Λειτουργικά Συστήματα

Ενότητα 4α: Σημαφόροι, Πρόβλημα Συνδαιτυμόνων Φιλοσόφων, 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.

Υλικό Συγχρονισμού

Ατομικός έλεγχος και τροποποίηση του περιεχομένου μίας λέξης.

```
function Test-and-Set(var target:boolean):boolean;
begin
      Test-and-Set := target;target := true;
end;
```

Αλγόριθμος Αμοιβαίου Αποκλεισμού

Διαμοιραζόμενα Δεδομένα
var lock : boolean (αρχικά false)

Διεργασία Pi

repeat
while Test-and-Set(lock) do no-op;
critical section
lock:= false;
remainder section
until false;

Απαγόρευση interrupts κατά την εκτέλεση TSL.

Κλειδώνει αρτηρία συστήματος κατά την εκτέλεση της.

Αμοιβαίος Αποκλεισμός Υποστήριξη Υλικού (1/3)

Απενεργοποίηση Διακοπών

- Μία διεργασίες εκτελείτε μέχρι να επικαλεσθεί μία υπηρεσία του λειτουργικού συστήματος ή μέχρι να διακοπεί.
- Η απενεργοποίηση των διακοπών εγγυάται τον αμοιβαίο αποκλεισμό.
- Ο επεξεργαστής είναι περιορισμένος στο να εναλλάσσει προγράμματα.
- Πολυεπεξεργασία

  Η απενεργοποίηση διακοπών σε έναν επεξεργαστή δεν εγγυάται τον αμοιβαίο αποκλεισμό.

Αμοιβαίος Αποκλεισμός Υποστήριξη Υλικού (2/3)

Ειδικές Οδηγίες Μηχανής

- Εκτελείται σε έναν ενιαίο κύκλο οδηγιών.
- Δεν υπόκειται σε παρεμβολές από άλλες οδηγίες.
- Ανάγνωση και εγγραφή.
- Ανάγνωση και έλεγχος.

Αμοιβαίος Αποκλεισμός Υποστήριξη Υλικού (3/3)

Οδηγίες Test and Set

```
boolean testset (int i){
      if (i == 0){
         i = 1;
         return true;
      }
      else{
        return false;
      }
}
until false;
```

Σημαφόροι (1/3)

- Μια ειδική μεταβλητή που χρησιμοποιείται για σηματοδότηση (signaling).
- Εάν μία διεργασία αναμένει για ένα σήμα (signal), αναστέλλεται (suspended) μέχρι να αποσταλεί αυτό το σήμα.
- Οι wait και signal ενέργειες δεν μπορούν να διακοπούν (interrupted).
- Χρησιμοποιείται ουρά για να κρατά τις διεργασίες που αναμένουν στο σημαφόρο.

Σημαφόροι (2/3)

Ακέραια μεταβλητή

- Μπορεί να αρχικοποιηθεί με έναν μη αρνητικό αριθμό.
- Η wait μειώνει την τιμή του σημαφόρου.
- Η signal αυξάνει την τιμή του σημαφόρου.

Σημαφόροι (3/3)

Εργαλείο συγχρονισμού που δεν απαιτεί ενεργό αναμονή (busy waiting).

Busy Waiting

- Η διεργασία ελέγχει συνεχώς να δει εάν μπορεί να μπει στο κρίσιμο τμήμα της.
- Η διεργασία δε μπορεί να κάνει τίποτα το παραγωγικό μέχρι να πάρει «άδεια» για να μπει στο κρίσιμο τμήμα της.

Σημαφόρος S

- Ακέραια μεταβλητή.
- Μπορεί να προσπελαθεί μέσω δύο ατομικών διακριτών πράξεων:

```
wait(S)
        S := S - 1;
        if S < 0 then block(S)
signal(S)
        S := S + 1;
        if S <= 0 then weakup(S)
```

block(S): έχει ως αποτέλεσμα την αναστολή της διεργασίας που την καλεί.

weakup(S): έχει ως αποτέλεσμα τη συνέχιση μιας ακριβώς διεργασίας που έχει καλέσει την block(S).

Παρέχονται ως βασικές κλήσεις συστήματος

Κρίσιμο Τμήμα για n Διεργασίες (1/3) (Παράδειγμα)

Διαμοιραζόμενες Μεταβλητές
var mutex: semaphore αρχικά mutex = 1

Διεργασία Pi
repeat **wait(mutex);** critical section **signal(mutex);** remainder section until false;

Κρίσιμο Τμήμα για n Διεργασίες (2/3) (Παράδειγμα)

Υλοποίηση της wait και signal έτσι ώστε να εκτελούνται ΑΤΟΜΙΚΑ.

