ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Λειτουργικά Συστήματα

Ενότητα # 11: Σχεδίαση λειτουργικών συστημάτων

Διδάσκων: Γεώργιος Ξυλωμένος

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

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Οικονομικό Πανεπιστήμιο Αθηνών**» 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.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

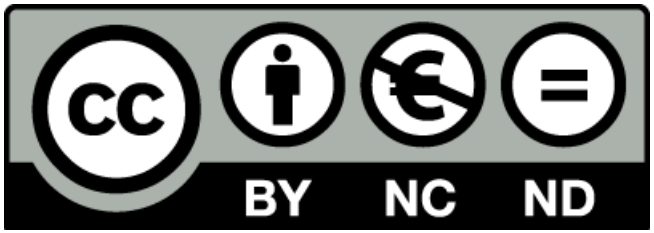
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

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.
- Οι εικόνες προέρχονται από το βιβλίο «Σύγχρονα Λειτουργικά Συστήματα», A.S. Tanenbaum, 3η έκδοση, 2009, Εκδόσεις Κλειδάριθμος.



Σκοποί ενότητας

- Κατανόηση των ιδιαιτεροτήτων της σχεδίασης ΛΣ και των βασικών διασυνδέσεων που πρέπει να σχεδιαστούν.
- Εξοικείωση με τα βασικά ζητήματα υλοποίησης και τα κύρια προβλήματα απόδοσης στα ΛΣ.
- Εισαγωγή στη διαχείριση έργων λογισμικού μεγάλης κλίμακας και στις σχεδιαστικές τάσεις των ΛΣ.

Περιεχόμενα ενότητας

- Το πρόβλημα σχεδίασης
- Σχεδίαση διασυνδέσεων
- Υλοποίηση
- Απόδοση
- Διαχείριση έργων
- Σχεδιαστικές τάσεις

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Το πρόβλημα σχεδίασης

Μάθημα: Λειτουργικά Συστήματα, **Ενότητα #11:** Σχεδίαση λειτουργικών συστημάτων

Διδάσκων: Γιώργος Ξυλωμένος, **Τμήμα:** Πληροφορικής

Στόχοι (1 από 5)

- Ποιος είναι ο στόχος ενός νέου ΛΣ;
  - Ένα μάθημα από τις γλώσσες προγραμματισμού
  - Κακό παράδειγμα: PL/I
    - Θα αντικαθιστούσε FORTRAN, COBOL και Algol
    - Εξαφανίστηκε πριν από όλες αυτές
    - Δεν υπήρχε ένα ενιαίο όραμα σχεδίασης
  - Καλό παράδειγμα: C
    - Δημιουργήθηκε μόνο για προγραμματισμό ΛΣ
    - Επιβίωσε περισσότερο από τις σύγχρονές της

Στόχοι (2 από 5)

- Στόχοι ενός ΛΣ γενικής χρήσης
  - Ορισμός αφαιρέσεων
  - Παροχή πρωτογενών λειτουργιών
  - Εξασφάλιση της απομόνωσης
  - Διαχείριση του υλικού
- Μπορεί να υπάρχουν και πιο ειδικοί στόχοι
  - Παράδειγμα: ΛΣ πραγματικού χρόνου

Στόχοι (3 από 5)

- Ορισμός αφαιρέσεων
  - Οι βασικές αφαιρέσεις είναι γνωστές
  - Διεργασίες, χώροι διευθύνσεων, αρχεία
  - Αλλά και νήματα, συγχρονισμός, επικοινωνία
  - Ο ακριβής ορισμός δεν είναι προφανής
    - Παράδειγμα: χειρισμός νημάτων μετά από fork()
    - Αντιγράφονται τα νήματα;
    - Αντιγράφονται τα εκκρεμή σήματα;

Στόχοι (4 από 5)

- Παροχή πρωτογενών λειτουργιών
  - Χειρίζονται τις αφαιρέσεις
    - Για την ακρίβεια, τις δομές που τις παριστάνουν
    - Δημιουργία, διαγραφή, τροποποίηση
  - Υλοποιούνται ως κλήσεις συστήματος
- Διαχείριση του υλικού
  - Έλεγχος υλικού σε χαμηλό επίπεδο
  - Περιβάλλον λειτουργίας οδηγών συσκευών

Στόχοι (5 από 5)

- Εξασφάλιση της απομόνωσης
  - Διάκριση των εργασιών των διάφορων χρηστών
    - Χρήση διεργασιών για ομαδοποίηση πόρων
    - Προστασία αρχείων και δομών δεδομένων
  - Ελεγχόμενος καταμερισμός πόρων
    - Επικοινωνία και συγχρονισμός διεργασιών
  - Απομόνωση των αστοχιών
    - Κάθε διεργασία αποτυγχάνει αυτόνομα
    - Ιδανικό, το ίδιο ισχύει και για τα τμήματα του ΛΣ

Δυσκολίες (1 από 4)

- Γιατί τα ΛΣ δεν βελτιώνονται όπως το υλικό;
  - Επιθυμία για συμβατότητα με προβληματικά ΛΣ
  - Κακή σχεδίαση και ανεπαρκής συντήρηση
  - Υπάρχουν όμως και θεμελιώδεις δυσκολίες
- Τα ΛΣ είναι τεράστια
  - Πολλά εκατομμύρια γραμμές κώδικα
  - Αδύνατον να τα κατανοήσει ένας σχεδιαστής
  - Τα τμήματά τους δεν απομονώνονται εύκολα

Δυσκολίες (2 από 4)

- Ο ταυτοχρονισμός είναι περίπλοκος
  - Σε κάθε στιγμή μπορεί να έχουμε διακοπές
  - Κίνδυνος αδιεξόδων και αγώνων δρόμου
- Οι χρήστες μπορεί να είναι εχθρικοί
  - Κλοπή ή αλλοίωση στοιχείων άλλων χρηστών
  - Κάθε χρήστης πρέπει να θεωρείται επικίνδυνος
- Πρέπει να επιτρέπεται ο καταμερισμός πόρων
  - Παρά το ότι οι χρήστες μπορεί να είναι εχθρικοί!

