Ελληνική Δημοκρατία
Τεχνολογικό Εκπαιδευτικό
Ίδρυμα Ηπείρου

Λειτουργικά Συστήματα

Ενότητα 5 : **Αμοιβαίος Αποκλεισμός**

Δημήτριος Λιαροκάπης

Τμήμα Μηχανικών Πληροφορικής Τ.Ε

Λειτουργικά Συστήματα

Ενότητα 5 : Αμοιβαίος Αποκλεισμός

Δημήτριος Λιαροκάπης
Καθηγητής Εφαρμογών

Άρτα, 2015

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

Αμοιβαίος Αποκλεισμός

1. Εισαγωγή
2. Κρίσιμα τμήματα (Critical Sections)
3. Υλοποίηση του αμοιβαίου αποκλεισμού
   i. *Προσεγγίσεις λογισμικού*
   ii. *Υποστήριξη εκ μέρους του υλικού*
   iii. *Σημαφόροι (σηματοφορείς)*

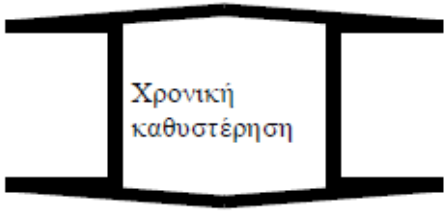
1. Εισαγωγή

- Αμοιβαίος αποκλεισμός: αποκλεισμός μιας διεργασίας από μια ενέργεια που επιτελεί ταυτοχρόνως κάποια άλλη διεργασία
- Απαιτεί μόνο μία διεργασία να πραγματοποιεί λειτουργίες σε κοινούς πόρους. Αν δύο διεργασίες προσπαθήσουν να αποκτήσουν ένα κοινό πόρο, τότε η μία θα πρέπει να αναμένει
- Για την αποφυγή των παρενεργειών λόγω ανταγωνισμού είναι ζωτικής σημασίας η πρόβλεψη και η αποτροπή χρήσης διαμοιραζόμενων πόρων από περισσότερες από μια διεργασίες την ίδια χρονική στιγμή.
- Είναι απαραίτητος για την προστασία των κρίσιμων τμημάτων των διεργασιών

Παράδειγμα (1)

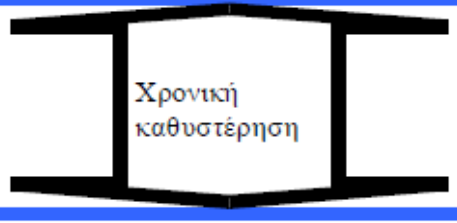
- Κατάθεση 100 €
- Ανάγνωση τρέχοντος υπολοίπου = 500 €

- Απαίτηση ανάληψης 100 €



- Ανάγνωση τρέχοντος υπολοίπου = 500 €
- Νέο υπόλοιπο= 500 - 100 = 400

- Νέο υπόλοιπο= 500 + 100 = 600
- Εγγραφή νέου υπολοίπου στο δίσκο = 600 €



- Εγγραφή νέου υπολοίπου στο δίσκο = 400 €

Παράδειγμα (2)

•Κατάθεση 100 € •Απαίτηση για κλείδωμα του λογαριασμού •Ανάγνωση υπολοίπου από το δίσκο = 500 €	•Απαίτηση ανάληψης 100 €
	•Απαίτηση για κλείδωμα του λογαριασμού •Αναμονή
•υπόλοιπο = 500 + 100 = 600 •Εγγραφή νέου υπολοίπου στο δίσκο = 600 € •Απελευθέρωση λογαριασμού	
	•Δέσμευση λογαριασμού •Ανάγνωση υπολοίπου = 600 •Νέο υπόλοιπο = 600 - 100 = 500 •Εγγραφή νέου υπολοίπου στο δίσκο = 500 €

2. Κρίσιμα τμήματα (Critical Sections)

- Κρίσιμο τμήμα: μια ακολουθία εντολών μιας διεργασίας που πρέπει να εκτελείται αδιαίρετα
- Η εκτέλεση του κρίσιμου τμήματος δεν θα πρέπει να διακόπτεται στο μέσον (“όλα ή τίποτα”)
- Η αποτελεσματικότητα της πολυεπεξεργασίας εξαρτάται από το μήκος του κρίσιμου τμήματος
  - Πρέπει να είναι όσο το δυνατόν μικρότερο
- Το πλέον αναποτελεσματικό σενάριο
  - Όλο το πρόγραμμα είναι ένα κρίσιμο τμήμα: δεν υπάρχει πολυπρογραμματισμός

