Λειτουργικά Συστήματα

Ενότητα 3: Δρομολόγηση Κεντρικής Μονάδας Επεξεργασίας

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.

Βασικές Έννοιες

Μεγιστοποίηση της χρήσης της CPU επιτυγχάνεται με τον πολυ-προγραμματισμό.

Η εκτέλεση των διεργασιών περιλαμβάνει έναν κύκλο από CPU εκτέλεση και αναμονή Εισόδου/Εξόδου (Ε/Ε).

Συνεχής εναλλαγή μεταξύ CPU και Ε/Ε “καταιγισμών” (burst).

Κατανομή καταιγισμού CPU

Μικροπρόθεσμος Δρομολογητής

Επιλέγει από ένα σύνολο διεργασιών της μνήμης, οι οποίες είναι έτοιμες για εκτέλεση και κατανέμει τη CPU σε μία από αυτές τις διεργασίες.

Οι αποφάσεις δρομολόγησης της CPU γίνονται όταν:

1. Εναλλαγή από την εκτελούμενη (running) κατάσταση στην κατάσταση αναστολής (waiting).
2. Εναλλαγή από την εκτελούμενη (running) κατάσταση στην έτοιμη κατάσταση (ready).
3. Εναλλαγή από την κατάσταση αναστολής (waiting) στην έτοιμη (ready) κατάσταση.
4. Τερματισμός.

Η δρομολόγηση λόγω 1. και 4. είναι μη-προεκχωρίσιμη.

Κάθε άλλη δρομολόγηση είναι προεκχωρίσιμη.

Κριτήρια Δρομολόγησης

- **Χρησιμοποίηση CPU** (CPU usage)**:** όσο γίνεται απασχολημένη.
- **Φόρτος εργασίας** (workload)**:** αριθμός διεργασιών που ολοκληρώνουν την εκτέλεση τους ανά μονάδα χρόνου.
- **Χρόνος επιστροφής** (turnaround time)**:** χρονικό διάστημα από την υποβολή μίας διεργασίας έως και την ολοκλήρωση της (δηλαδή άθροισμα χρόνων αναμονής για μνήμη, ουρά αναμονής, εκτέλεση στη CPU και Ε/Ε).
- **Χρόνος αναμονής** (waiting time) **:** ο χρόνος που περνά μία διεργασία στην ουρά έτοιμων διεργασιών.
- **Χρόνος απόκρισης** (response time)**:** χρονικό διάστημα που μεσολαβεί από τη στιγμή του αιτήματος έως την έναρξη της πρώτης απόκρισης.

Βελτιστοποίηση

- Μεγιστοποίηση Χρήσης CPU.
- Μεγιστοποίηση Φόρτου εργασίας.
- Ελαχιστοποίηση χρόνου επιστροφής.
- Ελαχιστοποίηση χρόνου αναμονής.
- Ελαχιστοποίηση χρόνου απόκρισης.

FCFS (First Come First Server) (1/2) Δρομολόγηση

Διεργασίας	Χρόνος καταιγισμού
P_1	24
P_2	3
P_3	3

Έστω ότι οι διεργασίες φθάνουν με τη σειρά P_1, P_2, P_3.
Το διάγραμμα Gantt για δρομολόγηση είναι:

Χρόνος αναμονής: $P_1 = 0$
$P_2 = 24$
$P_3 = 27$
Μέσος χρόνος αναμονής: (0+24+27)/3 = 17

FCFS (First Come First Server) (2/2) Δρομολόγηση

Έστω ότι οι διεργασίες φθάνουν με τη σειρά P_2, P_3, P_1.
Το διάγραμμα Gantt για δρομολόγηση είναι:

Χρόνος αναμονής: $P_1 = 6$
$P_2 = 0$
$P_3 = 3$
Μέσος χρόνος αναμονής: (6+0+3)/3 = 3

Φαινόμενο "φάλλαγας"
Οι σύντομες διεργασίες πριν από τις "χρονοβόρες".

SJF(Shortest Job First) (1/2) Δρομολόγηση

Συσχέτιση κάθε διεργασίας με το μήκος του επόμενου της CPU καταιγισμού.

Χρήση των μεγεθών αυτών για δρομολόγηση της πλέον σύντομης διεργασίας.