Δυσκολίες (3 από 4)

- Μεγάλη διάρκεια ζωής
  - Τα Windows (NT) έκλεισαν 2 δεκαετίες
  - Οι παραλλαγές του UNIX ακόμη περισσότερες
  - Κανείς δεν μπορεί να προβλέψει το μέλλον
- Ο τρόπος χρήσης είναι απρόβλεπτος
  - Windows και UNIX δεν σχεδιάστηκαν για WWW
  - Οι ανάγκες και οι κίνδυνοι αλλάζουν συνεχώς
    - Δεν βοηθάει και η μακροβιότητα των συστημάτων

Δυσκολίες (4 από 4)

- Ανάγκη συμβατότητας με υλικό
  - Χιλιάδες συσκευές εισόδου / εξόδου
  - Αλλαγή του βασικού υλικού του συστήματος
  - Μόνο η Apple ελέγχει στενά το υλικό
    - Εν μέρει, αυτό εξηγεί τη σταθερότητα του OS X
- Ανάγκη αναδρομικής συμβατότητας
  - Υποστήριξη παρωχημένων αφαιρέσεων
  - Αύξηση μεγέθους και πολυπλοκότητας

Σχεδίαση διασυνδέσεων

Μάθημα: Λειτουργικά Συστήματα, **Ενότητα #11:** Σχεδίαση λειτουργικών συστημάτων

Διδάσκων: Γιώργος Ξυλωμένος, **Τμήμα:** Πληροφορικής

Διασυνδέσεις

- Το ΛΣ καθορίζεται από τις διασυνδέσεις του
  - Διασυνδέσεις για προγραμματιστές
    - Η γραφική διεπαφή δεν περιλαμβάνεται
    - Μπορεί να αλλάζει ανάλογα με τον χρήστη
  - Διασυνδέσεις υψηλού επιπέδου
    - Παράδειγμα: glibc, Win32, DirectX
  - Διασυνδέσεις οδηγών συσκευών
    - Παράδειγμα: VxD, WDM

Καθοδηγητικές αρχές

- Απλότητα
  - Βοηθούν την κατανόηση και την υλοποίηση
- Πληρότητα
  - Καλύπτουν όλες τις ανάγκες των χρηστών
  - Αλλά: με τον ελάχιστον δυνατό μηχανισμό
    - Υπάρχουν και βιβλιοθήκες επιπέδου χρήστη!
- Αποδοτικότητα
  - Μόνο λειτουργίες με αποδοτική υλοποίηση

Υποδείγματα

- Δύο κατηγορίες πελατών
  - Χρήστες προγραμμάτων εφαρμογών
  - Προγραμματιστές εφαρμογών
- Σχεδίαση με βάση γραφική διεπαφή (Mac OS)
  - Ξεκινάμε από τις λειτουργίες του χρήστη
    - Καταλήγουμε σε μοντέλο οδηγούμενο από γεγονότα
- Σχεδίαση με βάση κλήσεις συστήματος (UNIX)
  - Ξεκινάμε από τις ανάγκες των προγραμματιστών

Υποδείγματα διασυνδέσεων χρήστη

- Ποιο είναι το βασικό παράδειγμα/μεταφορά;
  - Πρέπει να υποστηρίζεται από βιβλιοθήκες
    - Ώστε όλες οι εφαρμογές να το υπηρετούν
- Διασύνδεση επιφάνειας γραφείου
  - Υλοποιείται με το μοντέλο WIMP
  - Βασίζεται στο ποντίκι
- Διασύνδεση αφής
- Διασύνδεση φωνής

Υποδείγματα εκτέλεσης (1 από 2)

- Αλγοριθμικό υπόδειγμα
  - Είσοδος, επεξεργασία, έξοδος
  - Κλήσεις συστήματος για επικοινωνία με ΛΣ

```
main()
{
    int ...;

    init();
    do_something();
    read(...);
    do_something_else();
    write(...);
    keep_going();
    exit(0);
}
```

Υποδείγματα εκτέλεσης (2 από 2)

- Οδηγούμενο από συμβάντα υπόδειγμα
  - Αρχικοποίηση, αναμονή, εξυπηρέτηση
  - Το ΛΣ παρέχει τα συμβάντα

```
main()
{
    mess_t msg;


    init();
    while (get_message(&msg)) {
        switch (msg.type) {
            case 1: ...;
            case 2: ...;
            case 3: ...;
        }
    }
}
```

Υποδείγματα δεδομένων

- Ταινίες με εγγραφές (FORTRAN)
  - Αντιστοίχιση συσκευών σε λογικές ταινίες
- Αρχεία byte (UNIX)
  - Σωληνώσεις, συσκευές, αρχεία
- Αντικείμενα (Windows)
  - Χειριστήρια αντικειμένων με μεθόδους
- Έγγραφα (Παγκόσμιος Ιστός)
  - Προσδιορίζονται με URL

Διασύνδεση κλήσεων (1 από 2)

- Το υπόδειγμα δεδομένων είναι σημαντικό
  - Αν τα πάντα είναι αρχεία, χρειαζόμαστε μία read
- Πρέπει να είναι οι ελάχιστες δυνατές
  - Η βιβλιοθήκη μπορεί να παρέχει παραλλαγές
    - Οι execl/lp/le/v/vp/ve υλοποιούνται από την exec
  - Γιατί να μην είναι περισσότερες;
    - Παραπάνω κώδικας σημαίνει παραπάνω σφάλματα
  - Αν είναι πολύ περίπλοκες, πρέπει να σπάσουν
    - CreateProcess ή fork()/exec();

Διασύνδεση κλήσεων (2 από 2)

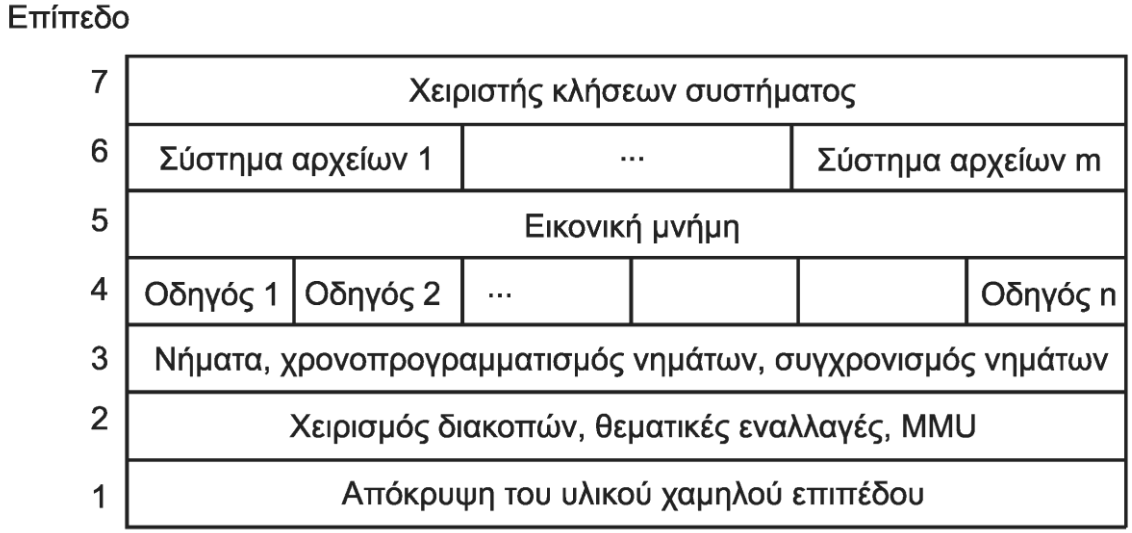
- Δεν κρύβουμε την υπολογιστική ισχύ
  - Οι αφαιρέσεις κρύβουν μόνο τα ανεπιθύμητα!
  - Τα δυνατά χαρακτηριστικά πρέπει να εκτίθενται
- Συνδεσμοστρεφείς ή ασυνδεσμικές κλήσεις;
  - Η χρήση συνδέσεων έχει κόστος
  - Πρέπει να οδηγεί σε μεταγενέστερο κέρδος
    - Παράδειγμα: άνοιγμα τοπικών αρχείων
    - Επιτρέπει απλούστερες κλήσεις στη συνέχεια

