ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

Σημειώσεις

Τμήμα Πληροφορικής – Πανεπιστήμιο Πειραιώς
ΜΠΣ «ΠΛΗΡΟΦΟΡΙΚΗ»

Διδάσκοντες: Επ.Καθηγητής Π. Κοτζανικολάου
Δρ. Ι. Ανδρέου

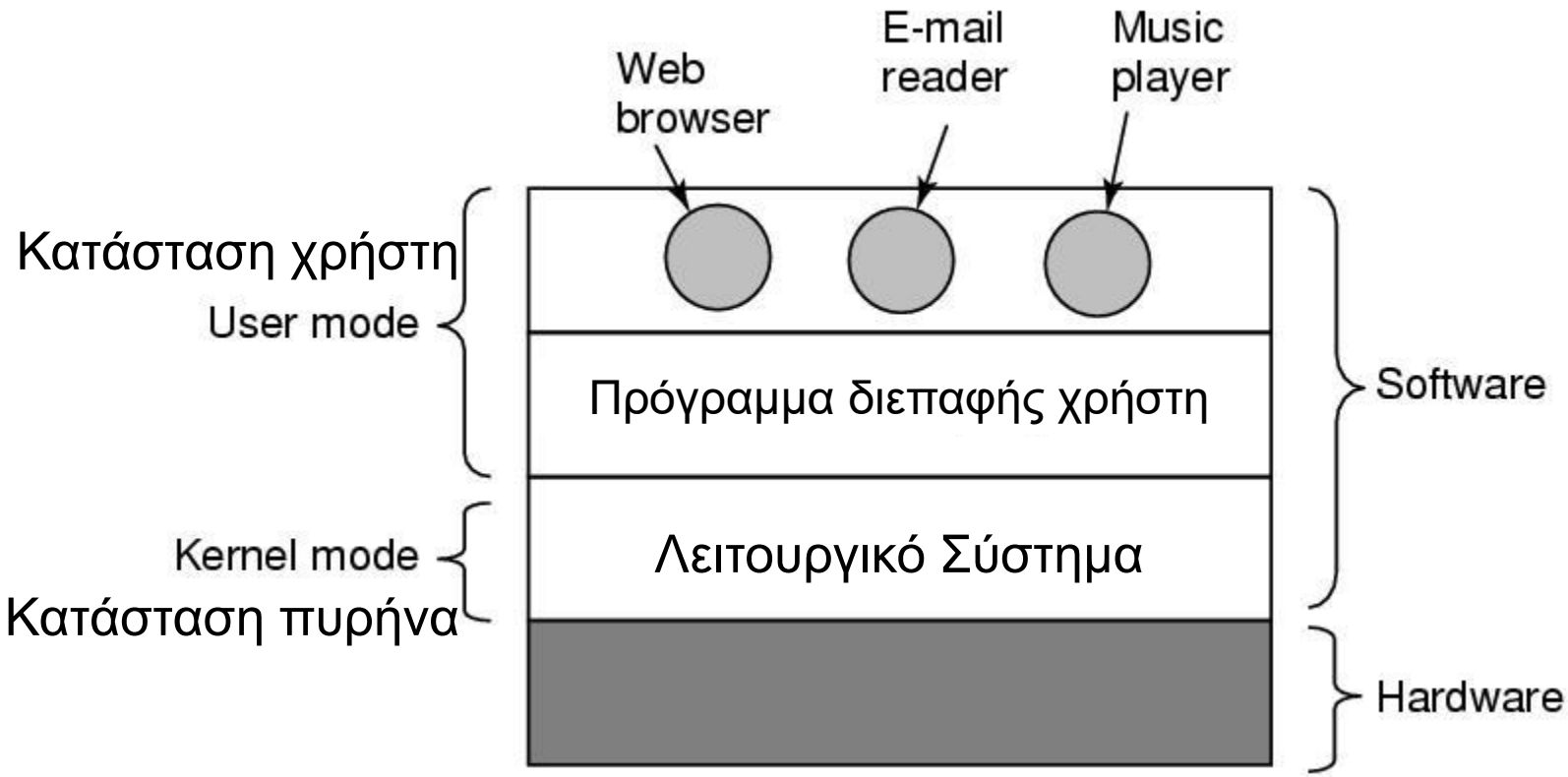
1η ενότητα

ΕΙΣΑΓΩΓΗ ΣΤΑ ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

Εισαγωγή στα ΛΣ

Περιεχόμενα

Εισαγωγή
Δομή υπολογιστικού συστήματος



Εικόνα 1-1. Που βρίσκεται το ΛΣ

1.1 Τι είναι το Λειτουργικό Σύστημα (ΛΣ)

Συστατικά Η/Υ

- Ένας σύγχρονος Η/Υ αποτελείται από:
  - Έναν ή περισσότερους επεξεργαστές
  - Κύρια μνήμη
  - Δίσκους
  - Εκτυπωτές
  - Διάφορες συσκευές εισόδου/εξόδου (Ε/Ε)

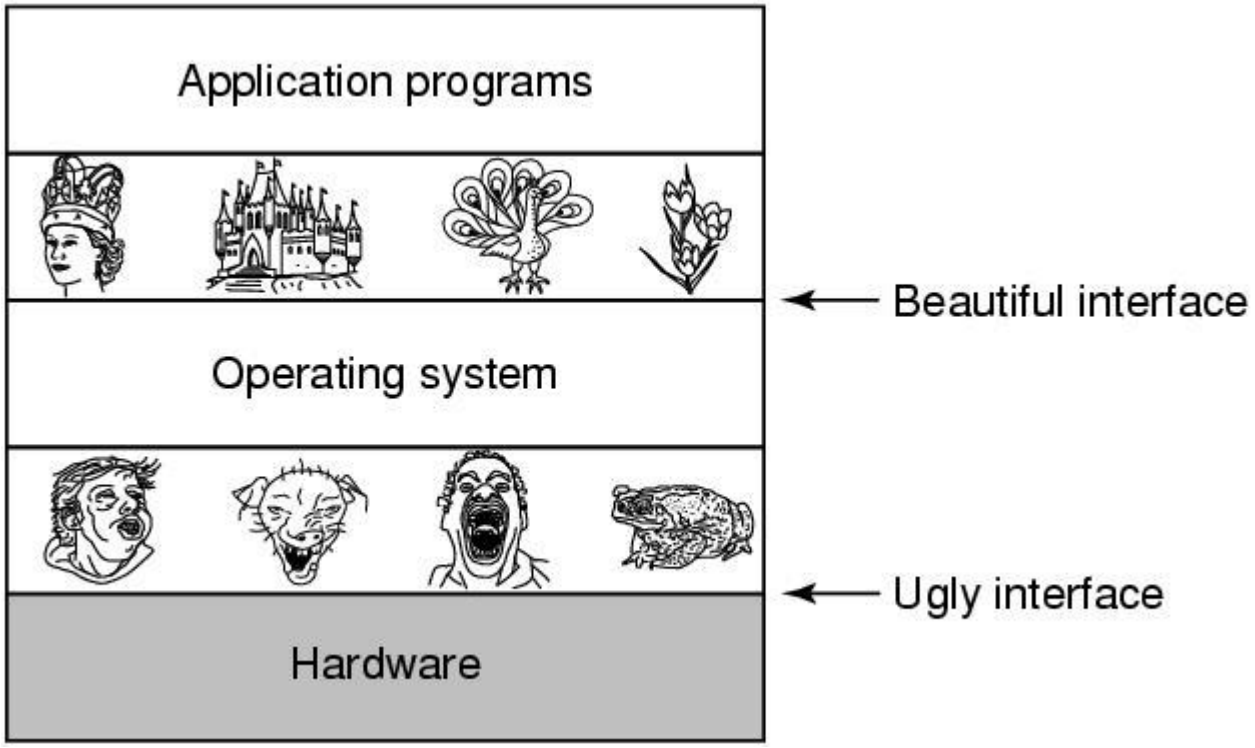
- Η διαχείριση αυτών των συστατικών, απαιτεί την ύπαρξη ενός στρώματος λογισμικού – του *Λειτουργικού Συστήματος*

Οι δύο όψεις των ΛΣ (1/2)

(Α) Μία εκτεταμένη μηχανή (extended machine)

- Η αρχιτεκτονική των Η/Υ είναι ιδιαίτερα σύνθετο θέμα (instruction set, διαχείριση μνήμης, Ε/Ε).
- Το ΛΣ **αποκρύπτει τις σύνθετες λεπτομέρειες** που εκτελεί ένας Η/Υ.
- Παρουσιάζει στον χρήστη μία ***«εικονική μηχανή»*** (virtual machine), πολύ πιο απλή στη χρήση της.
- Χρησιμοποιεί την έννοια της ***αφαίρεσης (abstraction)***
- Προσέγγιση top-down (από την κορυφή προς τη βάση).

(Α) Το ΛΣ ως μία εκτεταμένη μηχανή



Εικόνα 1-2. Το ΛΣ μετασχηματίζει το «άσχημο υλικό» σε «όμορφες αφαιρέσεις» / διεπαφές χρήστη.

Οι δύο όψεις των ΛΣ (2/2)

(Β) Ένας διαχειριστής πόρων

- Προσέγγιση bottom-up (από τη βάση προς την κορυφή).
- Το ΛΣ υπάρχει για να διαχειρίζεται τους πόρους του Η/Υ, (CPU, μνήμη, δίσκο κτλ).
- Ρόλος του ΛΣ: Η (δίκαιη) ***κατανομή των πόρων (resource allocation).***
  - ✓Τα προγράμματα των χρηστών **ανταγωνίζονται** να χρησιμοποιήσουν τους πόρους αυτούς.

(Β) Το ΛΣ ως ένας διαχειριστής πόρων

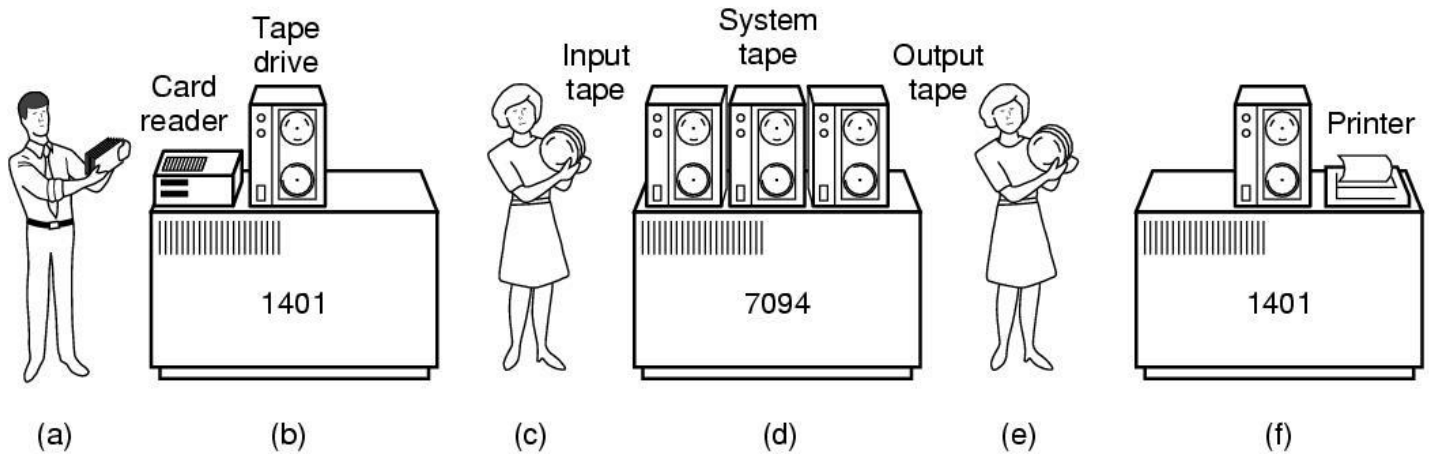
- Επιτρέπει την ταυτόχρονη εκτέλεση πολλών προγραμμάτων χρησιμοποιώντας **πολύπλεξη πόρων (multiplexing)** με δύο διαφορετικούς τρόπους:
  1. Χρονική πολύπλεξη (time multiplexing)
  2. Χωρική πολύπλεξη (space multiplexing)
- Η πολύπλεξη δημιουργεί θέματα ασφάλειας, προστασίας, αμεροληψίας κτλ.
  - Το ΛΣ διαχειρίζεται και προστατεύει τη μνήμη, τις συσκευές Εισόδου/Εξόδου (Ε/Ε) και άλλους πόρους.

1.2 Η ιστορία των ΛΣ

Ιστορία των ΛΣ

- **1η γενιά 1945 - 1955**
  - Λυχνίες κενού, καλωδίωση συνδέσεων
- **2η γενιά 1955 - 1965**
  - Κυκλώματα (transistors), Συστήματα δέσμης (batch systems)
- **3η γενιά 1965 – 1980**
  - Ολοκληρωμένα κυκλώματα (Integrated Circuits) και πολύ-προγραμματισμός (multiprogramming)
- **4η γενιά 1980 – σήμερα**
  - Προσωπικοί υπολογιστές

2η γενιά
Κυκλώματα και Συστήματα δέσμης



Εικόνα 1-3. Ένα πρώιμο σύστημα δέσμης.

(a) Ο προγραμματιστής φέρνει την κάρτα με το πρόγραμμα στο 1401.

(b) (b)Το 1401 περνάει τις δέσμες εργασιών στην ταινία εισόδου.

(c) Ο χειριστής φέρνει την ταινία εισόδου στο 7094.

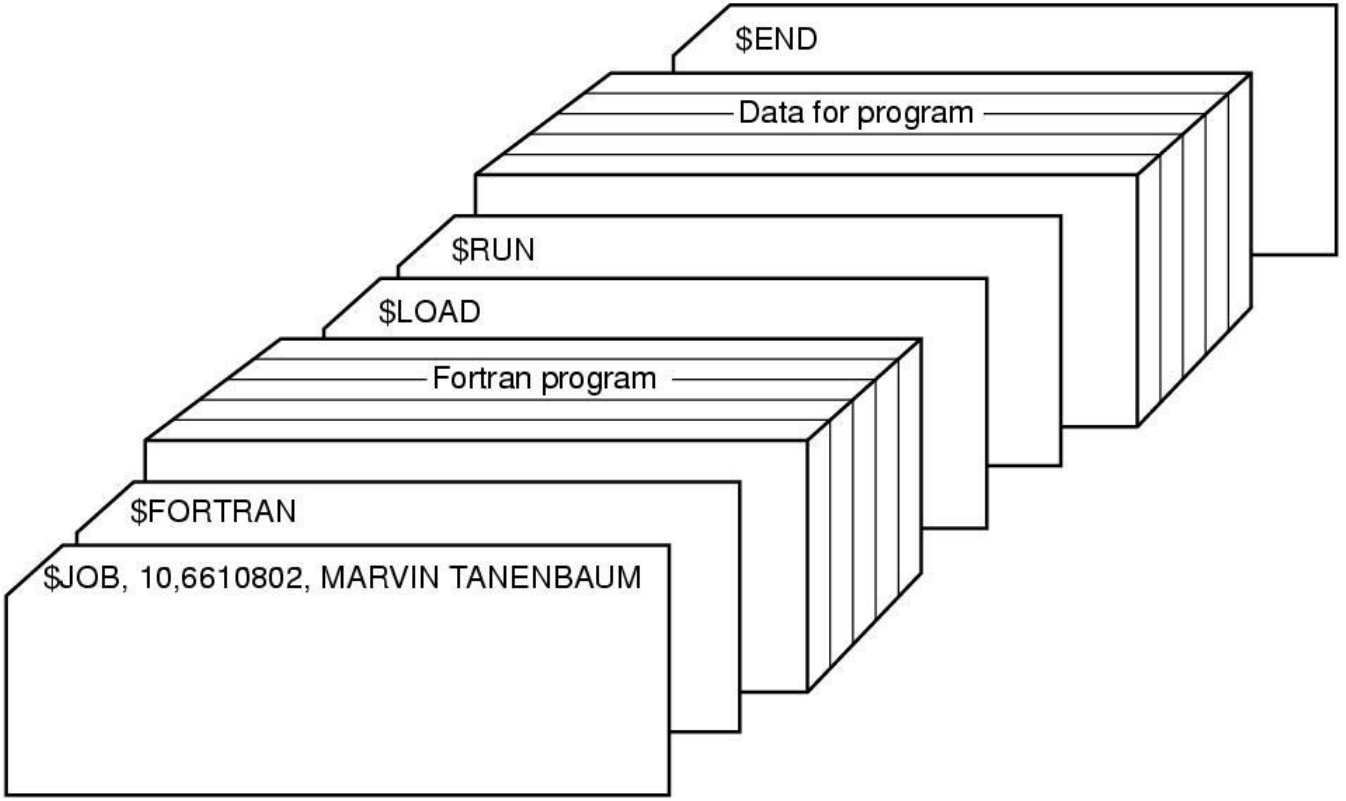
(d) Το 7094 κάνει τους υπολογισμούς.

(e) Ο χειριστής φέρνει την ταινία εξόδου στο 1401.

(f) Το 1401 τυπώνει τα αποτελέσματα εξόδου

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

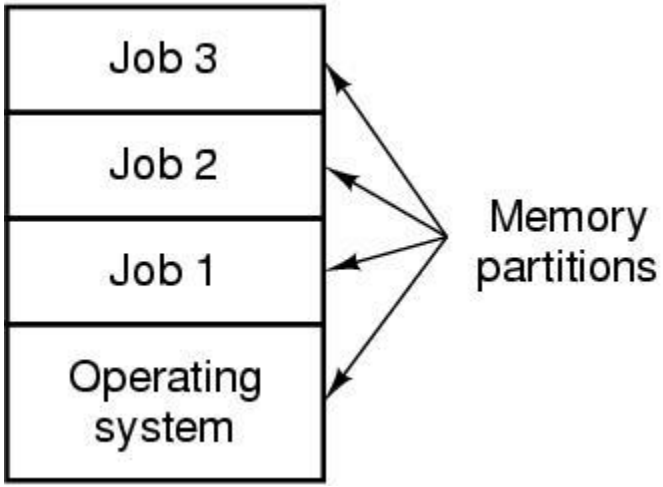
2η γενιά
Κυκλώματα και Συστήματα δέσμης(2)



Εικόνα 1-4. Δομή μιας τυπικής εργασίας FMS (Fortran Monitor System).

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

3η γενιά: Ολοκληρωμένα κυκλώματα και πολυπρογραμματισμός (1/2)



Εικόνα 1-5. Ένα σύστημα πολύπρογραμματισμού (multiprogramming) με τρεις εργασίες στη μνήμη.

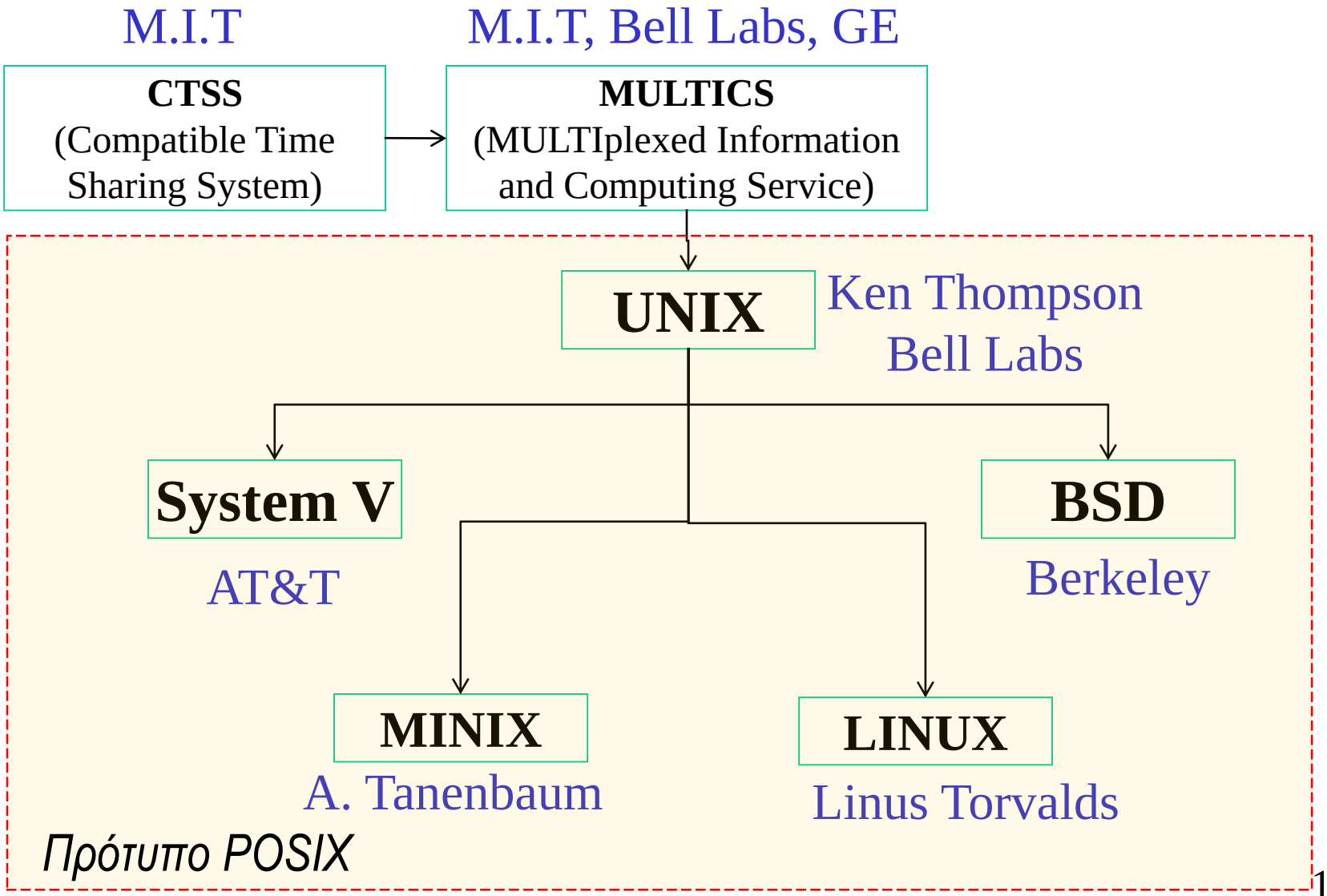
Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

3η γενιά: Ολοκληρωμένα κυκλώματα και πολυπρογραμματισμός (2/2)

- Άλλες καινοτομίες των ΛΣ 3ης γενιάς.
  - Παροχέτευση (spooling):
    - Οι εργασίες διαβάζονται από το δίσκο (πολλές στη σειρά). Με την ολοκλήρωση μίας εργασίας, φόρτωση της επόμενης σε ένα ελεύθερο τμήμα της μνήμης.
    - Πρόβλημα: μεγάλος χρόνο απόκρισης (λόγω της σειριακής εκτέλεσης).
  - Χρονομερισμός (time-sharing):
    - Κάθε χρήστης έχει για ένα μικρό διάστημα τη CPU.
    - Προτεραιότητα στις αλληλεπιδραστικές εργασίες των χρηστών. Οι βαριές εργασίες δέσμης τρέχουν στο παρασκήνιο, όταν η CPU είναι ελεύθερη.