Μη-προεκχωρημένη (nonpreemptive)

Από τη στιγμή που η CPU ανατίθεται σε μία διεργασία, η CPU δε μπορεί να προεκχωρηθεί μέχρι να ολοκληρωθεί ο CPU καταιγισμός.

Προεκχωρημένη (preemptive)

Εάν φθάσει μία διεργασία με καταιγισμό CPU μικρότερου μήκους από το χρόνο που απομένει για την τρέχουσα εκτελούμενη διεργασία έχουμε προεκχώρηση.

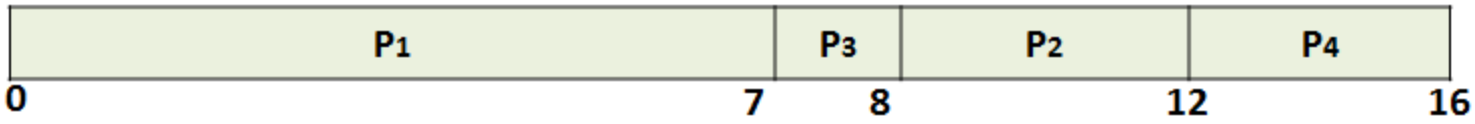
Shortest Remaining Time First (SRTF)

Ο SJF είναι βέλτιστος: καταλήγει σε ελάχιστο μέσο χρόνο αναμονής για δεδομένο σύνολο διεργασιών.

SJF(Shortest Job First) (2/2) Δρομολόγηση

Διεργασίας	Χρόνος άφιξης	Χρόνος CPU
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

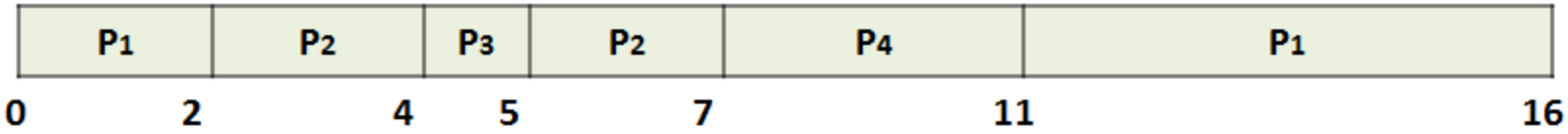
SJF, Μη-προεκχώρηση



Μέσος χρόνος αναμονής: (0+6+3+7)/4 = 4

SRTF(Shortest Remaining Time First) (1/2) Δρομολόγηση

SRTF, Προεκχώρηση



Μέσος χρόνος αναμονής: (9+1+0+2)/4 = 3

Επόμενος CPU καταιγισμός: μπορεί να γίνει μόνο εκτίμηση του μήκους.

SRTF(Shortest Remaining Time First) (2/2) Δρομολόγηση

- Χρήση του μήκους των προηγούμενων CPU καταιγισμών, με εκθετική προσέγγιση.
- T_n = πραγματικό μήκος του n-οστού CPU καταιγισμού.
- ψ_n = προβλεπόμενη τιμή του n-οστού CPU καταιγισμού.

$$0 \leq W \leq 1$$

Ορίζουμε: $\psi_{n+1} = W * T_n + (1 - W)\psi_n$

Παραδείγματα

$W = 0 \longrightarrow \psi_{n+1} = \psi_n$

Η πιο πρόσφατη συμπεριφορά δεν επηρεάζει.

$W = 1 \longrightarrow \psi_{n+1} = T_n$

Μόνο ο πραγματικός τελευταίος CPU καταιγισμός παίζει ρόλο.

Δρομολόγηση Προτεραιοτήτων (1/2)

Ένας ακέραιος αριθμός συνδέεται με κάθε διεργασία.

Η CPU κατανέμεται στη διεργασία σύμφωνα με την υψηλότερη προτεραιότητα. Συνήθως ο μικρότερος ακέραιος έχει τη μέγιστη προτεραιότητα.

- Μη-προεκχώρηση (nonpreemptive)
- Προεκχώρηση (preemptive)

Δρομολόγηση Προτεραιοτήτων (2/2)

SJN δρομολόγηση προτεραιοτήτων, όπου ως προτεραιότητα θεωρείται ο επόμενος προβλεπόμενος CPU καταιγισμός.

Πρόβλημα

Παρατεταμένη στέρηση (Starvation)

Οι διεργασίες χαμηλής προτεραιότητας μπορεί να μην εκτελεστούν ποτέ.