Υλοποίηση

Μάθημα: Λειτουργικά Συστήματα, **Ενότητα #11:** Σχεδίαση λειτουργικών συστημάτων

Διδάσκων: Γιώργος Ξυλωμένος, **Τμήμα:** Πληροφορικής

Δομή συστήματος (1 από 4)

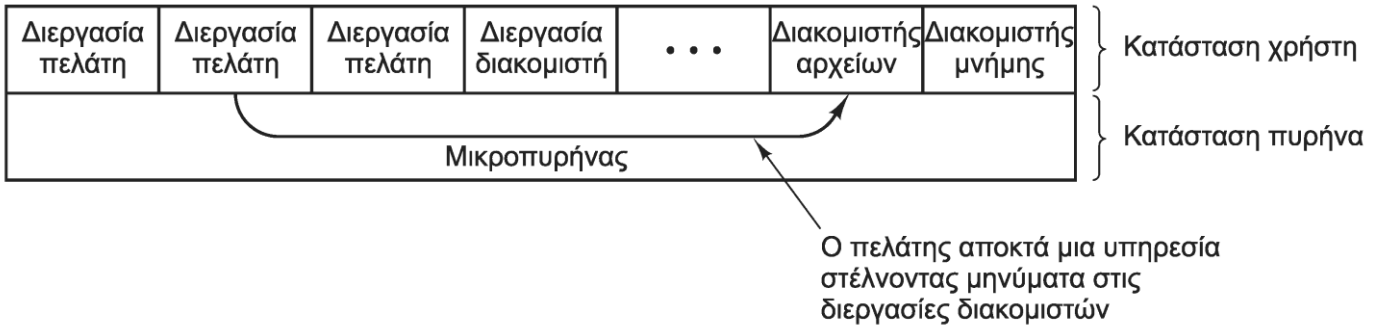


- Πολυεπίπεδα συστήματα
  - Ποια πρέπει να είναι τα επίπεδα;
  - Οδηγοί συσκευών ως ανεξάρτητα νήματα
  - Αρχεία πάνω από εικονική μνήμη

Δομή συστήματος (2 από 4)

- Εξωπυρήνες
  - Βασίζονται στο επιχείρημα των άκρων
    - Αν κάτι θα γίνει από τα άκρα, μην το κάνεις ενδιάμεσα
    - Εκτός αν έτσι κερδίζεις σε απόδοση
  - Το ΛΣ αρκεί να κατανέμει τους πόρους
    - Παράδειγμα: να δίνει μπλοκ δίσκου στους χρήστες
  - Οι αφαιρέσεις παρέχονται από βιβλιοθήκες
    - Παράδειγμα: σύστημα αρχείων επιπέδου χρήστη

Δομή συστήματος (3 από 4)



- Μικροπυρήνας
  - Ο πυρήνας παρέχει ελάχιστη λειτουργικότητα
  - Όλες οι υπηρεσίες παρέχονται από διεργασίες
    - Οι οποίες λειτουργούν σε επίπεδο χρήστη για ασφάλεια
  - Το πρόβλημα είναι η μείωση της απόδοσης
    - Πολλές εναλλαγές περιεχομένου

Δομή συστήματος (4 από 4)

- Επεκτάσιμα συστήματα
  - Προσθήκη υπομονάδων στον πυρήνα
    - Το ΛΣ εξειδικεύεται σε συγκεκριμένη εφαρμογή
  - Απαιτείται προστατευμένος τρόπος εκτέλεσης
    - Αμμοπαγίδες ή υπογραφή κώδικα
- Νήματα πυρήνα
  - Δεν ανήκουν σε διεργασίες χρήστη
  - Επιτρέπουν καλύτερη δόμηση του πυρήνα
    - Κατάλληλα για πολλά μοντέλα οργάνωσης

Μηχανισμός και πολιτική (1 από 2)

- Η πολιτική πρέπει να μπορεί να αλλάξει
  - Είτε αφήνεται στο επίπεδο χρήστη
  - Είτε παραμετροποιείται από τον χρήστη
  - Μεγαλύτερη ευελιξία και δυνατότητα εξέλιξης
- Χρονοπρογραμματισμός νημάτων
  - Μηχανισμός: ουρές πολλών επιπέδων
  - Πολιτική: προτεραιότητες χρηστών
    - Αρχική προτεραιότητα και δυνατότητα αλλαγής της

Μηχανισμός και πολιτική (2 από 2)

- Σελιδοποίηση
  - Μηχανισμός: διαχείριση σφαλμάτων και δομών
  - Πολιτική: αλγόριθμος αντικατάστασης
    - Τοπικός ή καθολικός, LRU, FIFO, …
- Υπομονάδες
  - Μηχανισμός: τρόπος φόρτωσης στον πυρήνα
  - Πολιτική: ποιος μπορεί να φορτώσει μονάδες
    - Και ποιες μονάδες μπορεί να φορτώσει