Κρίσιμα Τμήματα

- Όταν μια διεργασία εκτελεί το κρίσιμο τμήμα της (ΚΤ), σε καμιά άλλη δεν μπορεί να επιτραπεί να εκτελεί το δικό της κρίσιμο τμήμα.
- Αν μπορούσε να διασφαλισθεί ότι ποτέ δύο ή περισσότερες διεργασίες δεν θα βρίσκονται ταυτόχρονα σε κρίσιμα τμήματα, τότε θα είχε επιτευχθεί η αποφυγή των συνθηκών ανταγωνισμού
- Ωστόσο αυτό δεν είναι αρκετό για τη σωστή και αποδοτική συνεργασία δύο ή περισσότερων παράλληλων διεργασιών
- Μια καλή λύση προϋποθέτει τις παρακάτω συνθήκες:

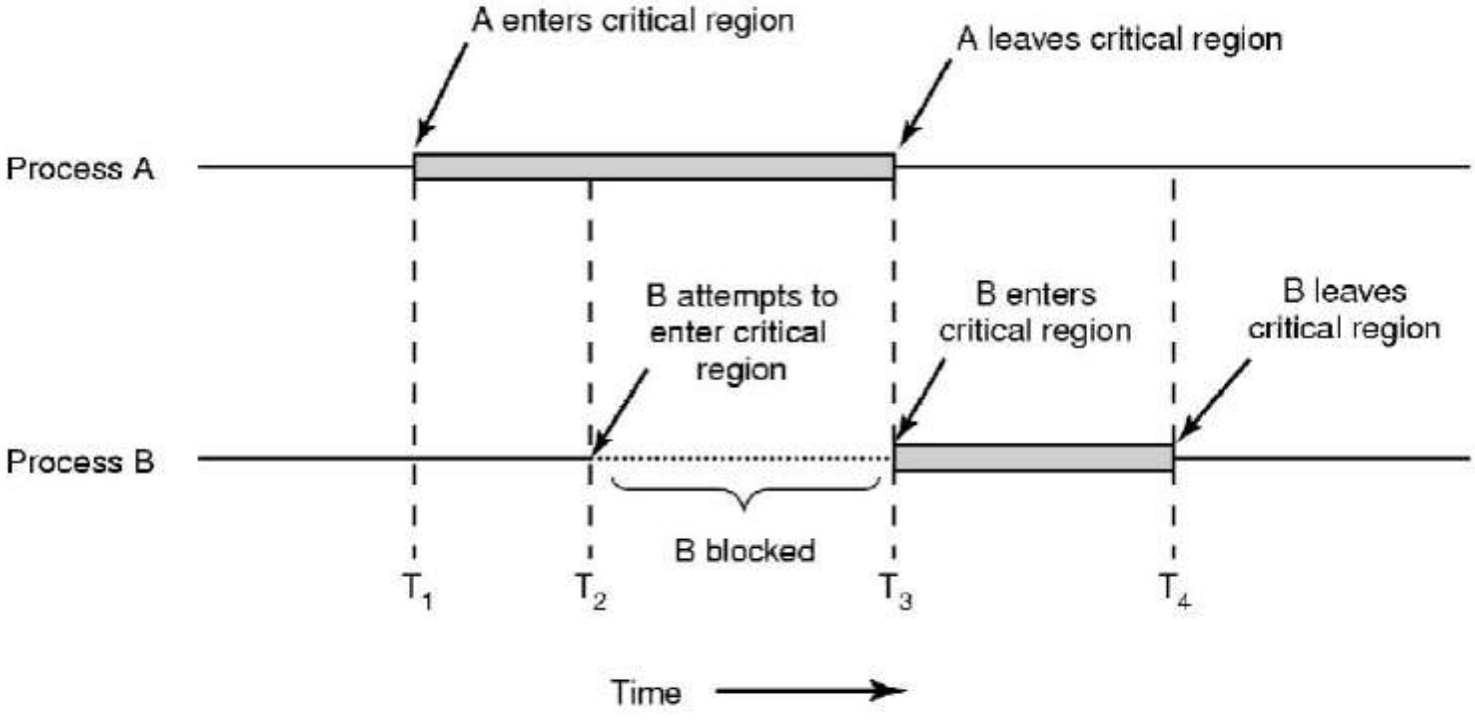
Συνθήκες για κρίσιμα τμήματα

1. Κάθε διεργασία μπορεί να διαιρεθεί σε δύο ακολουθίες (τμήματα ή περιοχές εντολών): το κρίσιμο τμήμα, δηλ. αυτό που προσπελαύνει τον κοινό πόρο και δεν πρέπει να καταμεριστεί και το μη κρίσιμο τμήμα
2. Δυο διεργασίες δεν μπορούν να βρίσκονται ταυτόχρονα στα κρίσιμα τμήματά τους (αμοιβαίος αποκλεισμός)
3. Μια διεργασία μπορεί να σταματήσει μόνον μέσα στο μη κρίσιμο τμήμα της, χωρίς να επηρεάσει (να αναστείλει) άλλες διεργασίες
4. Δεν επιτρέπονται υποθέσεις σε ότι αφορά την ταχύτητα ή το πλήθος των επεξεργαστών
5. Όταν μια διεργασία δεν εκτελεί το κρίσιμο τμήμα της δεν μπορεί να αναστείλει την εκτέλεση άλλης διεργασίας

Συνθήκες για αμοιβαίο αποκλεισμό

6. Το πρόγραμμα δεν επιτρέπεται να καταλήξει σε αδιέξοδο. Αν περισσότερες της μιας διεργασίες προσπαθήσουν να εισέλθουν ταυτόχρονα στις κρίσιμες περιοχές τους, τότε μόνο μια πρέπει να επιτύχει (πρόοδος).