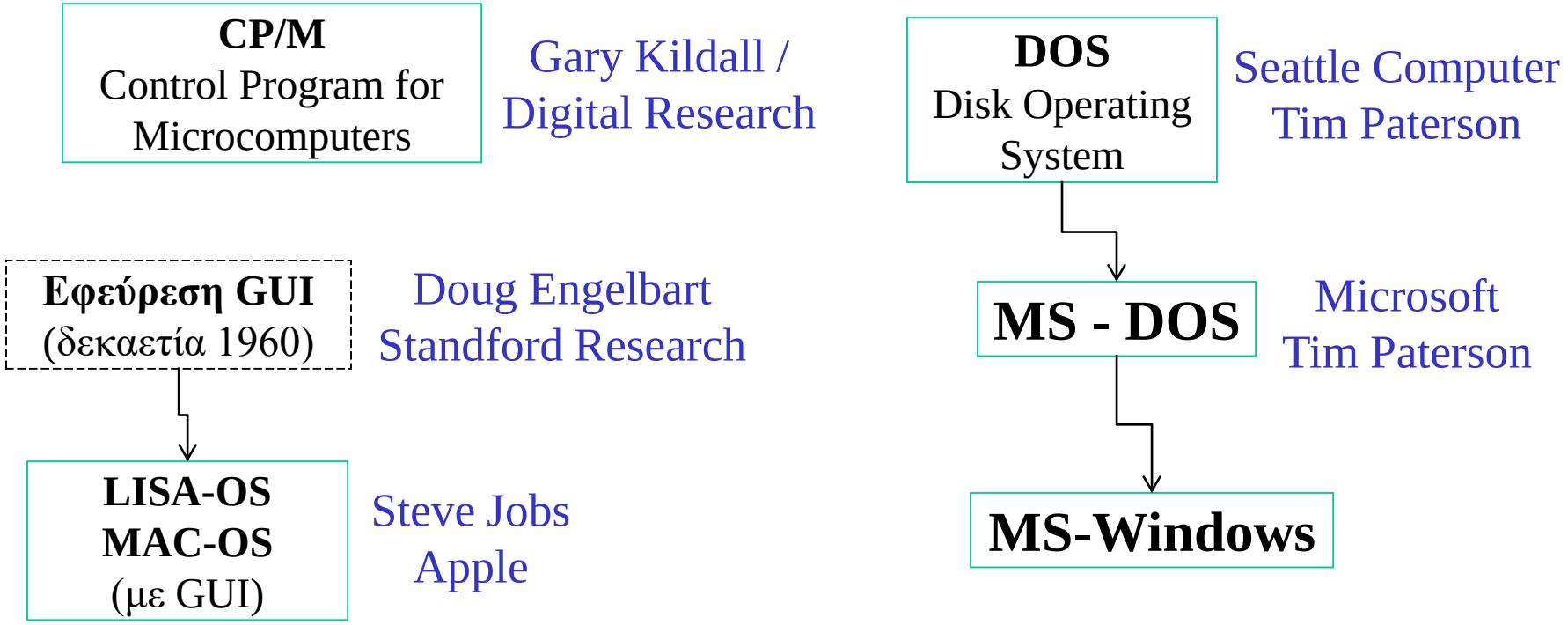
Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Γενεαλογικό δένδρο ΛΣ



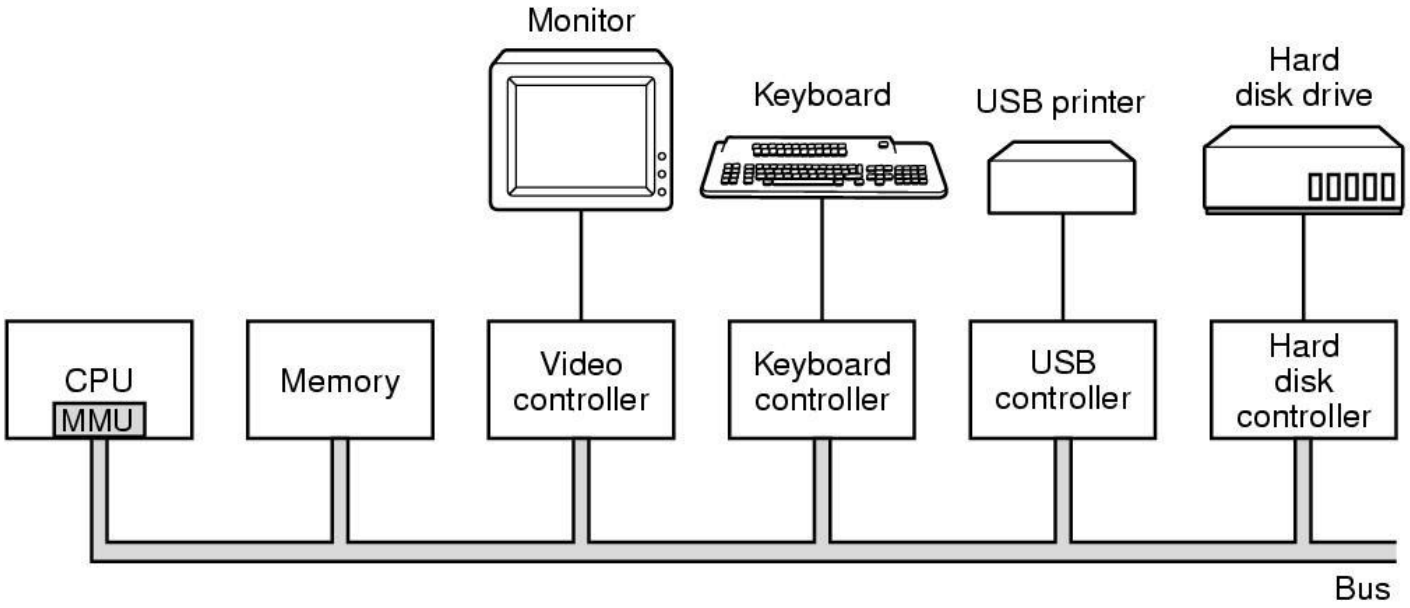
4η γενιά: Προσωπικοί υπολογιστές

Η ανάπτυξη κυκλωμάτων LSI (Large Scale Integation) οδήγησε στην δημιουργία των μικροϋπολογιστών



1.3 Το υλικό των Η/Υ

Το υλικό ενός σύγχρονου Η/Υ

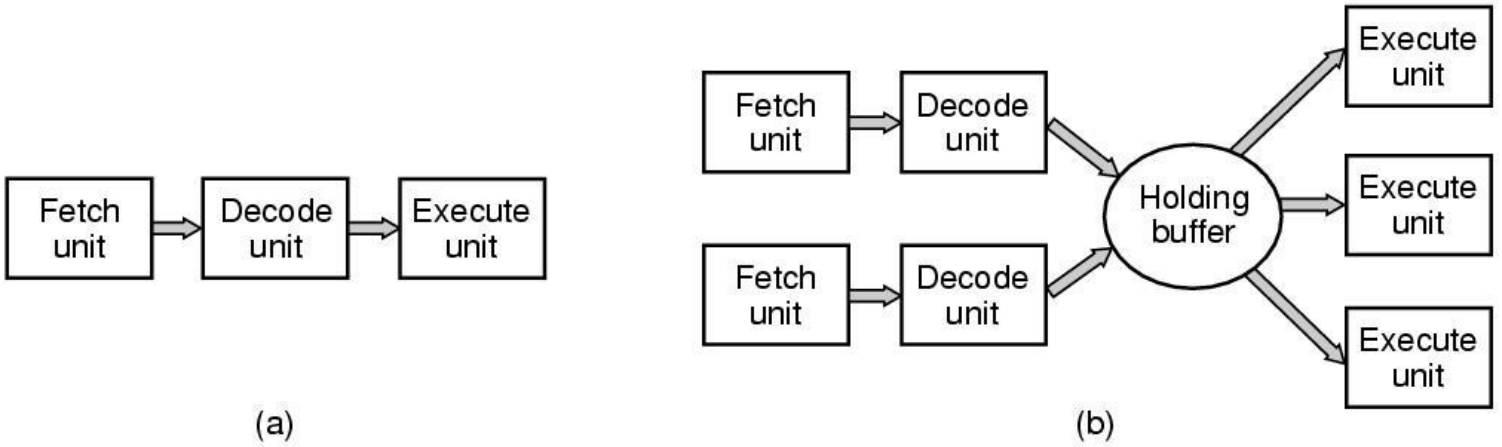


Εικόνα 1-6. Μια απλή αναπαράσταση των συστατικών ενός βασικού συστήματος Η/Υ.

Κεντρική μονάδα επεξεργασίας (CPU)

- Κύκλος λειτουργίας CPU: μεταφορά της επόμενης εντολής από τη μνήμη, την αποκωδικοποίηση του τύπου της, και την εκτέλεση.
- Η μεταφορά διαρκεί περισσότερο, γι' αυτό χρησιμοποιεί καταχωρητές (**registers**):
  1. Μετρητής προγράμματος (program counter): δείχνει τη διεύθυνση της επόμενης εντολής.
  2. Δείκτης στοίβας (stack pointer): δείχνει την κορυφή της τρέχουσας στοίβας στη μνήμη.
  3. Πλαίσιο στοίβας (stack frame): έχει τις παραμέτρους εισόδου, τοπικές και προσωρινές μεταβλητές που δεν διατηρούνται σε καταχωρητές.
  4. (Program Status Word) – PSW: δείχνει την κατάσταση λειτουργίας (πυρήνα ή χρήστη), την προτεραιότητα στη CPU, bit ελέγχου κτλ.

Διοχέτευση επεξεργαστή (CPU Pipelining)



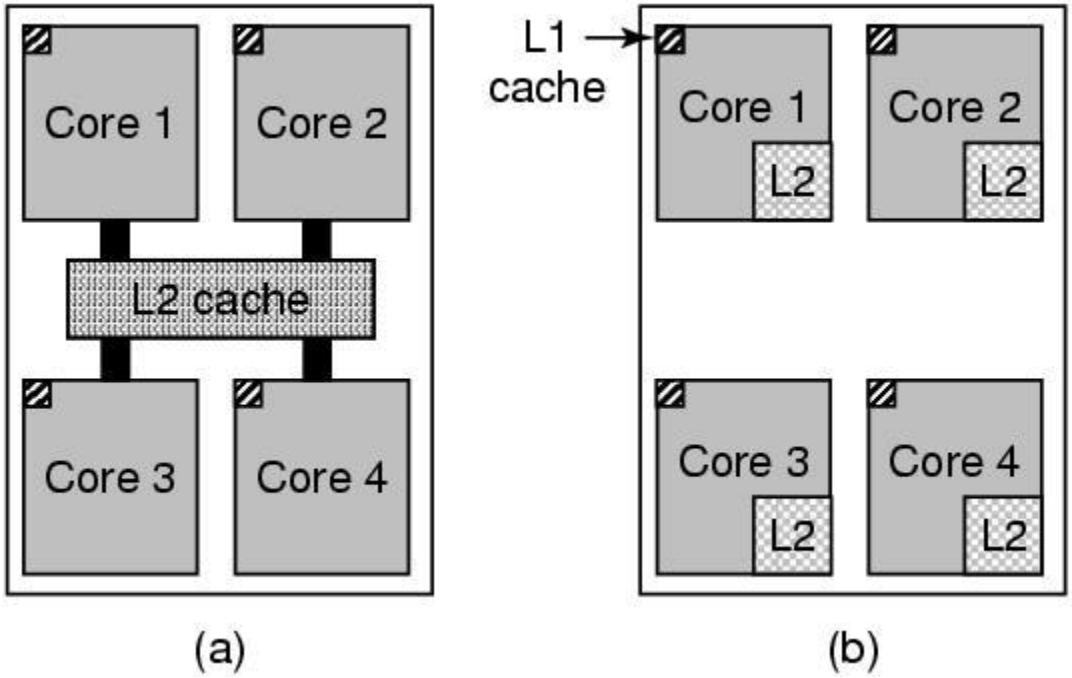
Εικόνα1-7. (a) Μία σωλήνωση τριών σταδίων. (b) Υπερβαθμωτή CPU.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

CPU και κατάσταση πυρήνα

- Η CPU έχει **κατάσταση πυρήνα** και **κατάσταση χρήστη.** Η κατάσταση εκτέλεσης κάθε εντολής ελέγχεται από bit ελέγχου στον καταχωρητή PSW.
- Στην κατάσταση πυρήνα εκτελούνται όλες οι εντολές, σε κατάσταση χρήστη μόνο ορισμένες.
- Το ΛΣ εκτελείται σε κατάσταση πυρήνα.
- Τα προγράμματα χρήστη που θέλουν ειδικές λειτουργίες πραγματοποιούν **κλήσεις συστήματος (system calls)**, η οποία προκαλεί **παγίδευση (trap)** και μεταβαίνει σε κατάσταση πυρήνα.
- Η εντολή TRAP προκαλεί την ανάμιξη του ΛΣ.

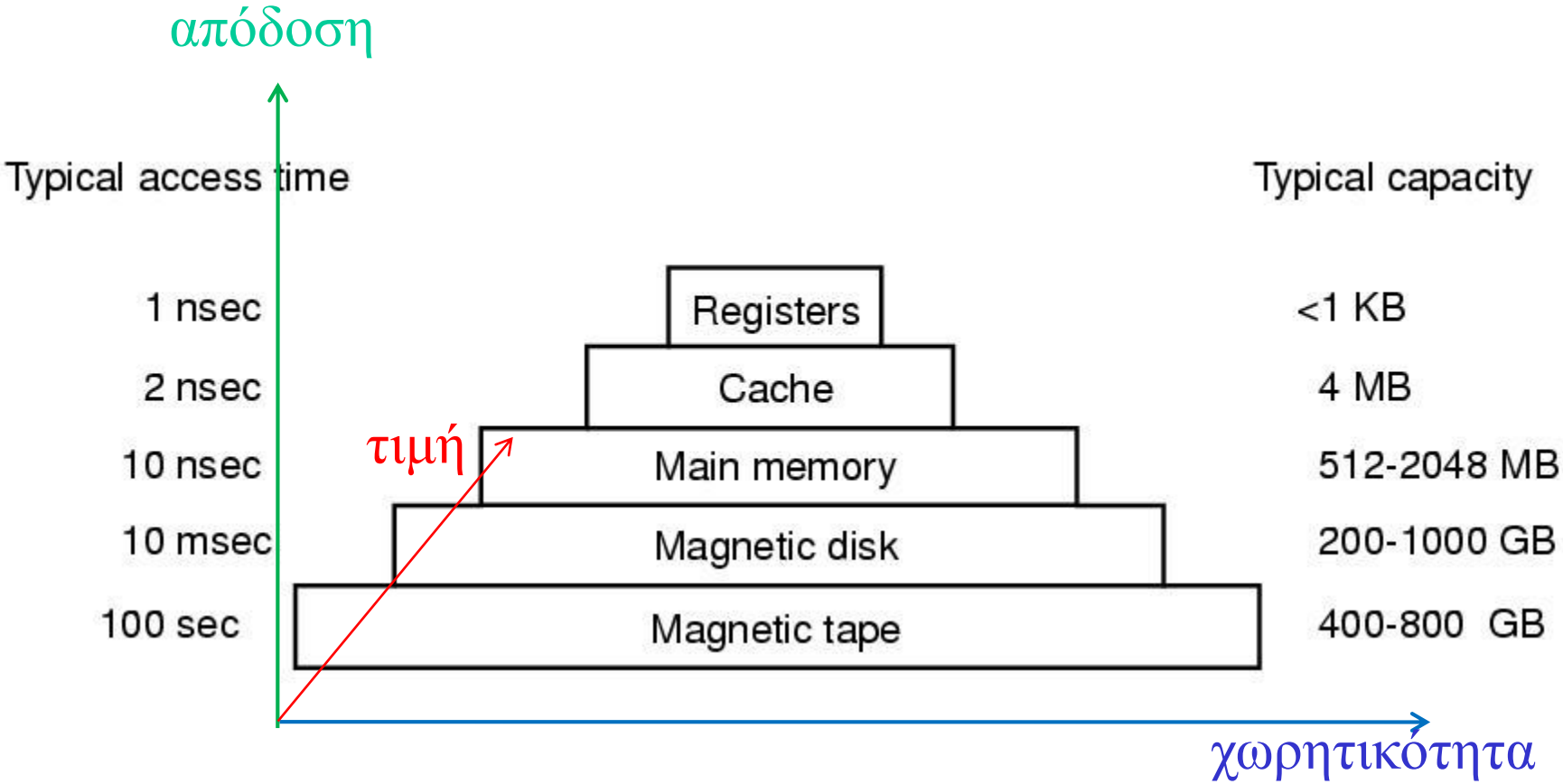
Επεξεργαστές πολλαπλών νημάτων (Multithread) και πολλαπλών πυρήνων (Multicore)



Εικόνα 1-8. (a) Chip τεσσάρων πυρήνων με διαμοιραζόμενη μνήμη cache L2.
(b) Chip τεσσάρων πυρήνων με ξεχωριστή μνήμη cache L2.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Μνήμη
(Ιεραρχία)



Εικόνα 1-9. Μία τυπική ιεραρχία της μνήμης.
Οι αριθμοί αποτελούν πολύ γενικές προσεγγίσεις.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Μνήμη
(Καταχωρητές)

1. Καταχωρητές (registers)

- Κορυφαίο επίπεδο μνήμης, πολύ υψηλό κόστος.
- Βρίσκονται στο εσωτερικό της CPU, άρα προκαλούν ελάχιστες καθυστερήσεις.
- Μέγεθος: (< 1KB) π.χ.,
  - 32 × 32 bit ή 64 × 64 bit.
- Τα προγράμματα αποφασίζουν ποια δεδομένα θα διατηρούνται εκεί.

Μνήμη
(Κρυφή μνήμη)

2. Κρυφή μνήμη (Cache):

- Δεύτερο επίπεδο, ελέγχεται από το υλικό.
- Χωρίζεται σε **γραμμές κρυφής μνήμης (cache lines)** των 64 byte.
- Όσες γραμμές χρησιμοποιούνται συχνότερα, διατηρούνται στην ταχύτερη κρυφή μνήμη που βρίσκεται **μέσα στην CPU.**
- Όταν το πρόγραμμα βρίσκει τα δεδομένα που ψάχνει μέσα στην μνήμη αυτή της CPU, τότε έχουμε ***ευστοχία μνήμης (cache hit).***
- Οι ευστοχίες διαρκούν κατά Μ.Ο. 2 κύκλους ρολογιού, μετά γίνεται αναζήτηση της ζητούμενης λέξης στη μνήμη με μεγάλη καθυστέρηση.

Μνήμη
(κρυφή μνήμη -επίπεδα)

Επίπεδα κρυφής μνήμης (cache memory layers):

L1 cache memory (συνήθως 16 KB):

- Βρίσκεται πάντα στο εσωτερικό της CPU και τροφοδοτεί τη λογική μονάδα εκτέλεσης με αποκωδικοποιημένες εντολές.
- Πολλά chip έχουν και δεύτερη L1 μνήμη για τις συχνότερα χρησιμοποιούμενες λέξεις δεδομένων.

L2 cache memory (μερικά MB)

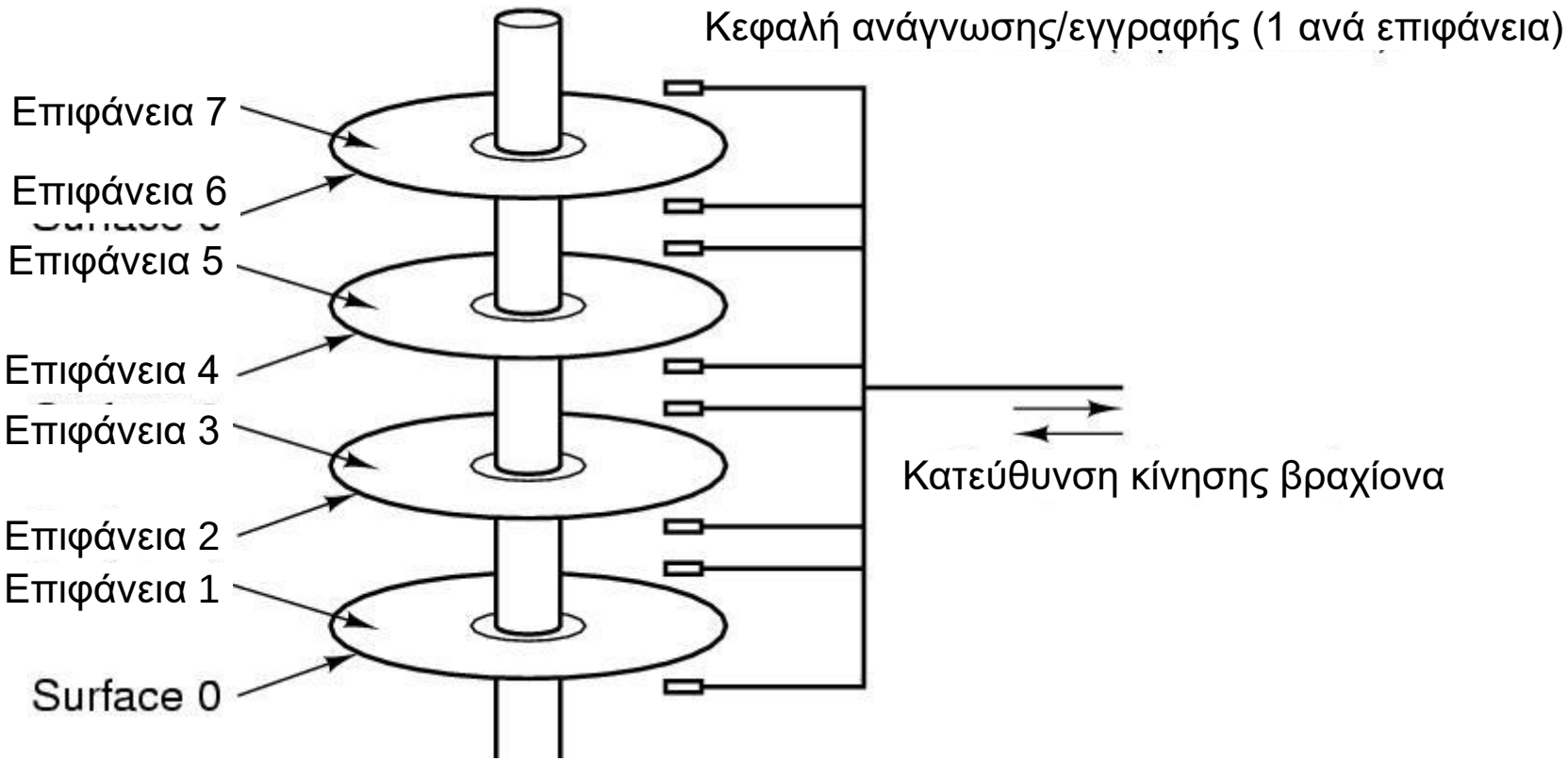
- Υπάρχει συχνά, είναι αρκετά μεγαλύτερη και διατηρεί τις πρόσφατα χρησιμοποιούμενες από τη CPU λέξεις.
- Η L2 έχει μεγαλύτερη καθυστέρηση από την L1 (π.χ. 1-2 κύκλους ρολογιού).