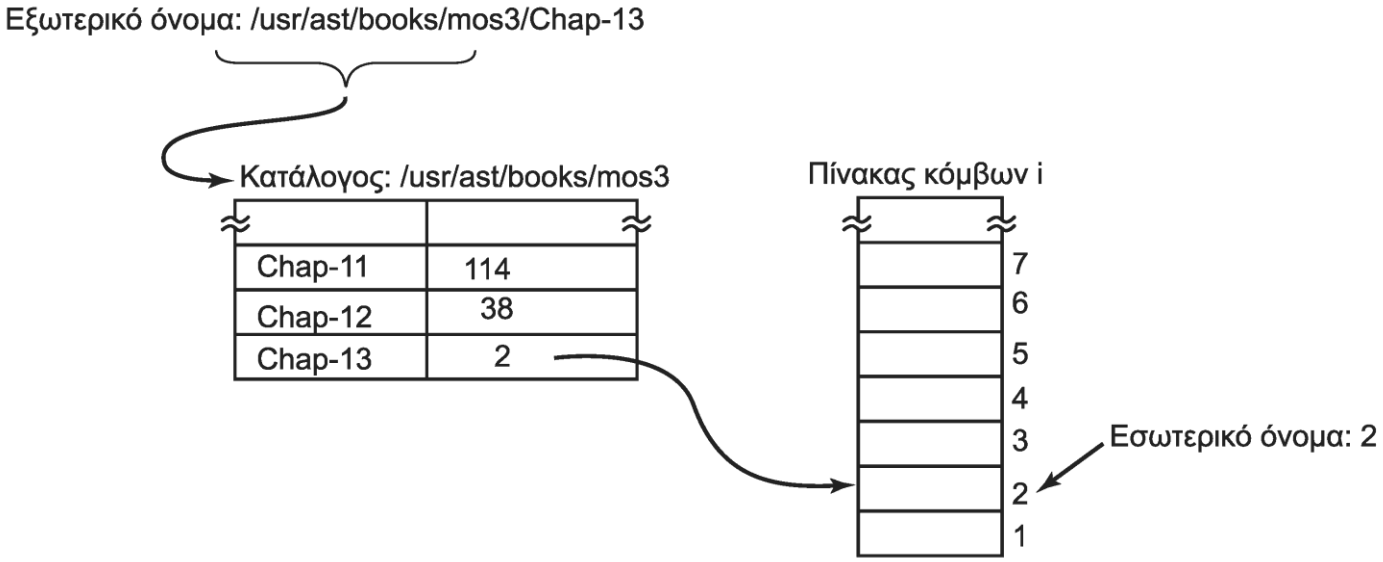
Ορθογωνικότητα

- Ελεύθερος συνδυασμός ανεξάρτητων εννοιών
  - Παράδειγμα: βασικοί και δομημένοι τύποι
- Κλήση clone() στο Linux
  - Ορίζουμε χωριστά τι θα γίνει για κάθε δομή
- Διεργασίες και νήματα στα Windows
  - Διεργασία: πόροι, Νήμα: εκτέλεση
- Fork() και exec() στο UNIX
  - Δημιουργία νέου και φόρτωση χώρου διευθύνσεων

Ονομασία (1 από 2)

- Οι περισσότερες δομές έχουν κάποιο όνομα
  - Αρχεία, διεργασίες, συσκευές
- Εξωτερικά ονόματα: συμβολοσειρές
  - Παράδειγμα: /usr/home/xgeorge
- Εσωτερικά ονόματα: αριθμοί
  - Παράδειγμα: κόμβος i του /usr/home/xgeorge
  - Τυπικά δείχνουν σε έναν πίνακα δομών

Ονομασία (2 από 2)



- Κατάλογοι: εξωτερικά σε εσωτερικά ονόματα
- Πολλαπλοί χώροι ονομάτων
  - Windows: αρχεία, αντικείμενα, μητρώο
  - UNIX: αρχεία, διεργασίες

Χρόνος δέσμευσης

- Πότε δεσμεύεται το όνομα με αντικείμενο;
  - Παράδειγμα: μεταβλητές στη C
    - Καθολικές: πρώιμη, τοπικές: όψιμη
- Κατανομή μνήμης
  - Παλιά κάθε πρόγραμμα δέσμευε στατικά μνήμη
  - Με τη σελιδοποίηση η δέσμευση είναι δυναμική
- Παραθυρικά συστήματα
  - Οι συντεταγμένες σχεδίασης είναι σχετικές

Στατικές ή δυναμικές δομές; (1 από 2)

```
found = 0;
for (p = &proc_table[0]; p < &proc_table[PROC_TABLE_SIZE]; p++) {
    if (p->proc_pid == pid) {
        found = 1;
        break;
    }
}
```

- Στατικός πίνακας διεργασιών
  - Περιορίζει το πλήθος εκτελούμενων διεργασιών
    - Πρέπει να έχουμε μια καλή ιδέα για το πλήθος τους
  - Απλός και αποδοτικός κώδικας
- Δυναμικός πίνακας διεργασιών
  - Λίστα πινάκων: δύσκολη αναζήτηση
  - Διπλασιασμός πίνακα: ανάγκη αντιγραφής

Στατικές ή δυναμικές δομές; (2 από 2)

- Στοίβα πυρήνα
  - Πρέπει να καταχωρηθεί εκ των προτέρων
  - Καταναλώνει χώρο μνήμης πυρήνα
- Χρονοπρογραμματισμός διεργασιών
  - Στατικός σε συστήματα πραγματικού χρόνου
  - Δυναμικός σε όλα τα άλλα
- Δομή πυρήνα
  - Δυναμική προσθήκη ενοτήτων
  - Εισάγει πρόσθετη πολυπλοκότητα

Αναλυτική ή συνθετική υλοποίηση;

- Αναλυτική υλοποίηση
  - Ξεκινάμε από κλήσεις συστήματος
  - Καταλήγουμε στο υλικό
  - Δύσκολο να δοκιμάσουμε οτιδήποτε
- Συνθετική υλοποίηση
  - Ξεκινάμε από υλικό και διακοπές
  - Προχωράμε σε πολυπρογραμματισμό
  - Καταλήγουμε σε μνήμη και συστήματα αρχείων
  - Δεν αντιστοιχεί στη μέθοδο σχεδίασης

Χρήσιμες τεχνικές (1 από 6)

- Απόκρυψη του υλικού
  - Πώς κάνουμε το υλικό να φαίνεται πιο απλό;
- Διαχείριση διακοπών
  - Δημιουργία αναδυόμενου νήματος σε διακοπή
  - Μετατροπή διακοπής σε mutex unlock
  - Μετατροπή διακοπής σε μήνυμα προς νήμα
  - Στόχος: χειρισμός διακοπών από σωστά νήματα
    - Όχι από ό,τι έτυχε να εκτελείται εκείνη τη στιγμή

Χρήσιμες τεχνικές (2 από 6)

- Χρήση ενιαίων αρχείων πηγαίου κώδικα
  - Επιτρέπει τη διόρθωση σφαλμάτων μία φορά
  - Ορισμένες διαφορές κρύβονται εύκολα
    - Υπολογισμός μεγέθους μνήμης στην εκκίνηση
    - Ρύθμιση δομών ανάλογα με τη μνήμη
  - Άλλες είναι πιο δύσκολες
    - Διάφοροι επεξεργαστές θέλουν διαφορετικό κώδικα
    - Χρήση μεταγλώττισης υπό συνθήκη

Χρήσιμες τεχνικές (3 από 6)

```
#include "config.h"

init()
{
#if (CPU == PENTIUM)
/* Κώδικας απόδοσης αρχικών τιμών Pentium */
#endif

#if (CPU == ULTRASPARC)
/* Κώδικας απόδοσης αρχικών τιμών UltraSPARC */
#endif
}
```

(α)

```
#include "config.h"

#if (WORD_LENGTH == 32)
typedef int Register;
#endif

#if (WORD_LENGTH == 64)
typedef long Register;
#endif

Register R0, R1, R2, R3;
```

(β)

- Παράδειγμα: Pentium και UltraSPARC
  - Επιλογή με σημαία ή αρχείο config.h
    - Διαφορετικός κώδικας αρχικοποίησης
    - Ορισμός κατάλληλων βασικών τύπων
    - Δεν μπλέκουμε το σύνολο εντολών με το μέγεθος λέξης!