7. Δεν επιτρέπεται η επ’ αόριστον αναμονή μιας διεργασίας για να εισέλθει στο κρίσιμο τμήμα της (λιμοκτονία)
8. Αν δεν υπάρχει ανταγωνισμός για είσοδο σε μια κρίσιμη περιοχή, μια διεργασία θα πρέπει να εισέρχεται με την ελάχιστη δυνατή επιβάρυνση
9. Μια διεργασία παραμένει μέσα στο κρίσιμο τμήμα της για ορισμένο χρονικό διάστημα

Αμοιβαίος αποκλεισμός με κρίσιμα τμήματα



Κίνδυνοι λόγω του αμοιβαίου αποκλεισμού

- Οι λανθασμένες ή ελλιπείς λύσεις στο πρόβλημα του αμοιβαίου αποκλεισμού μπορούν να οδηγήσουν σε
  - Παρατεταμένη στέρηση (starvation)
  - Αδιέξοδο (deadlock)
  - Ανεπάρκεια του συστήματος

3. Υλοποίηση του αμοιβαίου αποκλεισμού

- Προσεγγίσεις λογισμικού με παραχώρηση της ευθύνης στον προγραμματιστή
- Υποστήριξη εκ μέρους του υλικού με χρήση εντολών μηχανής ειδικού σκοπού (ατομικές λειτουργίες test και set)
- Παροχή κάποιου επιπέδου υποστήριξης από το ΛΣ ή τη γλώσσα προγραμματισμού:
  - σημαφόροι (semaphores)
  - παρακολουθητές(monitors)
  - μεταβίβαση μηνυμάτων
- Όλες οι λύσεις περιλαμβάνουν συνήθως κρίσιμα τμήματα

3.1. Προσεγγίσεις λογισμικού

- Υλοποιούνται για ταυτόχρονες διεργασίες που εκτελούνται σε έναν ή πολλαπλούς επεξεργαστές με διαμοιραζόμενη κύρια μνήμη
- Οι ταυτόχρονες προσβάσεις στην ίδια περιοχή της μνήμης διατάσσονται σειριακά, με κάποιο τρόπο διαχείρισης
- Η παραχώρηση της πρόσβασης δεν είναι εκ των προτέρων καθορισμένη.
- Δεν υπάρχει υποστήριξη σε επίπεδο υλικού, λειτουργικού συστήματος ή γλώσσας προγραμματισμού.

