Συστήματα Παράλληλης & Κατανεμημένης Επεξεργασίας

Ενότητα 8: Νήματα και Παραλληλισμός Διεργασιών

Δρ. Μηνάς Δασυγένης

mdasyg@ieee.org

Εργαστήριο Ρομποτικής, Ενσωματωμένων και Ολοκληρωμένων Συστημάτων

http://arch.ece.uowm.gr/mdasyg

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 Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» 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.

Σκοπός της Ενότητας

- Η κατανόηση της σημασίας των νημάτων για την αύξηση της παραλληλίας.
- Η παρουσίαση της τεχνικής της υπερνημάτωσης.

Η βασική μονάδα χρήσης είναι η διεργασία

- Στα παραδοσιακά λειτουργικά συστήματα η «μονάδα χρήσης» της CPU είναι η **διεργασία** (process).
- Κάθε διεργασία είναι ένα εκτελούμενο πρόγραμμα που έχει το δικό του «χώρο»:
  - **ιδιωτική περιοχή διευθύνσεων,**
  - **δικό της program counter,**
  - **δικό της stack,**
  - **δικούς της registers.**
- Ο χώρος κάθε διεργασίας είναι **προστατευμένος** από τις υπόλοιπες διεργασίες του συστήματος.
- Κάθε διεργασία αποτελείται από **ένα μοναδικό νήμα ελέγχου** (thread of control) μια και έχει ένα program counter.

Υπάρχει ένα πρόβλημα αν έχουμε μόνο διεργασίες

- Έστω μία διεργασία **file server** η οποία:
  - δέχεται κλήσεις και τις επεξεργάζεται
  - προσπελαύνει το δίσκο,
  - επεξεργάζεται τα δεδομένα (πχ τα συμπιέζει) και στέλνει απάντηση.
- Η προσπέλαση του δίσκου **απαιτεί προσωρινή αναστολή** του server και συνεπώς μείωση του throughput.
- Η αποδοτική υλοποίηση του server με **δύο ανεξάρτητες διεργασίες δεν είναι δυνατή**, διότι αυτές πρέπει να μοιράζονται μία ενδιάμεση κοινή μνήμη (buffer cache) η οποία προϋποθέτει κοινό χώρο διευθύνσεων.

Δύο είναι τα μοντέλα διεργασιών: (α) Μονονηματικό (β) Πολυνηματικό (1/2)

a. Μονονηματικό:
   - Ένα μόνο νήμα σε κάθε διεργασία.
- Διεργασία και νήμα **έχουν κοινά**:
  - χώρο διευθύνσεων,
  - μετρητή προγράμματος,
  - καταχωρητές,
  - στοίβα χρήστη και στοίβα πυρήνα.
- Η έννοια του νήματος ουσιαστικά δεν υφίσταται.

Δύο είναι τα μοντέλα διεργασιών: (α) Μονονηματικό (β) Πολυνηματικό (2/2)

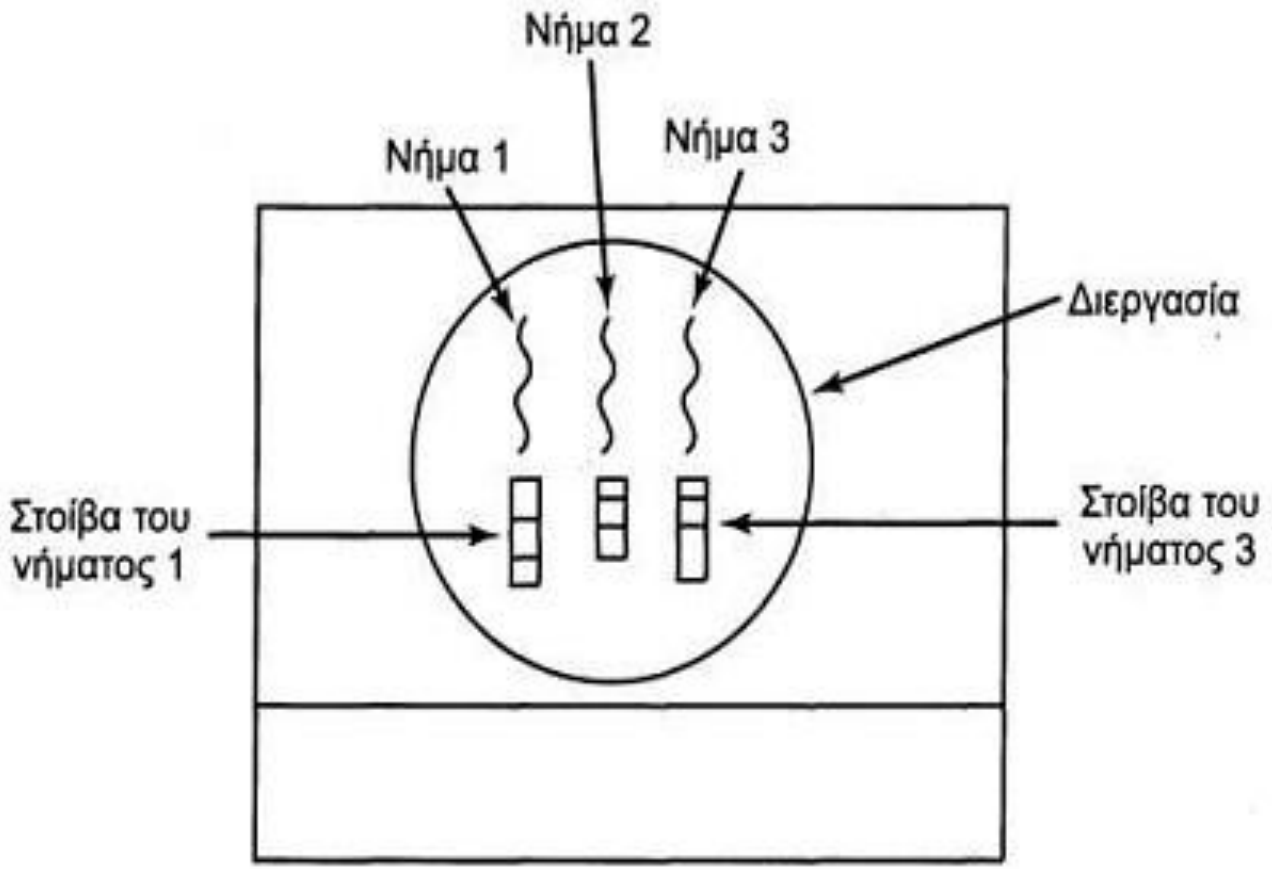
b. Πολυνηματικό:

- Ύπαρξη **πολλών νημάτων** σε κάθε διεργασία.
- Διεργασία και νήμα **έχουν κοινά**:
  - χώρο διευθύνσεων,
  - πόροι συστήματος.
- Κάθε νήμα έχει **ατομικό**:
  - μετρητή προγράμματος,
  - Καταχωρητές,
  - στοίβα χρήστη και στοίβα πυρήνα.

Το πρόβλημα του διακομιστή αρχείων λύνεται με πολλαπλά νήματα

- Λύση στο προηγούμενο πρόβλημα δίνουν τα *νήματα* (threads). Για το λειτουργικό σύστημα το νήμα είναι το αφαιρετικό μοντέλο μίας δραστηριότητας.
- Μία διεργασία μπορεί να έχει πολλά νήματα ελέγχου (ή νήματα, ή lightweight processes (lwp)).
- Κάθε νήμα μίας διεργασίας έχει τα δικά του:
  - program counter,
  - Stack,
  - Registers.
- Όλα τα νήματα μίας διεργασίας μοιράζονται:
  - τον χώρο διευθύνσεων,
  - τους πόρους συστήματος (πχ ανοιχτά αρχεία, semaphores, signals, child processes) της διεργασίας στην οποία περιέχονται.