Λύση

Ωρίμανση (Aging)

Σταδιακή αύξηση προτεραιότητας διεργασιών

Round Robin (RR) (1/3)

Κάθε διεργασία παίρνει μία μικρή ενότητα του CPU χρόνου (quantum) περίπου 10-100 millisecs. Μετά την πάροδο αυτού του χρόνου, η διεργασία προεκχωρείται και προστίθεται στο τέλος της ουράς έτοιμων διεργασιών.

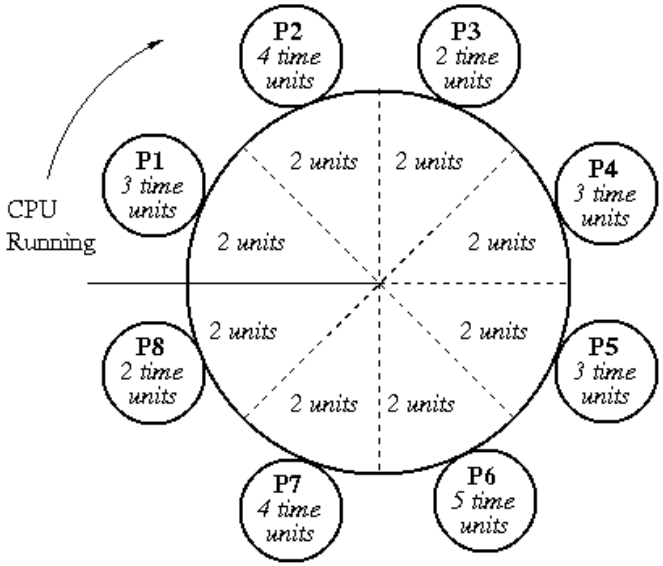
Εάν υπάρχουν n διεργασίες στην ουρά έτοιμων διεργασιών και η ενότητα χρόνου (quantum) είναι q, κάθε διεργασία παίρνει 1/ n του CPU χρόνου σε κομμάτια το πολύ έως q ενοτήτων χρόνου κάθε φορά.

Καμία διεργασία δεν περιμένει περισσότερο από (n-1) q ενότητες χρόνου.

Round Robin (RR) (2/3)

Επίδοση

- **Μεγάλο q:** FIFO
- **Μικρό q:** το q πρέπει να είναι μεγάλο σχετικά με την εναλλαγή περιεχομένου, διαφορετικά το κόστος (overhead) είναι μεγάλο.

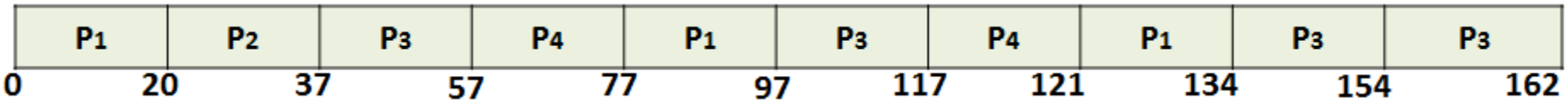


Round Robin [5]

Round Robin (RR) (3/3)

Διεργασίας	Χρόνοι CPU
P1	53
P2	17
P3	68
P4	24

Το διάγραμμα Gantt είναι:



Συνήθως μεγαλύτερο μέσο όρο χρόνου διακπαιρέωσης (turnaround) από τον SRT, αλλά καλύτερη αποκρισιμότητα.

Ουρά πολλαπλών επιπέδων (1/2)

- Η ουρά έτοιμων διεργασιών χωρίζεται σε 2 ξεχωριστές ουρές.

  Παράδειγμα :

  – προσκήνιο (διαλογική, interactive).

  – παρασκήνιο (ομαδική επεξεργασία, batch).

- Κάθε ουρά έχει τον δικό της αλγόριθμο δρομολόγησης.

  Παράδειγμα :

  – προσκήνιο (RR).

  – παρασκήνιο (FCFS).

Ουρά πολλαπλών επιπέδων (2/2)

- Δρομολόγηση πρέπει να επιβληθεί και μεταξύ των ουρών.
- Δρομολόγηση σταθερής προτεραιότητας.

  Παράδειγμα: εξυπηρέτηση όλου του προσκήνιου και στη συνέχεια από το παρασκήνιο.

  Πιθανότητα παρατεταμένης στέρησης.
- Δρομολόγηση κομματιών χρόνου.