Ο αλγόριθμος του Dekker

- Κάθε προσπάθεια για αμοιβαίο αποκλεισμό πρέπει να στηρίζεται σε βασικούς μηχανισμούς απαγορεύσεων σε επίπεδο υλικού
- Ο πλέον συνήθης:
  - Σε μια περιοχή (θέση) μνήμης επιτρέπεται μια πρόσβαση κάθε φορά.
- Υλοποίηση του αλγορίθμου Dekker με προσπάθειες αυξανόμενης δυσκολίας, όχι πάντοτε ορθές, για την ανίχνευση των λαθών

Dekker: 1η προσπάθεια

```
var turn: 0..1;
/* shared */
```

Διεργασία 0:

```
while (turn != 0) do
   nothing;
<critical section>
turn = 1;
```

Διεργασία 1:

```
while (turn != 1) do
   nothing;
<critical section>
turn = 0;
```

- Υλοποιείται ο αμοιβαίος αποκλεισμός αλλά
  - Καταναλώνει χρόνο του επεξεργαστή (ενεργός αναμονή)
  - Αυστηρή εναλλαγή των διεργασιών κατά την εκτέλεση των κρίσιμων τμημάτων τους

Dekker: 2η προσπάθεια

```
var flag: array[0..1] of Boolean;
/* initialize both to false */
/* shared */
```

Διεργασία 0:

```
while (flag[1]==true) do
   nothing;
flag[0] = true;
   <critical section>
flag[0] = false;
```

Διεργασία 1:

```
while (flag[0]==true) do
   nothing;
flag[1] = true;
   <critical section>
flag[1] = false;
```

- Μια διεργασία μπορεί να μεταβάλλει την κατάστασή της αφού έχει ελεγχθεί από την άλλη, αλλά πριν η άλλη διεργασία εισέλθει στο κρίσιμο τμήμα της.
- Και οι δύο διεργασίες θα βρίσκονται στο κρίσιμο τμήμα τους

3.2. Υποστήριξη εκ μέρους του υλικού

- Προβλήματα με τις προσεγγίσεις λογισμικού:
  - Ενεργός αναμονή (busy waiting)
    - επαναλαμβανόμενος έλεγχος μιας μεταβλητής με αποτέλεσμα να δαπανάται χρόνος της CPU
    - πρέπει να αποφεύγεται
  - Σύνθετοι αλγόριθμοι
- Οι αλγόριθμοι δουλεύουν μόνον με 2 διεργασίες. Μπορούν να επεκταθούν και σε n διεργασίες αλλά:
  - Το πλήθος n πρέπει να είναι εξ αρχής γνωστό
  - Οι αλγόριθμοι γίνονται ακόμη πιο σύνθετοι
- Η εφαρμογή ενός μηχανισμού αμοιβαίου αποκλεισμού είναι δύσκολη

Υλικό και αμοιβαίος αποκλεισμός

- Το υλικό (hardware) μπορεί να παρέχει αποτελεσματική και γενική λύση
- Το υλικό που απαιτείται για να υποστηρίξει κρίσιμα τμήματα πρέπει να έχει:
  - Αδιαίρετες εντολές
  - Ατομικές (atomic) εντολές load, store, test.
  - Αν δύο ατομικές εντολές εκτελεστούν ταυτόχρονα, πρέπει να συμπεριφερθούν σαν να εκτελούνται σειριακά.
- Οι σύγχρονοι μικροεπεξεργαστές διαθέτουν εντολές υλικού που υποστηρίζουν τον αμοιβαίο αποκλεισμό

Α. Απενεργοποίηση Διακοπών

```
while (true)
{ /* disable interrupts */;
/* critical section */;
/* enable interrupts */;
/* remainder */;}
```

- Οι διακοπές απενεργοποιούνται κατά τη χρονική περίοδο που απαιτείται ο αμοιβαίος αποκλεισμός
- Χωρίς διακοπές δεν μπορεί να συμβεί εναλλαγή διεργασιών
- Είναι επικίνδυνη: μια διεργασία μπορεί να αποκλείσει τη διαθεσιμότητα του συστήματος
- Χρησιμοποιείται σε συστήματα ειδικής χρήσης με περιορισμένο υλικό