- Μονο-επεξεργαστικό περιβάλλον.

  Αναστέλλει τις διακοπές γύρω από το τμήμα του κώδικα που υλοποιεί τις πράξεις wait και signal.

- Πολυ-επεξεργαστικό περιβάλλον.

  Εάν δεν παρέχεται ειδικό υλικό μέρος, χρήση λύσης μέσω του λογισμικού για το πρόβλημα του κρίσιμου τμήματος όπου τα κρίσιμα τμήματα περιλαμβάνουν τις πράξεις wait και signal.

Χρήση ειδικού hardware εάν είναι διαθέσιμο, δηλαδή Test-and-Set.

Κρίσιμο Τμήμα για n Διεργασίες (3/3) (Παράδειγμα)

Υλοποίηση της πράξης wait(S) με την εντολή Test-and-Set.

Διαμοιραζόμενες Μεταβλητές
var lock : boolean αρχικά lock = false

Κώδικας της Wait(S)

while Test-and-Set(lock) do no-op;
S := S-1;
if S < 0 then
begin
lock := false;
block(S);
end
else
lock := false;

Πρόβλημα Συγχρονισμού
Πρόβλημα Περιορισμένης Ενδιάμεσης Μνήμης

Διαμοιραζόμενα δεδομένα

```
type item = ...;
var buffer = ...;
    full, empty, mutex: semaphore;
    nextp, nextc : item; full:=0; empty:=n; mutex:=1;
```

Διαδικασία Παραγωγού

```
repeat
  ...
  produce an item in nextp;
  ...
  wait(empty);
  wait(mutex);
  ...
  add nextp to buffer;
  ...
  signal(mutex);
  signal(full);
until false;
```

Διαδικασία Καταναλωτή

```
repeat
    wait(full);
    wait(mutex);
    ...
    remove an item from buffer to
    nextc;
    ...
    signal(mutex);
    signal(empty);
    ...
    consume the item in nextc;
    ...
until false;
```

Πρόβλημα Αναγνωστών/ Συγγραφέων (1/2) (Readers/Writers Problem)

- Οποιοσδήποτε αριθμός αναγνωστών μπορεί να διαβάζει παράλληλα το αρχείο.
- Μόνο ένας συγγραφέας κάθε φορά μπορεί να γράφει στο αρχείο.
- Εάν ένας συγγραφέας γράφει στο αρχείο, κανένας αναγνώστης δε μπορεί να το διαβάζει.
- Όταν ο συγγραφέας ενημερώσει για τη λήξη της συγγραφής του, επιλέγεται ένας νέος αναγνώστης ή συγγραφέας μέσω ενός scheduler.

Πρόβλημα Αναγνωστών/ Συγγραφέων (2/2) (Readers/Writers Problem)

Διαμοιραζόμενα Δεδομένα

```
var mutex, wrt : semaphore
(=1);
readcount : integer (=0);
```

Διαδικασία Συγγραφέα

```
wait(wrt);
…
writing is performed;
…
signal(wrt);
```

Διαδικασία Αναγνώστη

```
wait(mutex);
readcount := readcount+1;
if readcount=1 then wait(wrt);
signal(mutex);
      …
reading is performed;
      …
wait(mutex);
readcount := readcount-1;
if readcount=0 then signal(wrt);
signal(mutex);
```

Πρόβλημα Συγχρονισμού
Πρόβλημα Συνδαιτυμόνων Φιλοσόφων

Κανόνες

- Κανείς δε μπορεί να σηκώσει και τα 2 chopsticks ταυτόχρονα και κατά τη διάρκεια του φαγητού δεν αφήνει τα chopsticks.
- **Απλή λύση:** 1 σημαφόρος ανά chopstick.

  Μπορεί να προκύψει αδιέξοδο;

Πρόβλημα Συνδαιτυμόνων Φιλοσόφων

Διαμοιραζόμενα Δεδομένα

```
var chopstick:array[0..4] of semaphore;
(αρχικά =1)
```

Φιλόσοφος i

```
repeat
      wait(chopstick[i]);
      wait(chopstick[i+1 mod 5]);
      …
      eat
      …
      signal(chopstick[i]);
      signal(chopstick[i+1 mod 5]);
      …
      think
      ...
until false;
```

Το γεύμα των φιλοσόφων (The dining philosophers)

Λύσεις για starvation

- Επιτρέπουμε το πολύ σε 4 φιλοσόφους να κάθονται ταυτόχρονα στο τραπέζι.
- Ένας φιλόσοφος μπορεί να πάρει chopstick μόνο όταν και τα δύο είναι διαθέσιμα.
- Χρήση μη-συμμετρικής λύσης:
  - Περιττός αριθμός φιλοσόφου: πρώτα αριστερό, μετά δεξί chopstick.
  - Άρτιος αριθμός φιλοσόφου: πρώτα δεξί, μετά αριστερό chopstick.

