Λειτουργικά Συστήματα Πραγματικού Χρόνου
2006-07

Λειτουργικά Συστήματα Πραγματικού Χρόνου

Βασικές Έννοιες

Μ.Στεφανιδάκης

Το μάθημα

- **ΛΣ Πραγματικού Χρόνου**
  - **Θεωρητικό υπόβαθρο**
  - **Αρχές Προγραμματισμού**
  - **Παραδείγματα ΛΣ**
- **Εργασίες**
  - **Ανά ομάδες**
  - **40% τελικού βαθμού**
- **Ύλη μαθήματος**
  - **Σημειώσεις παραδόσεων**
  - **Επιλεγμένη βιβλιογραφία**

Ενσωματωμένα Συστήματα
(embedded systems)

- **Αύξηση αριθμού ενσωματωμένων συστημάτων**
  - **>90% των παραγόμενων μικροεπεξεργαστών ανά έτος προορίζεται για ενσωματωμένα συστήματα**
- **Εφαρμογές ελέγχου**
- **Ειδικές απαιτήσεις**
  - **Χρονικές λειτουργίες**
  - **Αξιοπιστία**
- **Διαφορετικότητα**
  - **Στη σύνθεση των συστημάτων (υλικό)**
  - **Στη λειτουργικότητα (λογισμικό)**