  Κάθε ουρά έχει ένα συγκεκριμένο χρόνο CPU που μπορεί να δρομολογήσει μεταξύ των διεργασιών της.

  Παράδειγμα
  - 80% στο προσκήνιο με RR.
  - 20% στο παρασκήνιο με FCFS.

Ουρά Πολλαπλής Ανάδρασης

- Μία διεργασία μπορεί να μετακινηθεί μεταξύ των διαφόρων ουρών. Με αυτό τον τρόπο μπορεί να υλοποιηθεί η ωρίμανση.

- Η ουρά Πολλαπλής Ανάδρασης καθορίζεται από τις εξής παραμέτρους:
  - Αριθμό Ουρών.
  - Αλγόριθμος δρομολόγησης για κάθε ουρά.
  - Μέθοδος που καθορίζει πότε θα γίνει η αναβάθμιση μίας διεργασίας.
  - Μέθοδος που καθορίζει πότε θα γίνει η υποβάθμιση μίας διεργασίας.
  - Μέθοδος που καθορίζει την ουρά στην οποία θα εισαχθεί μία διεργασία που απαιτεί εξυπηρέτηση.

Δρομολόγηση Πολλαπλών Επεξεργαστών

- Η δρομολόγηση της CPU είναι πιο περίπλοκη όταν πολλαπλές CPU είναι διαθέσιμες.
- Ομογενείς επεξεργαστές με πολυ-επεξεργαστή.
- Διαμοίραση φορτίου.
- Ασύμμετρη πολυ-επεξεργασία: μόνο ένας επεξεργαστής έχει πρόσβαση στις δομές δεδομένων του συστήματος.
- Δρομολόγηση Πραγματικού Χρόνου.
  - "Αυστηρά" συστήματα Πραγματικού Χρόνου:
    Απαιτείται να ολοκληρώνουν μία κρίσιμη ενέργεια σε συγκεκριμένο χρόνο.
  - "Χαλαρά" συστήματα Πραγματικού Χρόνου:
    Απαιτείται να δίνεται προτεραιότητα στις κρίσιμες διεργασίες.

Αλγοριθμικός Υπολογισμός

- Η επιλογή του κατάλληλου αλγορίθμου Δρομολόγησης είναι περίπλοκη διαδικασία.

- Προσδιοριστικό μοντέλο (deterministic model)

  Ξεκινά με δεδομένο έναν προκαθορισμένο φόρτο εργασίας και καθορίζει τη συμπεριφορά κάθε αλγορίθμου για το συγκεκριμένο φόρτο.

- Μοντέλα Ουρών (Queueing model)

  Κανόνας του Little: $n = \lambda \times W$

  – n: μέσο μήκος ουράς.
  – λ: μέσος ρυθμός άφιξης νέων διεργασιών στην ουρά.
  – W: μέσος χρόνος αναμονής στην ουρά.

- Προσομοίωση (Simulation)

Προγραμματισμός ενός μοντέλου του υπολογιστικού συστήματος.

Ντετερμινιστικό Μοντέλο (1/2) (Παράδειγμα)

Θεωρούμε το παρακάτω σύνολο διεργασιών.

Διεργασίας	Χρόνος καταιγισμού
P1	10
P2	29
P3	3
P4	7
P5	12

Έστω ότι οι διεργασίες φθάνουν με τη σειρά P1, P2, P3, P4, P5 κατά τη χρονική στιγμή 0. Να σχεδιασθεί το διάγραμμα Gantt για δρομολόγηση FCFS, μη-προεκχωρήσιμη SJF, RR (quantum: 10msec).

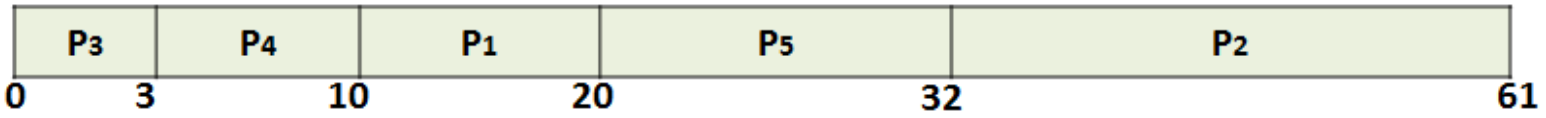
Να υπολογισθεί ο μέσος χρόνος αναμονής για κάθε μία διεργασία ανά διαφορετικό αλγόριθμο δρομολόγησης.