Μνήμη
(Κύρια Μνήμη)

3. Μνήμη Τυχαίας Προσπέλασης (Random Access Memory – RAM)

- Ο εργάτης του συστήματος.
- Όταν μία αίτηση της CPU δεν μπορεί να ικανοποιηθεί από την κρυφή μνήμη, τότε μεταφέρεται στην κύρια μνήμη.
- Η RAM είναι πτητική μνήμη (τα δεδομένα χάνονται όταν κλείσει το ρεύμα).
- Η **μνήμη ROM (Read Only Memory) είναι μη πτητική.**
  - Διατηρεί πληροφορία όπως τον boot loader.
- Η μνήμες EEPROM (Electrically Erasable Programmable ROM) μπορούν να ξαναγραφούν.

Δίσκοι



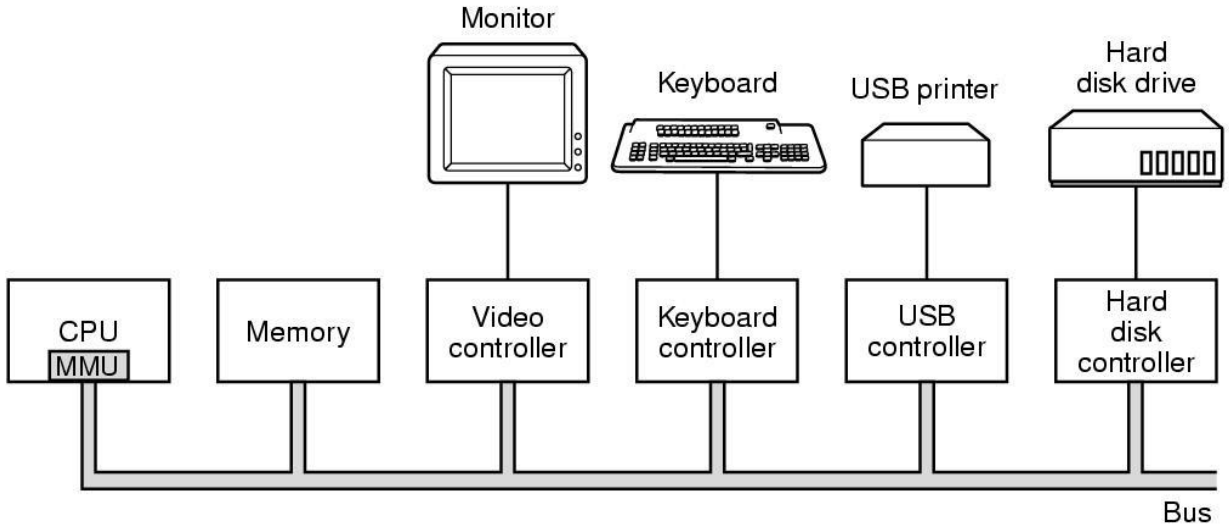
Εικόνα 1-10. Δομή ενός οδηγού δίσκου.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Συσκευές Εισόδου/Εξόδου (Ε/Ε)
(Ελεγκτής συσκευής)

Ελεγκτής συσκευής (controller): chip για το φυσικό έλεγχο της συσκευής.

- Ο ελεγκτής παρουσιάζει στο ΛΣ μία λιγότερο (αλλά και πάλι αρκετά) σύνθετη εικόνα.
- Το ΛΣ βλέπει **μόνο τη διασύνδεση με τον ελεγκτή**, και όχι τη διασύνδεση μεταξύ του ελεγκτή και της φυσικής συσκευής. (Π.χ. IDE controller, SCSI controller κ.ο.κ.)

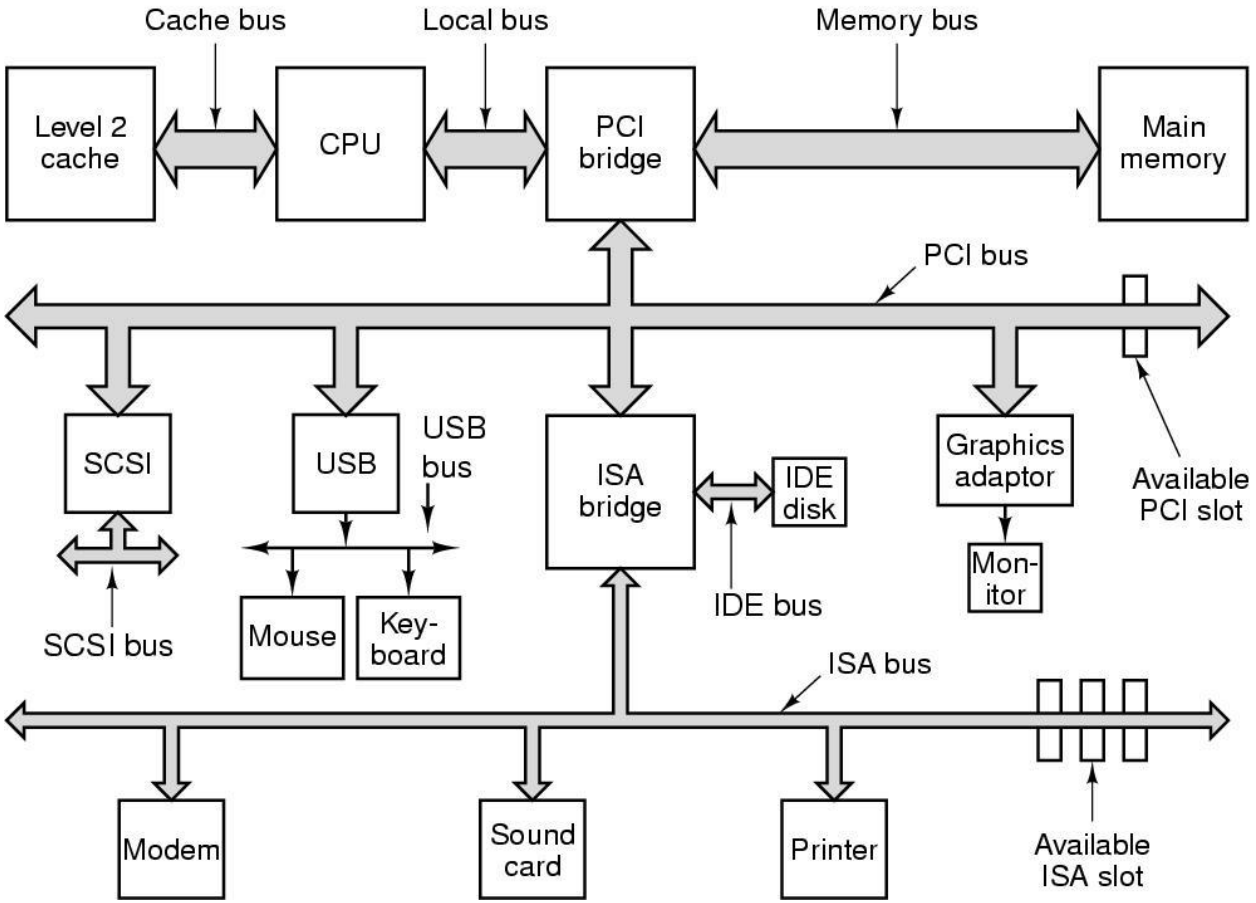


Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Συσκευές Εισόδου/Εξόδου (Ε/Ε)
(Οδηγός συσκευής)

- Η διασύνδεση του ΛΣ με τον ελεγκτή γίνεται μέσα από τον **οδηγό συσκευής (device driver)**.
- Τοποθέτηση driver στον πυρήνα του ΛΣ: (3 μέθοδοι)
  1. Επανασύνδεση του πυρήνα με το νέο οδηγό και επανεκκίνηση (UNIX).
  2. Δημιουργία μίας καταχώρησης για τον οδηγό της συσκευής σε κάποιο ειδικό αρχείο και επανεκκίνηση (Windows).
  3. Δυναμική φόρτωση (on-the-fly).
- Κάθε ελεγκτής συσκευής έχει έναν αριθμό καταχωρητών για να επικοινωνεί με τον έξω κόσμο.
  - Το σύνολο των καταχωρητών μίας συσκευής Ε/Ε αποτελεί το **χώρο θυρών Ε/Ε (I/O port space)**

Δίαυλοι (Buses)



Εικόνα 1-12. Η δομή ενός συστήματος Pentium

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Εκκίνηση του Η/Υ

1. Το πρόγραμμα BIOS ελέγχει τις συσκευές, τη RAM, τους διαύλους, τις συσκευές άμεσης λειτουργίας.
   - Το BIOS (Basic Input Output System) βρίσκεται επάνω στο motherboard και υποστηρίζει λειτουργίες Ε/Ε χαμηλού επιπέδου.
2. Το BIOS αποφασίζει ποια είναι η συσκευή εκκίνησης (σκληρός δίσκος, CD, flash κτλ).
3. Από τον τομέα εκκίνησης (boot sector) της συσκευής εκκίνησης, βρίσκεται ποιο είναι το ενεργό partition.
4. Φορτώνεται ο boot loader και από εκεί το ΛΣ του ενεργού partition.
5. Το ΛΣ βρίσκει από το BIOS τους οδηγούς των συσκευών και τους φορτώνει στον πυρήνα.
6. Το ΛΣ δίνει αρχικές τιμές στους πίνακές του, δημιουργεί τις απαραίτητες διεργασίες παρασκηνίου και ξεκινά το πρόγραμμα login.

1.4 Οι κατηγορίες των Λ.Σ.

Οι κατηγορίες των ΛΣ

1. ΛΣ μεγάλων υπολογιστών (Mainframe OS)
2. ΛΣ εξυπηρετητών (Server OS)
3. ΛΣ πολυεπεξεργαστικών συστημάτων (Multiprocessor OS)
4. ΛΣ προσωπικών υπολογιστών (Personal computer OS)
5. ΛΣ υπολογιστών χειρός (Handheld OS)
6. ΛΣ ενσωματωμένων συστημάτων (Embedded OS)
7. ΛΣ αισθητήρων (Sensor node OS)
8. ΛΣ πραγματικού χρόνου (Real-time OS)
9. ΛΣ έξυπνων καρτών (Smart card OS)

ΛΣ μεγάλων υπολογιστών
(Mainframes)

- Πολύ μεγάλοι υπολογιστές
- Βασική διαφορά: οι δυνατότητες Ε/Ε
- Προσανατολισμένα στην ταυτόχρονη επεξεργασία πολλών εργασιών με πολλή Ε/Ε
  1. Εργασίες δέσμης (batch processing)
  2. Εργασίες συναλλαγών (Transaction processing)
  3. Χρονομερισμός (Timesharing)
- Παράδειγμα ΛΣ: OS/360 (αντικαθίστανται σταδιακά από UNIX παραλλαγές – Linux)

ΛΣ εξυπηρετητών
(Servers)

- Μεγάλοι υπολογιστές.
- Εξυπηρετούν ταυτόχρονα πολλούς χρήστες, συνήθως μέσω διαδικτύου.
  - Εξυπηρετητές ιστού (web servers)
  - Εξυπηρετητές ταχυδρομείου (mail servers)
  - Εξυπηρετητές αρχείων (file servers) ...
- Τυπικά ΛΣ εξυπηρετητών:
  - Solaris, FreeBSD, Linux, Windows Server, κτλ.

ΛΣ πολυεπεξεργαστικών συστημάτων

- Σύνδεση πολλών CPU σε ένα σύστημα.
- Απαιτήσεις διαχείρισης πολλαπλών CPU από ένα μηχάνημα – προσπάθεια βέλτιστης εκμετάλλευσης της επεξεργαστικής ισχύος.
- Πλέον και οι απλοί υπολογιστές χρησιμοποιούν επεξεργαστές πολλών πυρήνων (multi-core processors).

ΛΣ προσωπικών υπολογιστών

- Εξυπηρετούν τις ανάγκες απλών χρηστών.
- Οι σύγχρονοι προσωπικοί υπολογιστές υποστηρίζουν πολυπρογραμματισμό, πολυνημάτωση, πολλαπλούς επεξεργαστές κτλ.
- Γνωστά ΛΣ προσωπικών υπολογιστών:
  - Windows, Linux, FreeBSD, MAC OS .

ΛΣ υπολογιστών χειρός
(Handheld computers)

- Ένα σύγχρονο PDA ή έξυπνο κινητό τηλέφωνο, είναι ένας μικρός Η/Υ.
- Υποστηρίζουν πλέον πολύ προγραμματισμό.
- Μικρότερη οθόνη, πληκτρολόγιο (πλέον και αφής)
- CPU 32 bit.
- Σκληρούς δίσκους μέχρι δεκάδων GB.
- Παραδείγματα: Symbian OS, Android, Iphone OS, Palm OS, Windows Mobile.

ΛΣ Ενσωματωμένων συστημάτων
(Embedded systems)

- ΛΣ που ελέγχουν συσκευές ειδικού σκοπού (τηλεόραση, οικιακές συσκευές, αυτοκίνητα κτλ).
- Εκτελούν μόνο συγκεκριμένο λογισμικό του κατασκευαστή.
  - Δεν είναι υπολογιστές γενικού σκοπού.
  - Δεν απειλείται από μη έμπιστο λογισμικό.
- Παραδείγματα: QNX, Vx Works και παραλλαγές του Linux.

ΛΣ κόμβων αισθητήρων
(Sensor nodes)

- Κόμβοι με πολύ χαμηλές δυνατότητες σε CPU,μνήμη, ενέργεια κτλ).
- Ασύρματη επικοινωνία.
- Απαιτείται μεγάλη ανθεκτικότητα σε απειλές.
- Τα ΛΣ αισθητήρων είναι πολύ μικρά:
  - Καθοδηγούμενα από γεγονότα (event-driven).
  - Οι χρήστες δεν μπορούν να εκκινήσουν προγράμματα της επιλογής τους.
- Παράδειγμα ΛΣ: Tiny OS.

ΛΣ πραγματικού χρόνου
(real-time systems)

- Ο χρόνος είναι η πιο σημαντική παράμετρος (βιομηχανικά συστήματα, αλυσίδες παραγωγής κτλ)
- Hard-time: Απόλυτη τήρηση του χρονοδιαγράμματος (π.χ. αεροσκάφη).
- Soft-time: Περιστασιακή μη τήρηση είναι αποδεκτή (π.χ. εικόνα, ήχος).
- Παράδειγμα ΛΣ: e-Cos.

ΛΣ έξυπνων καρτών
(smart cards)

- Κάρτες με περιορισμένη CPU και μνήμη.
- Εκτελούν συγκεκριμένες λειτουργίες (π.χ. πληρωμή).
- Συνήθως εκτελεί εφαρμογές Java (java applets) πάνω σε διερμηνευτή (Java Virtual Machine).
- Το ΛΣ είναι αρκετά «πρωτόγονο» για εργασίες όπως πολυπρογραμματισμός ή διαχείριση πόρων.

1.5 Βασικές έννοιες Λ.Σ.

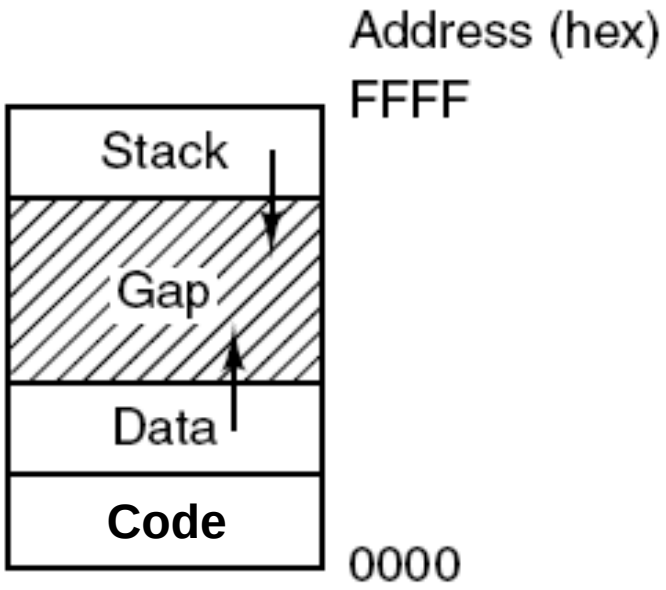
Βασικές έννοιες των ΛΣ

- Διεργασίες (Processes)
- Χώροι διευθύνσεων(Address spaces)
- Αρχεία (Files)
- Κέλυφος (shell)

Διεργασίες

- Διεργασία = ένα πρόγραμμα που εκτελείται.
  Αποτελείται από:
  - Το **χώρο διευθύνσεων (address space)** της διεργασίας στη μνήμη που περιλαμβάνει τον **κώδικα** του προγράμματος που εκτελείται, τα **δεδομένα** του προγράμματος και τη **στοίβα**.
  - **Καταχωρητές**: όπως ο **μετρητής προγράμματος,** και ο **δείκτης στοίβας.**
  - Ανοικτά αρχεία, προειδοποιήσεις, λίστα σχετιζόμενων διεργασιών κτλ.
- Το ΛΣ διαχειρίζεται πολλές διεργασίες, διατηρώντας ένα **πίνακα διεργασιών (process table).**

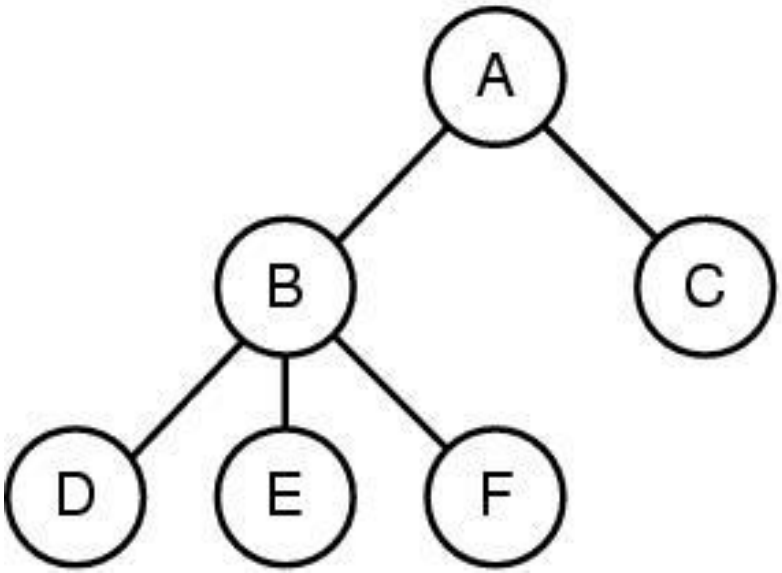
Η διεργασία στη μνήμη



Εικόνα 1-20. Οι διεργασίες έχουν τρία τμήματα μνήμης: πρόγραμμα (code), δεδομένα (data), και στοίβα (stack).

