Λειτουργικά Συστήματα (ΗΥ321)

Διάλεξη 2:
Διεργασίες και Νήματα

Χρήστος Δ. Αντωνόπουλος
6/3/2017

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Διαδικαστικά

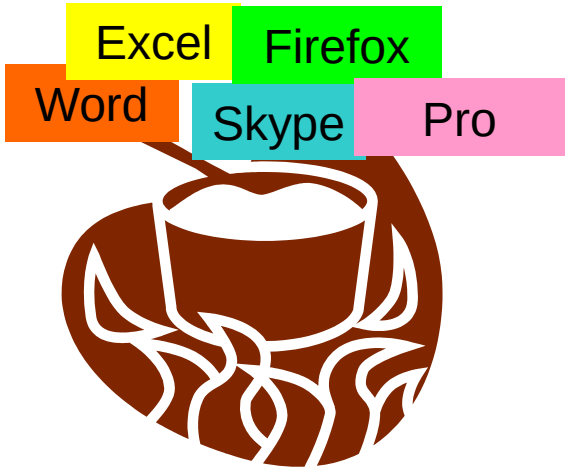
- Γραφτήκατε στη λίστα;
- Σχηματίστε ομάδες (για το project)!

Στο προηγούμενο επεισόδιο…

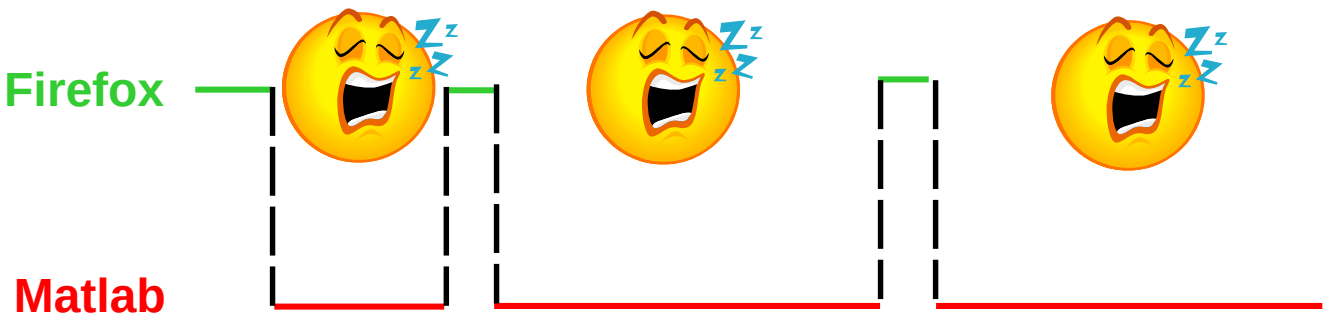
- Τι είναι λειτουργικό;
- Γενιές ΛΣ
- Βασικές υπηρεσίες ΛΣ
- Τύποι αρχιτεκτονικών υπολογιστών και αντίστοιχων ΛΣ
- Οργάνωση και δομή ΛΣ
- Μηχανισμοί και πολιτικές

Διεργασίες και Νήματα: Γιατί;

- Πολλά «πράγματα» γίνονται ταυτόχρονα.
  - Μπορούμε να τα απλοποιήσουμε για το λειτουργικό;
    - Πώς απλοποιούμε; ΑΠΟΣΥΝΘΕΣΗ
    - Βάλε καθένα από τα «πράγματα» να τρέχει σε μια «διεργασία» (process)
      - Τώρα κάθε διεργασία είναι απλή. Απλά (???) το ΛΣ πρέπει να χειριστεί τις διεργασίες και να τις αλληλοπροστατεύσει



Άλλοι Λόγοι; Επικάλυψη



Κανένας άλλος λόγος; Ταχύτητα (1/2)

- Μα πώς;
  - A (διάρκεια 100 sec, έτοιμη τη στιγμή 0)
  - B (διάρκεια 10 sec, έτοιμη τη στιγμή 0+dt)

  - Χρόνος ολοκλήρωσης A: 100 sec
  - Χρόνος ολοκλήρωσης B: 110 sec
  - Μέσος χρόνος ολοκλήρωσης: (100 + 110) / 2 = 105 sec.