Κρίσιμες περιοχές (1/2)

- Δομή υψηλού επιπέδου για συγχρονισμό.
- Μία διαμοιραζόμενη μεταβλητή v τύπου T ορίζεται:

  var v : shared T

- Η μεταβλητή v είναι προσβάσιμη μόνο μέσα στην εντολή:

  region v when B do S, όπου B είναι μία Boolean έκφραση.

  Κατά τη διάρκεια εκτέλεσης της εντολής S, καμία άλλη διεργασία δε μπορεί να έχει πρόσβαση στη μεταβλητή v.

Κρίσιμες περιοχές (2/2)

- Οι περιοχές που αφορούν στην ίδια διαμοιραζόμενη μεταβλητή, αποκλείουν χρονικά η μία την άλλη.

- Όταν μία διεργασία προσπαθεί να εκτελέσει την εντολή της περιοχής, υπολογίζεται η Boolean έκφραση B.
  - Εάν η B είναι αληθής, εκτελείται η εντολή S.
  - Αλλιώς, καθυστερείται η διεργασία μέχρι να γίνει η B αληθής και να μην υπάρχει διεργασία της περιοχής συσχετιζόμενη με τη μεταβλητή v.

Περιορισμένη Ενδιάμεση Μνήμη (buffer) (Παράδειγμα)

Διαμοιραζόμενα δεδομένα

```
var buffer : shared record
pool : array[0..n-1] of item;
count, in, out : integer;
```

Διαδικασία Παραγωγού

```
region buffer when count<n
      do begin
         pool[in]:=nextp;
         in := in+1 mod n;
         count := count +1;
      end;
```

εισάγει το nextp στη διαμοιραζόμενη ενδιάμεση μνήμη (buffer).

Διαδικασία Καταναλωτή

```
region buffer when count>0
      do begin
         nextc := pool[out];
         out := out+1 mod n;
         count := count -1;
      end;
```

εξάγει ένα αντικείμενο από τη διαμοιραζόμενη ενδιάμεση μνήμη (buffer) και το τοποθετεί στο nextc

Υλοποίηση της region x when B do S

- Συσχετίζουμε τις παρακάτω μεταβλητές, με τη διαμοιραζόμενη μεταβλητή x:

  var mutex, first-delay, second-delay : semaphore;

  first-count, second-count : integer;

- Αποκλειστική αμοιβαία πρόσβαση στο κρίσιμο τμήμα παρέχεται μέσω της μεταβλητής mutex.

- Εάν μία διεργασία δε μπορεί να μπει στο κρίσιμο τμήμα της επειδή η Boolean έκφραση B είναι false τότε:
  - αρχικά αναμένει στο σημαφόρο first-delay.
  - μετακινείται στο σημαφόρο second-delay πριν τον επαναπροσδιορισμό της έκφρασης B.

- Παρακολούθηση του αριθμού των αναμενόμενων διεργασιών στους σημαφόρους first-delay και second-delay με τις μεταβλητές first-count και second-count αντίστοιχα.

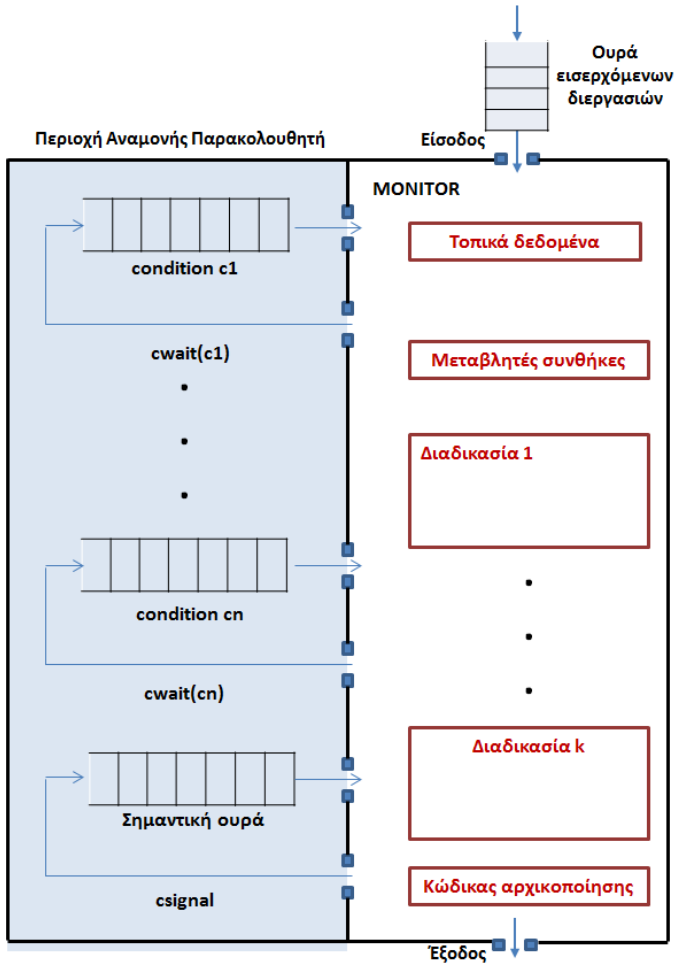
Παρακολουθητές (Monitors) (1/4)