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Δένδρο διεργασιών



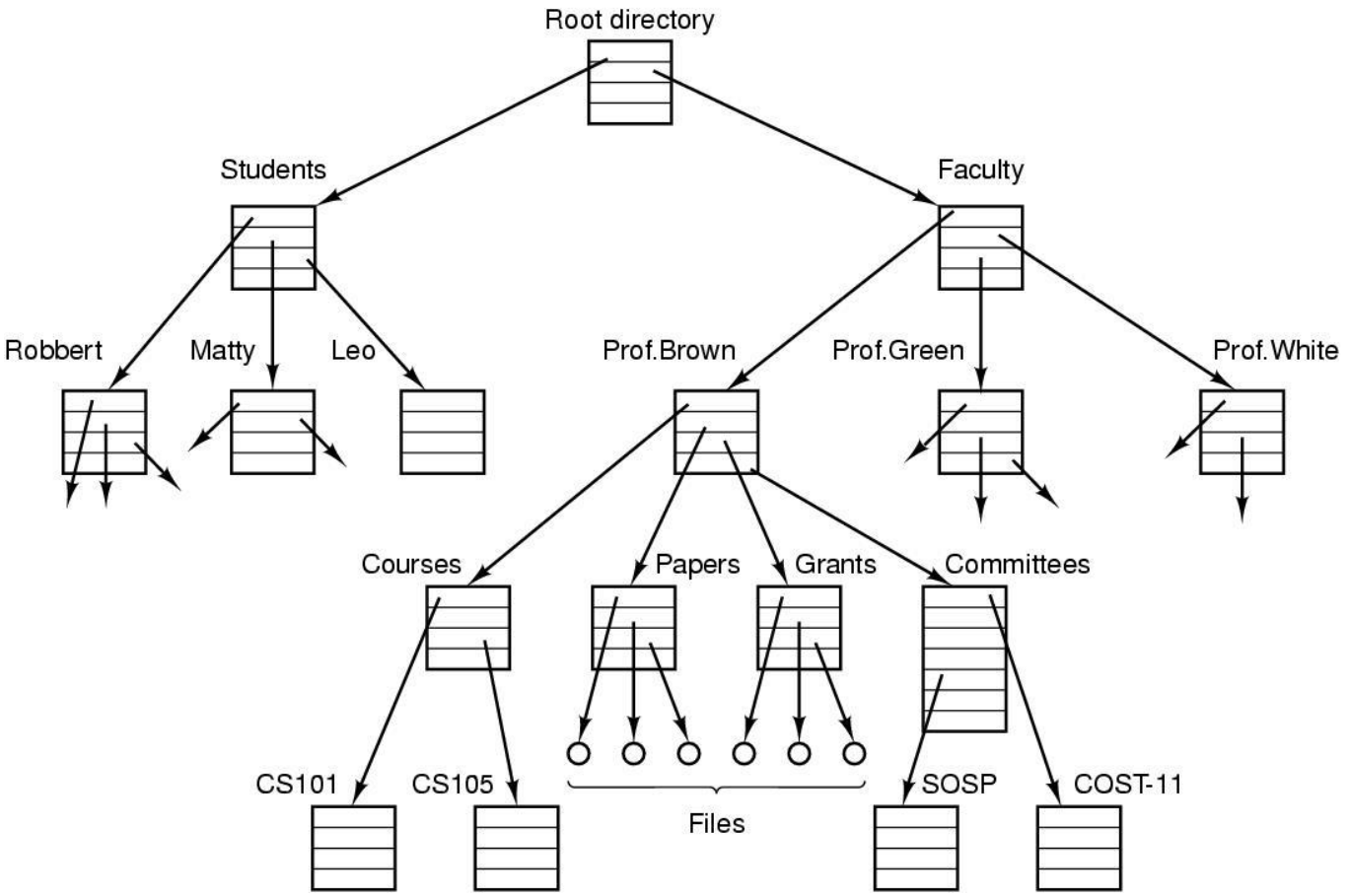
Εικόνα 1-13. Η διεργασία Α δημιουργεί δύο διεργασίες-παιδιά (ή θυγατρικές), την Β και την C. Η διεργασία Β δημιουργεί με τη σειρά της άλλες τρεις, τις D, E, και F.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Χώροι διευθύνσεων (Μνήμη)

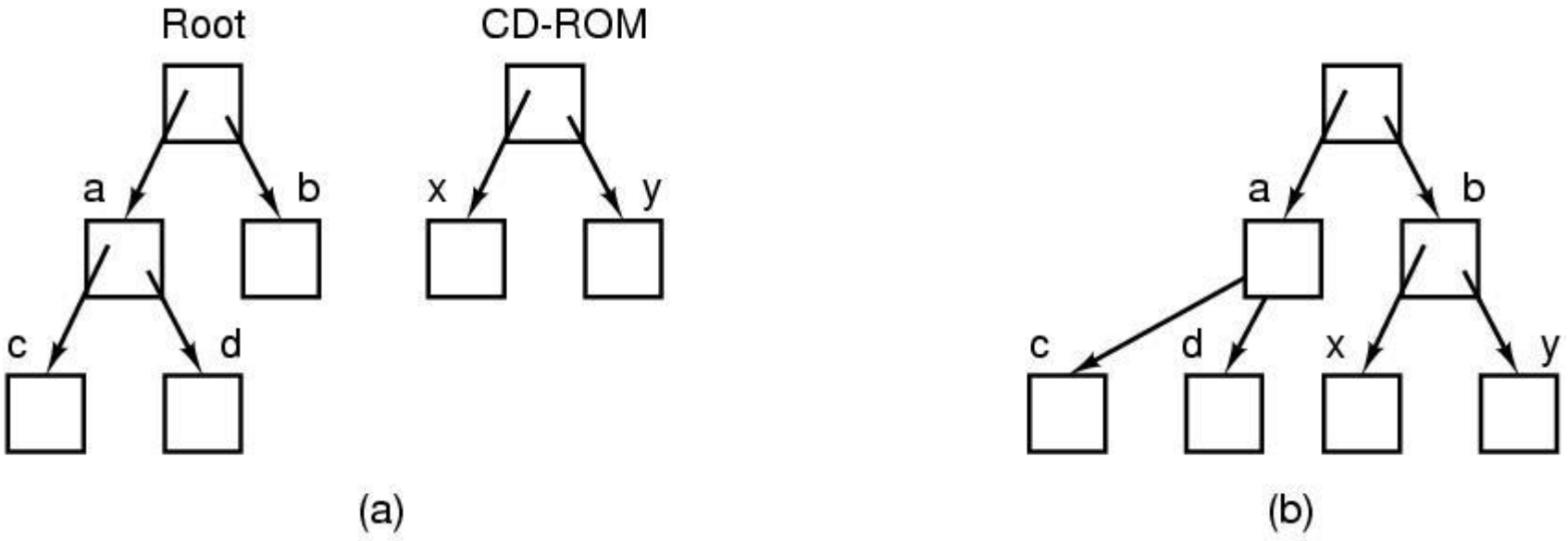
- Κάθε πρόγραμμα που εκτελείται, πρέπει να βρίσκεται στην κύρια μνήμη.
  - Το ΛΣ υποστηρίζει το χώρο διευθύνσεων των διεργασιών.
- Όταν βρίσκονται πολλά προγράμματα στη μνήμη ταυτόχρονα, απαιτείται προστασία.
- Τι συμβαίνει όταν ένα πρόγραμμα απαιτεί περισσότερο χώρο διευθύνσεων από ότι διαθέτει το σύστημα;
  - Χρήση **Εικονικής μνήμης (Virtual memory).**

Αρχεία



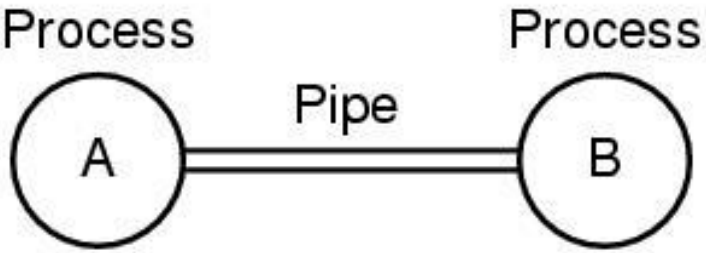
Εικόνα1-14. Ένα σύστημα αρχείων ενός τμήματος πανεπιστημίου.

Αρχεία
(ανάρτηση - mount)



Εικόνα1-15. (a) Πριν την φόρτωση (mounting), τα αρχεία στο CD-ROM δεν είναι προσβάσιμα. (b) Μετά τη φόρτωση (mounting), αποτελούν κλαδί της ιεραρχίας των αρχείων.

Αρχεία
(Αγωγοί ή σωληνώσεις - pipes)



Εικόνα 1-16. Δύο διεργασίες που συνδέονται με μία σωλήνωση (pipe).

Παράδειγμα:

cat file1 file2 file3 | sort >/dev/lp

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

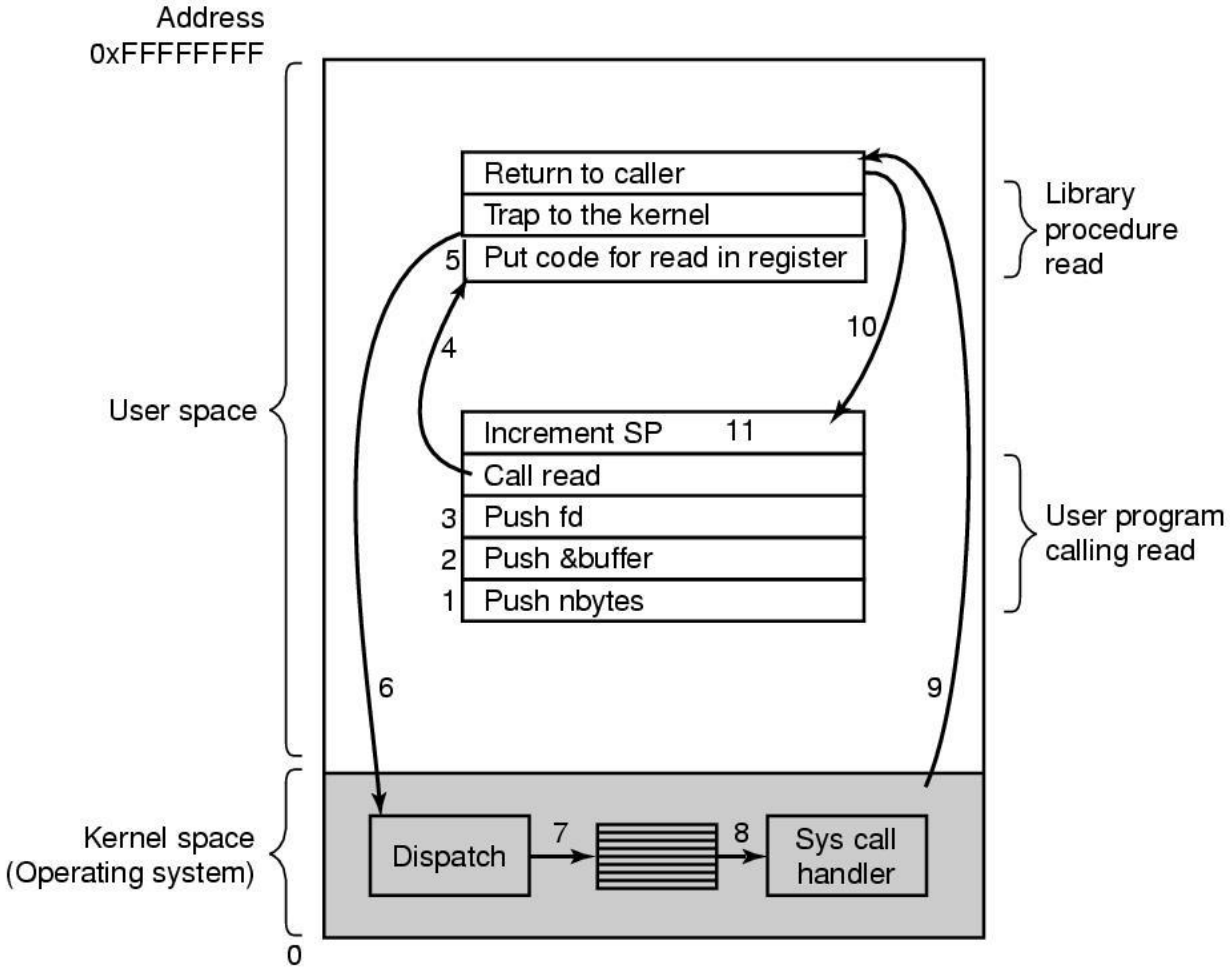
1.6 Κλήσεις συστήματος

Κλήσεις συστήματος (System Calls)

- Θυμηθείτε ότι το ΛΣ εκτελείται σε κατάσταση πυρήνα (kernel mode) και όλο το υπόλοιπο λογισμικό σε κατάσταση χρήστη (user mode).
- Για να χρησιμοποιήσει μία διεργασία μία υπηρεσία του ΛΣ (π.χ. ανάγνωση από αρχείο) θα εκτελέσει μία **κλήση συστήματος (system call)**.
- Κάθε κλήση συστήματος υλοποιείται μέσω μίας ρουτίνας που βρίσκεται σε μία βιβλιοθήκη διαδικασιών (συνήθως σε C).
- Η κλήση συστήματος προκαλεί μία **παγίδευση (trap)** και μεταβαίνει σε κατάσταση πυρήνα.
- Η εντολή TRAP προκαλεί την ανάμιξη του ΛΣ.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Κλήσεις συστήματος
(Παράδειγμα)



Εικόνα 1-17. Τα 11 βήματα της κλήσης συστήματος **read(fd, buffer, nbytes)**

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Κλήσεις συστήματος για τη Διαχείριση Διεργασιών

Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

Εικόνα1-18. Μερικές από τις κυριότερες κλήσεις συστήματος του POSIX.

pid: process ID (η ταυτότητα της διεργασίας)
s: κωδικός σφάλματος (-1 για λάθος)
fd: file descriptor (περιγραφέας αρχείου)

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Ένα απλό κέλυφος

```
#define TRUE 1

while (TRUE) {                                  /* repeat forever */
        type_prompt( );                         /* display prompt on the screen */
        read_command(command, parameters);      /* read input from terminal */

        if (fork( ) != 0) {                     /* fork off child process */
             /* Parent code. */
             waitpid(–1, &status, 0);           /* wait for child to exit */
        } else {
             /* Child code. */
             execve(command, parameters, 0);    /* execute command */
        }
}
```

Εικόνα 1-19. Παράδειγμα ενός απλού κελύφους (command shell).

(Παράδειγμα διαταγής)

cp file1 file2

- Το κύριο πρόγραμμα που υλοποιεί την εντολή cp περιέχει τη δήλωση: **main(argc, argv, envp)**
  - argc: το πλήθος των ορισμάτων (3)
  - argv: δείκτης σε ένα πίνακα
    - argv[0] → cp, argv[1] → file1, argv[2] → file2
  - envp: enviroment variables (π.χ. όνομα home directory)

Κλήσεις συστήματος για τη Διαχείριση Αρχείων

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing, or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Εικόνα 1-18. Μερικές από τις κυριότερες κλήσεις συστήματος του POSIX.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

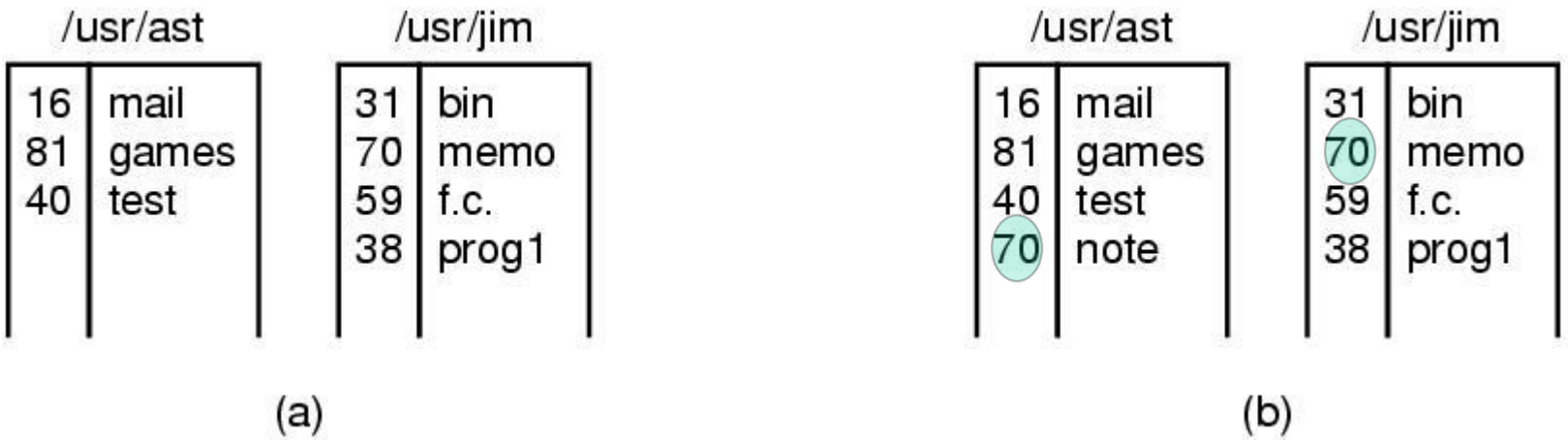
Κλήσεις συστήματος για τη Διαχείριση Καταλόγων

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Εικόνα 1-18. Μερικές από τις κυριότερες κλήσεις συστήματος του POSIX.

Σύνδεσμοι αρχείων (linking)

Παράδειγμα κλήσης link:
link (“usr/jim/memo”, “/usr/ast/note”);



Εικόνα 1-21. (a) Δύο κατάλογοι πριν τη σύνδεση του */usr/jim/memo* στον κατάλογο ast.
(b) Οι ίδιοι κατάλογοι μετά τη σύνδεση.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Φόρτωση αρχείων (mounting)

Παράδειγμα κλήσης mount:
mount(“/dev/hda”, “/mnt”, 0);

Εικόνα 1-22. (a) Το σύστημα αρχείων πριν τη φόρτωση. (b) Το σύστημα αρχείων μετά τη φόρτωση.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Διάφορες κλήσεις συστήματος

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

Εικόνα 1-18. Μερικές από τις κυριότερες κλήσεις συστήματος του POSIX.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Το API Win32 των Windows (Application Programming Interface)

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time

Εικόνα 1-23. Οι Win32 API κλήσεις σε, κατά προσέγγιση, αντιστοίχηση με τις κλήσεις συστήματος του UNIX.

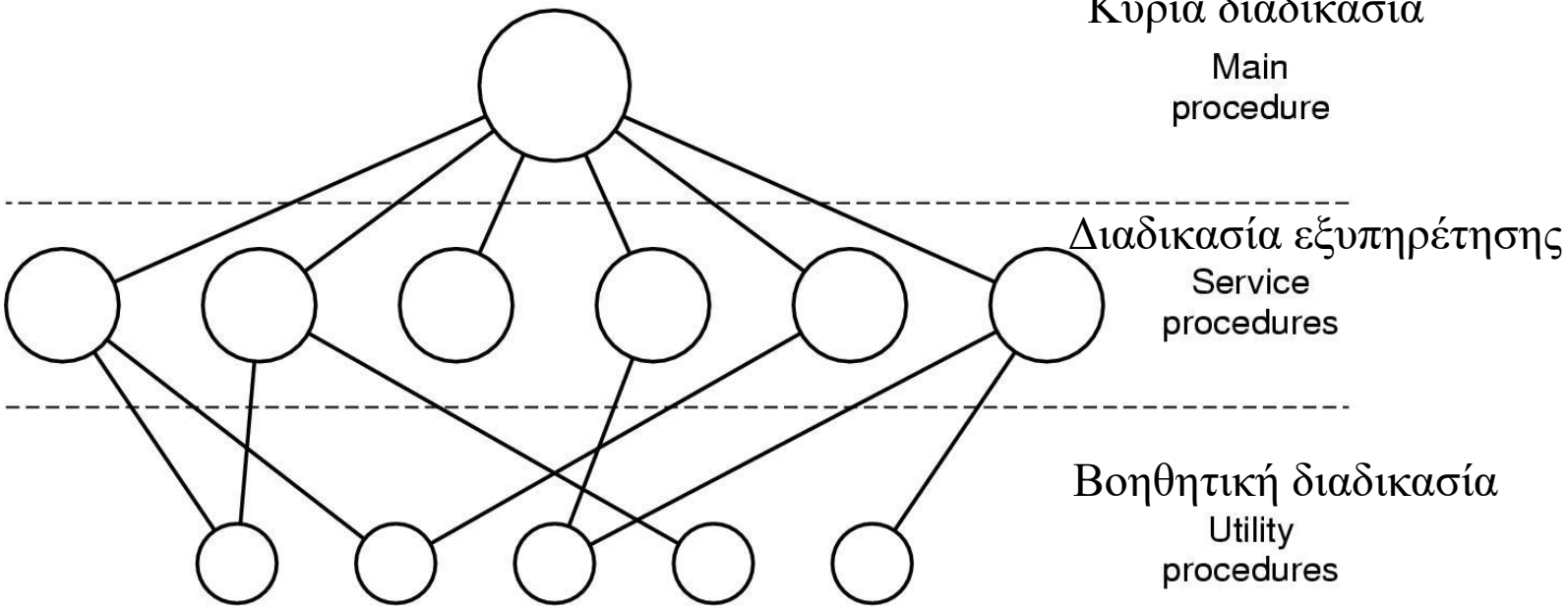
1.7 Η δομή των λειτουργικών συστημάτων

Δομές Λειτουργικών Συστημάτων

Μονολιθικά συστήματα – βασική δομή:

- Η πιο διαδεδομένη οργάνωση.
- Ένα κύριο πρόγραμμα το οποίο καλεί την αιτούμενη διαδικασία εξυπηρέτησης.
- Ένα σύνολο διαδικασιών υπηρεσιών (service procedures) οι οποίες εκτελούν τις κλήσεις συστήματος.
- Ένα σύνολο βοηθητικών διαδικασιών (utility procedures) οι οποίες βοηθούν τις διαδικασίες υπηρεσιών.