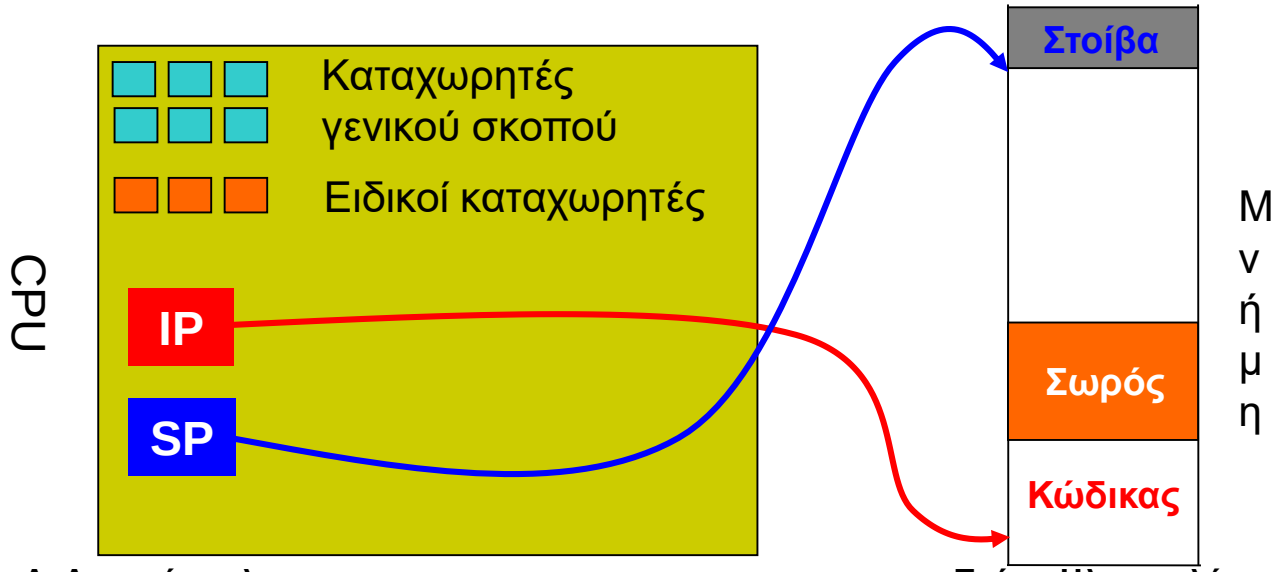
  - Χρόνος ολοκλήρωσης A: 110 sec
  - Χρόνος ολοκλήρωσης B: 20 sec
  - Μέσος χρόνος ολοκλήρωσης: (110 + 20) / 2 = 65 sec. **!!!**

Κανένας άλλος λόγος; Ταχύτητα (2/2)

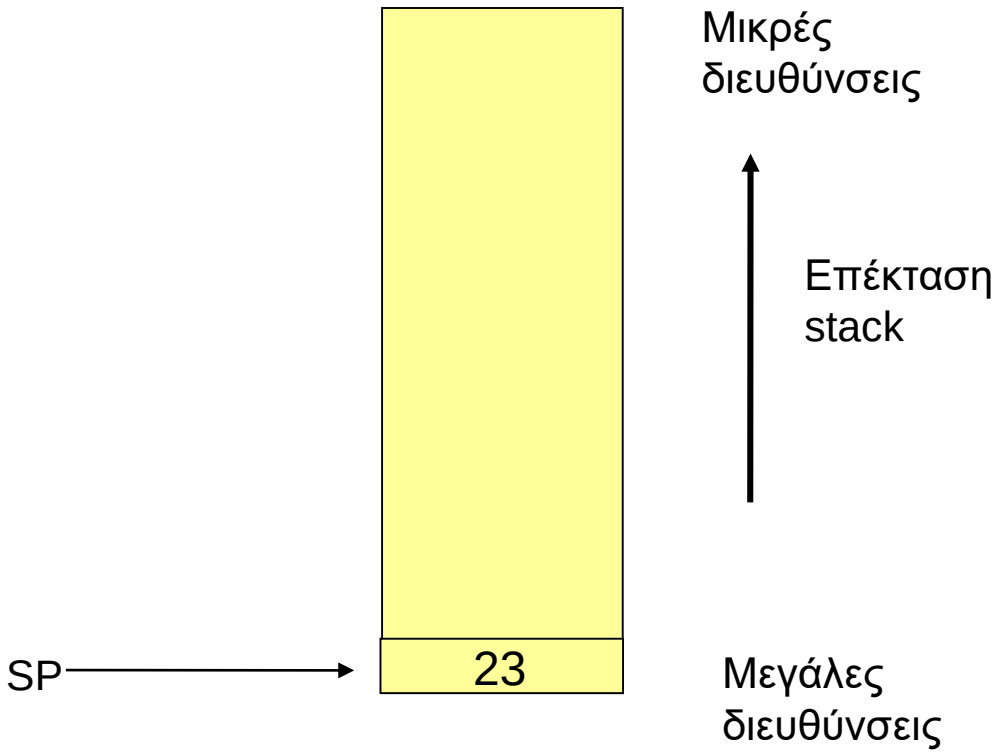
- Αν έχουμε πάνω από 1 μονάδες εκτέλεσης;
  - Κάθε διεργασία σε μια μονάδα εκτέλεσης: Παραλληλισμός ☺
  - Νέα ιδέα;
    - Πολλοί ταμίες (μονάδες εκτέλεσης) στην τράπεζα εξυπηρετούν πολλούς πελάτες, έναν πελάτη τη φορά ο καθένας.
- Πόσο πιο γρήγορα;
  - Ιδανικά N φορές πιο γρήγορα.
  - Όμως:
    - Επιβαρύνσεις.
    - Αν στο project είχατε ομάδες των 50 ατόμων θα τελειώνατε 50 φορές συντομότερα;

Τι είναι ένα νήμα;