Χρήσιμες τεχνικές (4 από 6)

- Εμμεσότητα
  - Το πληκτρολόγιο παράγει κωδικό press/release
    - Αντιστοίχιση σε διάφορα σετ χαρακτήρων
    - Δυνατότητα αυθαίρετων συνδυασμών πλήκτρων
  - Στην οθόνη στέλνουμε κωδικούς χαρακτήρων
    - Το αποτέλεσμα εξαρτάται από τη γραμματοσειρά
  - Αριθμοί συσκευών στο UNIX
    - Δείχνουν σε πίνακα οδηγών (χαρακτήρων ή μπλοκ)
    - Επιτρέπουν πολλές συσκευές του ίδιου τύπου

Χρήσιμες τεχνικές (5 από 6)

- Δυνατότητα επαναχρησιμοποίησης
  - Η διαχείριση χαρτών bit χρειάζεται παντού
    - Διαχείριση μπλοκ δίσκου, σελίδων μνήμης, κόμβων i
- Δυνατότητα επανεισόδου
  - Ορθή ταυτόχρονη εκτέλεση του ίδιου κώδικα
    - Απαιτεί προστασία των κοινών δομών
    - Ελαχιστοποίηση απενεργοποίησης διακοπών
  - Διευκολύνει την εκτέλεση με πολλούς πυρήνες

Χρήσιμες τεχνικές (6 από 6)

- Ωμή βία
  - Οι κατάλογοι γενικά δεν είναι ταξινομημένοι
    - Σε μικρούς καταλόγους συμφέρει γραμμική αναζήτηση
  - Πολυπλοκότητα μόνο όπου αξίζει πραγματικά
- Πρώτα έλεγχος για σφάλματα
  - Πρώτα ελέγχουμε αν η κλήση μπορεί να γίνει
  - Μετά δεσμεύουμε τους απαιτούμενους πόρους
  - Επιστροφή πόρων σε περίπτωση προβλήματος

Απόδοση

Μάθημα: Λειτουργικά Συστήματα, **Ενότητα #11:** Σχεδίαση λειτουργικών συστημάτων

Διδάσκων: Γιώργος Ξυλωμένος, **Τμήμα:** Πληροφορικής

Γιατί είναι αργά τα ΛΣ;

- Γιατί το Linux είναι πιο αργό από το Unix V7;
  - Γιατί κάνει πολύ περισσότερα πράγματα
  - Είναι όμως όλα απαραίτητα;
    - Το PnP ελέγχει για νέο υλικό σε κάθε εκκίνηση
    - Δεν αρκεί να γίνεται έλεγχος μόνο όποτε το ζητάμε;
    - Δυνατότητα που παρέχεται έτσι κι αλλιώς!
  - Αξίζει να βάλουμε ένα νέο χαρακτηριστικό;
    - Ή μήπως μπαίνει για να έχουμε νέα χαρακτηριστικά;
    - Ενώ κάνει το σύστημα πιο αργό και επιρρεπή σε σφάλματα;

Τι να βελτιστοποιήσουμε; (1 από 2)

- Αρχικά, μόνο τα προφανή!
  - Παράδειγμα: να έχουμε κρυφή μνήμη μπλοκ
- Όταν λειτουργεί το σύστημα το μετράμε
  - Πόσο γρήγορο και πόσο αποδοτικό είναι;
  - Πόσο δύσκολο είναι να βελτιωθεί;
  - Πόση βελτίωση περιμένουμε;
  - Παράδειγμα: μορφοποίηση σκληρού δίσκου
    - Εκτελείται τόσο σπάνια, που δεν αξίζει βελτιστοποίηση

Τι να βελτιστοποιήσουμε; (2 από 2)

- Υπάρχουν σημεία βελτιστοποίησης
  - Χρονοπρογραμματιστής
    - Εκτελείται μετά από κάθε διακοπή
  - Αλγόριθμος προσκόμισης σελίδων
    - Εκτελείται σε κάθε σφάλμα σελίδας
  - Αλγόριθμος αντικατάστασης σελίδων
    - Εκτελείται περιοδικά (όχι σε κάθε σφάλμα!)
  - Αν είναι αρκετά καλοί, αυτό αρκεί

Χώρος έναντι χρόνου (1 από 4)

```
#define BYTE_SIZE 8                 /* Ένα byte περιέχει 8 bit. */

int bit_count(int byte)             /* Μέτρηση των bit ενός byte */
{
    int i, count = 0;
    for (i = 0; i < BYTE_SIZE; i++) /* βρόχος σε όλα τα bit ενός byte */
        if ((byte >> i) & 1) count++; /* αν αυτό το bit είναι 1, πρόσθεση 1 στο count */
    return(count);                  /* επιστροφή του αθροίσματος */
}
```

- Κλασικό παράδειγμα: μακροεντολές
  - Μέτρηση των bit ενός byte
    - Συχνή χρήση όταν έχουμε χάρτες bit
  - Ορίζουμε μία διαδικασία για τη μέτρηση
  - Η διαδικασία χρησιμοποιεί έναν βρόχο
    - Σε κάθε βήμα ελέγχει ένα bit

Χώρος έναντι χρόνου (2 από 4)

```
/* Μακροεντολή για την πρόσθεση όλων των bit ενός byte και την επιστροφή του αθροίσματος */
#define bit_count(b) ((b&1) + ((b>>1)&1) + ((b>>2)&1) + ((b>>3)&1) + \
        ((b>>4)&1) + ((b>>5)&1) + ((b>>6)&1) + ((b>>7)&1))
```

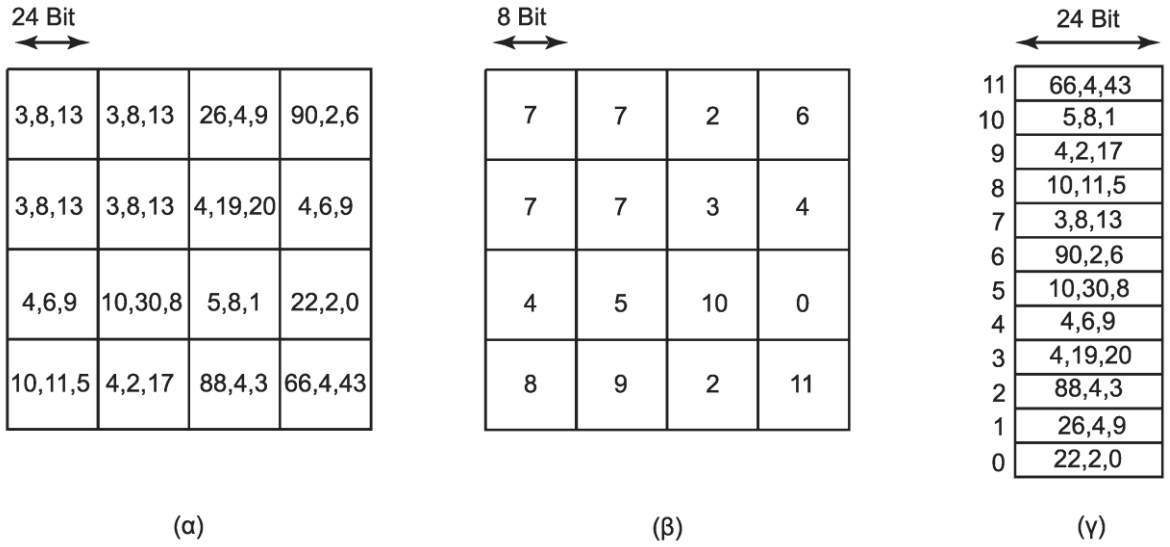
(β)

```
/* Μακροεντολή για την αναζήτηση του πλήθους bit σε έναν πίνακα */
char bits[256] = {0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4, 1, 2, 2, 3, 2, 3, 3, ...};
#define bit_count(b) (int) bits[b]
```