Μονολιθικά συστήματα



Εικόνα 1-24. Ένα απλό μοντέλο της δομής ενός μονολιθικού συστήματος

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

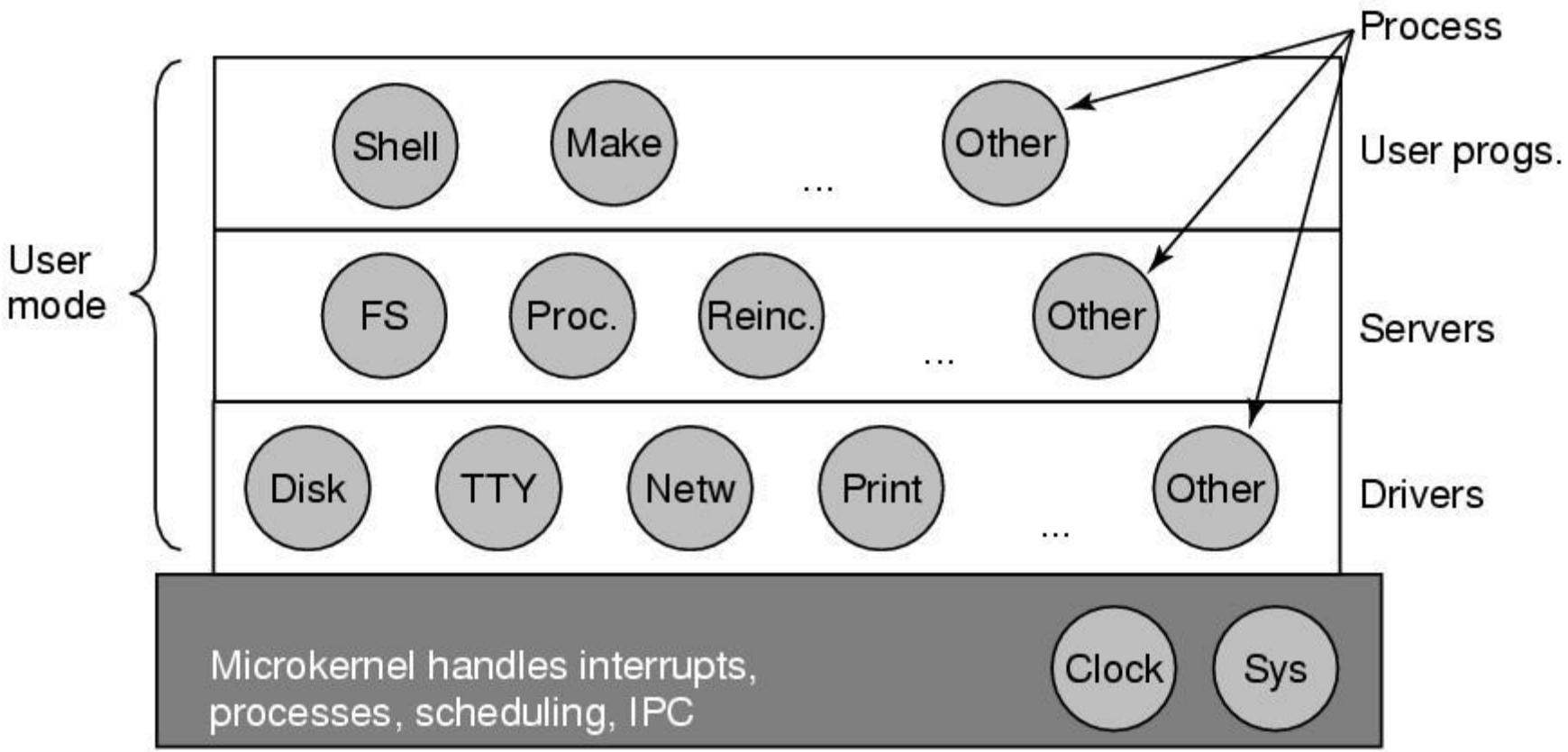
Πολύ-επίπεδα συστήματα

Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

Εικόνα 1-25. Η δομή του ΛΣ «THE».

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

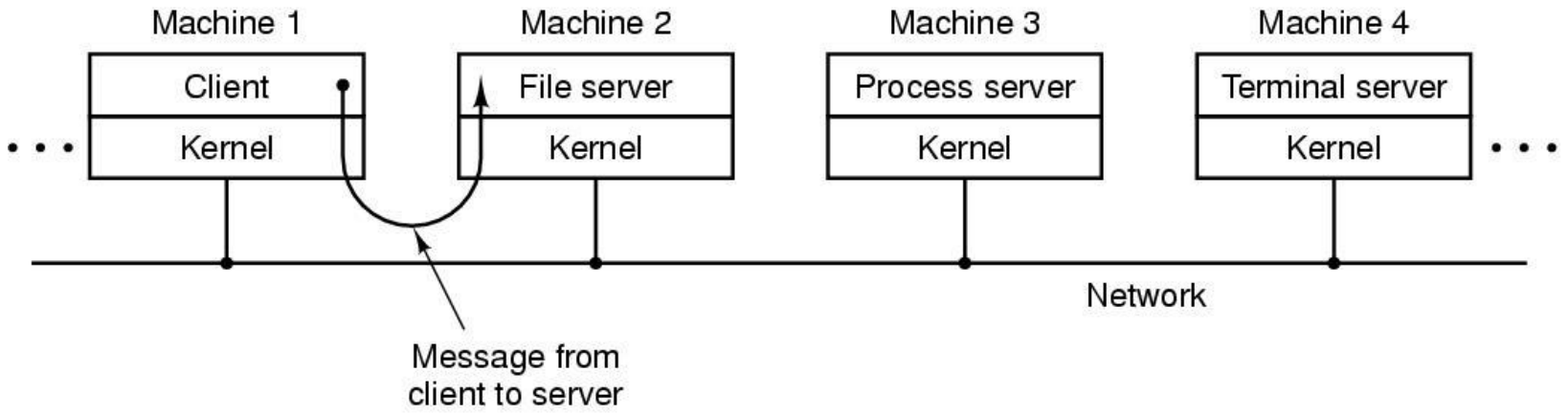
Μικροπυρήνες (Microkernels)



Εικόνα 1-26. Η δομή του ΛΣ «MINIX 3».

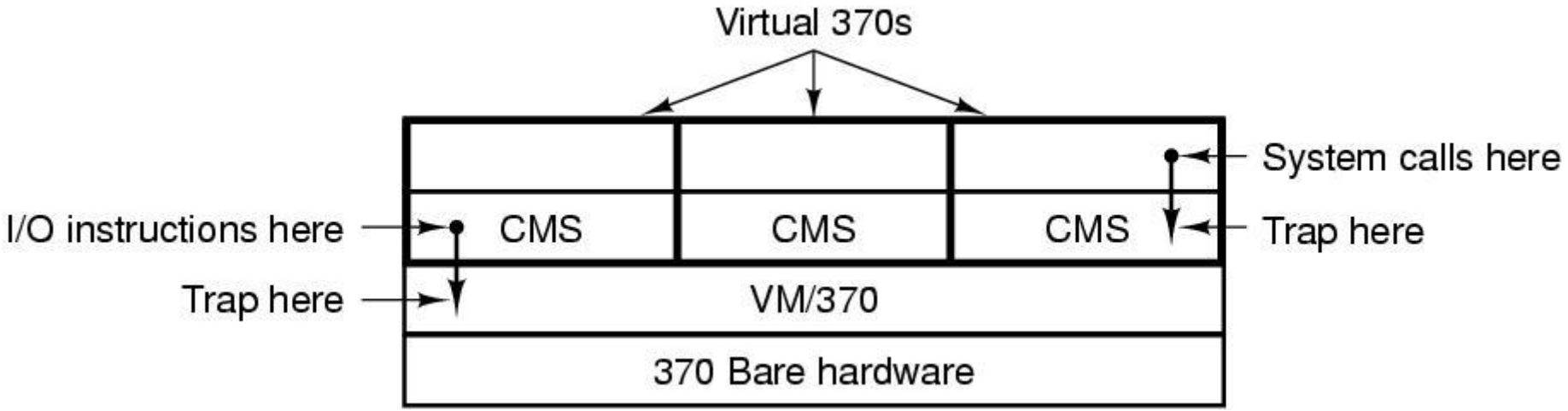
Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Μοντέλο πελάτη-εξυπηρετητή (Client-Server)



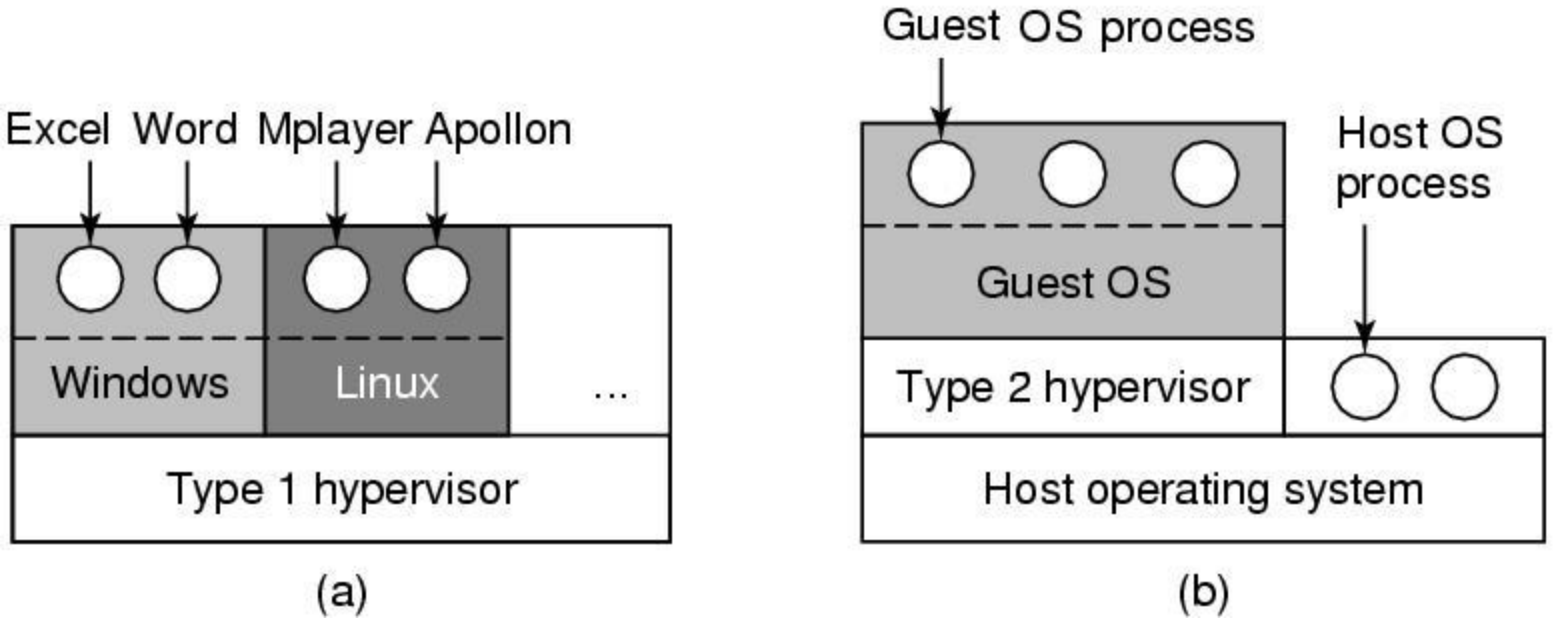
Εικόνα 1-27. Το μοντέλο πελάτη-εξυπηρετητή σε ένα δίκτυο.

Εικονικές μηχανές (1)



Εικόνα 1-28. Η δομή του VM/370 με CMS.

Εικονικές μηχανές (2)



Εικόνα 1-29. (a) Υπερεπόπτης τύπου-1 (type 1 hypervisor). (b) Υπερεπόπτης τύπου-2 (type 2 hypervisor).

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Μονάδες μέτρησης

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	milli	10^{3}	1,000	Kilo
10^{-6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

Περιγραφή των μονάδων μέτρησης

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

2η ενότητα

ΔΙΕΡΓΑΣΙΕΣ ΚΑΙ ΝΗΜΑΤΑ

Διεργασίες και Νήματα

Περιεχόμενα

2.1 Διεργασίες (Processes)

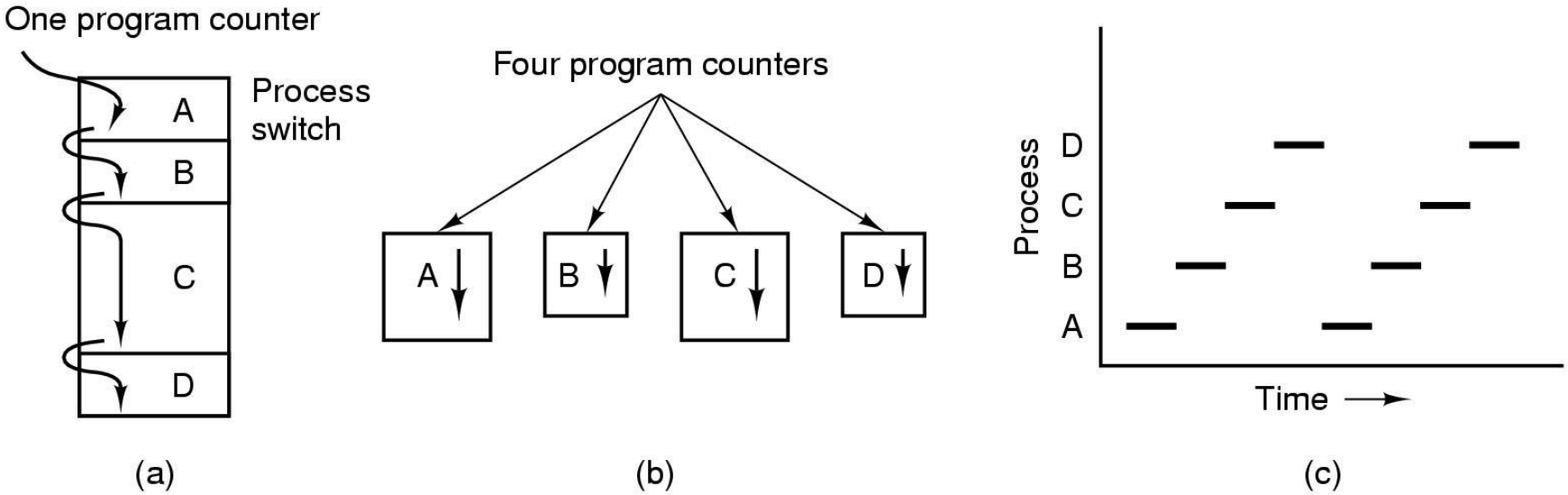
Διεργασίες

- Διεργασία = Η αφαίρεση (abstraction) ενός προγράμματος που εκτελείται. Αποτελείται από:
  - Το **χώρο διευθύνσεων (address space)** της διεργασίας στη μνήμη που περιλαμβάνει:
    - τον **κώδικα** του προγράμματος που εκτελείται,
    - τα **δεδομένα** του προγράμματος και
    - τη **στοίβα**.
  - **Καταχωρητές**: όπως ο **μετρητής προγράμματος,** και ο **δείκτης στοίβας.**
  - Ανοικτά αρχεία, προειδοποιήσεις, λίστα σχετιζόμενων διεργασιών κτλ.

Το μοντέλο των διεργασιών

- Όλο το εκτελέσιμο λογισμικό οργανώνεται σε ένα πλήθος σειριακών διαδικασιών (sequential processes).
- Εννοιολογικά, κάθε διεργασία έχει τη δική της CPU.
  - Στην πραγματικότητα υπάρχει μόνο 1 CPU.
- Η γρήγορη εναλλαγή των διεργασιών στη CPU λέγεται πολυπρογραμματισμός (βλ. Κεφ.1).
  - (Ψευδο)παράλληλη εκτέλεση προγραμμάτων με 1 CPU.
- Η κοινή χρήση της CPU απαιτεί τη χρήση αλγορίθμων **χρονοπρογραμματισμού (scheduling algorithm).**

Το μοντέλο των διεργασιών



Εικόνα 2-1. (a) Πολυπρογραμματισμός με τέσσερα προγράμματα. (b) Εννοιολογικό μοντέλο τεσσάρων ανεξάρτητων σειριακών διεργασιών. (c) Ενεργές διεργασίες στο χρόνο. Ένα πρόγραμμα είναι ενεργό κάθε στιγμή.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Το μοντέλο των διεργασιών

- Η διεργασία είναι ένα **στιγμιότυπο (instance)** ενός εκτελέσιμου προγράμματος.
  - Εάν ένα πρόγραμμα εκτελεστεί δύο φορές, οδηγεί στη δημιουργία δύο διαφορετικών διεργασιών.
- Δεν πρέπει να γίνονται υποθέσεις για το χρόνο εκτέλεσης μίας διεργασίας.
  - Ένα πρόγραμμα που εκτελείται δύο φορές, ενδέχεται να έχει μεγάλες αποκλίσεις στο χρόνο εκτέλεσης!

Δημιουργία διεργασίας

Γεγονότα που προκαλούν τη δημιουργία μίας διεργασίας:

(α) Εκκίνηση συστήματος.

(β) Εκτέλεση μίας κλήσης συστήματος για τη δημιουργία διεργασίας (από μία άλλη διεργασία που εκτελείται).

(γ) Αίτηση χρήστη για τη δημιουργία νέας διεργασίας.

(δ) Εκκίνηση εργασίας δέσμης (batch job).

Τερματισμός διεργασίας

Γεγονότα που προκαλούν τον τερματισμό μίας διεργασίας είναι:

(α) Κανονική έξοδος (εθελοντική).

(β) Έξοδος σφάλματος (error exit) (εθελοντική).

(γ) Έξοδος μοιραίου σφάλματος (fatal error) (μη εθελοντική).

(δ) Τερματισμός από άλλη διεργασία (μη εθελοντική).

Ιεραρχίες διεργασιών (UNIX)

- Μία (γονική) διεργασία μπορεί να δημιουργήσει μία ή περισσότερες θυγατρικές διεργασίες με την κλήση fork( ).
- Με αυτό τον τρόπο δημιουργούνται ιεραρχίες διεργασιών ή ομάδες διεργασιών (process group).
- Ένα σήμα από το πληκτρολόγιο προς μία διεργασία, θα διανεμηθεί σε όλη την ομάδα διεργασιών.
- Κάθε διεργασία της ομάδας επεξεργάζεται το σήμα.
  - Το δέχεται και εκτελεί κάποια προεπιλεγμένη ενέργεια.
  - Το αγνοεί.
  - Τερματίζεται.

Ιεραρχίες διεργασιών (Windows)

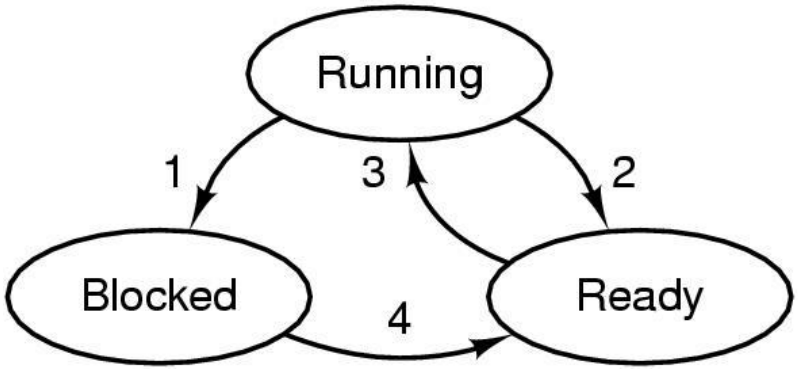
- Στα Windows δεν υπάρχει ιεραρχία διεργασιών .
- Όλες οι διεργασίες είναι ισοδύναμες.
- Όταν δημιουργείται μία διεργασία, η «γονική» της αποκτά ένα ειδικό χειριστή (handle), μέσω του οποίου μπορεί να διαχειρίζεται τη θυγατρική διεργασία.
- Μπορεί να μεταβιβάσει το χειριστή σε κάποια άλλη διεργασία, παραβιάζοντας την ιεραρχία.
  - Κάτι τέτοιο δεν είναι δυνατό στο UNIX.

Καταστάσεις διεργασιών

cat file1 file2 | grep tree

- Συνένωση δύο αρχείων και αποστολή στην έξοδο.
- Η σωλήνωση δίνει το αποτέλεσμα της διεργασίας ως είσοδο στην εντολή **grep** ώστε να αναζητηθεί μία λέξη.
- Εάν η **cat** δεν έχει ολοκληρωθεί, η **grep** παραμένει μπολκαρισμένη.

Καταστάσεις διεργασιών

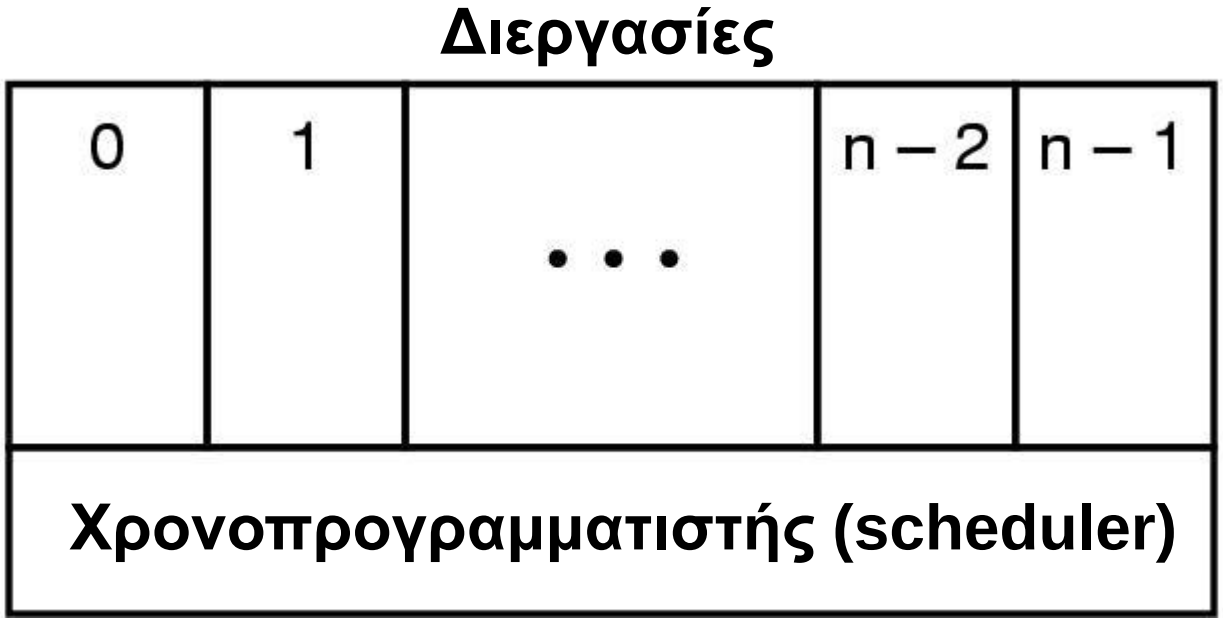


1. Η διεργασία μπλοκάρεται (περιμένει δεδομένα εισόδου).
2. Η διεργασία διακόπτεται από τον scheduler (χρονοπρογραμματιστή). Η CPU δώθηκε σε άλλη διεργασία.
3. Ο scheduler επιλέγει την διεργασία για να τρέξει στη CPU
4. Τα δεδομένα εισόδου που αναμένει η διεργασία είναι διαθέσιμα.

Εικόνα 2-2. Μία διεργασία μπορεί να εκτελείται, να είναι μπλοκαρισμένη ή να είναι έτοιμη για εκτέλεση.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Υλοποίηση των διεργασιών (1)



Εικόνα 2-3. Το κατώτερο στρώμα του ΛΣ χειρίζεται τις διακοπές και το χρονοπρογραμματισμό.