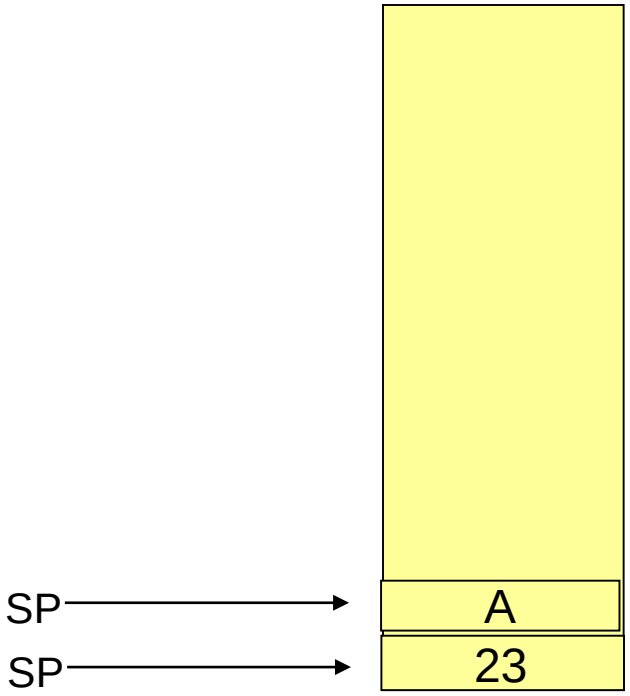
- Ότι χρειάζεται για να εκτελεστεί ένα πρόγραμμα στον επεξεργαστή.
  - + Κάποιες ακόμα λεπτομέρειες (file descriptors, page table …)
- Δηλαδή;
  - Γρήγορο μάθημα αρχιτεκτονικής…



Παρένθεση: Απλά Μαθήματα Stack… (IA32)



PUSH IA32

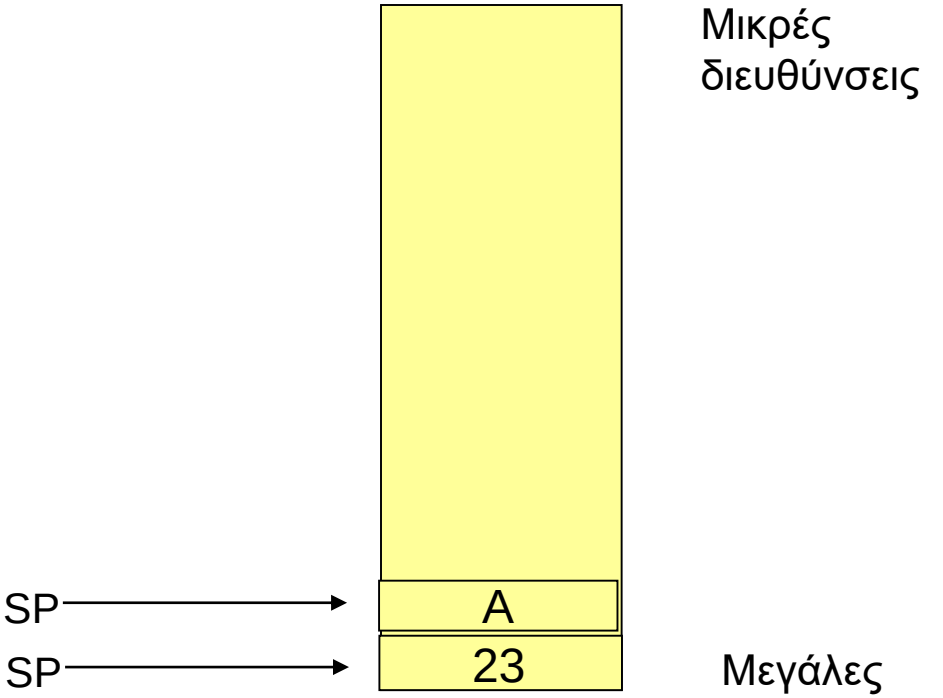


push A

SP = SP – 4

(SP) = A

POP (IA32)



pop <reg>

<reg> = (SP)

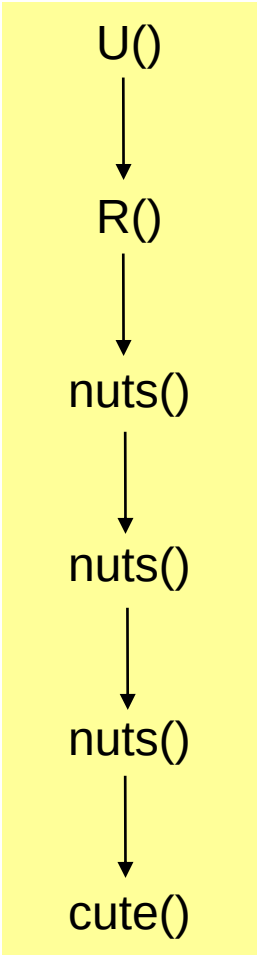
SP = SP + 4

Κλήση συνάρτησης (C)

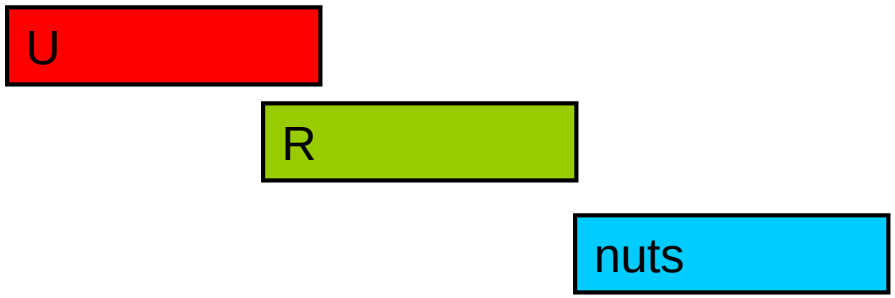

```
U() {
   …
   R()
   …
}
```

```
R() {
   …
   nuts()
   …
}
```

```
nuts() {
   …
   if (…)
     nuts()
   else
     cute()
   …
}
```