Μειονεκτήματα

- Υψηλό κόστος
- Μείωση αποτελεσματικότητας εκτέλεσης
  - Περιορισμός δυνατότητας εναλλαγής προγραμμάτων
- Χρήσιμη τεχνική για τον πυρήνα άλλα όχι για τις διεργασίες
- Σε συστήματα πολλών επεξεργαστών δεν εγγυάται ο αμοιβαίος αποκλεισμός:
  - Η απενεργοποίηση διακοπών επηρεάζει μόνον έναν επεξεργαστή
  - Οι υπόλοιποι επιτρέπουν τη μεταβολή των περιεχομένων της κοινής περιοχής μνήμης

Β. Ειδικές Εντολές Μηχανής

- Πολλές CPU παρέχουν εντολές υλικού για την ανάγνωση, τροποποίηση και την εγγραφή μιας θέσης μνήμης ατομικά. Οι πιο κοινές εντολές με αυτή τη δυνατότητα είναι :
  - TAS: test-and-set
  - xchg: exchange
- Βασική ιδέα: ανάγνωση και μεταβολή της τιμής μιας θέσης μνήμης σε έναν κύκλο εντολής που δεν διακόπτεται.
- Εκτέλεσης λειτουργιών σε έναν κύκλο μηχανής: αποφυγή παρεμβολών από άλλες εντολές

Αμοιβαίος Αποκλεισμός με Test and Set

- Η TAS είναι εντολή του επεξεργαστή η οποία εκτελείται αδιαίρετα σε ένα κύκλο μηχανής
  - Η TAS διαβάζει και επιστρέφει την τρέχουσα τιμή μιας μεταβλητής, ενώ της αναθέτει τιμή true
- Η TAS και η χρήση της για την επίτευξη αμοιβαίου αποκλεισμού μπορούν να περιγραφούν στη γλώσσα C ως:

```
int TAS(int lock){
    int tmp;
    tmp = lock;
    lock = true;
    return tmp;
    }
```

```
var lock: false;
/* shared */

while (TAS(lock)==true) do
    nothing;
<critical section>
lock = false;
<remainder>
```

Πλεονεκτήματα Ειδικών Εντολών

- Εφαρμογή σε οποιοδήποτε αριθμό διεργασιών
- Χρησιμοποίηση σε έναν ή πολλούς επεξεργαστές που διαμοιράζονται μια κοινή μνήμη
- Απλή και εύκολη στην επαλήθευση
- Υποστήριξη πολλαπλών κρίσιμων τμημάτων
  - π.χ. μια ξεχωριστή μεταβλητή για κάθε κρίσιμο τμήμα

Μειονεκτήματα

- Απασχόληση ενεργούς αναμονής (busy waiting)
  - Καθώς μια διεργασία αναμένει για πρόσβαση στο κρίσιμο τμήμα, συνεχίζει να καταναλώνει χρόνο του επεξεργαστή
- Παρατεταμένη στέρηση είναι πιθανή (starvation)
  - Η επιλογή για είσοδο σε μια διεργασία είναι αυθαίρετη όταν πολλαπλές διεργασίες συναγωνίζονται για να εισέλθουν στο κρίσιμο τμήμα
- Πιθανότητα αδιεξόδου
  - Αν μια διεργασία χαμηλής προτεραιότητας P1 εκτελεί την ειδική εντολή μηχανής, εισέρχεται στο κρίσιμο τμήμα της και μια μεγαλύτερης προτεραιότητας διεργασία P2 πάρει τον έλεγχο του επεξεργαστή, η P2 θα περιμένει να αποκτήσει τον έλεγχο του πόρου, εισερχόμενη σε βρόγχο ενεργού αναμονής
  - Το πρόβλημα αυτό εμφανίζεται και στις προσεγγίσεις λογισμικού

3.3 Προσεγγίσεις Αναστολής Εκτέλεσης

- Εναλλακτικά στην ενεργό αναμονή, που είναι αναποτελεσματική και επιρρεπής σε αδιέξοδα, μπορεί να χρησιμοποιηθεί η προσέγγιση προσωρινής αναστολής εκτέλεσης (sleep – wakeup)
- Εφαρμόζεται από τους σημαφόρους ή σηματοφορείς (semaphores) δηλ. γίνεται προσέγγιση χρησιμοποιώντας μηχανισμούς του ΛΣ και των γλωσσών προγραμματισμού

Σημαφόροι (semaphores)