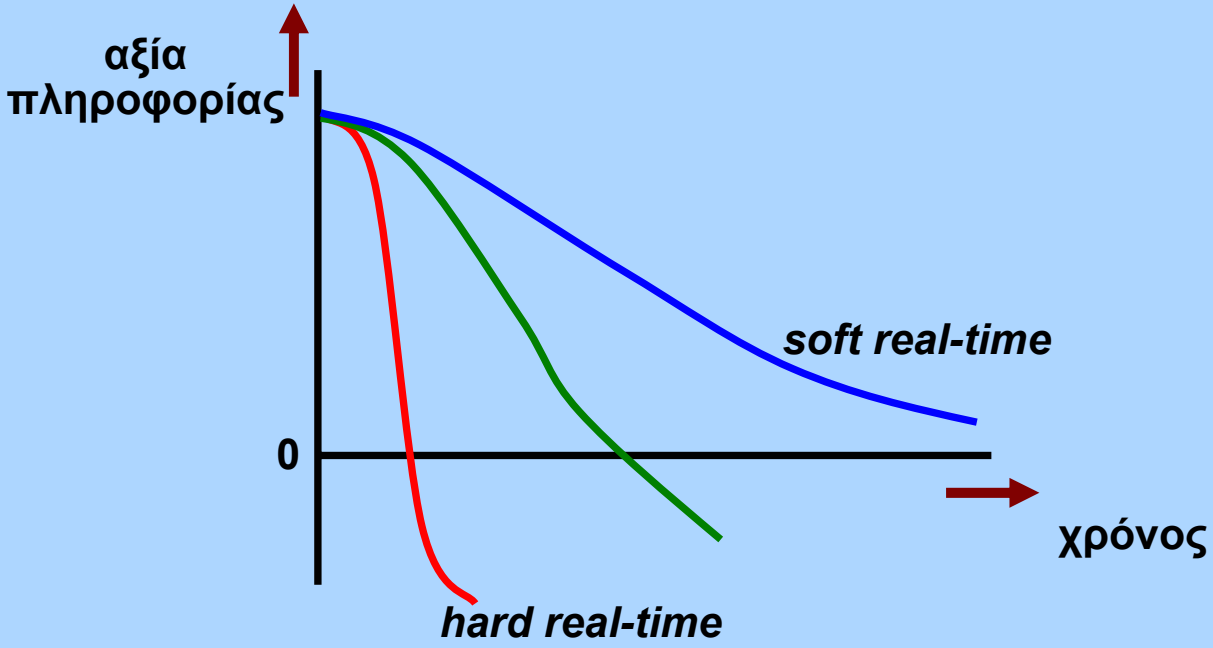
Παραδείγματα

- **Έλεγχος διεργασιών**
  - **επεξεργασία τροφίμων, εργοστάσια χημικών...**
- **Αυτοκίνητα**
  - **έλεγχος κινητήρα-φρένων...**
- **Μηχανήματα γραφείου**
  - **φαξ, φωτοτυπικά μηχανήματα...**
- **Περιφερειακά υπολογιστών**
  - **εκτυπωτές, τερματικά, μόντεμ...**
- **Ρομπότ**
- **Αεροπορία**
  - **οπλικά συστήματα, έλεγχος μάχης...**
- **Οικιακές συσκευές**
  - **φούρνοι μικροκυμάτων, πλυντήρια...**

ΛΣ Πραγματικού Χρόνου

- **Όπως ένα πολυδιεργασιακό ΛΣ γενικού σκοπού , με επιπλέον περιορισμούς:**
- **Χρόνος: *εγγυημένη χρονική απόκριση***
  - **Προσοχή: διαφορετικό από *γρήγορη απόκριση* !**
  - **Προθεσμίες (deadlines)**
    - ***Οι απαιτήσεις σχετικές των αναγκών της εφαρμογής!***
  - ***ΛΣΠΧ: η χρονική ορθότητα (timing correctness) είναι εξίσου σημαντική με (ίσως και σημαντικότερη από!) τη λειτουργική ορθότητα (functional correctness) του συστήματος***

Η αξία της πληροφορίας στον πραγματικό χρόνο



- **Hard - Soft real time**
  - **hard: η εκτέλεση *πρέπει* να τελειώσει εντός προθεσμίας**
  - **soft: η εκτέλεση *καλό θα ήταν* να τελειώσει εντός προθεσμίας**