Πάνω από αυτό το επίπεδο βρίσκονται οι σειριακές διεργασίες.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Υλοποίηση των διεργασιών (2)

Διαχείριση διεργασιών	**Διαχείριση μνήμης**	**Διαχείριση αρχείων**
Καταχωρητές (registers) Μετρητής (Program counter) Program Status Word (PSW) Δείκτης στοίβας (stack pointer) Κατάσταση διεργασίας Προτεραιότητα Παράμετροι scheduling Ταυτότητα διεργασίας (PID) Ταυτότητα γονικής (PPID) Ομάδα διεργασίας Σήματα Χρόνος εκκίνησης διεργασίας Χρόνος χρήσης CPU Χρόνος χρήσης CPU από θυγατρικές διεργασίες Χρονική στιγμή επόμενης ειδοποίησης	Δείκτης τμήματος κώδικα Δείκτης τμήματος δεδομένων Δείκτης τμήματος στοίβας	Βασικός κατάλογος Κατάλογος εργασίας Περιγραφείς αρχείων Ταυτότητα χρήστη (user ID) Ταυτότητα ομάδας (group ID)

Εικόνα 2-4. Μερικά πεδία μιας τυπικής καταχώρησης του **πίνακα διεργασιών (process table)**.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Υλοποίηση των διεργασιών (3)

1. Το υλικό τοποθετεί στη στοίβα τους μετρητές της διεργασίας που τρέχει.
2. Το υλικό φορτώνει το μετρητή του προγράμματος που δείχνει το διάνυσμα διακοπών (interrupt vector).
3. Η assembly διαδικασία αποθηκεύει τους καταχωρητές της τρέχουσας διεργασίας.
4. Η assembly διαδικασία δημιουργεί μία νέα προσωρινή στοίβα για το χειριστή διεργασιών.
5. Η C διαδικασία εξυπηρέτησης διακοπής (interrupt service) εκτελείται. Διαβάζει την είσοδο και την αποθηκεύει σε ένα buffer. Φέρνει σε ετοιμότητα μία διεργασία και καλεί τον χρονοπρογραμματιστή.
6. Ο χρονοπρογραμματιστής (scheduler) αποφασίζει ποια διεργασία θα εκτελεστεί.
7. Η C διαδικασία επιστρέφει τον έλεγχο στην αντίστοιχη assembly διαδικασία της διεργασίας που θα εκτελεστεί.
8. Η διαδικασία assembly εκκινεί τη νέα διεργασία.

Εικόνα 2-5. Οι ενέργειες που εκτελεί το χαμηλότερο επίπεδο του ΛΣ όταν συμβεί κάποια διακοπή.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

2.2 Νήματα (Threads)

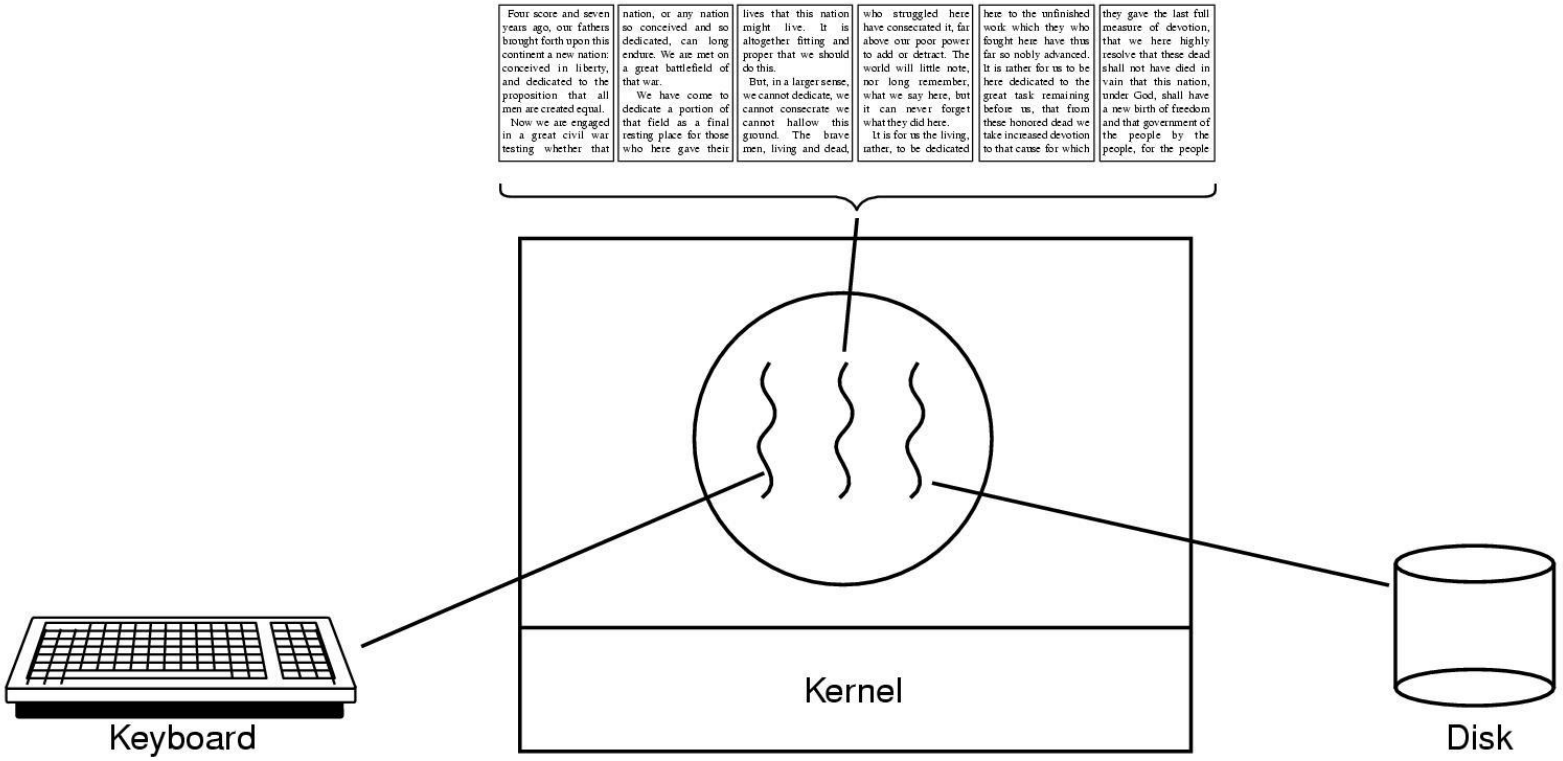
Τι είναι τα νήματα

- Διεργασία: ένα νήμα ελέγχου σε ένα χώρο διευθύνσεων.
- Σε πολλές περιπτώσεις υπάρχει η ανάγκη χρήσης περισσότερων νημάτων ελέγχου στον ίδιο χώρο διευθύνσεων
- Νήμα: ένα πρόγραμμα που εκτελείται, σε κοινό χώρο διευθύνσεων με άλλα νήματα.
- Επιτρέπει την (ψευδο)παράλληλη εκτέλεση εργασιών στον ίδιο χώρο διευθύνσεων.

Πλεονεκτήματα των νημάτων

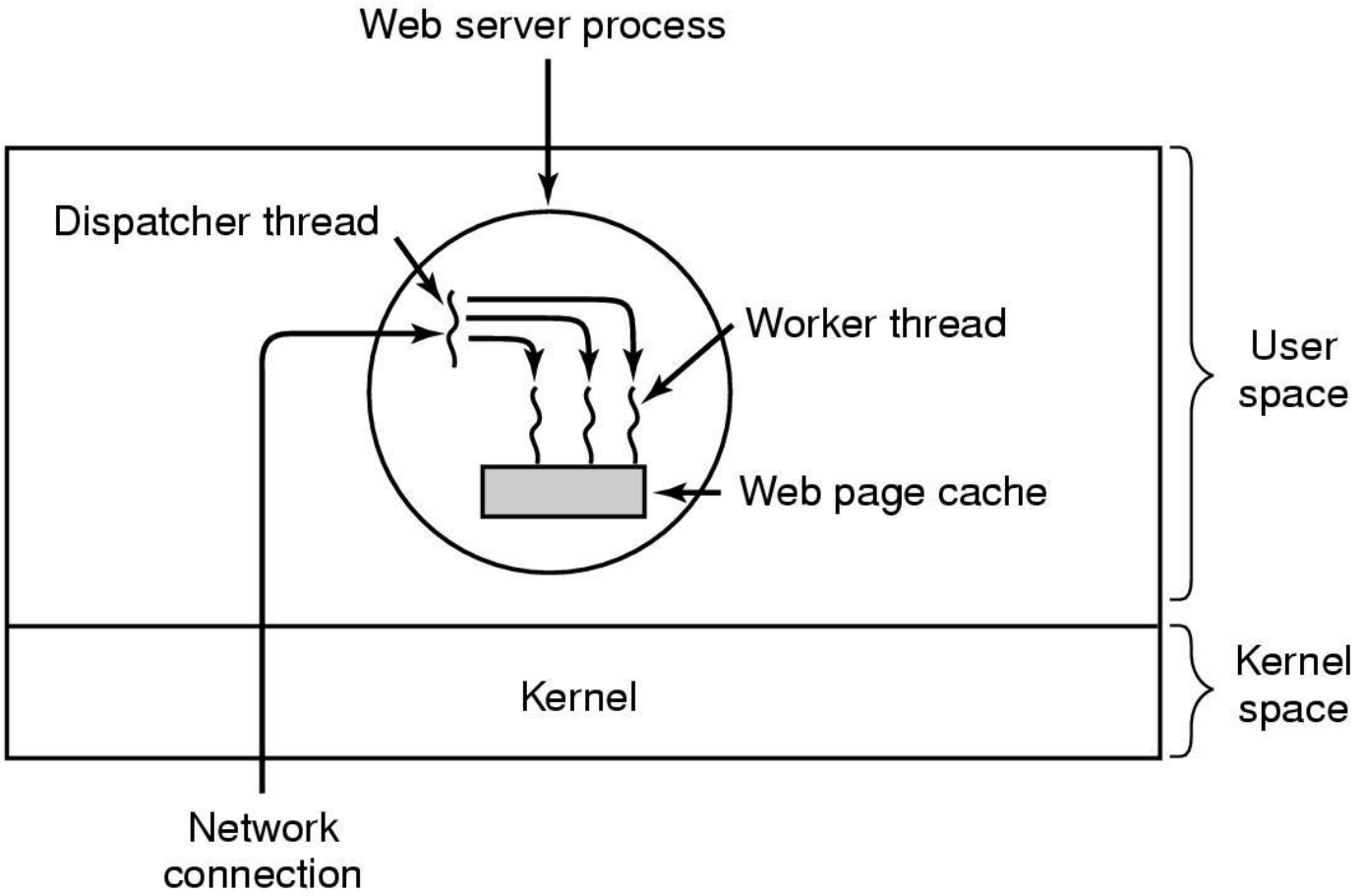
1. Απαραίτητη η κοινή χρήση χώρου διευθύνσεων σε ορισμένες εφαρμογές.
2. Ελαφρύτερα από τις διεργασίες (10-100 φορές η δημιουργία, καταστροφή τους).
3. Καλύτερη επικάλυψη εργασιών σε περίπτωση μεγάλου υπολογιστικού φόρτου και παράλληλα εκτεταμένης χρήσης Ε/Ε.

Χρήση των νημάτων (1)



Εικόνα 2-7. Ένας επεξεργαστής κειμένου με τρία νήματα.

Χρήση των νημάτων (2)



Εικόνα 2-8. Εξυπηρετητής ιστού με τη χρήση πολυνημάτωσης (multithreaded Web server).

Χρήση των νημάτων (3)

```
while (TRUE) {
    get_next_request(&buf);
    handoff_work(&buf);
}
```

(a)

```
while (TRUE) {
    wait_for_work(&buf)
    look_for_page_in_cache(&buf, &page);
    if (page_not_in_cache(&page))
        read_page_from_disk(&buf, &page);
    return_page(&page);
}
```

(b)

Εικόνα 2-9. Ένας γενικός σκελετός του κώδικα ενός multithreaded web server (Σχήμα 2-8).

(a) Το νήμα διεκπεραίωσης (Dispatcher thread).

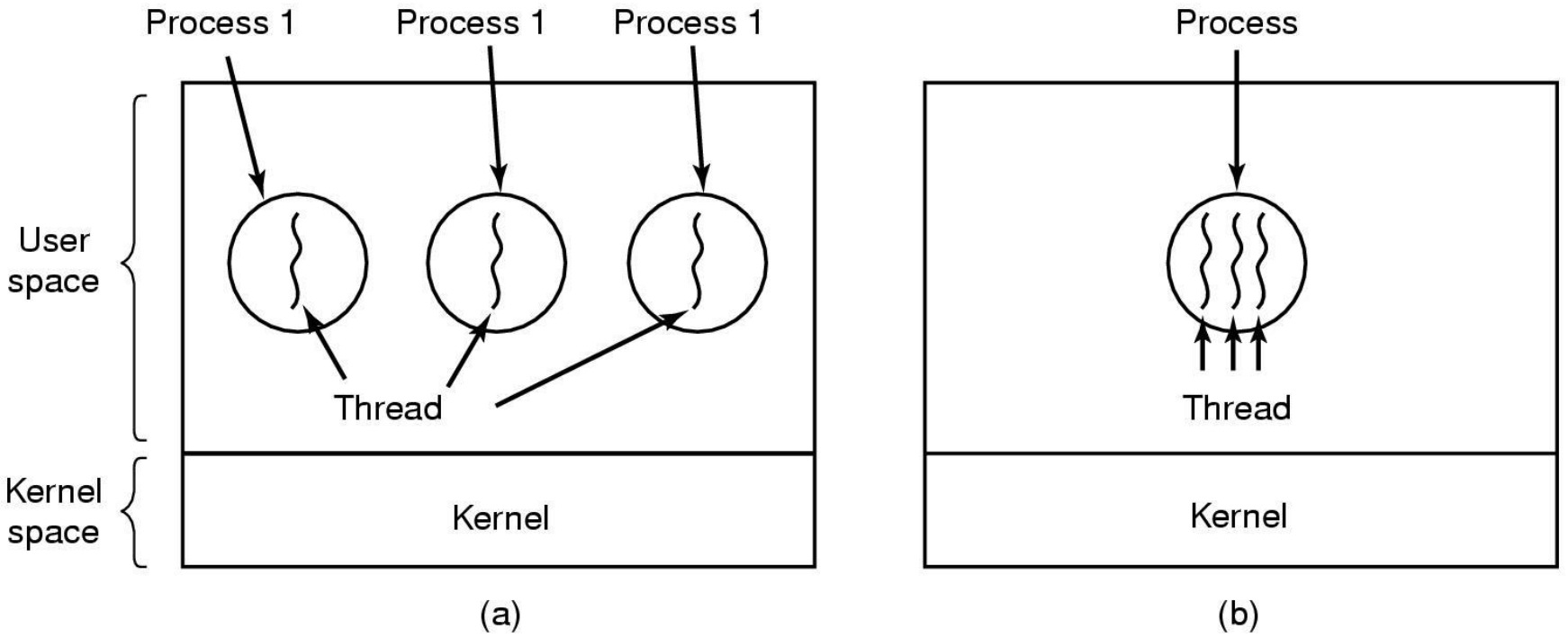
(b) Το νήμα εργασίας (Worker thread).

Το κλασικό μοντέλο των νημάτων (1)

- Κλασικό μοντέλο διεργασίας: ομαδοποίηση συναφών πόρων (π.χ. ανοικτά αρχεία, θυγατρικές διεργασίες, εκκρεμή σήματα, κτλ)
- Κάθε διεργασία με ένα νήμα εκτέλεσης διαθέτει:
  1. ένα μετρητή προγράμματος που δείχνει την επόμενη εντολή που θα εκτελεστεί,
  2. καταχωρητές για τις τρέχουσες μεταβλητές,
  3. μία στοίβα με το ιστορικό εκτέλεσης.
- Θα μπορούσαμε να έχουμε **πολλές ροές εκτέλεσης** στον ίδιο χώρο διευθύνσεων.
  - Άρα πολλά νήματα σε μία διεργασία.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Το κλασικό μοντέλο των νημάτων (2)



Εικόνα 2-11. (a) Τρεις διεργασίες με ένα νήμα η κάθε μία. (b) Μια διεργασία με τρία νήματα.

Το κλασικό μοντέλο των νημάτων (3)

Per process items	Per thread items
Address space (Χώρος διευθύνσεων)	Program counter (Μετρητής προγράμματος)
Global variables (Καθολικές μεταβλητές)	Registers (Καταχωρητές)
Open files (Ανοικτά αρχεία)	Stack (Στοίβα)
Child processes (Θυγατρικές διεργασίες)	State (Κατάσταση)
Pending alarms (Εκκρεμή σήματα)	
Signals and signal handlers	
Accounting information (Πληροφ. διαχείρισης)	

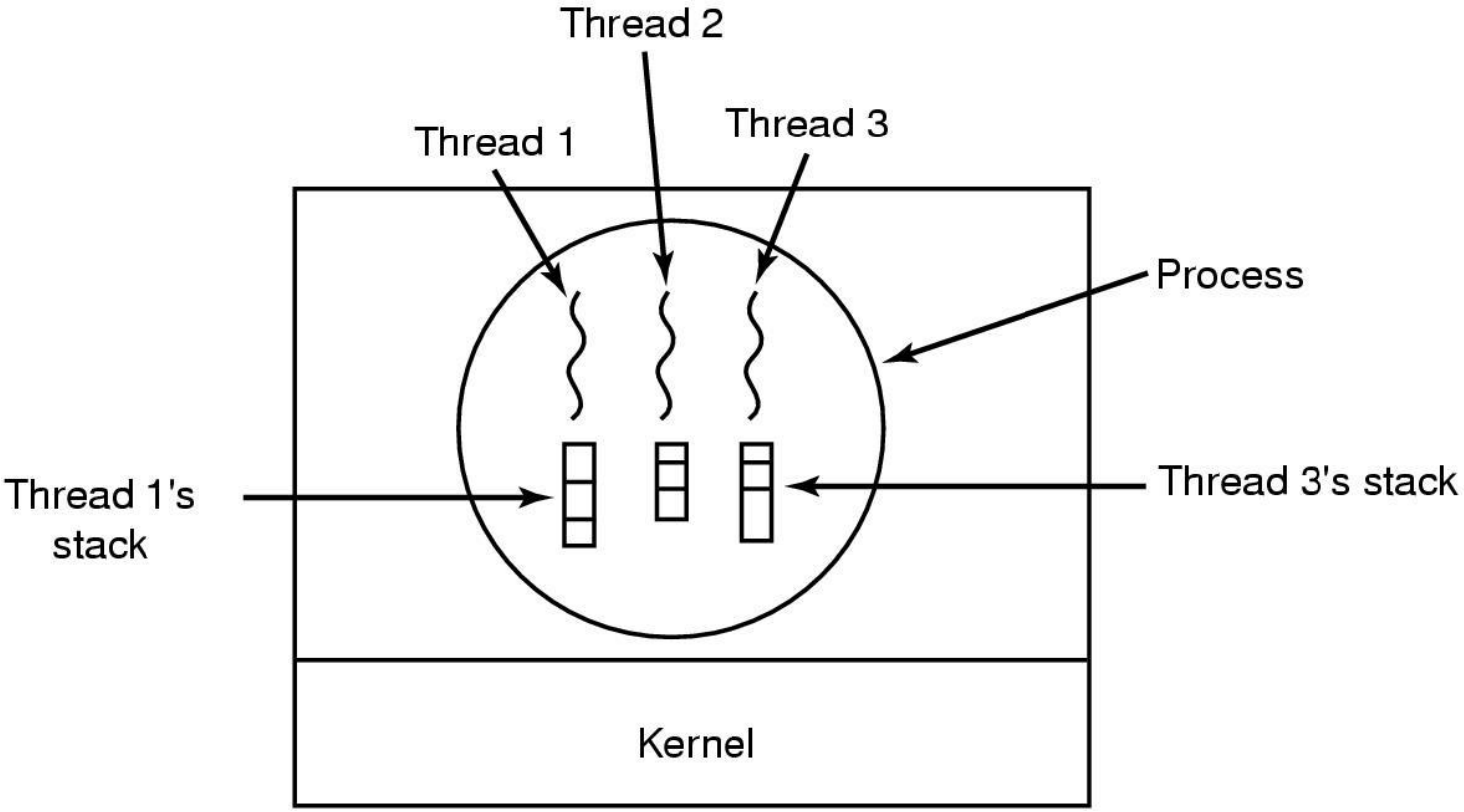
Εικόνα 2-12.

1η στήλη: τι μοιράζονται μεταξύ τους τα νήματα μίας διεργασίας.