Απεικόνιση διεργασίας & νήματος

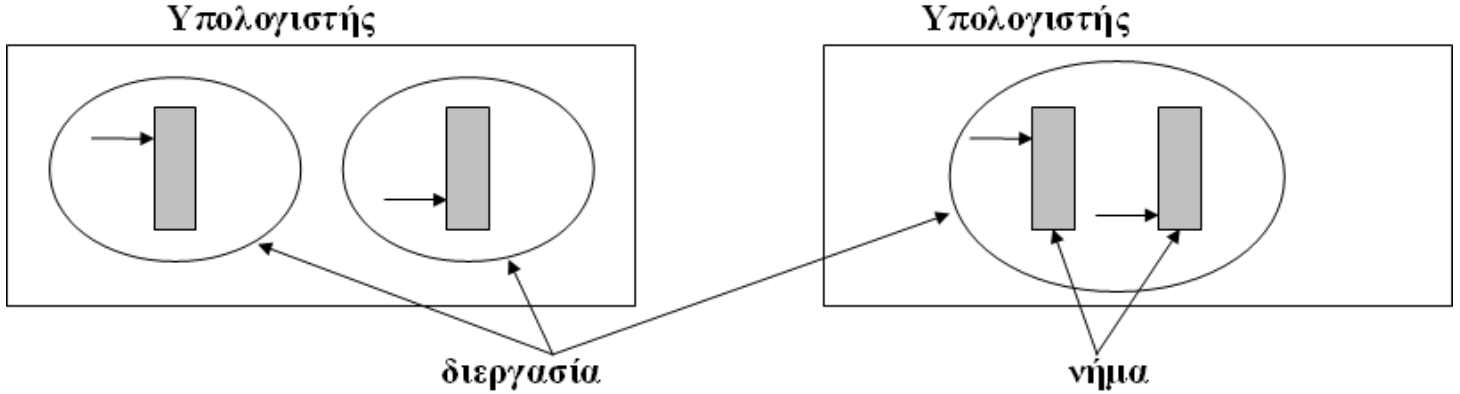


Τα νήματα χρησιμοποιούνται ευρέως!

- **Λειτουργικά συστήματα:**
  - Ένα kernel thread (νήμα πυρήνα) για κάθε διεργασία χρήστη.
- **Επιστημονικές εφαρμογές:**
  - Ένα thread ανά CPU (καλύτερος παραλληλισμός).
- **Κατανεμημένα συστήματα:**
  - Συγχρονισμός αιτήσεων διεργασιών.
- **GUI:**
  - Ανταπόκριση στον χρήστη.

Τα νήματα μιας διεργασίας συνεργάζονται

- Γενικά ισχύει η αναλογία ότι ένα νήμα είναι για μία διεργασία, ότι μία διεργασία για όλο τον υπολογιστή.
- Όμως, τα νήματα μίας διεργασίας **δεν είναι ανεξάρτητα** διότι μοιράζονται τον ίδιο χώρο διευθύνσεων. Πχ:
  - έχουν κοινές καθολικές μεταβλητές,
  - ένα νήμα μπορεί να αλλάξει το stack pointer ενός άλλου.
- Η ανεξαρτησία δεν είναι ούτε εφικτή, ούτε απαραίτητη, μια και τα **νήματα είναι πάντα συνεργαζόμενα** και όχι ανταγωνιστικά.



Τι είναι το νήμα; (1/3)

- Τα νήματα διεργασιών έχουν:
  - Ένα νήμα κατάστασης της εκτέλεσης.
  - Μια στοίβα αποθήκευσης.
  - Στατική αποθήκευση ανά νήμα τοπικών μεταβλητών.
  - Προσπέλαση στη μνήμη και τους πόρους της διεργασίας, μαζί με όλα τα άλλα νήματα της διεργασίας.

Τι είναι το νήμα; (2/3)

- **Multithreading:**
  - Εκτέλεση πολλών threads στον ίδιο χώρο.
  - Σε μια χρονική στιγμή το πρόγραμμα έχει τόσες ροές εκτέλεσης, όσα και τα threads.
  - (Ψευδο-ταυτοχρονισμός).
- **Με τη χρήση multithreading:**
  - Καλύτερη απόδοση.
  - Καλύτερη δομή.

Τι είναι το νήμα; (3/3)

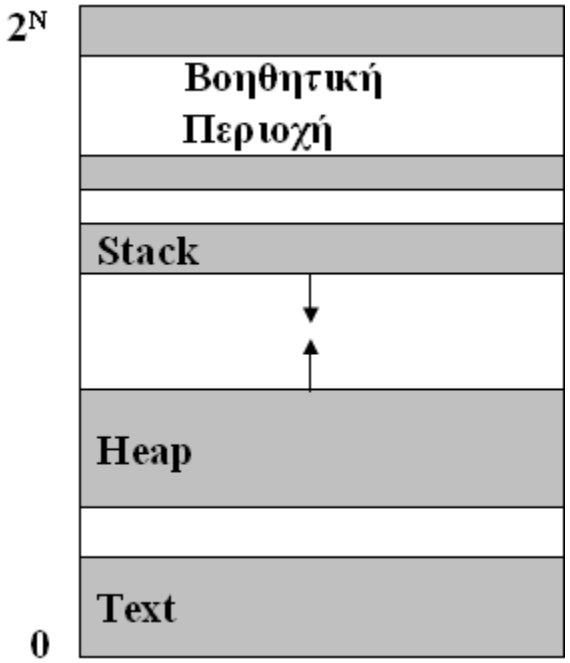
- Παράδειγμα εφαρμογής multithreading.
- Web Browser:
  - Scrolling.
  - Downloading.
  - Printing.
- =>Όλα μπορούν να γίνουν ταυτόχρονα (ή έτσι φαίνεται στο χρήστη).

Τα νήματα χρησιμοποιούνται όταν απαιτείται παράλληλη εκτέλεση

- Τα νήματα χρησιμοποιούνται ευρέως σε εφαρμογές όπου η εκτέλεση τους **πραγματοποιείται παράλληλα** ή ψευδοπαράλληλα.
- Όταν υπάρχει αίτημα για I/O, ένα νήμα μπορεί να περιμένει την ολοκλήρωση του αιτήματος ενώ άλλα εκτελούνται ταυτόχρονα.
- **Αποφεύγεται η αναμονή** άλλων αιτημάτων.

Πλεονεκτήματα Νημάτων (1/7)

- Μικρό overhead δημιουργίας:
  - το address space μίας νέας διεργασίας δημιουργείται από την αρχή (ή μέρος του κληρονομείται από την parent process),
  - είναι πολύ δαπανηρό (size, initial virtual memory page faults, empty cache),
  - αντίθετα ένα thread χρησιμοποιεί το address space της διεργασίας του, και η cache περιέχει «φρέσκα» δεδομένα.



Πλεονεκτήματα Νημάτων (2/7)

- Γρήγορη μετάβαση ελέγχου (switching) μεταξύ νημάτων:
  - η μετάβαση ελέγχου και εκτέλεσης (CPU switching) από μία διεργασία σε μία άλλη είναι πολύ πιο χρονοβόρα από τη μετάβαση ελέγχου μεταξύ των νημάτων μίας διεργασίας, διότι τα τελευταία έχουν κοινό address space.
- Αποδοτικός διαμοιρασμός πόρων:
  - ο διαμοιρασμός πόρων μεταξύ νημάτων μίας διεργασίας είναι πολύ πιο αποδοτικός λόγω κοινού address space.

