Παράλληλη Επεξεργασία
Κεφάλαιο 4
Επικοινωνία Διεργασιών

Κωνσταντίνος Μαργαρίτης
Καθηγητής
Τμήμα Εφαρμοσμένης Πληροφορικής
Πανεπιστήμιο Μακεδονίας
kmarg@uom.gr
http://eos.uom.gr/~kmarg

Αρετή Καπτάν
Υποψήφια Διδάκτορας
Τμήμα Εφαρμοσμένης Πληροφορικής
Πανεπιστήμιο Μακεδονίας
areti@uom.gr
http://eos.uom.gr/~areti

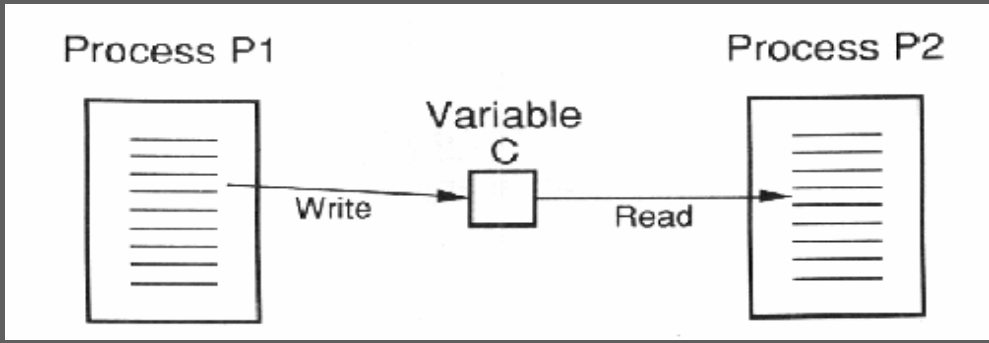
Περίληψη Κεφαλαίου

- Ασύγχρονοι αλγόριθμοι (Relaxed Parallel Algorithms)
- Επικοινωνία-αλληλεπίδραση διεργασιών – Μεταβλητές κανάλια (Channel Variables)
- Αλγόριθμοι διασωλήνωσης (Pipelined Algorithms)
- Επίλυση τριγωνικού συστήματος γραμμικών εξισώσεων με τη μέθοδο της προς-τα-πίσω-αντικατάστασης (Back Substitution)
- Διτονικός αλγόριθμος ταξινόμησης με συγχώνευση

Παράλληλες Διεργασίας και Επικοινωνία

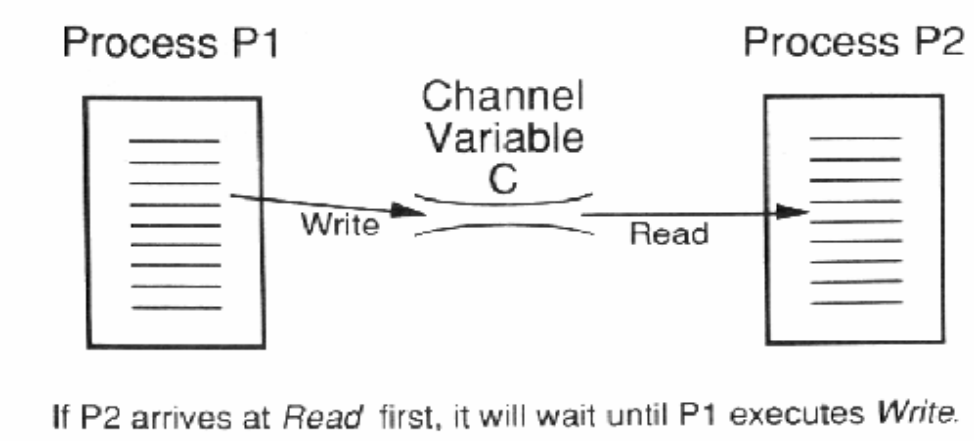
```
Fork For i:=1 to 20 do sum1:= sum1 +A[i];
Fork For j:=1 to 20 do sum2:= sum2 +B[j];
Join; Join;
```