Ντετερμινιστικό Μοντέλο (2/2) (Παράδειγμα)

FCFS

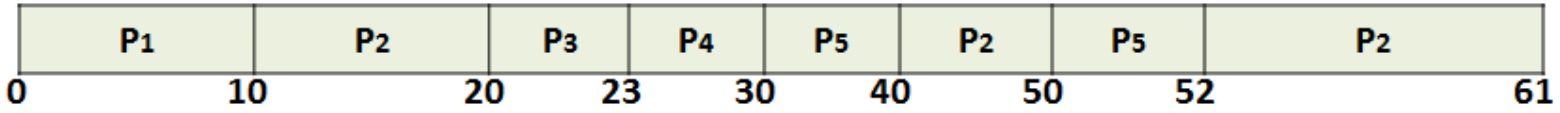
Μέσος χρόνος αναμονής: (0+10+39+42+49)/ = 28 msec.

SJF, Μη-προεκχώρηση



Μέσος χρόνος αναμονής: (10+32+0+3+20)/5 = 13 msec.

RR (quantum = 10msec)



Μέσος χρόνος αναμονής: (0+32+20+23+40)/5 = 23 msec.

Ερωτήσεις (1/2)

Θεωρούμε το παρακάτω σύνολο διεργασιών.

Διεργασίας	Χρόνος καταιγισμού	Προτεραιότητα
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

Έστω ότι οι διεργασίες φθάνουν με τη σειρά P1, P2, P3, P4, P5 κατά τη χρονική στιγμή 0. Να σχεδιασθεί το διάγραμμα Gantt για δρομολόγηση FCFS, μη-προεκχωρήσιμη SJF, RR (quantum: 1msec).

Να υπολογισθεί ο χρόνος αναμονής για κάθε μία διεργασία ανά διαφορετικό αλγόριθμο δρομολόγησης.

Ποιος αλγόριθμος καταλήγει στον ελάχιστο μέσο χρόνο αναμονής;

Ερωτήσεις (2/2)

Θεωρούμε το παρακάτω σύνολο διεργασιών.

Διεργασίας	Χρόνος καταιγισμού	Χρόνος άφιξης
P1	8	0.0
P2	4	0.4
P3	1	1.0

Να δρομολογηθούν οι διεργασίες χωρίς προεκχώρηση, με δρομολόγηση FCFS, και SJF.

Να υπολογισθεί ο μέσος χρόνος αναμονής για κάθε δρομολόγηση.

Συνεργαζόμενες Διεργασίες

- Οι ανεξάρτητες διεργασίες δε μπορούν να επηρεάσουν ή να επηρεαστούν από άλλες διεργασίες.

- Οι συνεργαζόμενες διεργασίες μπορούν να επηρεάσουν ή να επηρεαστούν από την εκτέλεση μίας άλλης διεργασίας.

Πλεονεκτήματα συνεργασίας διεργασιών:

- Διαμοίραση πληροφορίας.
- Επιτάχυνση υπολογισμών.
- Διαμόρφωση ενοτήτων.
- Ευκολία.

Πρόβλημα Παραγωγού-Καταναλωτή (1/2)

Παράδειγμα συνεργαζόμενων διεργασιών:

- Η διεργασία παραγωγός παράγει την πληροφορία που καταναλώνει η διεργασία καταναλωτής.
- Δύο γενικές περιπτώσεις:
  - μη-περιορισμένος ενδιάμεσος χώρος (unbounded buffer).
  - περιορισμένος ενδιάμεσος χώρος (bounded buffer).

Πρόβλημα Παραγωγού-Καταναλωτή (2/2)

Διαμοιραζόμενα δεδομένα

```
var n;
type item = …;
var buffer:
    array[0..n-1] of item;

in, out :0..n-1;
in:= 0;
out:= 0;
```

Διαδικασία Παραγωγού

```
repeat
   …
   produce an item in nextp;
   …
   while in+1 mod n=out do no-op;
   buffer[in]:= nextp;
   in:= in+1 mod n;
until false;
```

Διαδικασία Καταναλωτή

```
repeat
   while in = out do no-op;
   nextc:= buffer[out];
   out:= out+1 mod n;
   …
   consume the item in nextc
   ...
until false;
```