Πλεονεκτήματα Νημάτων (3/7)

- Ο κοινός χώρος διευθύνσεων έχει ως αποτέλεσμα:
  - Ίδιες καθολικές μεταβλητές.
  - Πρόσβαση στα ίδια δεδομένα.
  - Δεν απαιτείται προστασία μεταξύ των νημάτων.
  - Ανήκουν σε ένα χρήστη.
  - Σχεδιάζονται ώστε να συνεργάζονται και να είναι φιλικά μεταξύ τους.

Πλεονεκτήματα Νημάτων (4/7)

- Ο **χρόνος δημιουργίας** και τερματισμού ενός νήματος μιας υπάρχουσας διεργασίας είναι **μικρότερος** από τον αντίστοιχο μιας καινούργιας διεργασίας.
- Η **επικοινωνία** των εκτελέσιμων προγραμμάτων επιτυγχάνεται πιο αποδοτικά.
- Το μοντέλο προγραμματισμού είναι **πολύ πιο απλό** διότι όλα τα νήματα έχουν πρόσβαση στον ίδιο χώρο μνήμης.

Πλεονεκτήματα Νημάτων (5/7)

- Προσφέρουν συνδυασμό «παραλληλίας» (performance), σειριακής εκτέλεσης, και ανασταλτικών κλήσεων συστήματος (easy programming).
- Το πλεονέκτημα αυτό μπορεί να εξηγηθεί με ένα παράδειγμα διαφορετικών υλοποιήσεων ενός file server process:
- ***single-thread server process:***
  - χρησιμοποιείται 1 νήμα ελέγχου (σειριακή εκτέλεση) και ανασταλτικές κλήσεις για πρόσβαση στον δίσκο:
    - απλός προγραμματισμός,
    - χαμηλή απόδοση λόγω συχνής αναστολής.
- ***finite-state machine server process:***
  - χρησιμοποιείται 1 νήμα ελέγχου, και υποστηρίζεται «παραλληλία» μέσω μη-ανασταλτικών κλήσεων.... (επόμενη διαφάνεια).

Πλεονεκτήματα Νημάτων (6/7)

- Ο server λειτουργεί σαν finite-state machine, το οποίο χρησιμοποιεί ένα **event-queue** για τις κλήσεις των clients και τις απαντήσεις από τον δίσκο,
- αν το επόμενο event είναι νέα αίτηση, την επεξεργάζεται (check permissions, cached data) και αν χρειαστεί κάνει κλήση για πρόσβαση στο δίσκο,
- μετά από κάθε τέτοια κλήση, **ο server δεν αναστέλλεται**, αλλά σώζει την κατάσταση της κλήσης του client, και παίρνει το επόμενο μήνυμα από το event queue.
- Αν αυτό είναι από το δίσκο, ανακτά την κατάσταση του client και συνεχίζει την επεξεργασία της αίτησης του:
  - **καλύτερη απόδοση**, λόγω «παραλληλίας»,
  - **δυσκολότερος προγραμματισμός** με διαχείριση του event queue και client-state.

Πλεονεκτήματα Νημάτων (7/7)

- *group-of-threads server process:*
  - υπάρχει ένα νήμα που λαμβάνει κλήσεις (ο **dispatcher (διεκπεραιωτής)**),
  - υπάρχουν νήματα **«εργάτες»** (δημιουργημένα από την αρχή ή δυναμικά),
  - ο dispatcher ελέγχει permissions και **αναθέτει** την κλήση σε ένα νήμα-εργάτη,
  - κάθε νήμα εργάτης χρησιμοποιεί ανασταλτικές κλήσεις για να διαβάσει το δίσκο:
    - **ευκολία προγραμματισμού,**
    - **υψηλή απόδοση,**
    - **«παράλληλη» επεξεργασία κλήσεων.**

Χρονοδρομολόγηση νημάτων (1/2)

- Ένα thread κληρονομεί **την προτεραιότητα** του δημιουργού του.
- Ένα thread εκτελείται έως ότου:
  - Ένα thread μεγαλύτερης προτεραιότητας βρεθεί σε κατάσταση runnable.
  - Όταν κρεμάει ή τερματίζεται.
  - Όταν τελειώσει ο χρόνος εκτέλεσης που του έχει δοθεί από το σύστημα.

Χρονοδρομολόγηση νημάτων (2/2)

- Αν δυο threads έχουν την **ίδια προτεραιότητα**, το σύστημα επιλέγει ποιο θα εκτελεστεί.
- Οι προσπάθειες διακοπής προς χαμηλότερης προτεραιότητας threads αγνοούνται!

Συγχρονισμός των νημάτων

- Υπάρχουν **ασύγχρονα** και **μη threads.**
  - Τα ασύγχρονα εκτελούνται χωρίς να ενδιαφέρονται για την κατάσταση των άλλων threads.
- Μερικές φορές χρειάζεται η επίγνωση της κατάστασης άλλων threads.

Συγχρονισμός των νημάτων - Παράδειγμα

- Σε μια εφαρμογή ένα thread γράφει δεδομένα σε ένα αρχείο και ένα δεύτερο διαβάζει δεδομένα από το αρχείο.
- **Είναι εμφανής η ανάγκη επικοινωνίας** μεταξύ των δυο threads για να επιτευχθεί συγχρονιστικός και **να αποφευχθούν ανεπιθύμητες** καταστάσεις.

Συλλογές ή ομάδες νημάτων (1/2)

- Είναι μια **συλλογή διαχείρισης threads** διαθέσιμων για την εκτέλεση διαφόρων αποστολών.
- Παρέχουν:
  - Βελτιωμένη απόδοση σε περίπτωση μεγάλου αριθμού αποστολών.
  - Ένα μέσο **δέσμευσης πηγών** που εμπεριέχει threads που ‘καταναλώνονται’ κατά την εκτέλεση μιας συλλογής αποστολών.

Συλλογές ή ομάδες νημάτων (2/2)

- Με την χρήση των thread pools δεν είναι απαραίτητος ο έλεγχος του κύκλου ζωής των threads.
- Ο προγραμματιστής μπορεί να **επικεντρωθεί στην αποστολή** των threads και όχι στους μηχανισμούς τους.

Συλλογές νημάτων

- Thread pools ή δυναμική δημιουργία;
  - **Η δημιουργία threads είναι ακριβή (αλλά πιο φθηνή από τη δημιουργία διεργασίας).**
  - Απόσβεση του κόστους δημιουργίας **έρχεται μετά την διατήρηση** του thread για αρκετές αιτήσεις.
  - Μπορεί να **δημιουργηθεί ένας αριθμός threads** στον χρόνο εκκίνησης που προσδιορίζονται καθώς εισέρχονται οι αιτήσεις.