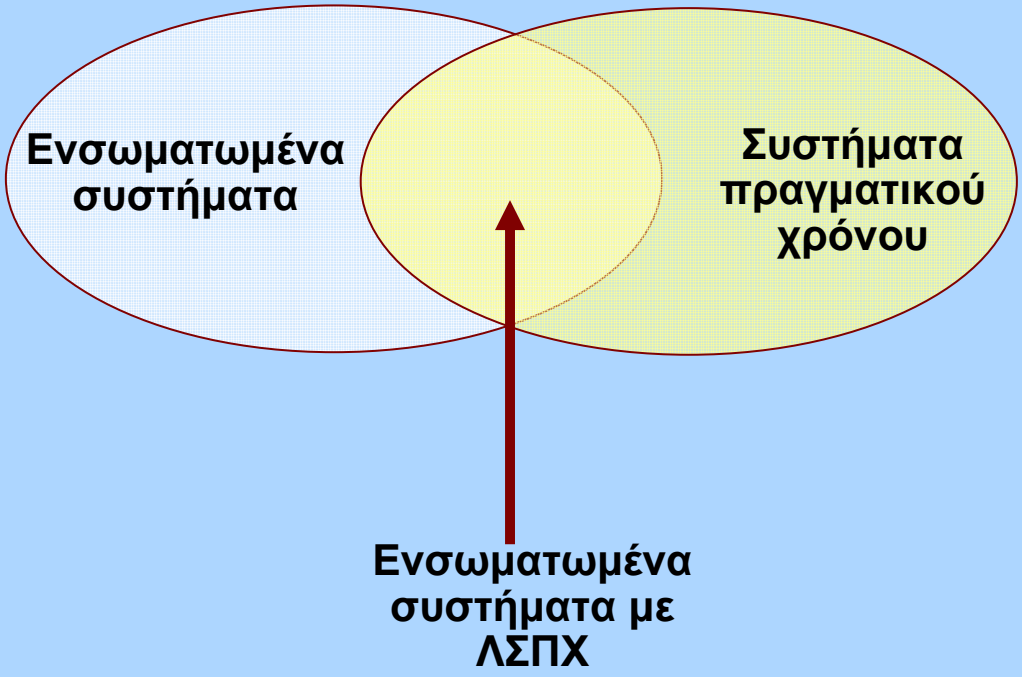
Άλλες Απαιτήσεις

- **Αξιοπιστία**
  - **Διαθεσιμότητα και αδιάλειπτη λειτουργία συστήματος**
    - **Συνδυασμός αξιοπιστίας όλων των τμημάτων του συστήματος, όχι μόνο του ΛΣΠΧ**
- **Προβλεψιμότητα**
  - **Πώς θα αντιδράσει το ΛΣΠΧ σε καταστάσεις σφάλματος**
    - **Εγγύηση εκτέλεσης διεργασιών με προκαθορισμένη σειρά**
- **Απόδοση**
  - **Ικανότητα τήρησης προθεσμιών**
    - **Για όλες τις εκτελούμενες διεργασίες**

Άλλες Απαιτήσεις

- **Περιεκτικότητα**
  - **Συμπαγή συστήματα για συσκευές με περιορισμένο υλικό**
    - **Π.χ. διαθέσιμο χώρο μνήμης**
- **Επεκτασιμότητα**
  - **Δυνατότητα προσαρμογής σε διάφορες πλατφόρμες εκτέλεσης**
    - **Δομημένη σχεδίαση (modules)**

ΛΣΠΧ και ενσωματωμένα συστήματα



Βασικά στοιχεία ΛΣ (και ΛΣΠΧ!)

(μια γρήγορη ανασκόπηση-επανάληψη)

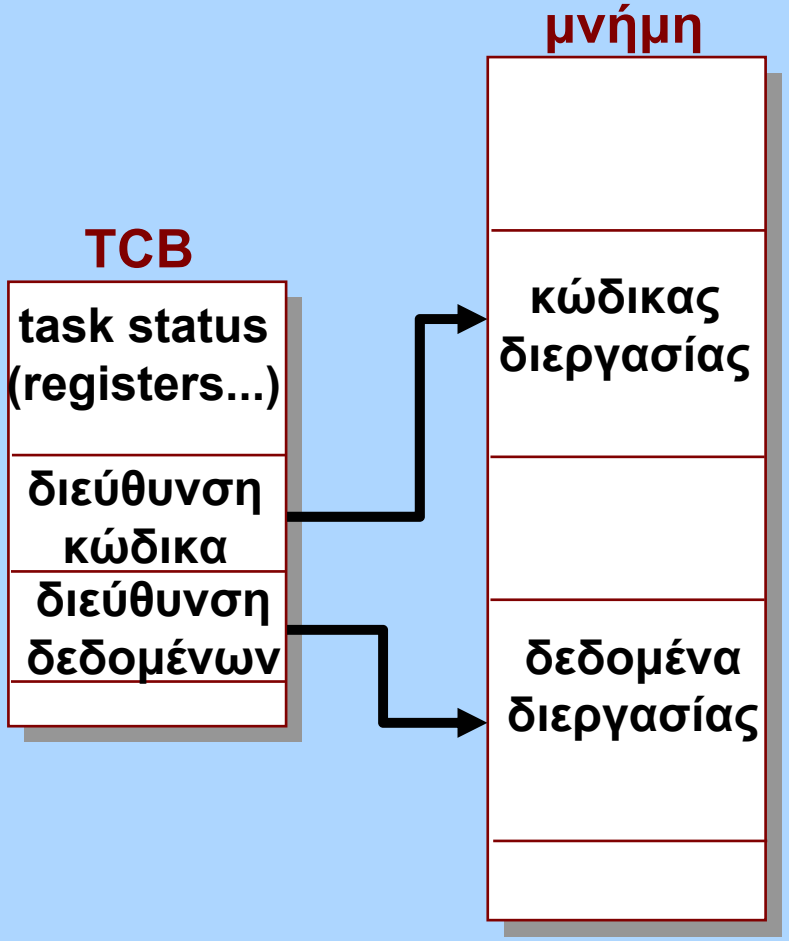
- **Απαιτήσεις από παραδοσιακά ΛΣ και από ΛΣΠΧ:**
- **Εκτέλεση Διεργασιών**
  - **Χρονοδρομολόγηση διεργασιών (scheduling)**
  - **Συγχρονισμός Διεργασιών**
  - **Εξυπηρέτηση αιτήσεων – επίπεδο αφαίρεσης για υλικό**
- **Διαχείριση πόρων συστήματος**
  - **Διαχείριση μνήμης**
  - **Συσκευές Εισόδου-Εξόδου**
  - **Εξυπηρέτηση διακοπών (interrupts)**