Βασικές Έννοιες Συγχρονισμού Διεργασιών

- Ταυτόχρονη πρόσβαση σε διαμοιραζόμενα δεδομένα μπορεί να έχει ως αποτέλεσμα την ασυνέπεια των δεδομένων.
- Η διατήρηση συνέπειας των δεδομένων απαιτεί την ύπαρξη μηχανισμών που να διασφαλίζουν τη διαδοχική εκτέλεση των συνεργαζόμενων διεργασιών.
- Η λύση της διαμοιραζόμενης μνήμης στο πρόβλημα του περιορισμένου ενδιάμεσου αποθηκευτικού χώρου (buffer),επιτρέπει την ταυτόχρονη ύπαρξη το πολύ n-1 αντικειμένων σε έναν buffer.
- Τροποποίηση του κώδικα του προβλήματος παραγωγού-καταναλωτή με προσθήκη μίας μεταβλητής counter που εκκινεί με την τιμή 0 και αυξάνεται κάθε φορά που ένα νέο αντικείμενο προστίθεται στον buffer.

Πρόβλημα Παραγωγού-Καταναλωτή

Διαμοιραζόμενα δεδομένα

```
type item = …;
var buffer: array[0..n-1] of item;

in, out : 0..n-1;
counter : 0..n;
 in := 0;
out := 0;
 counter :=0;
```

Διαδικασία Παραγωγού

```
repeat
…
produce an item in nextp
…
while counter=n do no-op;
buffer[in] := nextp;
in = in+1 mod n;
counter:= counter+1;
until false
```

Διαδικασία Καταναλωτή

```
repeat
while counter=0 do no-op;
nextc := buffer[out];
out := out+1 mod n;
counter := counter-1;
…
consume  item in nextc;
…
until false;
```

Οι εντολές

counter:= counter+1; counter:= counter-1;

πρέπει να εκτελεστούν ατομικά.

Πρόβλημα του Κρίσιμου Τμήματος (Critical-Section Problem) (1/7)

- n διεργασίες ανταγωνίζονται για τη χρήση κάποιων διαμοιραζόμενων δεδομένων.
- Κάθε διεργασία έχει ένα κομμάτι κώδικα, που καλείται κρίσιμο τμήμα, μέσω του οποίου γίνεται η πρόσβαση στα διαμοιραζόμενα δεδομένα.

ΠΡΟΒΛΗΜΑ: διαβεβαίωση ότι καμία άλλη διεργασία δεν επιτρέπεται να εκτελέσει το κρίσιμο τμήμα της, κατά τη διάρκεια εκτέλεσης του κρίσιμου τμήματος μίας διεργασίας.

Δομή Διεργασίας Pi

```
Repeat
    entry section
        critical section
    exit section
        remainder section
until false;
```

Πρόβλημα του Κρίσιμου Τμήματος (Critical-Section Problem) (2/7)

Απαραίτητες προϋποθέσεις για τη λύση του προβλήματος:

- **Αμοιβαίο Αποκλεισμό**

  Εάν η διεργασία Pi εκτελεί το κρίσιμο τμήμα της, καμία άλλη διεργασία δεν μπορεί να εκτελέσει το κρίσιμο τμήμα της.

- **Πρόοδος διεργασιών**

  Εάν καμία διεργασία δεν εκτελείται στο κρίσιμο τμήμα της και υπάρχουν κάποιες διεργασίες που θέλουν να προχωρήσουν στην εκτέλεση του κρίσιμου τμήματος τους, τότε η επιλογή της διεργασίας που θα προχωρήσει στο κρίσιμο τμήμα της δεν μπορεί να αναβάλλεται επ' αόριστον.

Πρόβλημα του Κρίσιμου Τμήματος (Critical-Section Problem) (3/7)

Απαραίτητες προϋποθέσεις για τη λύση του προβλήματος (Συνέχεια):

- **Περιορισμένη αναμονή**

  Πρέπει να υπάρχει όριο στο πόσες φορές επιτρέπεται οι άλλες διεργασίες να προχωρήσουν στο κρίσιμο τμήμα τους, μετά από το αίτημα εισαγωγής στο κρίσιμο τμήμα μίας διεργασίας και πριν από την ικανοποίηση του αιτήματος.

  Υποθέτουμε ότι κάθε διεργασία εκτελείται.

  – με μη-μηδενική ταχύτητα.