Σχεδίαση νημάτων στα κατανεμημένα ΛΣ - 1ος τρόπος

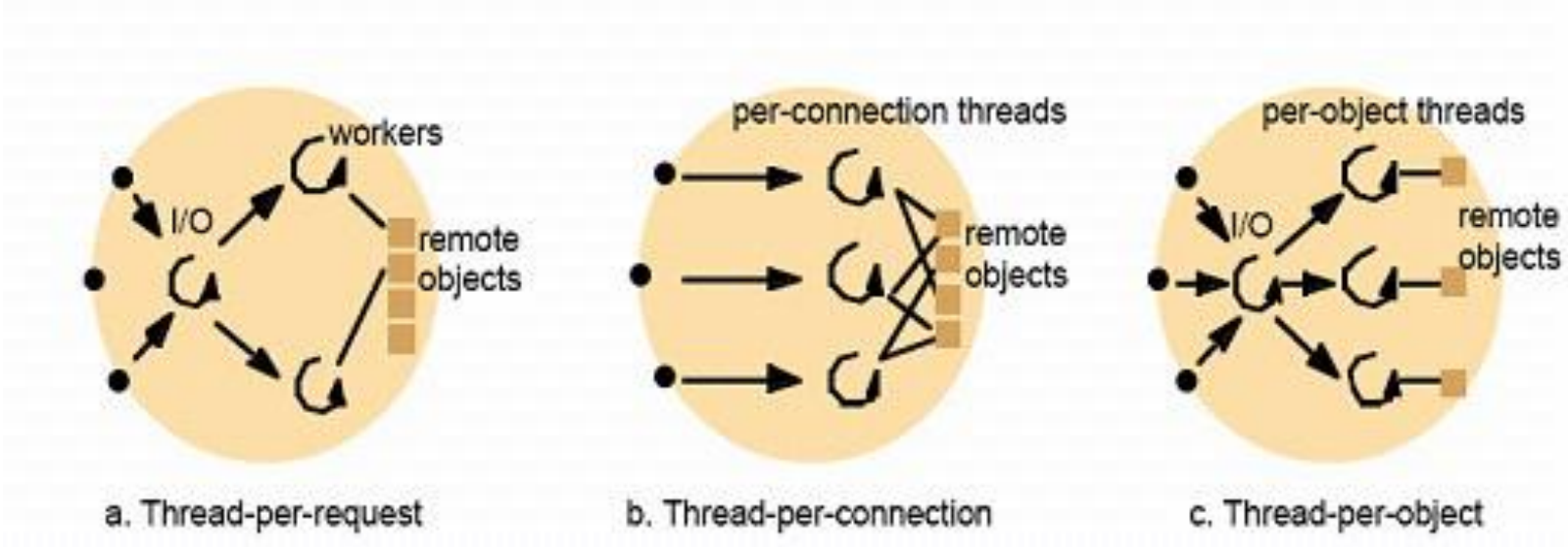
- Thread per connection.
  - Το thread συντονιστής ανιχνεύει καινούριο client και τον συνδέει με **ένα καινούργιο thread.**
  - Το καινούριο thread αποκωδικοποιεί την αίτηση, καλεί τον server και περιμένει νέα αίτηση από τον client.

Σχεδίαση νημάτων στα ΚΛΣ - 2ος τρόπος

- Thread per servant.
  - Κάθε υπηρέτης έχει δικό του thread και δικιά του ουρά.
  - Ο συντονιστής διαβάζει μια εισερχόμενη αίτηση και την προωθεί στην ουρά του κατάλληλου υπηρέτη.
  - Ο υπηρέτης παίρνει αιτήσεις από την ουρά του και τις εκτελεί.

Σχεδίαση νημάτων στα ΚΛΣ - Σύνοψη

- Εναλλακτικές αρχιτεκτονικές νηματισμού server.



Σχεδίαση νημάτων (1/2)

- Πολυεπεξεργαστές:
  - Το περιβάλλον εργασίας βρίσκεται στην κοινή μνήμη.
  - Τα νήματα μπορούν να τρέχουν παράλληλα σε διαφορετικά CPUs.
  - Χρειάζεται μόνο ο συγχρονισμός των προσβάσεων στην κοινή μνήμη.
- Περιβάλλοντα δικτύου:
  - Μπορεί μια διεργασία στον υπολογιστή X να έχει ένα νήμα στον υπολογιστή Y;
    - Όχι, χωρίς το περιβάλλον εργασιών.
  - Γιατί να νήματα αποτελούν πρόβλημα στα κατανεμημένα συστήματα;
    - Διότι το μοντέλο server απαιτεί πολλαπλά αντίγραφα να τρέχουν τον ίδιο κώδικα.

Σχεδίαση νημάτων (2/2)

- Νήματα για το μοντέλο πελάτη/εξυπηρετητή.
- Κάθε πελάτης κάνει ένα αίτημα στον εξυπηρετητή.
- Ο εξυπηρετητής εκτελεί μια παρόμοια διαδικασία για όλες τις κλήσεις.
  - η θεματική εναλλαγή, μπορεί να απαιτεί πάρα πολλούς κύκλους.
- Τα νήματα μειώνουν την επιβάρυνση.
- Ο προγραμματισμός του εξυπηρετητή, είναι ευκολότερος.

Μοντέλα αλληλεπίδρασης σε πολυ-υπολογιστές

- Τα περιβάλλοντα κατανεμημένου προγραμματισμού **περιορίζουν** όσον αφορά την αλληλεπίδραση των συνιστώμενων μερών.
- Τα RPC δημιουργούν ένα πιο γνώριμο περιβάλλον αλληλεπίδρασης.

Μοντέλα αλληλεπίδρασης

- Ένα κεντρικό πρόγραμμα συνήθως κατέχει ένα απλό νήμα ελέγχου, όταν περιορίζεται σε έναν χώρο μνήμης.
- Σε περίπτωση ταυτοχρονισμός τα νήματα επικοινωνούν με κοινόχρηστα δεδομένα.
- Για συγχρονισμό και αμοιβαίο αποκλεισμό χρησιμοποιούνται **σημαφόροι** και **monitors.**

Μοντέλα αλληλεπίδρασης σε Κ.Λ.Σ (1/2)

- Για Κατανεμημένα Συστήματα:
  - **Είναι πάντα πολυνηματικά.**
  - Τα RPC επιβάλλουν υπερβολική αστυνόμευση.
  - Τα middleware (βιβλιοθήκες ΚΛΣ) για ανάπτυξη παράλληλων προγραμμάτων προτιμούν την **ασύγχρονη μετάδοση δεδομένων.**
  - Αν το middleware δεν υποστηρίζει πάνω από έναν τρόπο αλληλεπίδρασης, θα πρέπει να τους υλοποιήσει ο προγραμματιστής με κίνδυνο λαθών και έλλειψης σαφήνειας.

Μοντέλα αλληλεπίδρασης σε Κ.Λ.Σ (2/2)

- Μοντέλο αλληλεπίδρασης.
  - Τα στοιχεία επικοινωνούν μέσω των τερματισμών που αποκαλύπτουν τις **διεπαφές** τους.
  - Η επικοινωνία μεταξύ δυο τερματισμών γίνεται με **ανταλλαγή μηνυμάτων.**
  - Τα μηνύματα που ανταλλάσσονται ορίζουν ένα πρωτόκολλο επιπέδου εφαρμογής.

Νήματα POSIX (1/5)

- Μοντέλο διεργασίας.
  - Το **συχνότερο προγραμματιστικό μοντέλο** για λειτουργικά συστήματα - παράγωγα UNIX.
  - Η διεργασία αποτελείται από αρκετούς πόρους του συστήματος και είναι δρομολογούμενη οντότητα του συστήματος.
  - Μια διεργασία ανά χώρο διευθύνσεων.
  - Η διεπαφή ελέγχου διεργασιών προσφέρεται μέσω των κλήσεων συστήματος **fork, exit, wait.**

Νήματα POSIX (2/5)

- Πολυεπεξεργαστικό ή Κατανεμημένο σύστημα.
- Παράλληλη εκτέλεση διεργασιών και επικοινωνία μέσω IPC.
- Τα νήματα επιτρέπουν την εκτέλεση πολλών ενεργειών σε έναν χώρο διευθύνσεων.
- Συστήματα που υλοποιούν κάποιον πολυνηματικό μηχανισμό : POSIX, DCE, Mach c, Solaris.