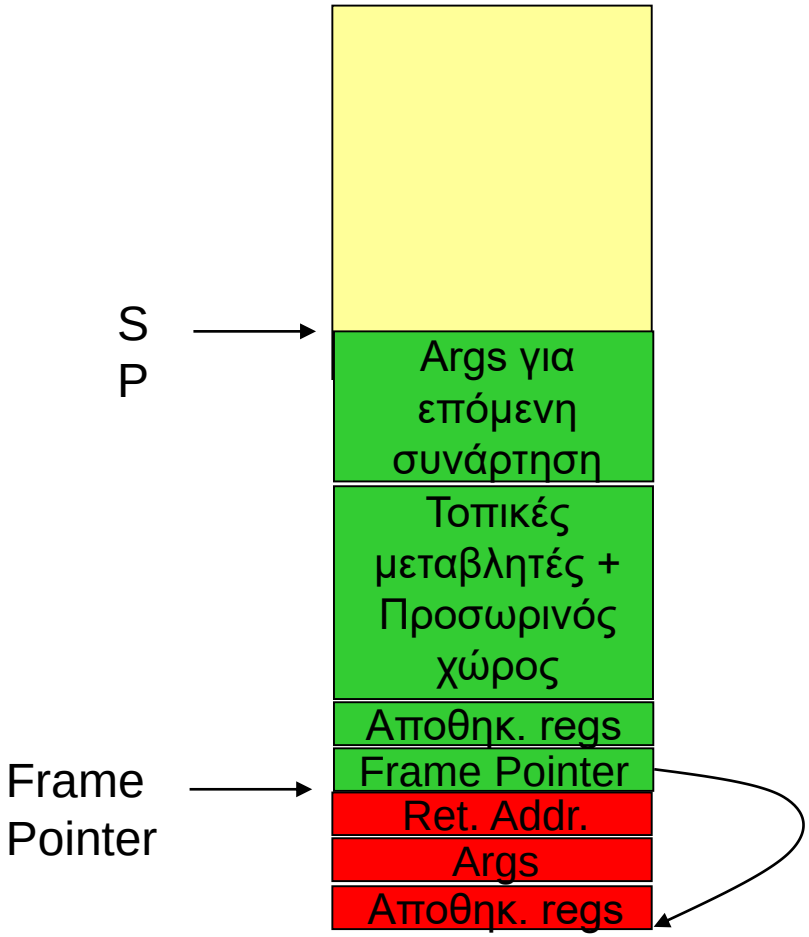


Stack Frames



- Περιέχουν
  - Παραμέτρους συνάρτησης
  - Τοπικές μεταβλητές
  - Χώρο (scratch) για προσωρινή αποθήκευση
  - Διευθύνσεις επιστροφής
- Διαχείριση
  - Δείκτης SP στην τρέχουσα θέση στη stack
  - Ενδεχομένως και δείκτης στην αρχή του stack frame

Στιγμιότυπο Stack (IA32)



Τι είναι διεργασία;

- Νήμα + Χώρος διευθύνσεων
  - Προστασία!!!
- Και γιατί νήματα και διεργασίες;
  - Περιπτώσεις όπου θέλουμε πολλαπλά νήματα σε κάθε διεργασία (π.χ. Servers, παράλληλα προγράμματα…)
- Διεργασία <> Πρόγραμμα
  - Πρόγραμμα: Κώδικας, στατικά δεδομένα
  - Διεργασία: Πρόγραμμα σε εκτέλεση (stack, regs)

Δημιουργία διεργασιών

- Πώς;
  - Πώς θα βρούμε όλη την πληροφορία που περιγράφει μια διεργασία; (registers, page table, file descriptors, …)
  - Κλωνοποίηση! (Dolly)
    - Αντίγραφο της τρέχουσας διεργασίας.
    - Στο Unix: fork()
      - Αντέγραψε όλη τη μνήμη
        - Ίσως και όχι όλη … (διαμοιραζόμενο τμήμα κώδικα, αντιγραφή κατά την εγγραφή).
      - Αντέγραψε τους καταχωρητές
        - Ίσως όχι και όλους: Ο καταχωρητής που περιέχει την τιμή που επιστρέφει η fork τίθεται στην τιμή 0 για τη νέα διεργασία και στον αριθμό της νέας διεργασίας για τον πατέρα.
      - Τοποθέτησε τη νέα διεργασία στη λίστα των διεργασιών του πυρήνα.