Επικοινωνία Μεταξύ Παράλληλων Διεργασιών



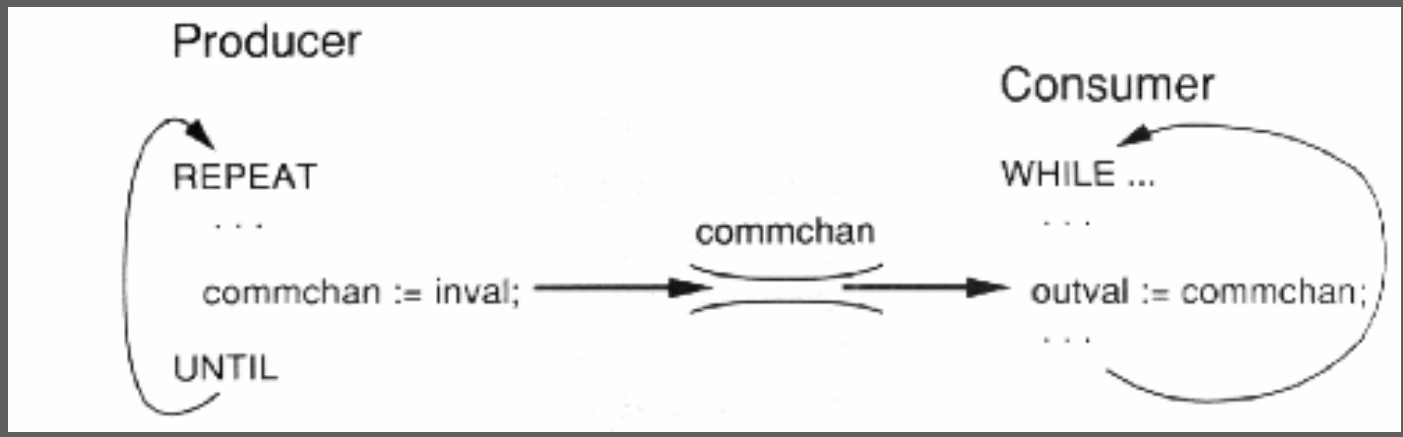
Αρχικοποίηση της μεταβλητής C από τη διεργασία P1.
Ανάγνωση της μεταβλητής C από τη διεργασία P2.

Ανάγνωση και Εγγραφή Καναλιών – Επικοινωνία με τη χρήση της μεταβλητής καναλιού



- Το κανάλι συμπεριφέρεται σαν μια ουρά με τιμές δεδομένων
- Οι τιμές αποθηκεύονται
- Ανάγνωση άδειου καναλιού συνεπάγεται αναμονή διεργασίας-αναγνώστη
- Εγγραφή-ανάγνωση μεταβλητής κανάλι
- Σύνταξη: `<όνομα καναλιού> : Channel of <τύπος δεδομένων>`,
  όπου <τύπος δεδομένων> μπορεί να είναι οι τύποι: Integer, Real, Boolean, Char
- Τελεστής ? για τον έλεγχο ενός καναλιού

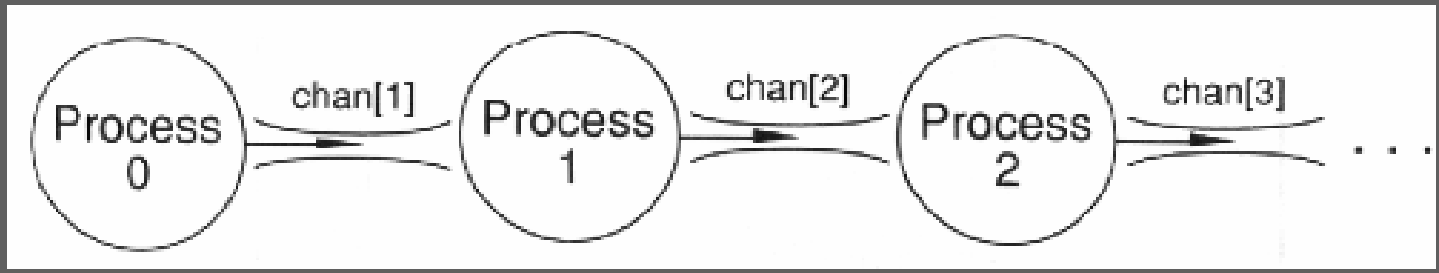
Αλληλεπίδραση Διεργασιών Παραγωγού - Καταναλωτή