(γ)

- Βήμα 1: αντικατάσταση με μακροεντολή
  - Γλυτώνουμε την κλήση διαδικασίας
  - Γλυτώνουμε την επιβάρυνση του βρόχου
- Βήμα 2: η μακροεντολή χρησιμοποιεί πίνακα
  - Γλυτώνουμε όλους τους υπολογισμούς

Χώρος έναντι χρόνου (3 από 4)



24 Bit

3,8,13	3,8,13	26,4,9	90,2,6
3,8,13	3,8,13	4,19,20	4,6,9
4,6,9	10,30,8	5,8,1	22,2,0
10,11,5	4,2,17	88,4,3	66,4,43

(α)

8 Bit

7	7	2	6
7	7	3	4
4	5	10	0
8	9	2	11

(β)

24 Bit

11	66,4,43
10	5,8,1
9	4,2,17
8	10,11,5
7	3,8,13
6	90,2,6
5	10,30,8
4	4,6,9
3	4,19,20
2	88,4,3
1	26,4,9
0	22,2,0

(γ)

- Πίνακες αναζήτησης χρωμάτων
  - Έστω ότι θέλω χρώμα 24 bit
  - Αποθηκεύω δείκτες στα χρώματα της εικόνας
  - Τα χρώματα αποθηκεύονται σε χωριστό πίνακα

Χώρος έναντι χρόνου (4 από 4)

- Γλώσσα PostScript
  - Αντί να στέλνω χάρτες bit, στέλνω πρόγραμμα
  - Το πρόγραμμα περιγράφει την εικόνα
  - Πρέπει όμως να εκτελείται στη συσκευή
- Πίνακες κατακερματισμού
  - Πολύ γρήγορη δομή αναζήτησης
  - Έχει όμως πάντα κενό χώρο

Κρυφή μνήμη (1 από 2)

Διαδρομή	Αριθμός κόμβου i
/usr	6
/usr/ast	26
/usr/ast/mbox	60
/usr/ast/books	92
/usr/bal	45
/usr/bal/paper.ps	85

- Αποθηκεύω τα αποτελέσματα που υπολογίζω
  - Αν τα χρειαστώ ξανά, δεν τα υπολογίζω ξανά
  - Παράδειγμα: ανάλυση ονομάτων καταλόγων
    - Κάθε τμήμα του ονόματος κοστίζει 2 αναγνώσεις
    - Διαβάζουμε και τον κόμβο i και το μπλοκ δεδομένων
    - Χρήση κρυφής μνήμης μετάφρασης ονομάτων σε κόμβους i

Κρυφή μνήμη (2 από 2)

- Η κρυφή μνήμη θέλει διαχείριση
  - Πρέπει να διαγράφουμε άκυρες τιμές
  - Παράδειγμα: διαγραμμένοι κατάλογοι
    - Ο κόμβος i μπορεί να χρησιμοποιηθεί αλλού
    - Αν ξαναφτιαχτεί το όνομα, θα δείχνει σε άλλο κόμβο i
- Πολλές χρήσεις της κρυφής μνήμης στο ΛΣ
  - Κρυφή μνήμη μπλοκ
  - Κρυφή μνήμη κόμβων i

Υποδείξεις

- Σαν κρυφή μνήμη, αλλά χωρίς συντήρηση
  - Δεν είναι εγγυημένα σωστές
  - Αν είναι όμως, γλυτώνουμε χρόνο
  - Παράδειγμα: σύνδεσμοι σε ιστοσελίδες
    - Αν αλλάξει μια ιστοσελίδα, δεν αλλάζουν οι σύνδεσμοι
    - Σε κάποια στιγμή θα παρατηρήσουμε το πρόβλημα
    - Θα το επιδιορθώσουμε με το χέρι

Εκμετάλλευση της τοπικότητας

- Τοπικότητα εκτέλεσης
  - Κάθε διεργασία χρησιμοποιεί μέρος των σελίδων
  - Οι σελίδες αυτές είναι το σύνολο εργασίας
  - Αρκεί να κρατάμε το σύνολο εργασίας στη μνήμη
- Τοπικότητα καταλόγων
  - Συχνά δουλεύουμε σε έναν κατάλογο
  - Συμφέρει τα αρχεία και οι κόμβοι i να είναι κοντά
  - Αξιοποίηση με τις ομάδες κυλίνδρων

Βελτιστοποίηση πιθανού (1 από 2)

- Πόσο πιθανό είναι κάθε αποτέλεσμα;
  - Αξίζει να βελτιστοποιήσουμε το πιο πιθανό
  - Τα ελάχιστα πιθανά δεν αξίζουν τον κόπο
- Κρίσιμες περιοχές σε Windows
  - Κλήση EnterCriticalSection σε Win32
    - Τις περισσότερες φορές πετυχαίνει άμεσα
    - Αρχικά κάνει έλεγχο σε επίπεδο χρήστη με TSL
    - Αν δεν πετύχει, κάνει down σε σηματοφόρο

Βελτιστοποίηση πιθανού (2 από 2)

- Χρονόμετρα σε UNIX
  - Κλήση alarm() για χρονόμετρο
  - Αν εκκρεμεί ήδη, ακυρώνεται το παλιό
    - Η κλήση δεν μας λέει αν εκκρεμεί
    - Πρέπει να ψάξουμε όλη τη λίστα χρονομέτρων
  - Τυπικά όμως δεν εκκρεμεί
    - Προσθέτουμε σημαία στον πίνακα διεργασιών
    - Σε κάθε alarm() θέτουμε τη σημαία
    - Αν είναι ενεργή, ψάχνουμε για το εκκρεμές

Διαχείριση έργων

Μάθημα: Λειτουργικά Συστήματα, **Ενότητα #11:** Σχεδίαση λειτουργικών συστημάτων

Διδάσκων: Γιώργος Ξυλωμένος, **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

Ο μυθικός ανθρωπομήνας (1 από 3)

- Κλασικό βιβλίο του Fred Brooks
  - Περιγράφει την εμπειρία υλοποίησης του OS/360
  - Δείχνει τη δυσκολία των έργων με ΛΣ
    - Πόσο κώδικα έγραφε ένας προγραμματιστής;
    - 1000 γραμμές το χρόνο (ελεγμένες, τεκμηριωμένες)
  - Τα μεγάλα έργα είναι εγγενώς δύσκολα
    - Προδιαγραφές, σχεδίαση, τμηματοποίηση
    - Έλεγχος κώδικα σε μονάδες και συνολικά
    - Απρόσμενες αλληλεπιδράσεις μεταξύ τους

Ο μυθικός ανθρωπομήνας (2 από 3)