Unix: fork()

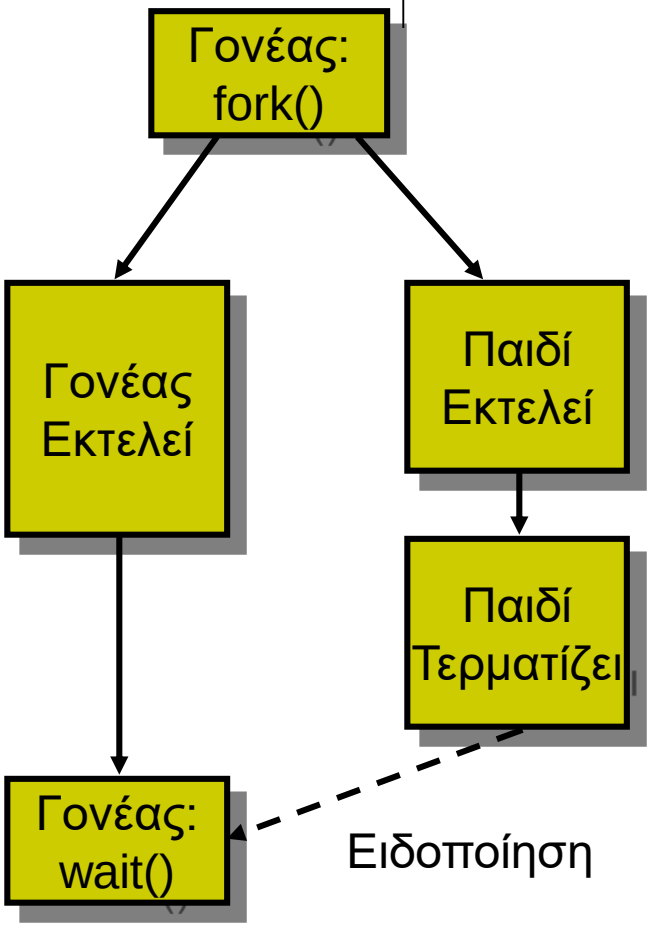

```
id = fork()
if (id == 0) {
    child_code()
}
else {
    printf("Child pid = %d\n", id);
    parent_code()
}
```

Γιατί Όλες οι Διεργασίες Μοιάζουν Ίδιες Μαμά; ☹

- Λύση: Η διεργασία στη «σχολική ηλικία»
  - Πάρε τον κλώνο
  - Άλλαξε τον κώδικα και τα στατικά δεδομένα.
  - Άλλαξε τους registers «κατάλληλα» ώστε να εκτελεστεί ο νέος κώδικας.
  - Κράτησε και κάτι από την παλιά διεργασία… (file descriptors)
- Όλα αυτά πώς;
  - Unix: execve(char *path, char *argv[], char *envp[])

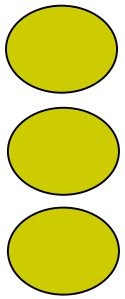
Ότι αρχίζει ωραίο… (τερματισμός διεργασιών)

- Τερματισμός:
  - Αυτοκτονία: Κλήση exit()
  - Δολοφονία: Kill από άλλη διεργασία
  - Ατύχημα: TRAP, σήμα που προκαλεί τερματισμό.
  - Γηρατειά: Επιστροφή από την πρώτη συνάρτηση που κάλεσε το παιδί μετά τη δημιουργία του.
- Γονέας μπορεί να (πρέπει να) περιμένει τον τερματισμό του παιδιού.
  - Διαφορετικά… zombie!
  - Ειδοποιείται από το ΛΣ

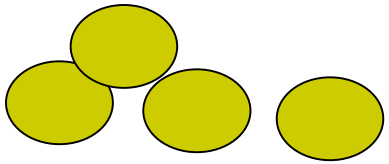


Η Κοινωνία μιας Διεργασίας

- Μονοπρογραμματιστικό περιβάλλον
  - Μία διεργασία εκτελείται μέχρι την ολοκλήρωσή της
  - Εύκολο για το ΛΣ, «κακό» για το χρήστη.
  - Χαλάει την απομόνωση μεταξύ διεργασιών.
    - Τι γίνεται αν μια διεργασία πέσει σε άπειρο loop;
- Πολυπρογραμματιστικό περιβάλλον
  - Χρονοδιαμοίραση επεξεργαστή.
  - Υποστηρίζεται από τα περισσότερα σύγχρονα ΛΣ.



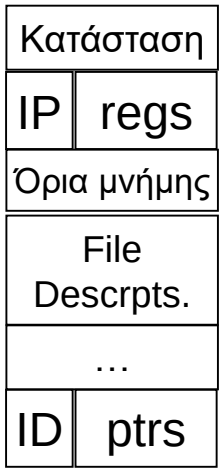
CPU

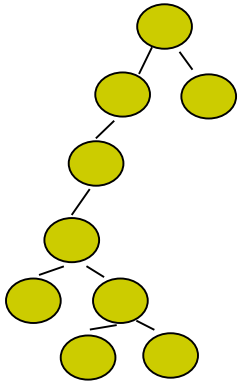


CPU

Να υπάρχει τάξη… (οργάνωση διεργασιών)

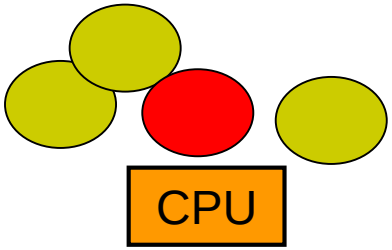
- Πώς αναπαριστάται η διεργασία στο λειτουργικό;
  - PCB:
    - Κάποια περίπλοκη χημική ουσία; (όχι!)
    - Process Control Block
- Και η τάξη που λέγαμε;
  - Ιεραρχική οργάνωση σε δέντρο.
  - Οργάνωση σε ουρές ανάλογα με την κατάσταση των διεργασιών