Παράδειγμα Παραγωγού - Καταναλωτή

```
PROGRAM Producer_Consumer;
CONST   endmarker=-1;
VAR     commchan:CHANNEL OF INTEGER;

        PROCEDURE Producer;
        VAR inval:INTEGER;
        BEGIN
        REPEAT
            ... (* Υπολογίζει το νέο αντικείμενο inval για το κανάλι *)

            commchan := inval;        (* Γράφει μέσα στο κανάλι *)

        UNTIL inval=endmarker;
        END;

        PROCEDURE  Consumer;
        VAL outval:INTEGER;
        BEGIN
        outval:=commchan;             (* Διαβάζει από το κανάλι *)
        WHILE outval<>endmarker DO
            BEGIN
            ... (* χρησιμοποίηση της outval σε ορισμένους υπολογισμούς *)

            outval := commchan; (* Διαβάζει την επόμενη τιμή από το κανάλι *)

            END;

BEGIN   (*  Κυρίως πρόγραμμα  *)

        FORK  Producer; (* Διεργασία Producer *)
        FORK  Consumer; (* Διεργασία Consumer *)

END.
```

Διασωλήνωση Διεργασιών



- Λήψη δεδομένων από τον αριστερό γείτονα – επεξεργασία – αποστολή αποτελεσμάτων στον δεξιό γείτονα
- Κάθε διεργασία εκτελεί μετασχηματισμούς την ίδια στιγμή με διαφορετικά δεδομένα
- Κάθε διεργασία ενεργεί ταυτόχρονα και ως παραγωγός και ως καταναλωτής
- Δήλωση:

```
Var  chan: Array [1..n] of Channel of Integer;
```

- Ο αριθμός των παράλληλων διεργασιών περιορίζεται από το πλήθος των μετασχηματισμών που πραγματοποιούνται στα δεδομένα

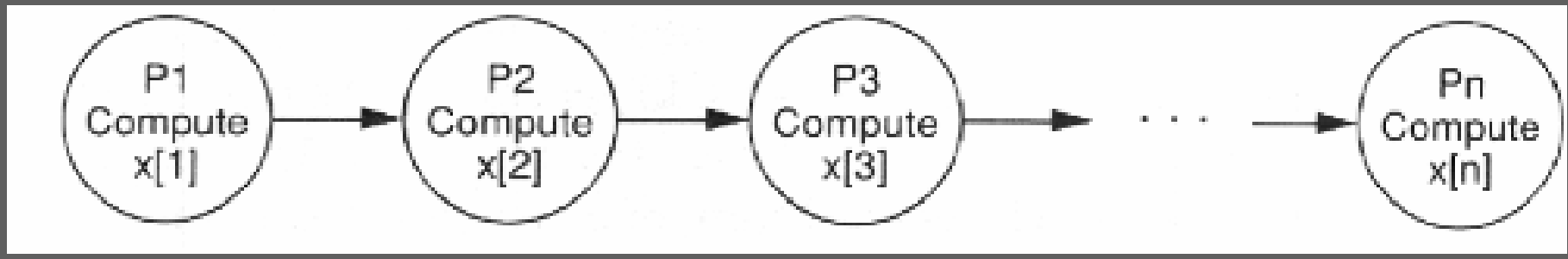
Δημιουργία Διεργασιών Διασωλήνωσης

```
PROCEDURE  PipeProcess(mynumber: INTEGER);
VAR inval, outval, i:INTEGER;
BEGIN
    FOR i:=1 TO m DO BEGIN
            inval:=chan[mynumber];  (* Διαβάζει από το αριστερό κανάλι *)
            ...   (* Χρήση της inval για να υπολογιστεί η inval *)
            chan[mynumber+1]:=outval;  (* Γράφει στο δεξιό κανάλι *)
    END;
END;
```

Σειριακός αλγόριθμος για επίλυση με προς-τα πίσω-αντικατάσταση

```
Sequential Back Substitution:

 FOR i:=1 TO n DO BEGIN
       sum:=0;
       FOR j:=1 TO i-1 DO
              sum:=sum+A[i,j]*x[j];
       x[i]:=(B[i]-sum)/A[i,i];
 END;
```

Διασωλήνωση διεργασιών στην προς-τα-πίσω-αντικατάσταση



Οι απαιτούμενες τιμές που μεταφέρονται μέσω της διασωλήνωσης πρέπει να φτάνουν με την σωστή σειρά.