- Η βασική αρχή: δύο ή περισσότερες διεργασίες μπορούν να συνεργαστούν με τη χρήση απλών σημάτων
  - Κάθε διεργασία επιβάλλεται να σταματήσει σε μια συγκεκριμένη θέση, μέχρι να παραλάβει ένα συγκεκριμένο σήμα.
- Για τη δημιουργία σημάτων χρησιμοποιούνται ειδικές μεταβλητές, οι σημαφόροι (ή σηματοφορείς).
  - Σημαφόρος: μεταβλητή συγχρονισμού που λαμβάνει ακέραιες θετικές τιμές ή μηδέν και μπορεί να αρχικοποιηθεί. Διαθέτει μια ουρά από συσχετιζόμενες διεργασίες τις οποίες εξυπηρετεί.
  - Ανακαλύφθηκαν στα μέσα της δεκαετίας του 1960 για να επιλύουν προβλήματα συγχρονισμού μεταξύ διεργασιών.
  - Οι σημαφόροι είναι απλοί και κομψοί και επιλύουν πολλά ενδιαφέροντα προβλήματα και όχι μόνον τον αμοιβαίο αποκλεισμό.

Λειτουργίες σημαφόρων

- Βασικές **ατομικές λειτουργίες** σε σημαφόρους
  - P ή wait
  - V ή signal
- P (semaphore) - wait:
  - Ατομική λειτουργία που αναμένει για τον σημαφόρο να γίνει >0, έπειτα τον μειώνει κατά 1
  - Αν ο σημαφόρος είναι μηδέν, η διεργασία μπλοκάρεται και αναμένει σε μία ουρά μπλοκαρισμένων διεργασιών για το σημαφόρο
  - Μόλις ο σημαφόρος γίνει >0 μία από τις διεργασίες της ουράς «ξυπνά», μειώνει το σημαφόρο κατά 1 και ολοκληρώνει τη wait
  - Καλείται πριν την είσοδο στο κρίσιμο τμήμα

Λειτουργίες σημαφόρων

- V (semaphore) - signal:
  - Ατομική λειτουργία που αυξάνει τον σημαφόρο κατά 1
  - Η λειτουργία αυτή επιτρέπει στις διεργασίες που έχουν μπλοκαριστεί στη wait να ξεκολλήσουν
  - Καλείται κατά την έξοδο από το κρίσιμο τμήμα

Είδη σημαφόρων

- Δυαδικοί σημαφόροι (binary semaphores) - τιμές 0 και 1
  - Χρησιμοποιούνται για να επιτευχθεί ο αμοιβαίος αποκλεισμός
- • Μετρητές σημαφόροι (counting semaphores)
  - Μπορούν να λάβουν οποιαδήποτε μη αρνητική τιμή
  - Χρησιμοποιούνται για τη διαχείριση των περιορισμένων πόρων των συστημάτων

Ιδιότητες σημαφόρων

- Ανεξάρτητοι του υλικού
- Απλοί
- Λειτουργούν με πολλές διεργασίες
- Μπορούν να υπάρχουν πολλά κρίσιμα τμήματα στον κώδικα, το καθένα με τον δικό του σημαφόρο
- Μπορούν να αποκτούν πολλούς πόρους ταυτόχρονα
- Κάθε σημαφόρος έχει μια ακέραια τιμή και μια ουρά από συσχετιζόμενες διεργασίες τις οποίες εξυπηρετεί
- Μη ύπαρξη ενεργούς αναμονής