Ο «Φανταστικός» Κόσμος του Πολυπρογραμματισμού

- Κάθε διεργασία έχει την ψευδαίσθηση της αποκλειστικής χρήσης του επεξεργαστή
  - Όμως σε ένα μονοεπεξεργαστικό σύστημα όλοι μοιράζονται τον ίδιο επεξεργαστή
- Πώς;



While (1) {

Διέκοψε τρέχουσα διεργασία

Σώσε κατάσταση (στο PCB)

Βρες επόμενη διεργασία

Φόρτωσε κατάσταση (από PCB)

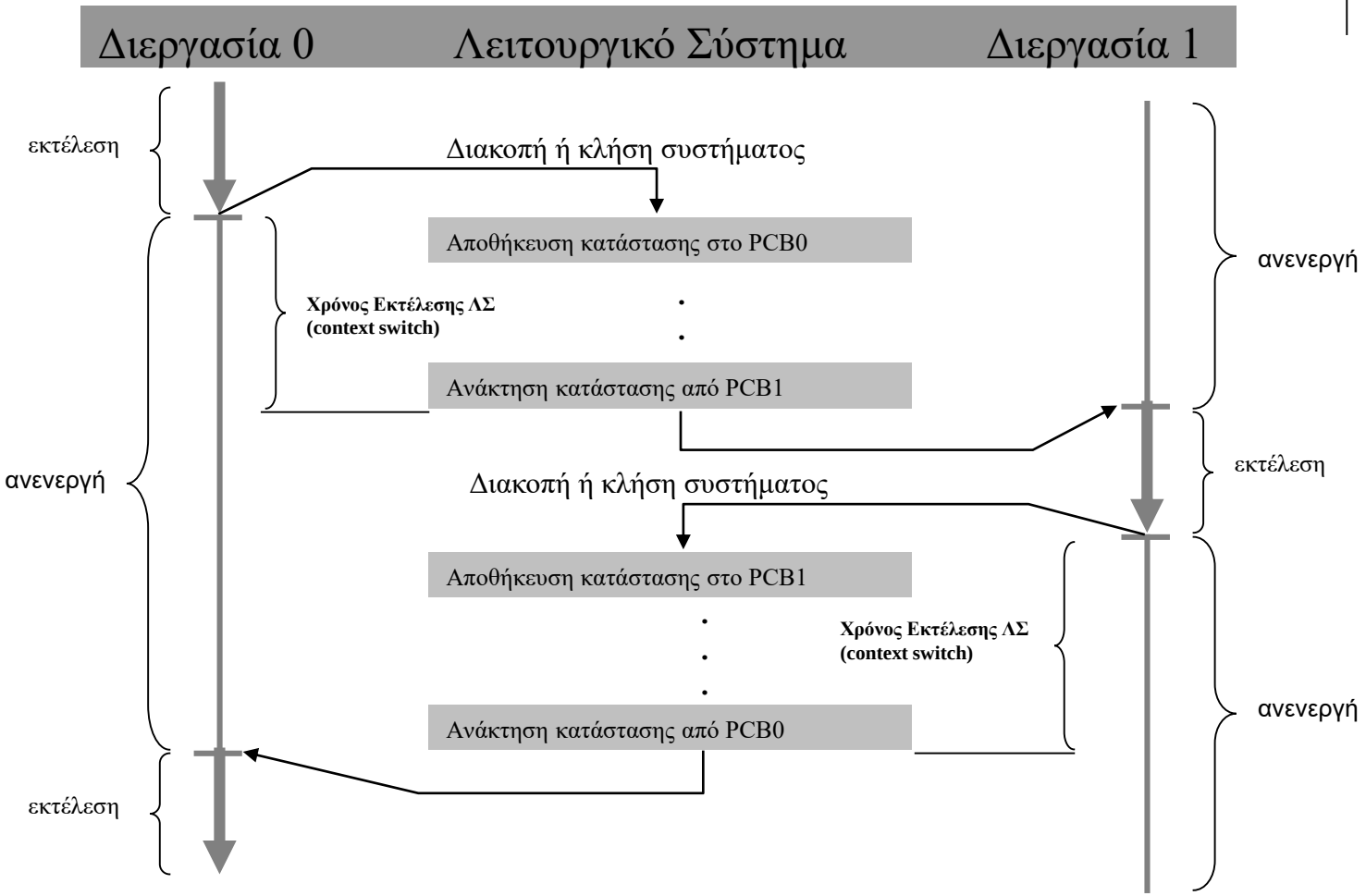
Τρέξε τη!