Νήματα POSIX (3/5)

- Πρόβλημα η δυνατότητα εφαρμογής μόνο σε στενά συνδεδεμένες αρχιτεκτονικές.
  - Επεκτείνονται μέχρι **κάποιον αριθμό επεξεργαστών.**
- Για περισσότερους επεξεργαστές, NORMA.
- Επικοινωνία με κάποιο είδος εσωτερικού δικτύου.

Νήματα POSIX (4/5)

- Για Κατανεμημένα Συστήματα υπάρχουν 2 προγραμματιστικές τάσεις.
- **Ανταλλαγή μηνυμάτων:**
  - Ανάγκη επανασυγγραφής και αλλαγής της αλγοριθμικής δομής των προγραμμάτων.
- **Διαμοιραζόμενη μνήμη:**
  - Εφαρμόζεται η αφαιρετικότητα μιας διαμοιραζόμενης μνήμης.

Νήματα POSIX (5/5)

- Συνήθως τα νήματα σε συστήματα Unix & Linux είναι POSIX threads ή αλλιώς **pthreads.**
  - **Τα pthreads είναι kernel space threads.**
- Υπάρχουν και τα user level threads για τα οποία ο πυρήνας έχει πλήρη άγνοια.

User level threads

- Η διαχείριση της εκτέλεσης των νημάτων μιας διεργασίας γίνεται μέσω του πίνακα νημάτων:
  - είναι ένα πακέτο από ρουτίνες,
  - είναι μοναδικός για κάθε διεργασία,
  - περιέχει τα περιεχόμενα του μετρητή προγράμματος,
  - των καταχωρητών του νήματος και τους δείκτες από τις στοίβες.
- Είναι κοινός για όλα τα νήματα μιας διεργασίας.

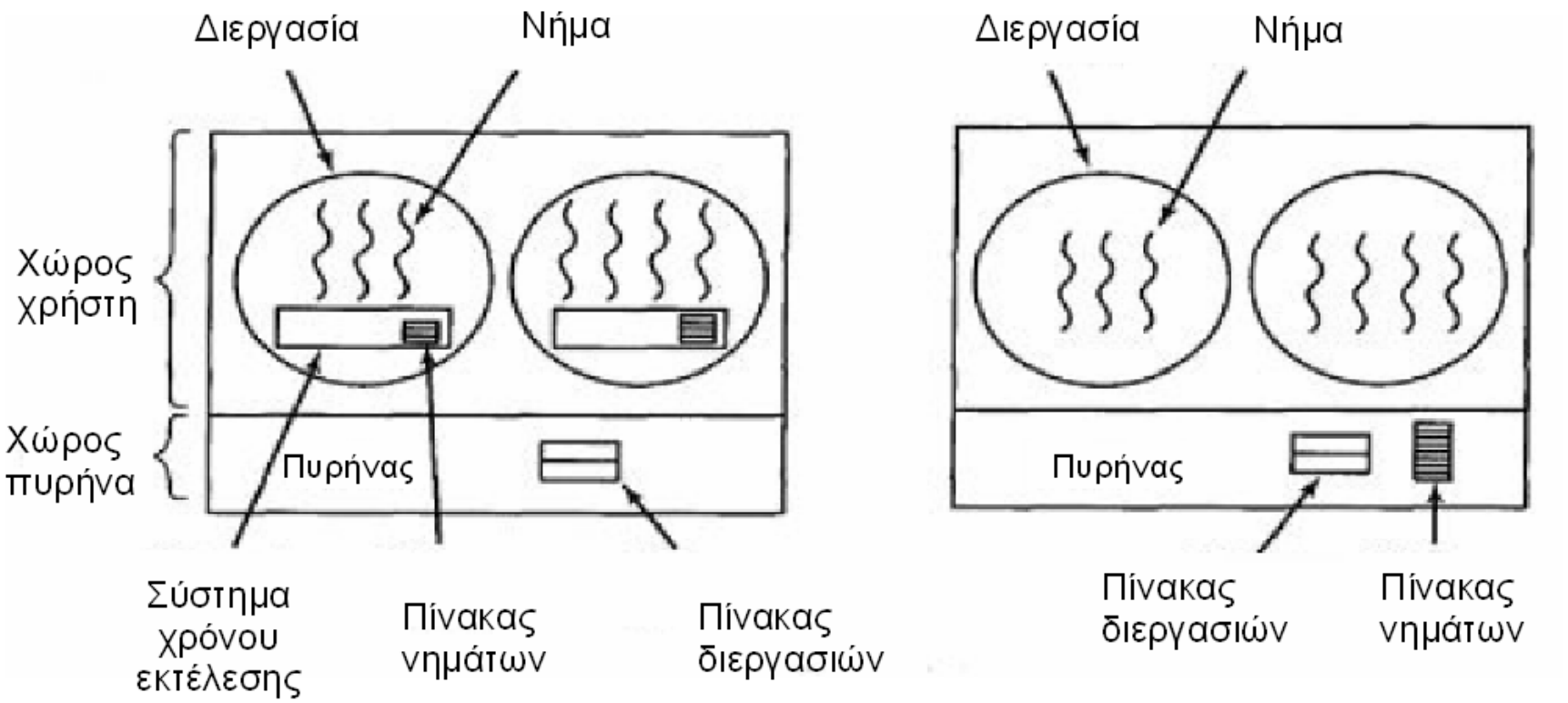
User level threads - Πλεονεκτήματα

- Τα νήματα επιτρέπουν σε κάθε διεργασία να χρησιμοποιεί **διαφορετικό αλγόριθμο χρονοδρομολόγησης.**
- Η διαχείριση του νήματος γίνεται μέσω του πίνακα νημάτων της διεργασίας **χωρίς την παρεμβολή του πυρήνα.**
  - Οι διαδικασίες του συστήματος χρόνου εκτέλεσης και ο χρονοπρογραμματιστής είναι τοπικές διαδικασίες.
  - Η επίκληση τους είναι **πιο αποδοτική** από την κλήση του πυρήνα.
- Εφαρμογές που απαιτούν **μεγάλο αριθμό νημάτων** εκτελούνται πιο αποδοτικά.
- Οι καταχωρητές και οι στοίβες **δεσμεύουν χώρο** στον πίνακα νημάτων της διεργασίας ο οποίος δεν βρίσκεται στον πυρήνα.
- Επιτρέπει την εκτέλεση των διεργασιών **σε λειτουργικά συστήματα που δεν υποστηρίζουν νήματα.**

User level threads - Μειονεκτήματα

- Όταν ένα νήμα εκτελεί μία κλήση συστήματος που **προκαλεί αναστολή αναστέλλονται και τα υπόλοιπα νήματα της διεργασίας.**
- Μόνο μία **διεργασία ανατίθεται για εκτέλεση** από τον πυρήνα ανά χρονική στιγμή σε έναν επεξεργαστή.
- Οι πολυνηματικές εφαρμογές **δεν επωφελούνται από τα πλεονεκτήματα του πολυπρογραμματισμού**.
- Όταν ένα νήμα εισέλθει στη **CPU κανένα άλλο νήμα δε μπορεί να αποκτήσει τον έλεγχο της** εκτός και αν αυτό παραδώσει τον έλεγχο οικειοθελώς.