Διασωλήνωση διεργασιών στην προς-τα-πίσω-αντικατάσταση

```
Pipeline Process i:

  sum:=0;
  FOR j:=1 TO i-1 DO BEGIN
        Διαβάζει την τιμή του x[j] από αριστερά;
        Στέλνει την τιμή του x[j] προς τα δεξιά;
        sum:=sum+A[i,j]*x[j];
  END;
  x[i]:=(B[i]-sum)/A[i,i];  (* Υπολογίζει την τιμή του x[i] *)

  Στέλνει το x[i] προς τα δεξιά;
```

```
PROGRAM BackSubstitution;
CONST
  n=50;
VAR
  A:ARRAY [1.. n,1.. n] OF REAL;
  B,x:ARRAY [1.. n] OF REAL;
  pipechan : ARRAY [1.. n+1] OF CHANNEL OF REAL;
  i : INTEGER;

PROCEDURE PipeProcess(i:INTEGER);
(*  Επιλύει την εξίσωση i για να υπολογίσει το x[i] *)
VAR
  j:INTEGER; sum,xvalue:REAL;
BEGIN
  sum:=0;
  FOR j:=1 TO i-1 DO BEGIN
    xvalue:=pipechan[i];   (* Διαβάζει το x[j] από τα αριστερά *)
    pipechan[i+1]:=xvalue; (* Στέλνει το x[j] προς τα δεξιά *)
    sum := sum + A[i,j]*xvalue;
  END;
  x[i]:=(B[i]-sum)/A[i,i];    (*  Υπολογίζει την τιμή για το x[i]  *)
  pipechan[i+1]:=x[i];            (*Στέλνει το x[i] προς τα δεξιά  *)
END;

BEGIN    (*  Κυρίως πρόγραμμα  *)

  ....   (*  Αρχικοποίηση των πινάκων A και B  *)

  FORALL i:=1 TO n DO         (*  Δημιουργία διεργασιών διασωλήνωσης  *)
    PipeProcess(i);

END.
```

Διτονική ταξινόμηση με συγχώνευση

- Επικοινωνία πινάκων κατά φάσεις (Phased-Array communications)
- Κάθε διεργασία επικοινωνεί με πολλές διαφορετικές διεργασίες. Η σειρά τους παίζει καθοριστικό ρόλο.
- Κάθε διεργασία έχει το δικό της πίνακα καναλιών
- Διτονική λίστα, τοπικό μέγιστο, τοπικό ελάχιστο

Διτονική λίστα με ένα τοπικό μέγιστο

Βήματα δυαδικής διάσπασης:

- Σε κάθε διεργασία του πρώτου μισού της λίστας ανατίθεται ως σύντροφος μια διεργασία που βρίσκεται στην ίδια σχετική θέση του δεύτερου μισού της λίστας. Αυτό απλά σημαίνει ότι σε κάθε διεργασία i του πρώτου μισού της λίστας ανατίθεται η διεργασία i+n/2 που ανήκει στο δεύτερο μισό της λίστας.
- Κάθε ζευγάρι διεργασιών-συντρόφων συγκρίνει τις τιμές της λίστας που της έχει ανατεθεί. Αν το στοιχείο της διεργασίας από το πρώτο μισό της λίστας είναι μεγαλύτερο από το στοιχείο της διεργασίας του δεύτερου μισού της λίστας, τότε οι διεργασίες ανταλλάσσουν τα στοιχεία τους ενώ σε αντίθετη περίπτωση δεν συμβαίνει καμιά ενέργεια.

Ιδιότητες της Δυαδικής Διάσπασης

- Κάθε στοιχείο του πρώτου μισού της λίστας είναι μικρότερο από κάθε στοιχείο του δεύτερου μισού της λίστας.
- Καθένα από τα δύο μισά της λίστας είναι από μόνα τους μια διτονική λίστα με μήκος n/2.

Αύξουσα Ταξινόμηση Διτονικής Λίστας

```
(* Ο αριθμός αναγνώρισης της διεργασίας είναι το  myid *)
FOR j:=d-1 DOWNTO 0 DO BEGIN (* Χρήση του j bit του myid *)
```

- Καθορισμός της διεργασίας-συντρόφου αντιστρέφοντας το j bit του myid;
- Αποστολή αντιγράφου του στοιχείου της διεργασίας προς τη διεργασία-σύντροφο;
- Παραλαβή αντίγραφου του στοιχείου της διεργασίας-συντρόφου;

```
    IF (το j bit του myid)=0 THEN
        κράτησε το μικρότερο από τα δυο στοιχεία
    ELSE
        κράτησε το μεγαλύτερο από τα δυο στοιχεία;
END;
```