2η στήλη: ατομικά στοιχεία κάθε νήματος.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Το κλασικό μοντέλο των νημάτων (4)



Εικόνα 2-13. Κάθε νήμα έχει τη δική του στοίβα.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Τα νήματα στο POSIX (1)

Thread call	Description
Pthread_create	Create a new thread
Pthread_exit	Terminate the calling thread
Pthread_join	Wait for a specific thread to exit
Pthread_yield	Release the CPU to let another thread run
Pthread_attr_init	Create and initialize a thread's attribute structure
Pthread_attr_destroy	Remove a thread's attribute structure

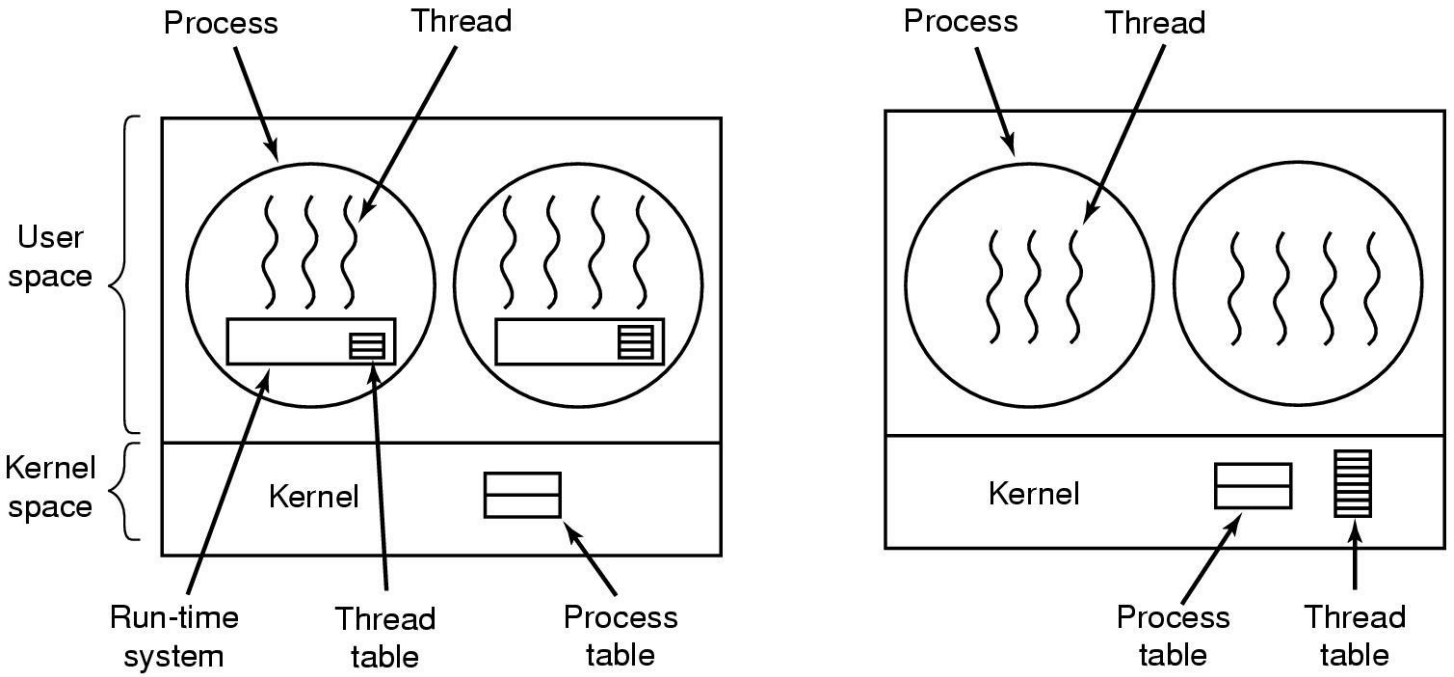
Εικόνα 2-14. Μερικές βασικές κλήσεις συστήματος της ομάδας Pthreads.

Τα νήματα στο POSIX (2)

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

#define NUMBER_OF_THREADS     10

void *print_hello_world(void *tid)
{
        /* This function prints the thread's identifier and then exits. */
        printf("Hello World. Greetings from thread %d0, tid);
        pthread_exit(NULL);
}

int main(int argc, char *argv[])
{
        /* The main program creates 10 threads and then exits. */
        pthread_t threads[NUMBER_OF_THREADS];
        int status, i;

        for(i=0; i < NUMBER_OF_THREADS; i++) {
                printf("Main here. Creating thread %d0, i);
                status = pthread_create(&threads[i], NULL, print_hello_world, (void *)i);

                if (status != 0) {
                        printf("Oops. pthread_create returned error code %d0, status);
                        exit(-1);
                }
        }
        exit(NULL);
}
```

Εικόνα 2-15. Παράδειγμα χρήσης νημάτων.

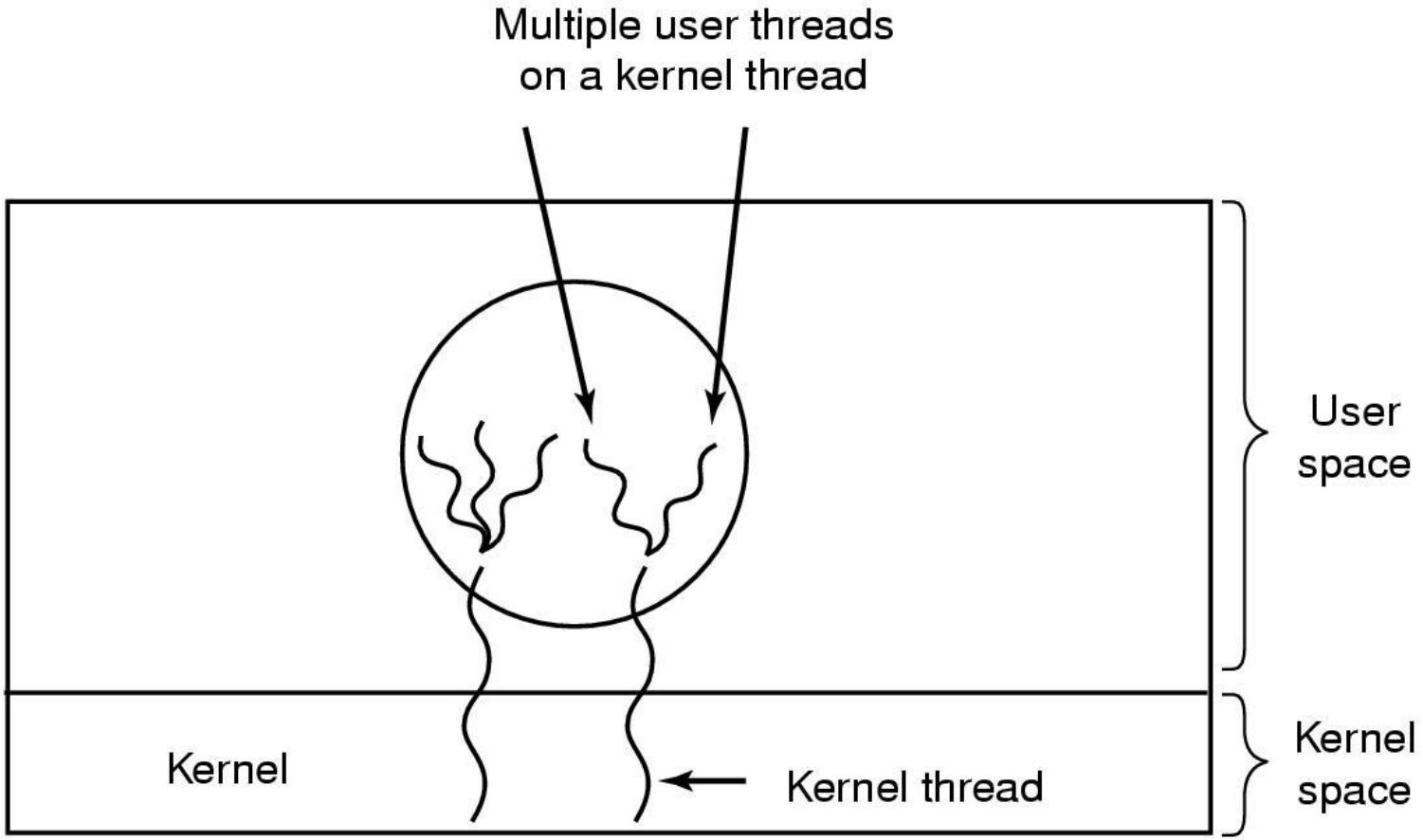
Υλοποίηση νημάτων στο χώρο του χρήστη (user space)



Εικόνα 2-16. (a) Πακέτο νημάτων επιπέδου χρήστη. (b) Πακέτο νημάτων που διαχειρίζεται από τον πυρήνα.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Υβριδικές υλοποιήσεις



Εικόνα 2-17. Πολύπλεξη νημάτων επιπέδου χρήστη σε νήματα επιπέδου πυρήνα.

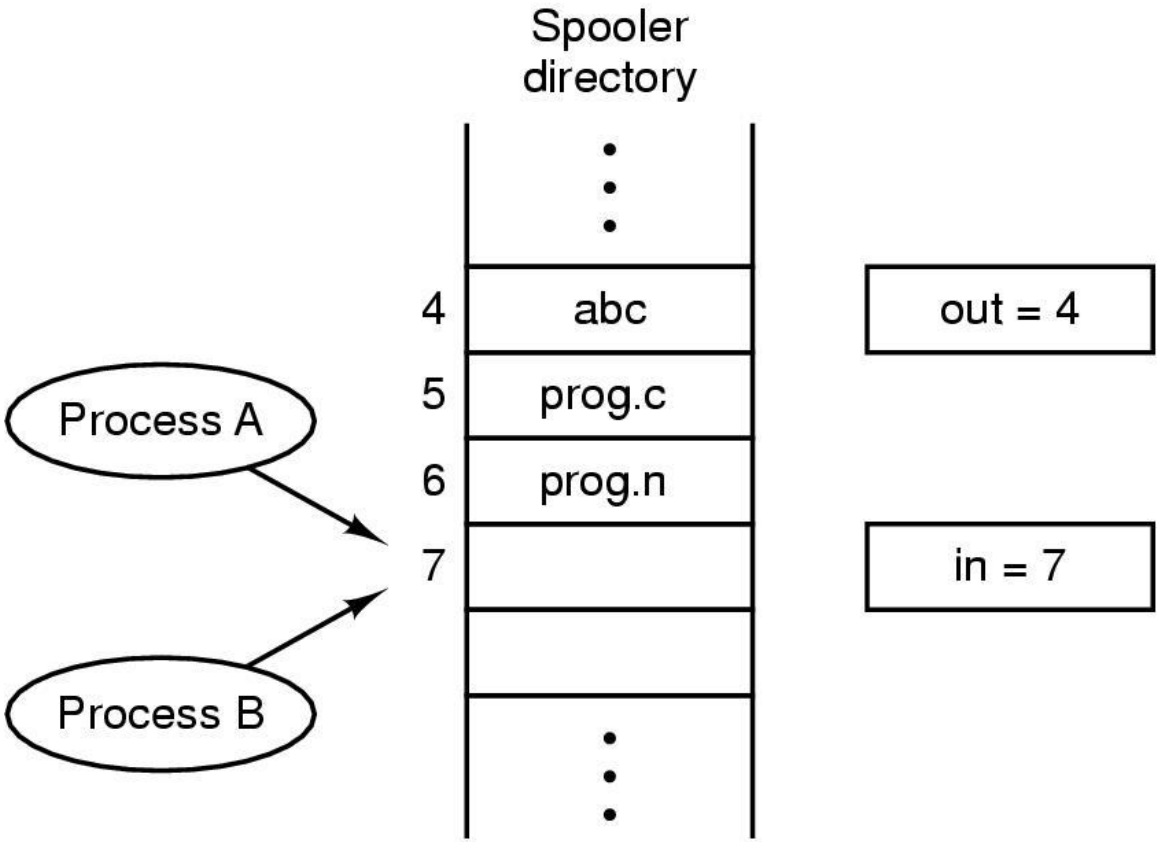
Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

2.3. Διαδιεργασιακή Επικοινωνία (Inter Process Communication)

Διαδιεργασιακή επικοινωνία

1. Με **ποιο τρόπο επικοινωνούν μεταξύ τους** δύο διεργασίες;
   - Μία σωλήνωση (pipe) επιτρέπει για παράδειγμα την επικοινωνία μεταξύ δύο διεργασιών.
   - Πολλά ΛΣ δημιουργούν «κοινόχρηστους» χώρους διευθύνσεων.
2. Πώς μπορούμε να εξασφαλίσουμε ότι μία διεργασία **δεν εμποδίζεται από μία άλλη**, όταν η 1η εκτελεί μία *κρίσιμη λειτουργία*;
3. Πως εξασφαλίζουμε το **σωστό συγχρονισμό** μεταξύ διεργασιών που πρέπει να επικοινωνήσουν;

Συνθήκες ανταγωνισμού



Εικόνα 2-21. Δύο διεργασίες που θέλουν ταυτόχρονα να προσπελάσουν κοινή μνήμη.

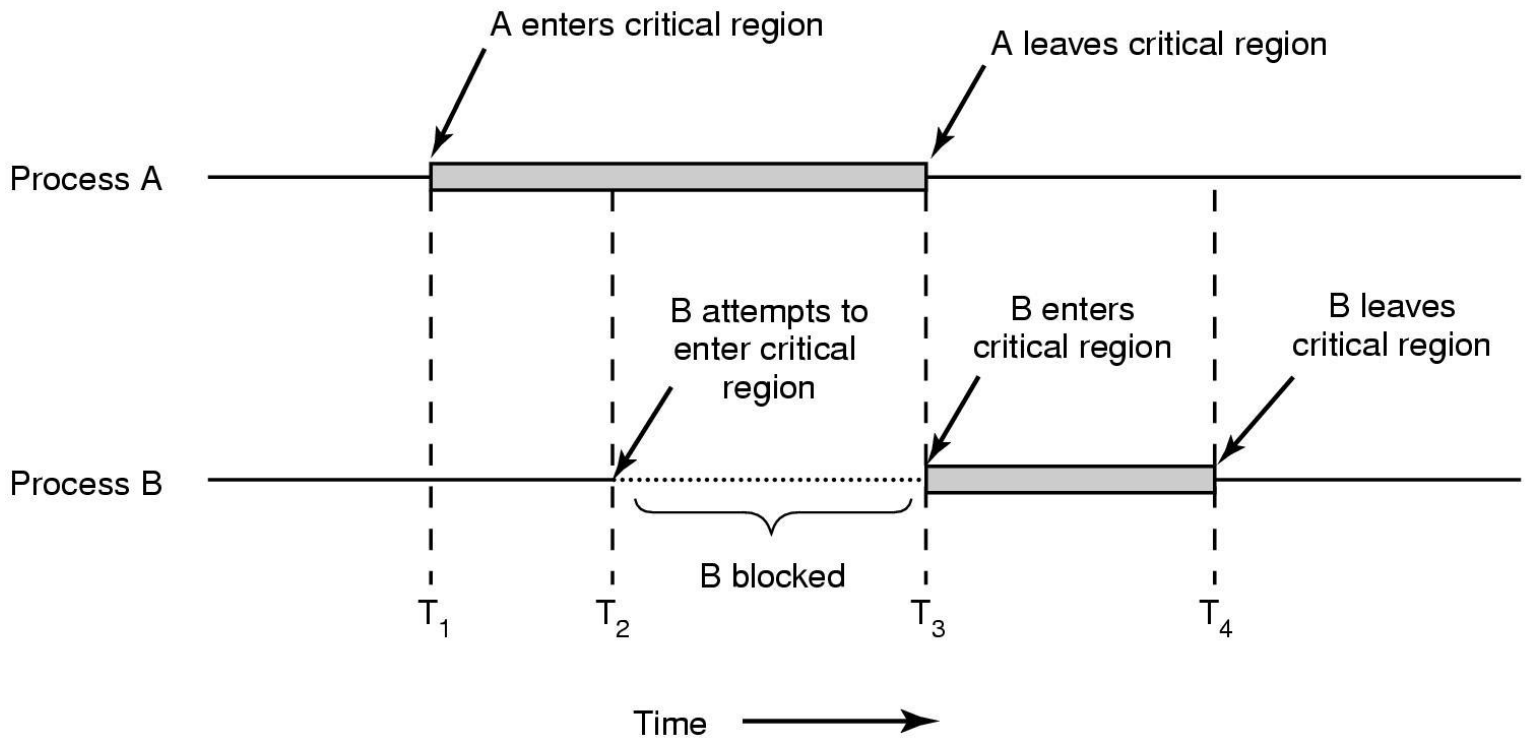
Κρίσιμες περιοχές (1)

Κρίσιμη περιοχή = τμήμα προγράμματος που γίνεται προσπέλαση κοινόχρηστης μνήμης.

Συνθήκες για την αποφυγή την ανταγωνισμού:

1. Δύο διεργασίες δεν βρίσκονται ***ποτέ ταυτόχρονα*** στις κρίσιμες περιοχές τους **(αμοιβαίος αποκλεισμός)**.
2. Δεν επιτρέπονται παραδοχές σχετικά με την ταχύτητα ή το πλήθος των CPU.
3. Αν μία διεργασία δεν βρίσκεται σε κρίσιμο τμήμα, δεν επιτρέπεται να μπλοκάρει άλλες διεργασίες.
4. Δεν επιτρέπεται η επ' αόριστο αναμονή μίας διεργασίας, για την είσοδό της στην κρίσιμη περιοχή της.

Κρίσιμες περιοχές (2)



Εικόνα 2-22. Αμοιβαίος αποκλεισμός με τη χρήση κρίσιμων περιοχών.

Αμοιβαίος αποκλεισμός μέσω αναμονής με απασχόληση (busy waiting)

Μέθοδοι για την επίτευξη αμοιβαίου αποκλεισμού:

1. Απενεργοποίηση διακοπών
2. Μεταβλητές κλειδώματος
3. Αυστηρή εναλλαγή
4. Η λύση του Peterson
5. Η εντολή TSL

Απενεργοποίηση διακοπών

- Όταν μπαίνει μία διεργασία στην κρίσιμη περιοχή της απενεργοποιεί όλες τις διακοπές. Τις ενεργοποιεί ξανά μόλις βγει από την κρίσιμη περιοχή της.
- Επιτρέπεται στις διεργασίες να απενεργοποιούν τις διακοπές ρολογιού.
- Μία κακόβουλη διεργασία μπορεί να μην επαναφέρει ποτέ τις διακοπές (συνεπώς να μην μπορέσει ποτέ να δοθεί η CPU σε άλλη διεργασία!)

Μεταβλητές κλειδώματος

- Χρήση κοινόχρηστης μεταβλητής κλειδώματος (lock variable).
- Κάθε διεργασία ελέγχει τη μεταβλητή κλειδώματος:
  - Αν είναι 0 μπαίνει στην κρίσιμη περιοχή και την αλλάζει σε 1.
  - Αν είναι 1, περιμένει για λίγο και ξαναδοκιμάζει.
- Πρόβλημα:
  - Η Α διαβάζει την μεταβλητή και τη βρίσκει 0. Συμβαίνει μία διακοπή ρολογιού.
  - Η Β διαβάζει τη μεταβλητή και τη βρίσκει 0. Την αλλάζει σε 1 και μπαίνει στην κρίσιμη περιοχή της. Συμβαίνει διακοπή ρολογιού.
  - Η Α συνεχίζει από εκεί που είχε μείνει. Αλλάζει την μεταβλητή σε 1 (*παρόλο που είναι ήδη δεν το ξέρει*) και μπαίνει και αυτή στην κρίσιμη περιοχή της!

Αυστηρή εναλλαγή

```
while (TRUE) {
    while (turn != 0)        /* loop */ ;
    critical_region( );
    turn = 1;
    noncritical_region( );
}
```

(a)

```
while (TRUE) {
    while (turn != 1)        /* loop */ ;
    critical_region( );
    turn = 0;
    noncritical_region( );
}
```

(b)

Εικόνα 2-23. Μία λύση στο πρόβλημα των κρίσιμων περιοχών. (a) Διεργασία 0. (b) Διεργασία 1. Και στις δύο περιπτώσεις, προσέξτε το ; στο τέλος της εντολής while.

Η λύση του Peterson

```
#define FALSE  0
#define TRUE   1
#define N      2                        /* number of processes */

int turn;                               /* whose turn is it? */
int interested[N];                      /* all values initially 0 (FALSE) */

void enter_region(int process);         /* process is 0 or 1 */
{
    int other;                          /* number of the other process */

    other = 1 – process;                /* the opposite of process */
    interested[process] = TRUE;         /* show that you are interested */
    turn = process;                     /* set flag */
    while (turn == process && interested[other] == TRUE) /* null statement */ ;
}

void leave_region(int process)          /* process: who is leaving */
{
    interested[process] = FALSE;        /* indicate departure from critical region */
}
```

Εικόνα 2-24. Η λύση του Peterson για την επίτευξη αμοιβαίου αποκλεισμού.

Η εντολή TSL

```
enter_region:
        TSL REGISTER,LOCK       | copy lock to register and set lock to 1
        CMP REGISTER,#0         | was lock zero?
        JNE enter_region        | if it was nonzero, lock was set, so loop
        RET                     | return to caller; critical region entered

leave_region:
        MOVE LOCK,#0            | store a 0 in lock
        RET                     | return to caller
```

Εικόνα 2-25. Είσοδος και έξοδος από την κρίσιμη περιοχή με την εντολή TSL.

Η ανάγνωση της λέξης LOCK και η αποθήκευση τιμής σε αυτή είναι ***αδιαίρετη πράξη***.

Αναμονή με απασχόληση και το πρόβλημα της αντιστροφής προτεραιοτήτων

- Οι διεργασίες Χ και Υ έχουν την ίδια κρίσιμη περιοχή (η Υ έχει υψηλή προτεραιότητα και η Χ χαμηλή προτεραιότητα)
- Η Χ έχει μπει στην κρίσιμη περιοχή της, αλλά δεν πρόλαβε να καλέσει την leave_region( )
- Γίνεται μία διακοπή (interrupt) και εκτελείται μία τρίτη διεργασία (η Ζ). Και οι δύο (Υ και Χ) είναι έτοιμες για εκτέλεση (ready).
- Γίνεται μία διακοπή (interrupt) και καλείται η Υ (έχει υψηλή προτεραιότητα). Η Υ κάνει αναμονή με απασχόληση μέχρι να μπορέσει να μπει στην κρίσιμη περιοχή της.
- Η Χ όμως δεν μπορεί να χρονοπρογραμματιστεί εφόσον εκτελείται η Υ!

Λήθαργος και αφύπνιση (sleep & wakeup)

- Για την ***αποφυγή της αναμονής με απασχόληση***, γίνεται χρήση του ζεύγους κλήσεων συστήματος sleep( ) και wakeup( ).
- sleep( )
  - Μπλοκάρει («κοιμίζει») την καλούσα διεργασία, μέχρι κάποια άλλη να την ξυπνήσει.
- wakeup(process)
  - Ξυπνάει μία διεργασία.

Το πρόβλημα παραγωγού-καταναλωτή (1)

- Γνωστό και ως *το πρόβλημα της περιορισμένης προσωρινής μνήμης (bounded-buffer problem)*.
- Δύο (ή περισσότερες) διεργασίες μοιράζονται μία σταθερού μήκους προσωρινή μνήμη (ένα buffer).
- Η διεργασία - παραγωγός τοποθετεί δεδομένα στον buffer.
  - Εάν ο buffer έχει γεμίσει, ο παραγωγός πρέπει να **κοιμηθεί**, μέχρις ότου αδειάσει τουλάχιστον μία θέση στον buffer.
- Η διεργασία – καταναλωτής αφαιρεί δεδομένα από τον buffer.
  - Εάν ο buffer είναι τελείως άδειος, ο καταναλωτής πρέπει να **κοιμηθεί**, μέχρις ότου γεμίσει τουλάχιστον μία θέση στον buffer.
- Είναι ευθύνη της διεργασίας παραγωγού (καταναλωτή) να **ξυπνήσει** την διεργασία καταναλωτή (παραγωγό).

Το πρόβλημα παραγωγού-καταναλωτή