(a) user level threads
(b) kernel level threads



Kernel level threads

- Ο **πυρήνας γνωρίζει την ύπαρξη των νημάτων** και εκτελεί τη δρομολόγησή τους με βάση το νήμα.
- Όλες οι διεργασίες **έχουν κοινό σύστημα χρόνου εκτέλεσης** και πίνακα νημάτων.
- Ο πυρήνας έχει έναν πίνακα διεργασιών για την παρακολούθησή τους.
- Ο **πίνακας νημάτων του πυρήνα διαχειρίζεται** τα νήματα όλων των διεργασιών.
- Η δημιουργία και η καταστροφή των νημάτων **πραγματοποιείται με μία κλήση στον πυρήνα**, ο οποίος ενημερώνει τον πίνακα νημάτων.

Kernel level threads - πλεονεκτήματα

- Νήματα της ίδιας διεργασίας **μπορούν να εκτελεστούν ταυτόχρονα** σε διαφορετικούς επεξεργαστές
- **Επωφελούμαστε** από τα πλεονεκτήματα του πολυπρογραμματισμού.
- **Όταν ένα νήμα μπλοκάρεται** ο πυρήνας μπορεί να δρομολογήσει άλλο νήμα της ίδιας διεργασίας.

Kernel level threads - μειονεκτήματα

- Οι εναλλαγές των νημάτων για εκτέλεση μέσω του πυρήνα επιφέρουν **μεγάλο κόστος.**
- Η δημιουργία και καταστροφή των νημάτων επιφέρει **μεγάλο κόστος.**
  - Μια λύση είναι η **ανακύκλωση των νημάτων.**
    - Δεν καταστρέφονται οι δομές δεδομένων των νημάτων όταν αλλά σημαδεύονται ως μη εκτελέσιμα.
    - Κατά τη δημιουργία ενός νέου νήματος δεσμεύονται οι ήδη υπάρχουσες δομές και επανενεργοποιείται ένα παλιό νήμα.
    - Μειώνεται η επιβάρυνση του συστήματος.

User level or kernel level threads

- Υπάρχει **σημαντική επιτάχυνση** χρησιμοποιώντας νήματα στο χώρο του χρήστη απ’ ότι στο χώρο του πυρήνα.
- Εάν η εκτέλεση μιας εφαρμογής απαιτεί **πολλές προσπελάσεις στον πυρήνα, τότε είναι πιο αποδοτική η υλοποίησή της στο χώρο του πυρήνα.**

Νήματα χρήστη vs. Νήματα πυρήνα

- **Νήματα χρήστη:**
  - Ο πυρήνας χρονοδρομολογεί τις διεργασίες.
  - Οι διεργασίες χρονοδρομολογούν τα ανεξάρτητα νήματα.
  - Επιτρέπει την προσαρμοσμένη χρονοδρομολόγηση των νημάτων:
    - Πρέπει να γίνει στη διεργασία που χρονοδρομολογήθηκε από τον πυρήνα.
  - Πρόβλημα:
  - Πώς χρονοδρομολογούνται τα υπόλοιπα νήματα;
    - Μόνο αν το τρέχων νήμα απελευθερώσει την CPU.
  - Μερικά συστήματα δεν έχουν νήματα πυρήνα:
    - Υπάρχουν πακέτα νημάτων για το UNIX

- **Νήματα πυρήνα:**
  - Ο πυρήνας χρονοδρομολογεί και διευθύνει τα νήματα.
  - Όταν κάποιο νήμα αδρανοποιείται επιλέγει κάποιο άλλο.
    - Ο πυρήνας μπορεί επίσης και να προεκχωρεί τα νήματα.
  - Είτε από την ίδια είτε από διαφορετική διεργασία.
  - Προσοχή στις θεματικές εναλλαγές.
    - Οι εναλλαγές μεταξύ διεργασιών είναι επιβαρυντικές.
    - Χρειάζεται η ταυτόχρονη χρονοδρομολόγηση των νημάτων από την ίδια διεργασία.

Νήματα χρήστη - Νήματα πυρήνα

- Νήματα χρήστη: καλές επιδόσεις
  - Δεν υπάρχουν χρονοβόρες εναλλαγές μεταξύ χώρου χρήστη και πυρήνα.
- Νήματα πυρήνα: ευκολία στον προγραμματισμό.
  - Δεν χρειάζονται no-blocking reads κτλ.

Υβριδικές υλοποιήσεις

- Έχουμε συνδυασμό πλεονεκτημάτων υλοποίησης νημάτων στο χώρο χρήστη και πυρήνα.
- Χρησιμοποιούνται νήματα επιπέδου πυρήνα αλλά και νήματα επιπέδου χρήστη.
  - τα τελευταία πολυπλέκονται σε ένα ή σε όλα τα νήματα πυρήνα.
  - ο πυρήνας μπορεί να διαχειρίζεται μόνο τα νήματα επιπέδου πυρήνα
  - τα πολυπλεγμένα νήματα σχεδιάζονται, καταστρέφονται και χρονοπρογραμματίζονται όπως τα κοινά νήματα του επιπέδου του χρήστη σε σύστημα που δεν έχει δυνατότητες πολυνημάτωσης.
- Κάθε νήμα επιπέδου πυρήνα εξυπηρετεί εκ περιτροπής τα νήματα επιπέδου χρήστη.

Σχεδιαστικά Ζητήματα για Πακέτα Νημάτων

- Η υποστήριξη νημάτων σε ένα σύστημα γίνεται μέσω ενός συνόλου θεμελιωδών λειτουργιών (πχ κλήσεις σε βιβλιοθήκες), το οποίο αποκαλείται πακέτο νημάτων (threads package).
- Τα βασικά ζητήματα σχεδίασης ενός τέτοιου πακέτου είναι:
  - δημιουργία νημάτων,
  - τερματισμός νημάτων,
  - συγχρονισμός νημάτων,
  - χρονοπρογραμματισμός νημάτων,
  - διαχείριση σημάτων.

Δημιουργία Νημάτων

- **Τα νήματα μπορούν να δημιουργούνται:**
  - **στατικά**
  - **ή δυναμικά**
- **Στην στατική προσέγγιση:**
  - ο αριθμός νημάτων μίας διεργασίας είναι **σταθερός** για όλη τη ζωή της και καθορίζεται όταν γράφεται το πρόγραμμα ή κατά το compilation,
  - κάθε νήμα έχει stack σταθερού μεγέθους.
- **Στην δυναμική προσέγγιση:**
  - νέα νήματα **δημιουργούνται κατά την εκτέλεση της** διεργασίας,
  - το μέγεθος του stack, το scheduling priority και άλλες παράμετροι καθορίζονται στην κλήση δημιουργίας του νέου νήματος,
  - η κλήση επιστρέφει ένα thread identifier.

