κατοχής και αναμονής:
    - Όταν μια διεργασία δεν μπορεί να αποκτήσει έναν πόρο, τότε δεν μπορεί να ολοκληρωθεί και αποτυγχάνει
- Όρια (Quotas)
- Υψηλής απόδοσης τεχνικές προγραμματισμού:
  - Απαίτηση χρήσης σημαφόρων με καθορισμένη προτεραιότητα.

Στρατηγικές όταν ανιχνευθεί αδιέξοδο

- Τερματισμός διεργασίας & προεκχώρηση πόρων
  - Διακοπή όλων των διεργασιών που περιήλθαν σε αδιέξοδο
  - Διαδοχική διακοπή όλων των διεργασιών που βρίσκονται σε αδιέξοδο ώστε να μην υπάρχει πλέον αδιέξοδο
  - Δημιουργία αντιγράφου ασφαλείας για κάθε διεργασία μέσω περιοδικών σημείων ελέγχου. Οι διεργασίες που βρίσκονται σε αδιέξοδο επιστρέφουν σε κάποιο προηγούμενο σημείο ελέγχου και συνεχίζουν από εκεί.
    - Οι πόροι που οδήγησαν στο αδιέξοδο αποδεσμεύονται
    - Το αρχικό αδιέξοδο μπορεί να ξανασυμβεί
  - Διαδοχική προεκχώρηση πόρων έως ότου δεν θα υπάρχει αδιέξοδο

Στρατηγικές όταν ανιχνευθεί αδιέξοδο

- Κριτήρια επιλογής των διεργασιών που βρίσκονται σε αδιέξοδο και θα τερματιστούν
  - 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
    - τον μεγαλύτερο εκτιμώμενο χρόνο
    - το μικρότερο πλήθος πόρων που της έχουν εκχωρηθεί
    - τη μικρότερη προτεραιότητα

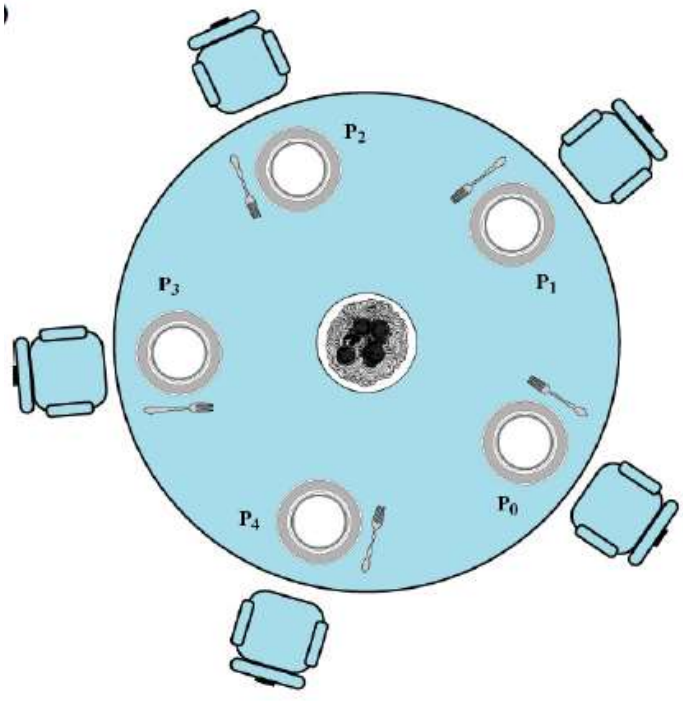
Συνδυασμένη ανίχνευση αδιεξόδου

- Ο συνδυασμός των τριών προσεγγίσεων
  - Πρόληψης,
  - Αποφυγής,
  - Ανίχνευσης

  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 των πλέον κατάλληλων τεχνικών για τη διαχείριση αδιεξόδων μέσα σε κάθε κλάση.

5. Πρόβλημα συνδαιτυμόνων 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 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.