Πολυδιεργασία (multitasking)

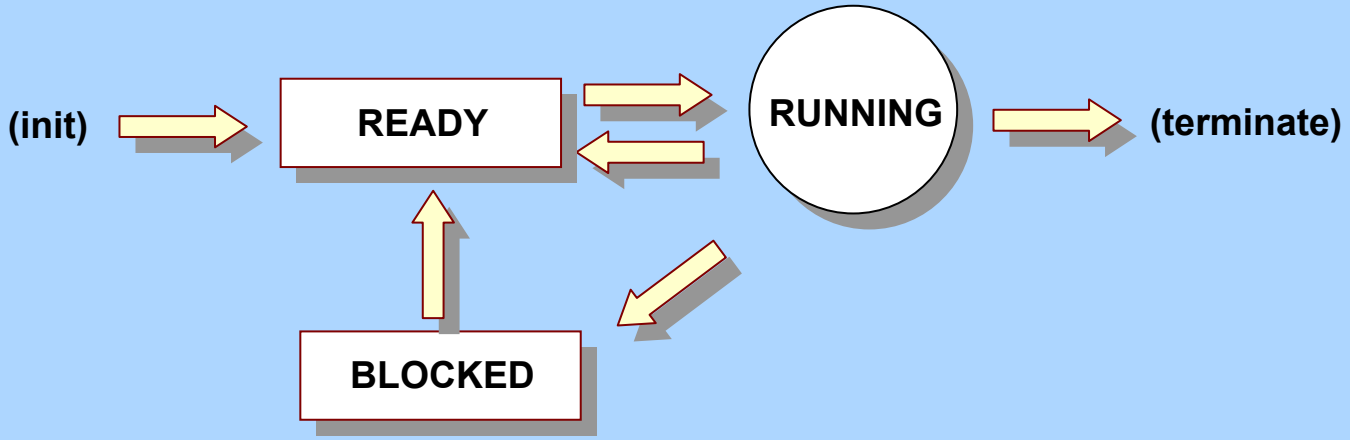
- **Περιοδική εκτέλεση περισσότερων από μία διεργασίες (tasks)**
  - **Με εναλλαγή μεταξύ τους (για μία CPU)**
  - **Μεγιστοποίηση χρήσης (utilization)**
    - **Π.χ. σε αναμονή για I/O**
  - **Δομημένη κατασκευή προγραμμάτων**
    - **Αντιμετώπιση πολυπλοκότητας εφαρμογών πραγματικού χρόνου**

Διεργασίες (tasks)

- Αυτόνομα τμήματα εκτέλεσης
- Περιβάλλον διεργασίας (context)
- Task Control Block (TCB)
- Προτεραιότητες



Καταστάσεις διεργασιών



- **Διαχείριση από ΛΣ**
  - **Ουρά εκτέλεσης (ready queue)**
  - **Blocked / waiting: αναμονή για event**
    - **Π.χ. διαθεσιμότητα πόρων**

Διεργασίες συστήματος

- **Συνήθως το ΛΣ δημιουργεί ειδικές διεργασίες:**
  - **Αρχικοποίησης**
    - **Για την εκκίνηση του συστήματος**
  - **Αδρανούς κατάστασης**
    - **Όταν δεν υπάρχει άλλη διεργασία έτοιμη προς εκτέλεση**
    - **Εκτός αν το σύστημα περνά σε κατάσταση χαμηλής κατανάλωσης ισχύος**
  - **Καταγραφής συμβάντων (logging)**
  - **Χειρισμού σφαλμάτων (exception handling)**
  - **Ειδικές διεργασίες debugging**