Αμοιβαίος αποκλεισμός με σημαφόρους

- Χρήση δυαδικού σημαφόρου που αρχικοποιείται στην τιμή 1

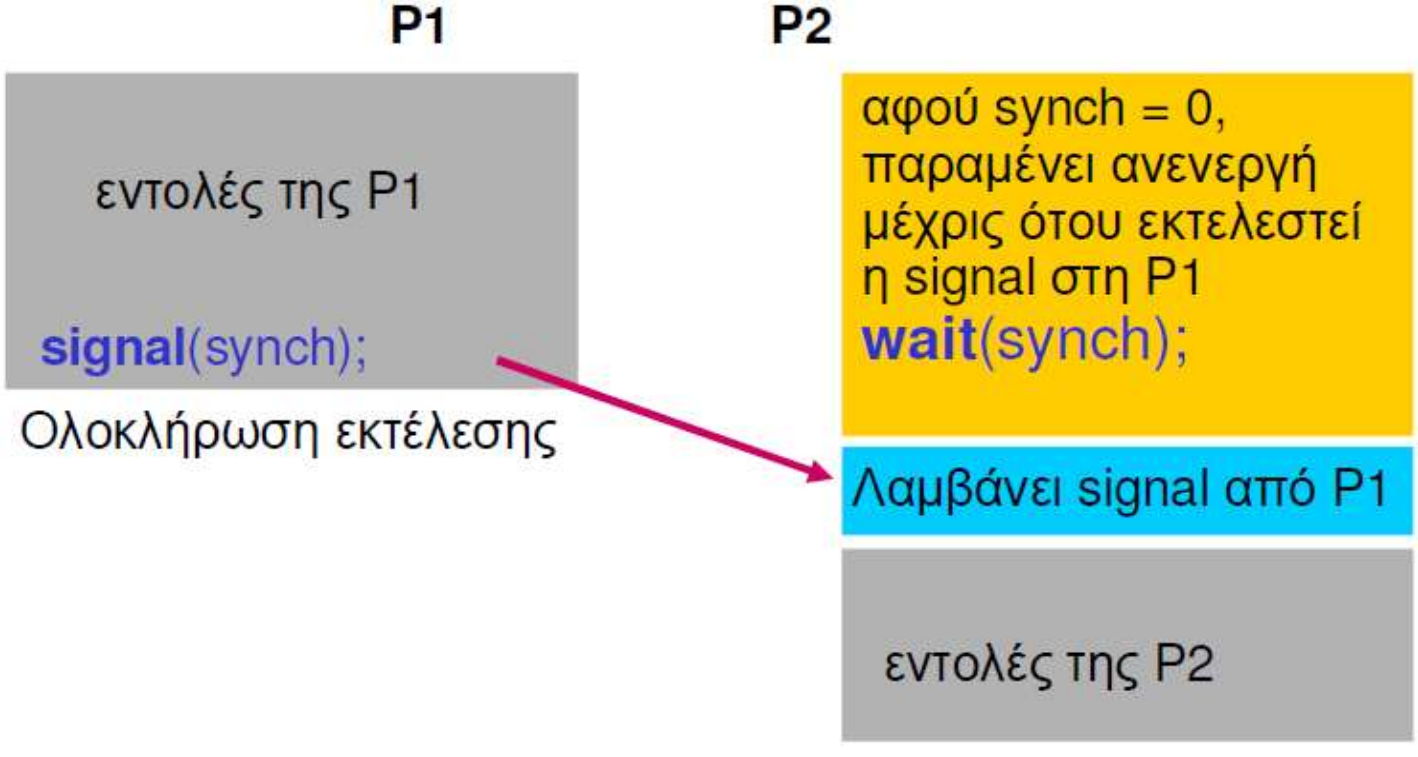
```
Semaphore Q: 1;
/* shared */

wait(Q);
<critical section>
signal(Q);
<remainder>
```

Παράδειγμα σημαφόρου

- Έστω οι διεργασίες P1 και P2
- Στόχος: η διεργασία P2 να εκτελεστεί υποχρεωτικά μετά από την P1
- Χρησιμοποιείται ένας κοινός σημαφόρος με όνομα synch για να συγχρονίσει τις λειτουργίες των δύο σύγχρονων διεργασιών
  - Οι εντολές wait, signal χρησιμοποιούνται για να καθυστερήσουν την P2 μέχρι να ολοκληρωθεί η P1
  - Ο σημαφόρος synch αρχικοποιείται με τιμή 0

Παράδειγμα σημαφόρου



Συγχρονισμός Εκτέλεσης

- Εκτέλεση του τμήματος Β στη διεργασία P2 μετά από την εκτέλεση του τμήματος Α στη διεργασία P1
- Χρήση ενός σημαφόρου flag με αρχική τιμή 0
- Κώδικας :

P1	P2
A	wait(flag)
signal(flag)	B

Αδιέξοδο με σημαφόρους

- Οι σημαφόροι πρέπει να χρησιμοποιούνται με προσοχή (αρχικά S=Q=1)