- Κατανομή χρόνου υλοποίησης ΛΣ
  - 1/3 σχεδίαση
  - 1/6 κωδικοποίηση
  - 1/4 έλεγχος μονάδων
  - 1/4 έλεγχος συστήματος
  - Το κόστος δεν είναι η κωδικοποίηση!
- Ο χρόνος δεν είναι εναλλάξιμος με ανθρώπους
  - Έστω ότι ένα έργο θέλει 10 άτομα για 2 χρόνια
  - Δεν σημαίνει ότι γίνεται με 20 άτομα σε 1 χρόνο

Ο μυθικός ανθρωπομήνας (3 από 3)

- Γιατί δεν ισχύει η εναλλαξιμότητα;
  - Το έργο δεν γίνεται όλο παράλληλα
    - Δεν μπορούμε να γράψουμε κώδικα πριν τη σχεδίαση
  - Τα καθήκοντα δεν παραλληλίζονται πλήρως
    - Δεν μπορούμε να σπάσουμε τις ενότητες αυθαίρετα
  - Η εκσφαλμάτωση είναι βασικά σειριακή
    - Ο έλεγχος γίνεται με κάποια σειρά
- Η προσθήκη ανθρώπων, καθυστερεί το έργο

Δομή ομάδας (1 από 2)

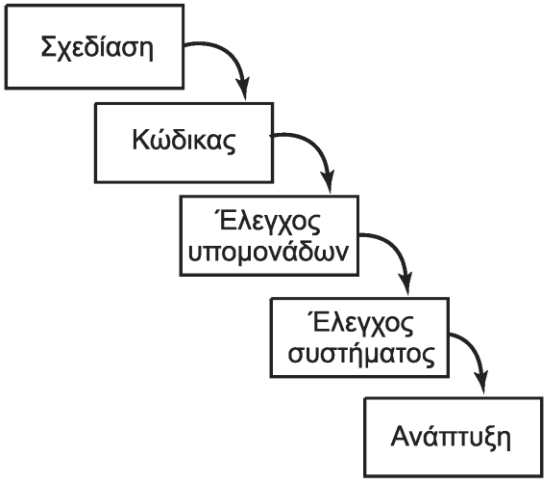
Τίτλος	Καθήκοντα
Αρχιπρογραμματιστής	Σχεδιάζει την αρχιτεκτονική και γράφει τον κώδικα
Συγκυβερνήτης	Βοηθάει τον αρχιπρογραμματιστή και δρα ως φερέφωνό του
Διαχειριστής	Διαχειρίζεται το ανθρώπινο δυναμικό, τους μισθούς, το χώρο, τον εξοπλισμό, τις αναφορές, κλπ.
Συντάκτης	Γράφει ή διορθώνει την τεκμηρίωση, η οποία μπορεί να έχει γραφεί από τον αρχιπρογραμματιστή
Γραμματείς	Ο διαχειριστής και ο συντάκτης χρειάζονται γραμματέα
Γραφέας προγράμματος	Συντηρεί τον κώδικα και τα αρχεία τεκμηρίωσης
Κατασκευαστής εργαλείων	Παρέχει στον αρχιπρογραμματιστή τα εργαλεία που χρειάζεται
Δοκιμαστής	Δοκιμάζει τον κώδικα του αρχιπρογραμματιστή
Σύμβουλος γλώσσας	Μερικώς απασχολούμενος που συμβουλεύει τον αρχιπρογραμματιστή σε θέματα γλώσσας προγραμματισμού

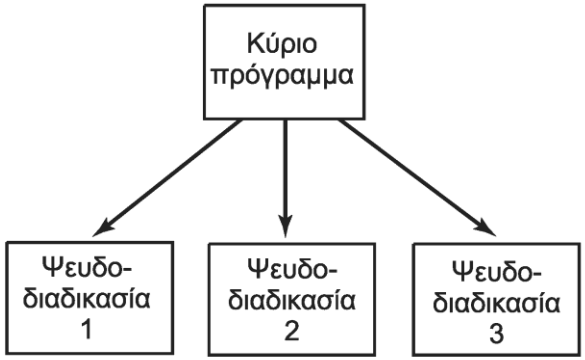
- Υπάρχουν λίγο εξαιρετικοί προγραμματιστές
  - Χρειαζόμαστε μια δομή που να τους υποστηρίζει
    - Ο αρχιπρογραμματιστής είναι ο ένας αρχιτέκτονας

Δομή ομάδας (2 από 2)

- Η δομή αυτή ισχύει ακόμη
  - Ίσως με κάπως μικρότερη ομάδα
    - Λόγω αξιοποίησης αυτοματοποιημένων εργαλείων
- Γενίκευση σε μεγάλα έργα
  - Οι ομάδες οργανώνονται ιεραρχικά
  - Ο προϊστάμενος έχει έως 10 υφιστάμενους
- Ισχύει και στο ανοιχτό λογισμικό
  - Ο Linus Torvalds ελέγχει τον πυρήνα του Linux
  - Μικρές ομάδες ελέγχουν τους πυρήνες xBSD

Ο ρόλος της πείρας (1 από 2)





- Τα περισσότερα λάθη είναι στη σχεδίαση
  - Και τα δυσκολότερα να εντοπιστούν
    - Οι προγραμματιστές υλοποιούν το λάθος πράγμα
    - Αλλά εντοπίζεται μόνο στον έλεγχο συστήματος
  - Προτείνεται υλοποίηση ενδιάμεσων συστημάτων

Ο ρόλος της πείρας (2 από 2)

- Το πρόβλημα του δεύτερου συστήματος
  - Ενίοτε το πρώτο σύστημα είναι καλό (CTSS)
    - Οι σχεδιαστές είναι συγκρατημένοι
    - Αποφεύγουν την πολυπλοκότητα
  - Το δεύτερο όμως αποτυγχάνει (MULTICS)
    - Η επιτυχία του πρώτου τους ενθαρρύνει
    - Βάζουν στο δεύτερο τα πάντα
  - Στο τρίτο έχουν μάθει το μάθημά τους (UNIX)