Context Switch

- ***Context Switch*: εναλλαγή εκτέλεσης διεργασιών**
  - **και του περιβάλλοντός τους**
- **Διαδικασία**
  - **Αποθήκευση του TCB της τρέχουσας διεργασίας**
    - **Δυναμική πληροφορία κατά την εκτέλεση**
  - **Επαναφορά TCB νέας διεργασίας προς εκτέλεση**
    - **Αποθηκευμένο από προηγούμενη περίοδο εκτέλεσης**
  - **Εκκίνηση νέας διεργασίας**
- **Χρονική επιβάρυνση**
  - **Αποφυγή συχνού context switching**

Χρονοδρομολόγηση (scheduling)

- ***Χρονοδρομολόγηση*: βασική λειτουργία του ΛΣ**
  - **ποια διεργασία θα εκτελεστεί στη συνέχεια;**
- **Εκτέλεση διεργασιών βάσει *προτεραιοτήτων***
  - **οι προτεραιότητες ορίζονται από εφαρμογή**
  - **προηγείται η διεργασία με την υψηλότερη προτεραιότητα**
- **Εναλλαγή διεργασιών**
  - **Context Switching: Εκτέλεση από dispatcher**
- **Πότε θα εκτελεστεί μια διεργασία;**
  - **προεκτοπιστικά και μη- ΛΣ**

Προεκτοπισμός (Preemption)

- **Μη προεκτοπιστικό (non-preemptive) ΛΣ**
  - **Κάθε διεργασία είναι υπεύθυνη για να παραχωρεί τον έλεγχο της CPU σε άλλες διεργασίες**
  - **Διακοπή μόνο από interrupt, επιστροφή στην ίδια διεργασία**
  - **Η παραχώρηση ελέγχου πρέπει να είναι συχνή**
  - **Απλούστερο ΛΣ (δομές και διαχείριση διεργασιών)**
- **Αλλά: μη εγγυημένος χρόνος εκτέλεσης**
  - **Όχι κατάλληλο για ΛΣΠΧ**

Προεκτοπισμός

- **Προεκτοπιστικό (preemptive) ΛΣ:**
  - **η διεργασία υψηλότερης προτεραιότητας, όταν είναι έτοιμη να εκτελεστεί, αποκτά τον έλεγχο της CPU.**
  - **Εγγυημένος χρόνος εκτέλεσης διεργασιών**
  - **Πολυπλοκότερα ΛΣ**
- **Απαιτείται συγχρονισμός διεργασιών για την προσπέλαση κοινών πόρων**
  - **Η εκτέλεση των διεργασιών μπορεί να διακοπεί ανά πάσα στιγμή!**

Κρίσιμες περιοχές (critical regions)

- Τμήμα κώδικα που πρέπει να εκτελεστεί ατομικά
  - Μόνο μία διεργασία εισέρχεται στην κρίσιμη περιοχή
  - Ενδεχομένως απενεργοποίηση interrupts
- Αμοιβαίος αποκλεισμός διεργασιών (mutual exclusion)
  - Προσπέλαση κοινών πόρων
- Υλοποίηση με δομές ελέγχου προσπέλασης
  - Διαχείριση από το λειτουργικό σύστημα
    - Π.χ. semaphores

Reentrancy

- **Reentrant κώδικας:** χρησιμοποίηση “ταυτόχρονα” από πολλές διεργασίες χωρίς κίνδυνο καταστροφής δεδομένων
  - Η εκτέλεση μπορεί να διακοπεί οποιαδήποτε στιγμή
  - Χρησιμοποιεί τοπικές μεταβλητές
- Non-reentrant κώδικας
  - Πρέπει να προστατεύεται από ταυτόχρονη προσπέλαση
  - Απενεργοποίηση interrupts

Deadlock

- **Παράδειγμα:**
  - **όταν δύο διεργασίες ζητούν αποκλειστική πρόσβαση η κάθε μία σε πόρους που κατέχει η άλλη**
- **Αποφυγή**
  - **Απόκτηση συνόλου πόρων πριν τη συνέχιση**
  - **Απόκτηση των πόρων στην ίδια σειρά**
  - **Ειδικά πρωτόκολλα αποφυγής deadlock**