}

Και πώς Αλλάζει η Διεργασία στον Επεξεργαστή;

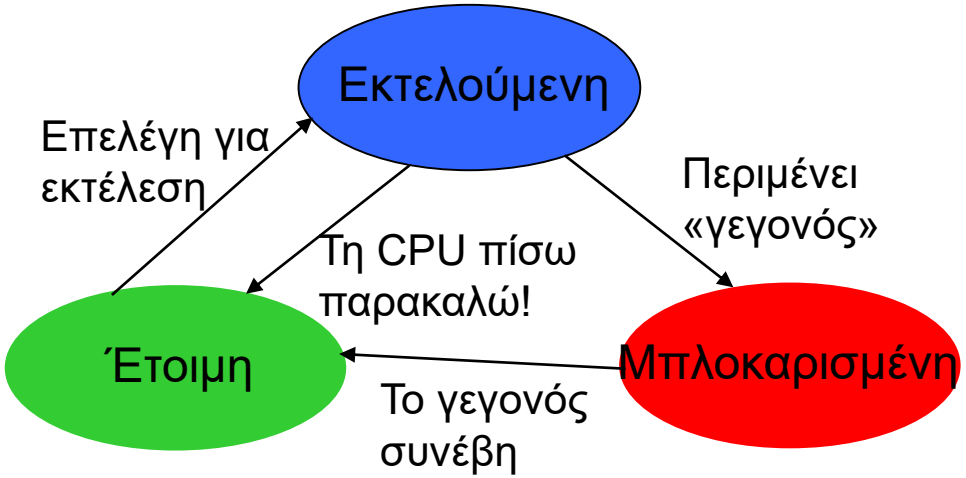
- Μεταγωγή περιβάλλοντος (context switch)
  - Τι;
    - Σώσε ότι μπορεί να σου «χαλάσει» η καινούρια διεργασία
      - Καταχωρητές γενικού σκοπού
      - Καταχωρητές ειδικού σκοπού
      - Καταχωρητές κινητής υποδιαστολής
      - Ενίοτε τεμπέλικη προσέγγιση… Σώσε μόνο όταν πάει η νέα διεργασία να χαλάσει κάτι (Linux, καταχωρητές κινητής υποδιαστολής).
  - Πώς;
    - Δύσκολο… Πρέπει να εκτελεστεί κώδικας χωρίς να αλλάξει τους καταχωρητές!
    - Εξαρτάται από την αρχιτεκτονική
      - Ειδικές εντολές
      - Δεσμευμένοι registers…
  - Πόσο;
    - Ακριβό!
      - Κόστος για την ίδια τη μεταγωγή.
      - Γραμμάτια για το μέλλον (καθαρισμός TLB, «μόλυνση» των μνημών cache).

Μεταγωγή Περιβάλλοντος (από ψηλά)



Καταστάσεις διεργασιών

- Τρέχουσα
  - Εκτελείται αυτή τη στιγμή.
- Έτοιμη
  - Έτοιμη προς εκτέλεση
- Μπλοκαρισμένη
  - Κάτι περιμένει…



- Επιλέγεται διεργασία προς εκτέλεση πάντα από τις έτοιμες
  - Αν 1; Εύκολο…
  - Αν 0; Idle loop
  - Αν >1; Ουπς… Πολιτική!

Προβληματάκια;

- Βεβαίως!
  - Ποια διεργασία θα επιλεγεί για να τρέξει κάθε φορά;
    - Λίγο υπομονή
  - Προστασία
    - Δεν πρέπει καμία διεργασία να έχει πρόσβαση στην «κατάσταση» της άλλης
  - Δικαιοσύνη
    - Όλοι πρέπει κάποια στιγμή να τρέξουν…
    - Και μάλιστα σχετικά «ισότιμα» (δεν θέλουμε γκρίνιες στο σύστημα…)

Έχω μια Διεργασία που όλο όλο τρέχει…

- … και που θα σταματήσει;
  - Trap: Γεγονός που προκλήθηκε από την ίδια τη διεργασία
    - Κλήση συστήματος
    - Πρόβλημα (λάθος εντολή, λάθος ορίσματα…)
    - Σφάλμα σελίδας
  - Διακοπή:
    - Τελείωσε μια ενέργεια I/O
    - Τέλος χρόνου, ο επόμενος παρακαλώ…
      - Κάθε πότε;
  - Από την οπτική της διεργασίας:
    - Άμεσα: Χαρίζω τον επεξεργαστή μου στο σύστημα
    - Έμμεσα: Τι έγινε; (Η κάτι έκανα και μπλόκαρα, ή κάποιος με σταμάτησε).