Πρόβλημα συνδαιτυμόνων φιλοσόφων

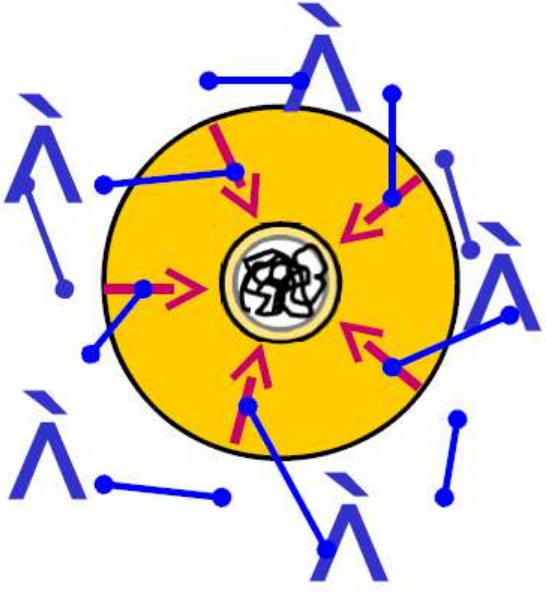
- Κάθε φιλόσοφος είναι μια διεργασία και κάθε πηρούνι είναι ένας διαμοιραζόμενος πόρος με ενέργειες δέσμευσης και απελευθέρωσης. Αν ένας φιλόσοφος πεινάσει, πρέπει πρώτα να πάρει τα πηρούνια δεξιά και αριστερά του για να μπορέσει να ξεκινήσει να τρώει.
- Το πρόβλημα αυτό είναι ένα πρότυπο που χρησιμοποιείται 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

Πρόβλημα συνδαιτυμόνων φιλοσόφων

- Δεν υπάρχει συμμετρική λύση
- Λύσεις
  - 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 (κρίσιμο τμήμα) - (κατοχή και αναμονή)
  - Σχεδιασμός του συστήματος έτσι ώστε ένας φιλόσοφος να «κλέψει» ένα πηρούνι που δεν είναι γειτονικό του.

Πρόβλημα συνδαιτυμόνων φιλοσόφων

```
/* 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]);
        }
}
```



Πρόβλημα συνδαιτυμόνων φιλοσόφων

```
/* 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);
        }
}
```

Πρόβλημα συνδαιτυμόνων φιλοσόφων

```
#define N 5  /* Number of philosphers */
#define RIGHT(i) (((i)+1) %N)
#define LEFT(i) (((i)==N) ? 0 : (i)+1)

typedef enum { THINKING, HUNGRY, EATING } phil_state;

phil_state state[N];
semaphore mutex =1;
semaphore s[N];       /* one per philosopher, all 0 */

void test(int i) {
    if ( state[i] == HUNGRY &&
     state[LEFT(i)] != EATING &&
     state[RIGHT(i)] != EATING ) {state[i] = EATING; V(s[i]);}
}
 void get_forks(int i) {
     P(mutex);
     state[i] = HUNGRY;
     test(i);
     V(mutex);
     P(s[i]);
 }
```

P->wait
V->signal

Πρόβλημα συνδαιτυμόνων φιλοσόφων

```
void put_forks(int i) {
    P(mutex);
    state[i]= THINKING;
    test(LEFT(i));
    test(RIGHT(i));
    V(mutex);
}

void philosopher(int process) {
    while(1) {
    think();
    get_forks(process);
    eat();
    put_forks(process);
    }
}
```

Βιβλιογραφία

Λειτουργικά Συστήματα, 8η Έκδοση, Stallings William

Λειτουργικά Συστήματα 9η Εκδ., Abraham Silberschatz, Peter Baer Galvin, Greg Gagne

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

[1] http://creativecommons.org/licenses/by-nc-nd/4.0/deed.el

Τέλος Ενότητας

Επεξεργασία: Ευάγγελος Καρβούνης

Άρτα, 2015

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Τέλος Ενότητας

Αδιέξοδο$_{2/2}$

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
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης
ΕΣΠΑ 2007-2013
Πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