Ανατροπή προτεραιοτήτων

- **Όταν διεργασία υψηλής προτεραιότητας αναγκάζεται να περιμένει διεργασία χαμηλότερης προτεραιότητας!**
- **Συμβαίνει όταν η διεργασία υψηλής προτεραιότητας:**
  - **περιμένει να απελευθερωθούν πόροι που χρησιμοποιεί μια διεργασία χαμηλής προτεραιότητας (semaphores)**
  - **βρίσκεται σε φάση δημιουργίας διεργασίας χαμηλής προτεραιότητας**
  - **βρίσκεται μπλοκαρισμένη σε επικοινωνία με διεργασία χαμηλής προτεραιότητας**
- **Πιθανή εκτέλεση εκτός προθεσμίας!**

Επικοινωνία μεταξύ διεργασιών

- Οι διεργασίες συγχρονίζονται ή ανταλλάσσουν δεδομένα (intertask communication)
  - Μέσω σφαιρικών μεταβλητών
    - Πρέπει να εξασφαλίζεται η αποκλειστική πρόσβαση
  - Μέσω δομών ελέγχου ("αντικειμένων") που παρέχει το ΛΣ
    - Semaphores
    - Messages
    - Event flags
    - …

Semaphores

- **Κρίσιμες περιοχές, αμοιβαίος αποκλεισμός**
- **Συγχρονισμός διεργασιών – σηματοδότηση συμβάντων**
- **Συναρτήσεις:**
  - **create, delete**
  - **wait (για πάντα ή με timeout, για αποφυγή deadlocks)**
  - **release**
- **Είδη semaphores:**
  - **binary**
  - **counting: δυνατότητα πολλαπλής προσπέλασης**

Messages

- **Ανταλλαγή δεδομένων μεταξύ διεργασιών**
  - **Ουρές μηνυμάτων**
- **Λειτουργίες**
  - **create, delete**
  - **sendmsg, getmsg (blocking / non-blocking)**
- **Άλλες Χρήσεις**
  - ***αντί για semaphores (συγχρονισμός)***
  - ***για ανάθεση πόρων***

Event Flags

- **Συγχρονισμός – Σηματοδότηση συμβάντων**
- **διαχείριση *event flags* από το ΛΣΠΧ αλλά: περιέχουν πληροφορία για την κατάσταση της εφαρμογής**
  - **create, delete**
  - **wait (για πάντα ή με timeout)**
  - **set (τώρα ή σε κάποιο χρόνο)**
  - **Η πληροφορία των *event flags* δεν καταναλώνεται**

Απόκριση σε Διακοπές (Interrupts)

- **Πρέπει να είναι μετρήσιμη**
  - **Αναγκαίο για εφαρμογές πραγματικού χρόνου**
- **Ρουτίνα Εξυπηρέτησης Διακοπής (ISR)**
  - **όσο πιο σύντομη γίνεται**
- **Αναγνώριση και εξυπηρέτηση διακοπών**
  - **Latency (CPU): μέγιστος χρόνος με απενεργοποιημένα interrupts (ΛΣΠΧ!) + χρόνος εκκίνησης ρουτίνας ISR (CPU)**
  - **Response (ΛΣΠΧ): latency + χρόνος αλλαγής context + χρόνος έναρξης εκτέλεσης επιθυμητής λειτουργίας**

Η Χρήση του Χρόνου

- **Clock Tick:** διακοπή που συμβαίνει περιοδικά
  - ο "παλμός" του ΛΣΠΧ (μs - ms)
  - συχνότερα = μεγαλύτερη επιβάρυνση
- Εκτέλεση λειτουργιών σε καθορισμένο χρονικό διάστημα
  - Χρονοδρομολόγηση
  - Timeouts - Delays
    - Παράδειγμα χρήσης: `sleep(10msec);`
- Επιτυγχάνεται ακριβής τήρηση των χρονικών διαστημάτων;