Διεργασίες vs. Νήματα

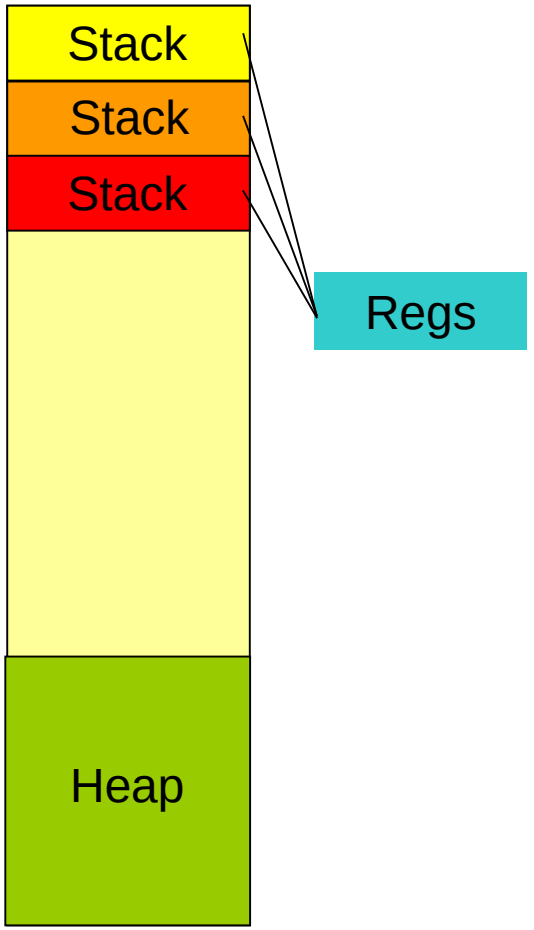
- Διεργασίες: Διαφορετικοί χώροι διευθύνσεων
  - Αλλαγή πίνακα σελίδων (page table)
  - Επικοινωνία;
  - Προστασία;
    - Πρέπει η κατάσταση να σώζεται σε προστατευμένο χώρο! (ΛΣ)
    - Χρειάζεται υποστήριξη από το υλικό για να μπορούμε να «κλέψουμε» τον επεξεργαστή.
- Νήματα: Ίδιος χώρος διευθύνσεων
  - Πιο «ελαφριά»!
  - Γιατί;
    - Πρόσβαση σε κοινές δομές δεδομένων
    - Καλύτερη απόκριση
    - Χρονοβελτίωση σε πολυεπεξεργαστικά συστήματα
  - Μπορεί να γίνει μεταγωγή περιβάλλοντος σε επίπεδο χρήστη;

Τύποι Νημάτων

- Νήματα επιπέδου χρήστη (N:1)
- Νήματα επιπέδου πυρήνα (1:N)
- Πολυεπίπεδος πολυνηματισμός (M:N)

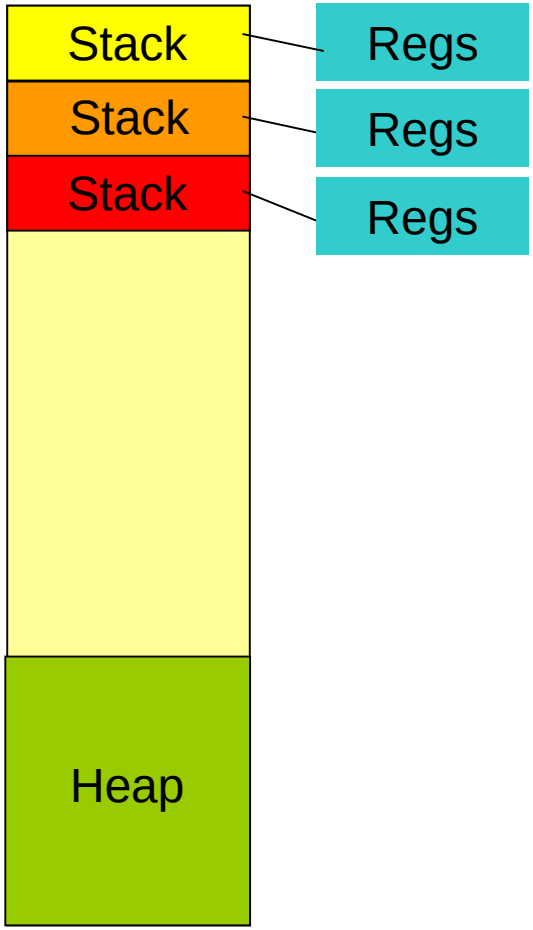
Νήματα Χρήστη

- Προσφέρονται από βιβλιοθήκη αποκλειστικά σε επίπεδο χρήστη
  - Τρέχουν πάνω από ένα νήμα επιπέδου πυρήνα
  - Το ΛΣ δεν ξέρει τίποτα…
- Δεν χρειάζονται αλλαγές στον πυρήνα
- Μια ενέργεια από ένα μπορεί να τα μπλοκάρει όλα…
- Δεν μπορούν να εκμεταλλευτούν παραπάνω επεξεργαστές



Νήματα Πυρήνα

- Το ΛΣ ξέρει τα πάντα…
  - Χειρίζεται τα νήματα
    - Χρονοδρομολογεί
    - Αναλαμβάνει μεταγωγές περιβάλλοντος
- Τα νήματα ανεξάρτητα
- Μπορούν να εκμεταλλευτούν πολυεπεξεργαστικά συστήματα
- Όμως:
  - Βιβλιοθήκες χρήστη; (προβλήματα, πρέπει να ξαναγραφτούν …)
  - Παραπάνω πόροι (μνήμη)



Η Χρυσή Τομή (M:N)

- Η μέση λύση
  - Πολλαπλά νήματα χρήστη τρέχουν πάνω από κάθε νήμα πυρήνα
  - Αλλά έχουμε και πολλαπλά νήματα πυρήνα…
- Χρονοδρομολογητές:
  - Χρήστη: Ποιο νήμα χρήστη θα τρέξει πάνω από κάθε νήμα πυρήνα;
  - Πυρήνα (ΛΣ): Ποιο νήμα πυρήνα θα τρέξει πάνω από κάθε CPU;
  - Όλα καλά αν οι χρονοδρομολογητές χρήστη και πυρήνα «συνεργάζονται» αρμονικά
    - Δεν είναι και εύκολο
    - Συνήθως δεν «κουβεντιάζουν»…