Δεν υπάρχει προστασία ανάμεσα στα νήματα (χρειάζεται όμως;)

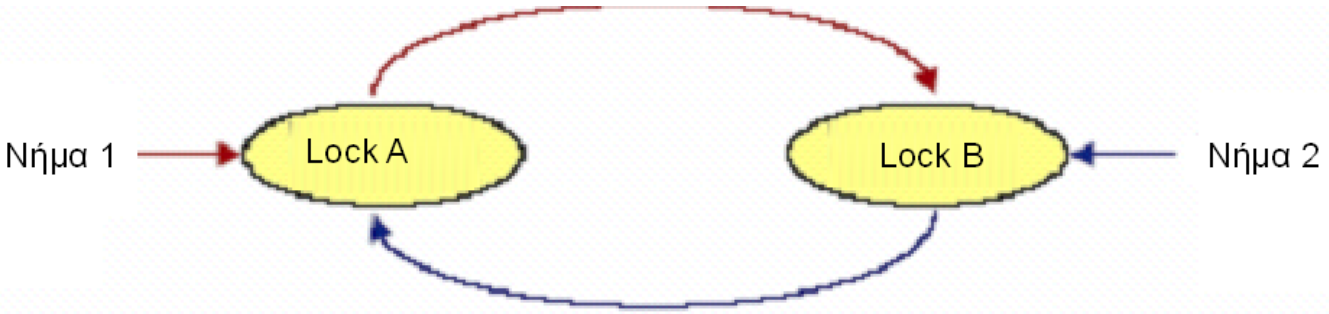
- Δεν υπάρχει προστασία ανάμεσα στα νήματα.
- Υποθέτοντας ότι τα νήματα έχουν σχεδιαστεί για να συνεργάζονται:
  - Έχουν πρόσβαση στα κοινά δεδομένα (global data).
  - Race conditions.
- Εξαρτάται από την στρατηγική χρονοπρογραμματισμού των νημάτων:
  - Μη προληπτικό: Το νήμα πρέπει οπωσδήποτε να παραχωρήσει την CPU.
  - Προληπτικό: Μπορεί να προκύψουν race conditions.

Τερματισμός Νημάτων

- Ένα νήμα μπορεί να τερματίσει τον εαυτό του με ένα **exit call** ή να τερματιστεί από εξωτερικούς παράγοντες (πχ από τη διεργασία του με κατάλληλη κλήση και το thread id).
- **Συχνά τα νήματα δεν τερματίζονται ποτέ**, όπως για παράδειγμα σε περιπτώσεις στατικής δημιουργίας τους.

Συγχρονισμός νημάτων (1/3)

- Η χρήση νημάτων είναι δύσκολη γιατί:
  - Παρουσιάζονται προβλήματα στον συγχρονισμό και στα locks.
  - Deadlocks.



Συγχρονισμός νημάτων (2/3)

- Τα νήματα μίας διεργασίας μοιράζονται κοινό address space.
- Συνεπώς **απαιτούνται μηχανισμοί αμοιβαίου αποκλεισμού** σε περιπτώσεις ταυτόχρονης προσπάθειας χρήσης κοινόχρηστων πόρων.
- Χρησιμοποιούνται 2 μηχανισμοί:
  - **mutex variables,**
  - **condition variables (μεταβλητές συνθήκης).**

Συγχρονισμός νημάτων (3/3)

- Mutex:
  - Ουσιαστικά είναι ένα δυαδικό semaphore.
  - Στο πακέτο νημάτων της C:
    - mutex_lock(mutexId).
    - mutex_unlock(mutexId).
  - non-blocking mutex: “trylock”.
  - Μπορεί να προκύψουν deadlocks με το κλείδωμα των σηματοφόρων.
- Μεταβλητές συνθήκης:
  - Χρήση για μεγάλες αναμονές.
  - Χρησιμοποιώντας σήματα στις μεταβλητές συνθήκης μειώνουμε την πιθανότητα για deadlocks.
  - condition_wait (conditionId, mutexId):
    - Αίτημα queue στη conditionId.
    - Μετά το mutex ξεκλειδώνεται.
  - condition_signal(conditionId)
    - Άν ένα νήμα είναι σε αναμονή για την συνθήκη conditionId, ξεκλειδώνεται.

Το πρόβλημα των global data

- Πρόβλημα μεταβλητών global:
  - Οι μεταβλητές είναι κοινές σε όλα τα νήματα.
  - Μεταβλητή errno στο Unix.
  - Η C χρησιμοποιεί απλό buffer για κάθε stream:
    - Τα νήματα μπορούν να γράφουν στο buffer ή να αλλάζουν το pointer χωρίς συνοχή.

- Λύση: Κάθε νήμα έχει τις δικές του private μεταβλητές:
  - create_global(“data1”)
    - Δημιουργεί τοπικό χώρο για την μεταβλητή.
    - Τα υπόλοιπα νήματα καλούν την συνάρτηση create_global για να έχουν ξεχωριστό χώρο διευθύνσεων.
  - set_global(“data1”)
    - Γράφει στην τοπική διεύθυνση μνήμης.
  - x = read_global(“data1”)
    - Ανάγνωση δεδομένων από τοπική διεύθυνση.

Συγχρονισμός Νημάτων με mutex (1/2)

- Ένα **mutex είναι σαν ένας δυαδικός σημαφόρος** και βρίσκεται πάντα σε κατάσταση locked, ή unlocked.
- Ένα νήμα που θέλει να μπει σε critical region προσπαθεί να κάνει lock στο κατάλληλο mutex variable.
- **Αν δεν είναι ήδη κλειδωμένο, μπαίνει στην κρίσιμη περιοχή.**
- **Αν είναι ήδη κλειδωμένο τότε:**
  - το νήμα μπλοκάρεται και μπαίνει σε ουρά περιμένοντας το ξεκλείδωμα του mutex,
  - ή επιστρέφεται ένα failure status code, οπότε το νήμα αποφασίζει το ίδιο αν θα συνεχίσει την προσπάθεια ή αν θα κάνει κάποια άλλη επεξεργασία.

Συγχρονισμός Νημάτων με mutex (2/2)

- Συχνά **υποστηρίζονται και οι δύο τρόποι.**
- Αν προσπαθήσουν ταυτόχρονα δύο διαφορετικά νήματα να κάνουν lock στο ίδιο mutex, επιτυγχάνει μόνον το ένα.
- Μόλις ένα νήμα βγει από την κρίσιμη περιοχή κάνει unlock στο αντίστοιχο mutex.
- Η υλοποίηση των μεταβλητών mutex **είναι εύκολη** διότι έχουν μόνον δύο καταστάσεις.
- Όμως **η χρησιμότητά τους περιορίζεται** στην φύλαξη της εισόδου σε κρίσιμες περιοχές.
- Για γενικότερο συγχρονισμό χρησιμοποιούνται οι **μεταβλητές συνθήκης.**

Συγχρονισμός Νημάτων με Μεταβλητές Συνθήκης (1/2)

- Κάθε μεταβλητή συνθήκης (ΜΣ) συνδέεται με μια μεταβλητή mutex και αντανακλά την κατάστασή της.
- Οι δύο βασικές λειτουργίες σε μια ΜΣ **είναι wait και signal.**
- Όταν ένα νήμα εκτελέσει wait σε μία ΜΣ, το αντίστοιχο mutex γίνεται unlock, και το νήμα αναστέλλεται μέχρι κάποιο άλλο νήμα να εκτελέσει signal στην ΜΣ αυτή.
- Όταν ένα νήμα κάνει signal σε μία ΜΣ, το αντίστοιχο mutex γίνεται lock, και το νήμα που περίμενε συνεχίζει την εκτέλεσή του.

Συγχρονισμός Νημάτων με Μεταβλητές Συνθήκης (2/2)

- Οι ΜΣ χρησιμοποιούνται ευρέως από **συνεργαζόμενα** νήματα τύπου producer-consumer.
- Στο μοντέλο αυτό ο producer τροφοδοτεί τον consumer με δεδομένα, μέσω ενός buffer ορισμένου μεγέθους.
- Ο **producer** ελέγχει αν το buffer είναι γεμάτο. Αν είναι, κάνει wait σε μία ΜΣ *nonfull,* αλλιώς βάζει δεδομένα και κάνει signal σε μία ΜΣ *nonempty.*
- Ο **consumer** ελέγχει αν το buffer είναι άδειο. Αν είναι, κάνει wait στην ΜΣ *nonempty,* αλλιώς παίρνει δεδομένα και κάνει signal στην ΜΣ *nonfull.*

Διαχείριση νημάτων

- Τα σήματα παρέχουν **διακοπές και αποκλεισμό.**
- Οι κοινές καθολικές μεταβλητές **προκαλούν πρόβλημα στην διαχείριση** τους επειδή τροποποιούνται με κάθε πρόσβαση που γίνεται σε αυτές από άλλα νήματα.
- Μια **λύση** είναι:
  - η δημιουργία ενός νήματος που θα χειρίζεται τις διακοπές αυτές μέσα στην κάθε διεργασία,
  - και αναθέτουμε σε κάθε νήμα ιδιωτικές καθολικές μεταβλητές για την αποθήκευση των καταστάσεων αμοιβαίου αποκλεισμού.