Όχι ασημένιες σφαίρες

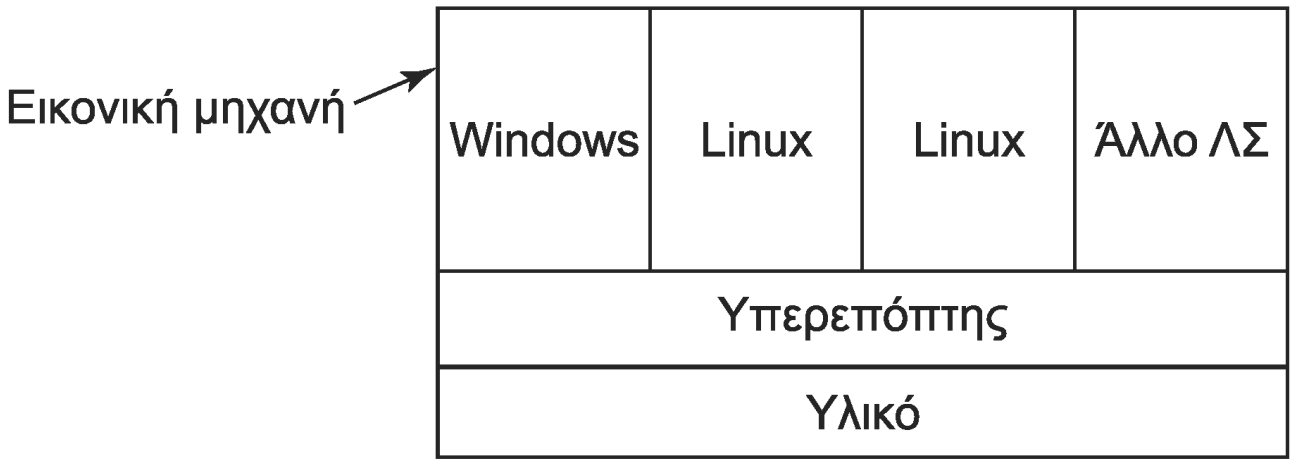
- Κλασικό άρθρο του Fred Brooks
  - Συνέχεια προτείνονται «λύσεις» για το λογισμικό
    - Γλώσσες υψηλότερου επιπέδου
    - Αντικειμενοστρεφής προγραμματισμός
    - Επαλήθευση λογισμικού
  - Καμία δεν έχει βελτιώσει θεαματικά τα πράγματα
    - Δεν είναι ασημένια σφαίρα που θα σκοτώσει το τέρας!
  - Οι βελτιώσεις είναι πολύ πιο σταδιακές
    - Αυτό δεν σημαίνει ότι δεν τις θέλουμε!

Σχεδιαστικές τάσεις

Μάθημα: Λειτουργικά Συστήματα, **Ενότητα #11:** Σχεδίαση λειτουργικών συστημάτων

Διδάσκων: Γιώργος Ξυλωμένος, **Τμήμα:** Πληροφορικής

Εικονικοποίηση



- Η εικονικοποίηση κυριαρχεί
  - Μεγάλη εξέλιξη σε υπερεπόπτες γυμνού υλικού
  - Τροποποίηση των ΛΣ ώστε να εικονικοποιούνται
  - Προσθήκη υλικού ειδικά για εικονικοποίηση

Πολυπύρηνοι επεξεργαστές

- Τα παραδοσιακά ΛΣ υποθέτουν έναν πυρήνα
  - Κλειδώνουν μεγάλα μέρη του ΛΣ
  - Δεν είναι επανεισαγόμενα σε μεγάλο βαθμό
  - Χρειάζεται πολύ δουλειά για να βελτιωθούν
- Αξιοποίηση πυρήνων για εικονικοποίηση
  - Ένα ΛΣ είναι δύσκολο να αξιοποιήσει 64 πυρήνες
  - Μπορούν να καταμεριστούν σε εικονικές μηχανές

Μεγάλοι χώροι διευθύνσεων

- Τι μπορεί να γίνει με διευθύνσεις 64 bit;
  - Κατάργηση του συστήματος αρχείων
    - Όλα τα δεδομένα είναι στην εικονική μνήμη
    - Η αποθήκευση είναι μόνο χώρος ανταλλαγής
  - Μόνιμη αποθήκη αντικειμένων
    - Διατήρηση όλων στην εικονική μνήμη
    - Διαγραφή όταν δεν αναφέρονται πια
  - Χρειάζονται όμως και νέες υλοποιήσεις
    - Όπως οι ανεστραμμένοι πίνακες σελίδων

Δικτύωση

- Η δικτύωση προστίθεται εκ των υστέρων
  - Σαφής διάκριση τοπικών/μη τοπικών δεδομένων
    - Ανάγκη χρήσης διαφορετικών πρωτοκόλλων
- Θα μπορούσαμε να έχουμε δικτυακά ΛΣ
  - Που δεν θα λειτουργούν χωρίς δίκτυο
  - Ενιαίος χειρισμός όλων των δεδομένων
  - Απευθείας πρόσβαση σε όλο τον ιστό
  - Οργάνωση των δεδομένων ως ιστοσελίδες

Παράλληλα και κατανεμημένα

- Παράλληλα συστήματα
  - Τα πολυπύρηνα συστήματα είναι παράλληλα
  - Ακόμη πιο σύνθετη η διαχείριση πολλών ΚΜΕ
  - Χρειάζεται δουλειά σε ΛΣ και εφαρμογές
- Κατανεμημένα συστήματα
  - Καταμερισμός έργου σε πολλές μηχανές
  - Απόκρυψη καθυστερήσεων και σφαλμάτων
  - Ενσωμάτωση κατανεμημένων μηχανισμών στο ΛΣ

Πολυμέσα

- Οι συσκευές πολυμέσων γίνονται έξυπνες
  - Τηλεοράσεις, media players
- Αλλά και οι υπολογιστές γίνονται πολυμεσικοί
  - Home theater PCs
- Και τα δύο θέλουν δουλειά
  - Διαχείριση τεράστιων όγκων δεδομένων
  - Μικτός χρονοπρογραμματισμούς
    - Πραγματικού και μη πραγματικού χρόνου
  - Βελτίωση αξιοπιστίας και ανθεκτικότητας

Υπολογιστές με μπαταρία

- ΛΣ για πραγματικά φορητές συσκευές
  - Android, iOS, Chrome OS
  - Παρόμοια αλλά όχι ίδια με Linux και OS X
  - Διαφορετικά μοντέλα λειτουργίας
    - Μία ενεργή εφαρμογή στην οθόνη
    - Γρήγορο πάγωμα και ξεπάγωμα εργασιών
    - Υποβάθμιση λειτουργίας για οικονομία
    - Λειτουργία με ή χωρίς δίκτυο

Ενσωματωμένα συστήματα

- Συστήματα με συγκεκριμένο σκοπό
  - Δεν επεκτείνονται με συσκευές
  - Δεν εκτελούν τυχαίες διεργασίες
  - Χαμηλό κόστος αλλά προβλέψιμη απόδοση
- Πώς θα διευκολυνθεί η συγγραφή τους;
  - Εξωπυρήνες: πολύ ελαφρά ΛΣ
  - Επεκτάσιμα συστήματα με βάση μικροπυρήνες

Κόμβοι αισθητήρων

- Εξαιρετικά απλά συστήματα
- Ανάγκη για μεγάλη διάρκεια μπαταρίας
- Χρησιμοποιούν πολύ απλά ΛΣ
  - Εξειδικευμένα ΛΣ αισθητήρων (Tiny OS)
- Σταδιακά θα περάσουν σε άλλα ΛΣ
  - Πιθανόν εξωπυρήνες και επεκτάσιμα ΛΣ
  - Ιδέες από ενσωματωμένα και φορητά ΛΣ

Τέλος Ενότητας # 11

Μάθημα: Λειτουργικά Συστήματα, **Ενότητα #11:** Σχεδίαση λειτουργικών συστημάτων

Διδάσκων: Γιώργος Ξυλωμένος, **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