Ο Παρακολουθητής είναι μια δομή γλώσσας προγραμματισμού.

Κύρια χαρακτηριστικά

- Οι μεταβλητές των τοπικών δεδομένων είναι προσβάσιμες μόνο από διαδικασίες του παρακολουθητή και όχι από οποιαδήποτε εξωτερική διαδικασία.
- Μία διεργασία εισέρχεται στον παρακολουθητή, καλώντας κάποιες από τις διαδικασίες του.
- Μόνο μία διεργασία μπορεί να εκτελεστεί σε έναν παρακολουθητή κάθε φορά.

Παρακολουθητές (Monitors) (2/4)



Η δομή ενός παρακολουθητή

Παρακολουθητές (Monitors) (3/4)

Δομή υψηλού επιπέδου που διασφαλίζει την ασφαλή διαμοίραση ενός αφηρημένου τύπου δεδομένων μεταξύ ταυτόχρονων διεργασιών.

```
type monitor-name = monitor
    variable declarations
    procedure entry P1(…);
      begin .. end;

    procedure entry P2(…);
      begin .. end;
    .
    .
    .

    procedure entry Pn(…);
      begin .. end;

    begin
      κώδικας αρχικοποίησης
    end;
```

Παρακολουθητές (Monitors) (4/4)

- Για να επιτρέπεται σε μία διεργασία να αναμένει μέσα σε έναν παρακολουθητή, πρέπει να δηλωθεί μία μεταβλητή συνθήκης:

  var x, y : condition;

- Η μεταβλητή συνθήκης μπορεί να χρησιμοποιηθεί μόνο με τις πράξεις wait και signal.

  Η πράξη x.wait; σημαίνει ότι η διεργασία που καλεί αυτήν την πράξη, αναστέλλεται έως κάποια άλλη διεργασία να καλέσει την πράξη x.signal;

  Η πράξη x.signal; επανεκκινεί ακριβώς μία από τις διεργασίες που είχαν ανασταλεί. Εάν δεν ανασταλεί καμία διεργασία, τότε η πράξη signal δεν έχει κανένα αποτέλεσμα.

Παρακολουθητές
Το γεύμα των φιλοσόφων

```
var state : array[0..4] of (thinking, hungry, eating);
var self : array[0..4] of condition;
procedure entry pickup (i:0..4);
begin
      state[i] = hungry;
      test (i);
      if state[i] = eating then self[i].wait;
end;
procedure entry putdown (i:0..4);
begin
      state[i] = thinking;
      test (i+4 mod 5);
      test (i+1 mod 5);
end;
procedure test (k:0..4);
begin
      if state[k+4 mod 5] = eating and
      state[k] = hungry and
      state[k+1 mod 5] = eating
      then begin
         state[k] = eating;
         self[k].signal;
      end;
end;
begin
      for i := 0 to 4  do state[i] := thinking;
end.
```

Γεύμα των φιλοσόφων - Παρακολουθητές

Το πρόβλημα του αδιεξόδου (Deadlock)

Ένα σύνολο από μπλοκαρισμένες διεργασίες κάθε μία από τις οποίες κρατά έναν πόρο και περιμένει να χρησιμοποιήσει κάποιον άλλο πόρο που κατέχεται από άλλη διεργασία του συνόλου.

Παράδειγμα 1

- Το σύστημα έχει 2 οδηγούς μαγνητοταινιών.
- P1 και P2 κάνουν χρήση του ενός οδηγού μαγνητοταινίας και η κάθε μία χρειάζεται τον άλλο οδηγό μαγνητοταινίας.

Παράδειγμα 2

- Οι σημαφόροι A και B, έχουν τιμή εκκίνησης το 1.

P_0	P_1
wait(A)	wait(B)
wait(B)	wait(A)

Αναφορές

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

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

[3] Raymond W. Turner, "Operating systems: design and implementation", New York: Macmillan Publ.

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

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