Χρονοπρογραμματισμός Νημάτων

- Συνήθως τα πακέτα νημάτων δίνουν την δυνατότητα καθορισμού αλγορίθμου χρονοπρογραμματισμού εκτέλεσής των νημάτων.
- Οι πιο συνηθισμένοι μηχανισμοί είναι:
  - **priority assignment:**
    - κάθε νήμα έχει τη δική του προτεραιότητα και το επόμενο νήμα που εκτελείται είναι αυτό με την υψηλότερη. Υποστηρίζεται preemptive και non-preemptive λειτουργία.
  - **dynamic time quantum size:**
    - απλό round robin, όπου το time-slot εκτέλεσης του νήματος είναι μεταβλητό. Χρησιμοποιείται όταν ο αριθμός CPU's είναι μεγαλύτερος από τον αριθμό νημάτων.
  - **handoff scheduling:**
    - κάθε νήμα καθορίζει το επόμενο νήμα στο οποίο θα περάσει ο έλεγχος.

Ο προγραμματισμός με νήματα είναι δύσκολος

- **Δύσκολο debugging** λόγω εξαρτήσεων χρόνου και δεδομένων.
- Δε μπορούν να σχεδιαστούν **ανεξάρτητα modules** (σπάει η αφαιρετικότητα).
- Δε μπορούν να δουλέψουν τα **callbacks** με locks.

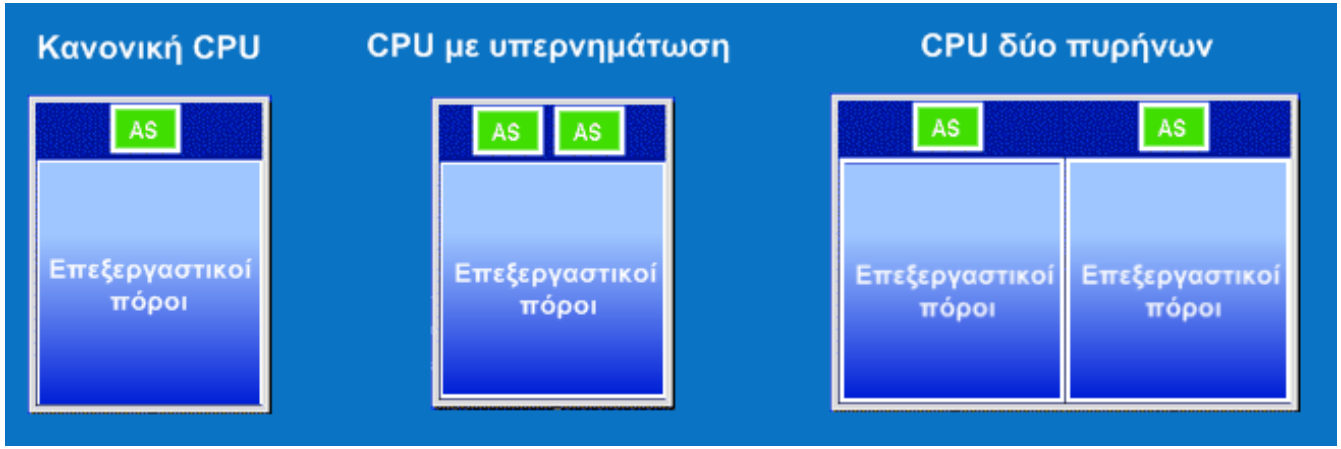
Hyperthreading (υπερνημάτωση)

Hyper threading στους επεξεργαστές (1/2)

- Τεχνολογία της Intel – επεξεργαστές Pentium4.
- Δημιουργείται **ένα virtual σύστημα** με δυο επεξεργαστές άνισης ταχύτητας.
- Είναι χρήσιμο με:
  - Multithreading εφαρμογές.
  - Διαφορετικά threads που χρησιμοποιούν διαφορετικό μέρος της CPU.

Υπερ-νημάτωση (Hyperthreading)

- Παρουσιάζει στο λογισμικό 2 λογικούς επεξεργαστές, ακόμα και όταν μόνο ένας φυσικός επεξεργαστής είναι παρών. Διπλασιάζει ουσιαστικά τον αριθμό των επεξεργαστών που αναγνωρίζει το λειτουργικό σύστημα.
- Το λειτουργικό σύστημα αναγνωρίζει 2 επεξεργαστές επειδή ένας επεξεργαστής HT έχει 2 σύνολα αρχιτεκτονικών πόρων.
- Όμως τεχνικά είναι μόνο ένας επεξεργαστής επειδή οι υπολογιστικοί πόροι δεν διπλασιάζονται.



Υπερ-νημάτωση (HT)

- **Διαμοιρασμός πόρων**:
- Οι λογικοί επεξεργαστές μοιράζονται τις μονάδες εκτέλεσης. Μία επιτάχυνση γίνεται αντιληπτή μόνο όταν ένα νήμα είναι σε θέση να εκμεταλλευτεί τις μονάδες εκτέλεσης που αφέθηκαν ελεύθερες από το άλλο νήμα. Για παράδειγμα όταν ένα νήμα υπολογίζει ακέραιους, και το άλλο πραγματικούς αριθμούς, τότε είναι σίγουρο ότι θα επιτύχουν την μέγιστη χρήση των υπολογιστικών πόρων της CPU.
- **Διαχωρισμός πόρων**:
- Η CPU έχει πόρους που είναι στατικά κατατετμημένες μεταξύ των νημάτων, όπως οι ουρές μ-op, ουρές load-store, re-order buffers, κτλ. Αυτοί οι πόροι μπορούν να κατανεμηθούν σε ένα λογικό επεξεργαστή όταν η υπερνημάτωση είναι απενεργοποιημένη, αλλά μοιράζονται 50/50 όταν είναι ενεργοποιημένη.

HT στον Intel Atom

- Ο Atom παρουσιάζει μια καινούργια αρχιτεκτονική m (mArchitecture (mobile)) χαμηλών ενεργειακών απαιτήσεων.

- Μία από τις αξιοσημείωτες αλλαγές που έχουν ως αποτέλεσμα δραστική μείωση της ενεργειακής κατανάλωσης είναι η αφαίρεση της εκτέλεσης εντολών εκτός σειράς. Μία CPU που εκτελεί εντολές εκτός σειράς μπορεί να κρατάει τις μονάδες εκτέλεσής του απασχολημένες. Σαν αποτέλεσμα ο Atom έχει **πολύ μεγαλύτερη αύξηση επιδόσεων** με την υπερνημάτωση από τους P4, ή τους Core i7.

Hyper threading στους επεξεργαστές (2/2)

- **Οφέλη:**
  - Μικρότερο overhead λειτουργικού συστήματος.
  - Η αλληλεπίδραση ανάμεσα σε εφαρμογές και client μπορεί να αλλάξει την απόδοση και των δύο.

Τέλος Ενότητας