  Δεν υπάρχει υπόθεση αναφορικά με τη σχετική.

  – ταχύτητα των n διεργασιών.

Πρόβλημα του Κρίσιμου Τμήματος (Critical-Section Problem) (4/7)

Ιχνηλάτηση των αρχικών προσπαθειών επίλυσης του προβλήματος.

Μόνο 2 διεργασίες P0 και P1.

Γενική Δομή Διεργασίας Pi (άλλη διεργασία Pj)

```
Repeat
    entry section
        critical section
    exit section
        remainder section
until false;
```

Οι διεργασίες πρέπει να διαμοιράζονται κάποιες κοινές μεταβλητές για να συγχρονίζουν τις ενέργειες τους.

Πρόβλημα του Κρίσιμου Τμήματος (Critical-Section Problem) (5/7)

Αλγόριθμος 1

Διαμοιραζόμενες Μεταβλητές

var turn: (0..1);
αρχικά turn = 0

turn = i;
η Pi μπορεί να μπει στο κρίσιμο τμήμα της

Διεργασία Pi

```
Repeat
    while turn = i do no-op;
        critical section
    turn := j;
        remainder section
until false;
```

Ικανοποιεί τον αμοιβαίο αποκλεισμό, αλλά όχι την πρόοδο διεργασιών.

Πρόβλημα του Κρίσιμου Τμήματος (Critical-Section Problem) (6/7)

Αλγόριθμος 2

Διαμοιραζόμενες Μεταβλητές
var flag: array[0..1] of boolean; αρχικά flag[0] = flag[1] = false; flag[i] = true; η Pi μπορεί να μπει στο κρίσιμο τμήμα της

Διεργασία Pi

Repeat
flag[i] :=true;
while flag[j] do no-op;
critical section
flag[i] :=false;
remainder section
until false;

Δεν ικανοποιεί την πρόοδο των διεργασιών.

Πρόβλημα του Κρίσιμου Τμήματος (Critical-Section Problem) (7/7)

Αλγόριθμος 3

Συνδυασμός των διαμοιραζόμενων μεταβλητών των Αλγορίθμων 1 και 2.

Διεργασία Pi

```
Repeat
    flag[i] :=true;
    turn := j;
    while (flag[j] and turn = j) do no-op;
        critical section
    flag[i] :=false;
        remainder section
until false;
```

Ικανοποιεί και τις 3 προϋποθέσεις.

Επιλύει το πρόβλημα του κρίσιμου τμήματος για 2 διεργασίες.

Ο αλγόριθμος του αρτοποιείου (Κρίσιμο Τμήμα για n διεργασίες)

- Πριν την είσοδο μίας διεργασίας στο κρίσιμο τμήμα της η διεργασία λαμβάνει έναν αριθμό.
- Η διεργασία με το μικρότερο αριθμό εισάγεται στο κρίσιμο τμήμα της.

Εάν οι διεργασίες i και Pj έχουν πάρει τον ίδιο αριθμό
    εάν i < j
        τότε εξυπηρετείται πρώτα η Pi
    αλλιώς
        εξυπηρετείται πρώτα η Pj

- Το αριθμητικό σχήμα πάντα παράγει αριθμούς σε αύξουσα τάξη, για παράδειγμα1, 2, 3, 3, 3, 3, 4, 5, ...

(ticket #, process id #)
    (a, b) < (c, d) εάν a < c, ή εάν a = c και b < d
    max(a0, …, an-1) είναι ένας ακέραιος k, όπου $k \geq a_i$ για i = 0, …, n-1.

Αλγόριθμος αρτοποιείου (Bakery algorithm)

Διαμοιραζόμενα Δεδομένα

```
var choosing: array[0..1] of boolean;
number: array[0..1] of integer;
αρχικά choosing[i] = false; i = 0, …, n-1
number[i] = 0; i = 0, …, n-1
```

Διεργασία Pi

```
Repeat
    choosing[i] :=true;
    number[i] =max(number[0],…,number[n-1])+1;
    choosing[i] :=false;
    for j := 0 to n-1
    do begin
       while choosing[j] do no-op;
       while number[j] =0 and
       (number[j], j) < (number[i], i) do no-op;
    end;
       critical section
    number[i]:= 0;
       remainder section
until false;
```

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;

Σημαφόρος (Semaphore)

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

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).

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;

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

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

[5]. Scheduling and CPU Scheduling.

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