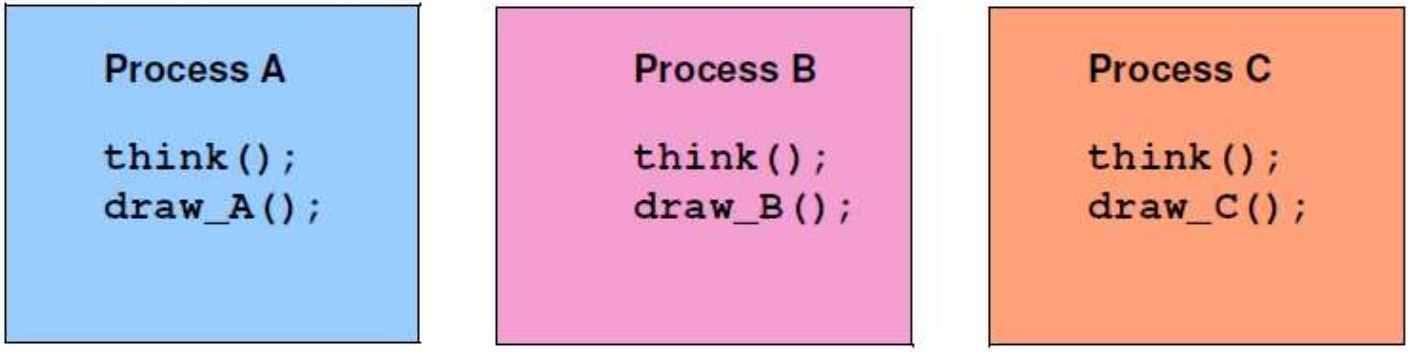
P0 περιμένει την P1 να εκτελέσει signal(Q)

P1 περιμένει την P0 να εκτελέσει signal(S)

Και οι δύο διεργασίες βρίσκονται σε αδιέξοδο

Παράδειγμα

- Τρεις διεργασίες διαμοιράζονται έναν πόρο στον οποίο
  - Μία σχεδιάζει ένα A
  - Μία σχεδιάζει ένα B
  - Μία σχεδιάζει ένα C
- Να υλοποιήσετε μια μορφή συγχρονισμού έτσι ώστε να εμφανίζονται κατά σειρά τα γράμματα A B C



Λύση με Σημαφόρους

```
Semaphore b = 0, c = 0;
```

Process A

```
think();
draw_A();
signal(b);
```

Process B

```
wait(b);
think();
draw_B();
signal(c);
```

Process C

```
wait(c);
think();
draw_C();
```

Ερωτήσεις$_{1/2}$

- Διεργασία (process) είναι:
  a) ένα πρόγραμμα.
  b) ένας επεξεργαστής.
  c) ένα ειδικό αρχείο.
  d) ένα πρόγραμμα σε εκτέλεση.
- Η λίστα έτοιμων διεργασιών (ready list) περιλαμβάνει:
  a) τις διεργασίες (processes) που περιμένουν για είσοδο/έξοδο.
  b) τις διεργασίες που περιμένουν σε κάποιο σημαφόρο (semaphore).
  c) τις διεργασίες που περιμένουν επιπλέον μνήμη.
  d) τις διεργασίες που περιμένουν για εξυπηρέτηση από την κεντρική μονάδα επεξεργασίας.

Ερωτήσεις$_{2/2}$

- Κρίσιμα τμήματα (critical sections) διεργασιών (processes):

  a) απαγορεύεται να εκτελούνται ταυτόχρονα από πολλές διεργασίες, χωρίς να αποκλείεται κάποιες να εκτελούν ταυτόχρονα μη κρίσιμα τμήματά τους.

  b) απαγορεύεται να εκτελούνται ταυτόχρονα από πολλές διεργασίες και αποκλείεται κάποιες να εκτελούν ταυτόχρονα μη κρίσιμα τμήματά τους.

  c) επιτρέπεται να εκτελούνται ταυτόχρονα από πολλές διεργασίες.

  d) όταν εκτελούνται τερματίζονται οι αντίστοιχες διεργασίες.

- Οι σηματοφόροι (semaphores) χρησιμεύουν για να:

  a) γίνεται πιο εύκολα η μετάφραση προγραμμάτων.

  b) συγχρονίζονται οι διεργασίες.

  c) επιταχύνεται η εκτέλεση προγραμμάτων.

  d) αντικαταστήσουν πολύπλοκες δομές δεδομένων.

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
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Τέλος Ενότητας

Αμοιβαίος Αποκλεισμός