```
#define N 100                                   /* number of slots in the buffer */
int count = 0;                                  /* number of items in the buffer */

void producer(void)
{
     int item;

     while (TRUE) {                             /* repeat forever */
          item = produce_item( );               /* generate next item */
          if (count == N) sleep( );             /* if buffer is full, go to sleep */
          insert_item(item);                    /* put item in buffer */
          count = count + 1;                    /* increment count of items in buffer */
          if (count == 1) wakeup(consumer);     /* was buffer empty? */
     }
}


void consumer(void)
{
     int item;

     while (TRUE) {                             /* repeat forever */
          if (count == 0) sleep( );             /* if buffer is empty, got to sleep */
          item = remove_item( );                /* take item out of buffer */
          count = count – 1;                    /* decrement count of items in buffer */
          if (count == N – 1) wakeup(producer); /* was buffer full? */
          consume_item(item);                   /* print item */
     }
}
```

Εικόνα 2-27. Το πρόβλημα παραγωγού-καταναλωτή με μία μοιραία συνθήκη συναγωνισμού.

Σηματοφόροι ή σημαφόροι (Semaphores)

- *Νέος τύπος μεταβλητής*
  - 0: δεν έχουν αποθηκευτεί σήματα αφύπνισης.
  - x>0: εκκρεμούν x σήματα αφύπνισης.
- Λειτουργία **down**:
  - Ελέγχει την τιμή του σημαφόρου.
    - Εάν >0, τον μειώνει κατά 1 και συνεχίζει.
    - Εάν =0 τότε η καλούσα διεργασία κοιμάται χωρίς να κάνει την μείωση κατά 1 (*η μείωση εκκρεμεί*).
  - Ο έλεγχος, η αλλαγή τιμής και η πιθανή κλήση της sleep είναι μία ***αδιαίρετη ενέργεια (atomic action)***.

Σηματοφόροι ή σημαφόροι (Semaphores)

- Λειτουργία **up**:
  - Αυξάνει την τιμή του σημαφόρου κατά 1.
  - Εάν κάποια λειτουργία down δεν είχε ολοκληρωθεί (*εκκρεμεί από πριν μία λειτουργία down*) τότε ολοκληρώνεται αυτή η λειτουργία (μειώνει κατά 1).
  - Μετά από μία λειτουργία up σε ένα σημαφόρο που έχει μπλοκάρει κάποιες διεργασίες, θα εξακολουθήσει να έχει την τιμή 0 αλλά *οι διεργασίες που βρίσκονται σε λήθαργο* ***θα έχουν μειωθεί κατά μία****!*
  - Οι λειτουργίες αύξησης του σημαφόρου και αφύπνισης της διεργασίας είναι επίσης **αδιαίρετες**.

Σηματοφόροι (Semaphores)

```
#define N 100                                   /* number of slots in the buffer */
typedef int semaphore;                          /* semaphores are a special kind of int */
semaphore mutex = 1;                            /* controls access to critical region */
semaphore empty = N;                            /* counts empty buffer slots */
semaphore full = 0;                             /* counts full buffer slots */

void producer(void)
{
     int item;

     while (TRUE) {                             /* TRUE is the constant 1 */
          item = produce_item( );                /* generate something to put in buffer */
          down(&empty);                          /* decrement empty count */
          down(&mutex);                          /* enter critical region */
          insert_item(item);                     /* put new item in buffer */
          up(&mutex);                            /* leave critical region */
          up(&full);                             /* increment count of full slots */
     }
}


void consumer(void)
{
     int item;

     while (TRUE) {                             /* infinite loop */
          down(&full);                           /* decrement full count */
          down(&mutex);                          /* enter critical region */
          item = remove_item( );                 /* take item from buffer */
          up(&mutex);                            /* leave critical region */
          up(&empty);                            /* increment count of empty slots */
          consume_item(item);                    /* do something with the item */
     }
}
```

Εικόνα 2-28. Επίλυση του προβλήματος παραγωγού-καταναλωτή με τη χρήση σηματοφόρων.

Τα Mutex

```
mutex_lock:
        TSL REGISTER,MUTEX      | copy mutex to register and set mutex to 1
        CMP REGISTER,#0         | was mutex zero?
        JZE ok                  | if it was zero, mutex was unlocked, so return
        CALL thread_yield       | mutex is busy; schedule another thread
        JMP mutex_lock          | try again
ok:     RET                     | return to caller; critical region entered

mutex_unlock:
        MOVE MUTEX,#0           | store a 0 in mutex
        RET                     | return to caller
```

Εικόνα 2-29. Υλοποίηση των διαδικασιών *mutex_lock* and *mutex_unlock*.

Χρήση mutex σε νήματα

- Όταν τα νήματα υλοποιούνται στο χώρο του χρήστη, τα mutex μπορούν να χρησιμοποιηθούν για αμοιβαίο αποκλεισμό των νημάτων.
- Κλήσεις συστήματος για τη δημιουργία, έλεγχο και καταστροφή των mutex σε pthreads
- Κλήσεις συστήματος για τη δημιουργία, έλεγχο και καταστροφή μεταβλητών συνθήκης (condition variables)
  - Οι μεταβλητές συνθήκης επιτρέπουν ή απαγορεύουν την πρόσβαση στην κρίσιμη περιοχή ανάλογα με κάποια συνθήκη (αν δεν ικανοποιείται η συνθήκη, δεν επιτρέπεται η πρόσβαση στο critical region).

Χρήση των Mutex σε νήματα Pthreads

Thread call	Description
Pthread_mutex_init	Create a mutex
Pthread_mutex_destroy	Destroy an existing mutex
Pthread_mutex_lock	Acquire a lock or block
Pthread_mutex_trylock	Acquire a lock or fail
Pthread_mutex_unlock	Release a lock

Εικόνα 2-30. Μερικές από τις κλήσεις Pthreads που σχετίζονται με τα mutex.

Ελεγκτές (Monitors) (1)

- Οι σηματοφόροι και τα mutex λειτουργούν σε χαμηλό επίπεδο
- Οι ελεγκτές (monitors) είναι μία λύση για συγχρονισμό μεταξύ διεργασιών και αποφυγή συνθηκών συναγωνισμού, σε υψηλότερο επίπεδο.
  - Αφορά γλώσσες προγραμματισμού υψηλότερου επιπέδου, όπως η java.
- Μέσα σε ένα ελεγκτή (monitor) περιλαμβάνονται πολλές διεργασίες. Κάθε φορά όμως μπορεί να είναι ενεργή μόνο μία διαδικασία!
- Χρησιμοποιούν μεταβλητές συνθήκης και κλήσεις wait και signal για μπλοκάρισμα και αφύπνιση.

Ελεγκτές (Monitors) (2)

```
monitor example
        integer i;
        condition c;

        procedure producer( );
        .
        .
        .
        end;

        procedure consumer( );
        .           .           .
        end;
end monitor;
```

Εικόνα 2-33. Παράδειγμα ελεγκτή.

Ελεγκτές (Monitors) (3)

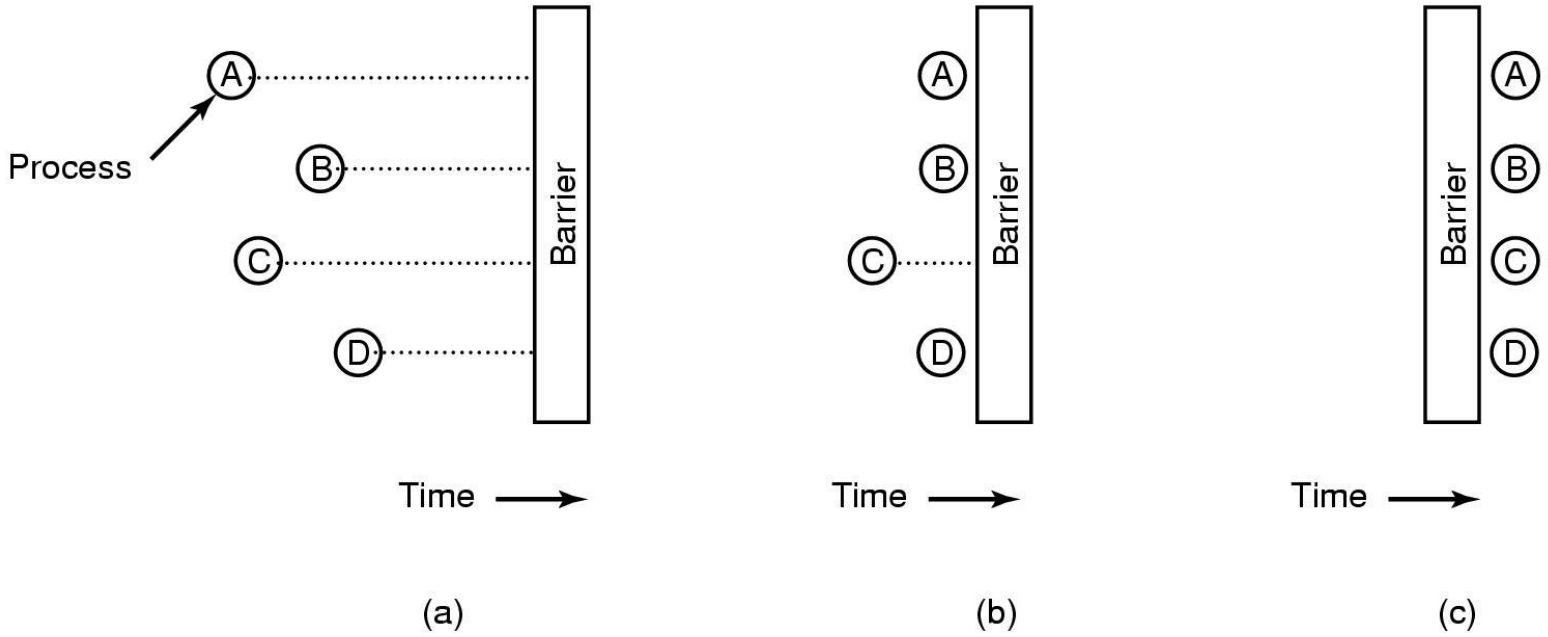
```
monitor ProducerConsumer
        condition full, empty;
        integer count;

        procedure insert(item: integer);
        begin
                if count = N then wait(full);
                insert_item(item);
                count := count + 1;
                if count = 1 then signal(empty)
        end;

        function remove: integer;
        begin
                if count = 0 then wait(empty);
                remove = remove_item;
                count := count – 1;
                if count = N – 1 then signal(full)
        end;

        count := 0;
end monitor;

procedure producer;
begin
        while true do
        begin
                item = produce_item;
                ProducerConsumer.insert(item)
        end
end;

procedure consumer;
begin
        while true do
        begin
                item = ProducerConsumer.remove;
                consume_item(item)
        end
end;
```

Εικόνα 2-34. Σε κάθε χρονική στιγμή χρησιμοποιείται μία μόνο διαδικασία του ελεγκτή.

Φράγματα (Barriers) (1)

- Μέθοδος συγχρονισμού που αφορά ομάδες διεργασιών.
- Ένα φράγμα σταματά όσες διεργασίες της ομάδας έχουν ολοκληρώσει κάποια φάση.
- Κάθε διεργασία που φτάνει στο φράγμα, εκτελεί την εργασία της και καλεί την κλήση barrier για να μπλοκαριστεί.
- Όταν όλες οι διεργασίες της ομάδας έχουν καλέσει την barrier, τότε μπορούν να συνεχίσουν.

Φράγματα (Barriers)



Εικόνα 2-37. Χρήση φράγματος.

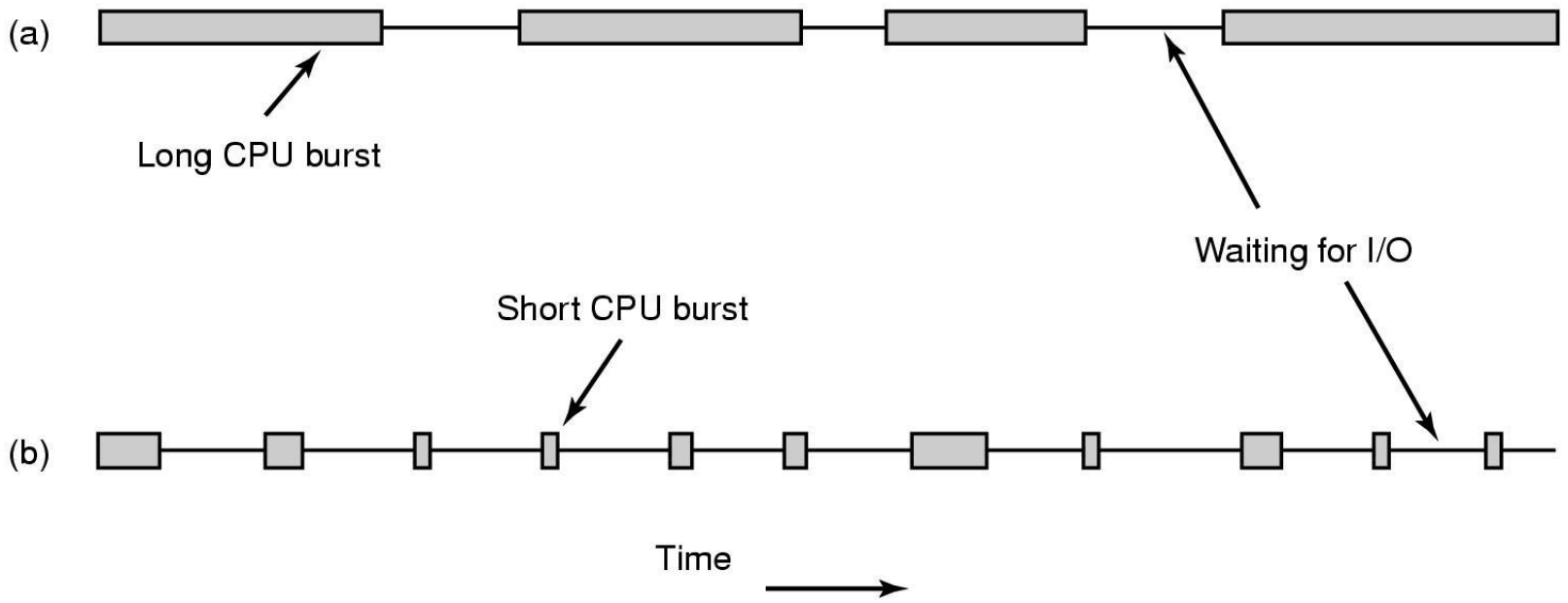
(a) Οι διεργασίες πλησιάζουν το φράγμα.

(b) Όλες οι διεργασίες εκτός από μία έχουν μπλοκαριστεί στο φράγμα.

(c) Όταν φτάσει και η τελευταία στο φράγμα, επιτρέπεται πλέον σε όλες να περάσουν.

2.4. Χρονοπρογραμματισμός (Scheduling)

Η χρονοπρογραμματιστική συμπεριφορά των διεργασιών



Εικόνα 2-38. Τα διαστήματα χρήσης της CPU εναλλάσσονται με περιόδους αναμονής για Ε/Ε. (a) Μια διεργασία εξαρτημένη από τη CPU. (b) Μία διεργασία εξαρτημένη από την Ε/Ε.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Πότε γίνεται χρονοπρογραμματισμός

1. Όταν μία διεργασία που εκτελείται τερματίσει.
2. Όταν περισσότερες από μία διεργασίες είναι έτοιμες για εκτέλεση (ready).
3. Όταν μία διεργασία που εκτελείται μπλοκάρεται από Ε/Ε, σημαφόρο κτλ.
4. Όταν συμβαίνει μία διακοπή Ε/Ε (interrupt).
5. Το ρολόι του συστήματος προκαλεί περιοδικές διακοπές (π.χ. στα 50Hz). Σε κάθε διακοπή λαμβάνεται μία απόφαση χρονοπρογραμματισμού.

Προεκτοπιστικοί και μη-προεκτοπιστικοί αλγόριθμοι χρονοπρογραμματισμού

- *Μη προεκτοπιστικοί αλγόριθμοι* (non-preemtive)
  - Σε κάθε διακοπή ρολογιού επιλέγουν μία διεργασία η οποία θα εκτελεστεί μέχρι να επιστρέψει εθελοντικά τη CPU ή μέχρι να μπλοκαριστεί για Ε/Ε.
  - Δεν λαμβάνονται αποφάσεις χρονοπρογραμματισμού στις διακοπές ρολογιού.
- *Προεκτοπιστικοί αλγόριθμοι* (preemtive)
  - Σε κάθε διακοπή ρολογιού επιλέγουν μία διεργασία η οποία θα εκτελεστεί για ένα μέγιστο χρονικό διάστημα.
  - Εάν δεν έχει ολοκληρωθεί σε αυτό το χρονικό διάστημα, αναστέλλεται και θα συνεχίσει κάποια άλλη στιγμή.

Κατηγορίες αλγορίθμων χρονοπρογραμματισμού

(Α) Αλγόριθμοι για περιβάλλοντα δέσμης (batch)

(Β) Αλγόριθμοι για αλληλεπιδραστικά περιβάλλοντα (interactive)

(Γ) Αλγόριθμοι για περιβάλλοντα πραγματικού χρόνου (real-time)

Στόχοι αλγόριθμων χρονοπρογραμματισμού

Για όλα τα συστήματα

- **Δικαιοσύνη:** Να εκχωρείται σε κάθε διεργασία ένα μερίδιο της CPU.
- **Επιβολή της πολιτικής:** Να παρακολουθείται εάν εφαρμόζεται η καθορισμένη πολιτική.
- **Ισορροπία:** Να διατηρούνται ενεργά όλα τα τμήματα του συστήματος.

Για τα συστήματα δέσμης

- **Διεκπεραιωτική ικανότητα:** Μεγιστοποίηση των εργασιών που ολοκληρώνονται ανά ώρα.
- **Χρόνος διεκπεραίωσης:** Ελαχιστοποίηση του χρόνου μεταξύ υποβολής και ολοκλήρωσης μιας διεργασίας.
- **Αξιοποίηση της CPU:** Συνεχής απασχόληση της CPU.

Για τα αλληλεπιδραστικά συστήματα

- **Χρόνος απόκρισης:** Ταχύτατη απόκριση στις αιτήσεις.
- **Τήρηση αναλογικότητας:** Ικανοποίηση των λογικών προσδοκιών των χρηστών.

Για τα συστήματα πραγματικού χρόνου

- **Τήρηση προθεσμιών:** Αποφυγή απώλειας δεδομένων.
- **Προβλεψιμότητα:** Αποφυγή υποβιβασμού ποιότητας πολυμέσων.

Εικόνα 2-39. Μερικοί στόχοι των αλγορίθμων χρονοπρογραμματισμού για διαφορετικές συνθήκες.

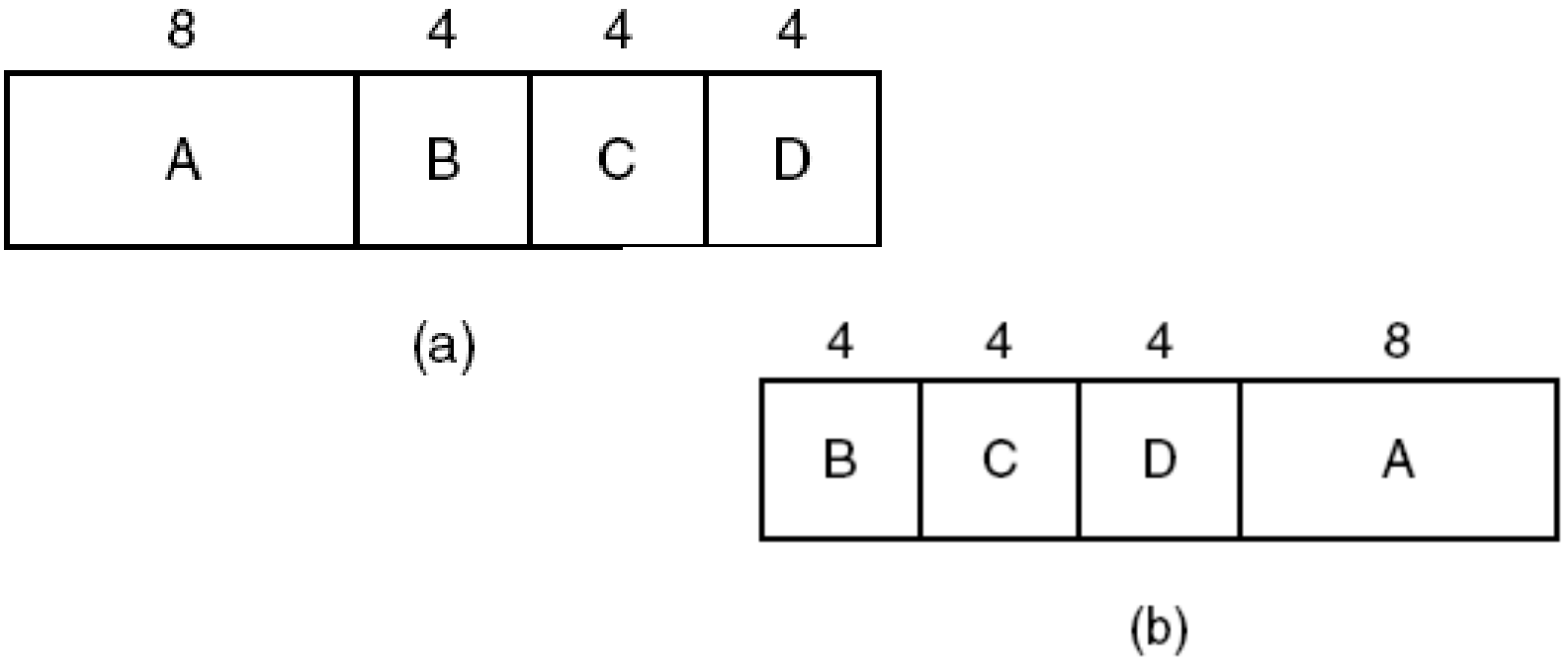
(Α) Χρονοπρογραμματισμός σε συστήματα δέσμης

- **Με βάση τη σειρά άφιξης** (First-come first-served)
- **Με βάση τη μικρότερη διάρκεια** (Shortest job first)
- **Με βάση τη μικρότερη υπολειπόμενη διάρκεια** (Shortest remaining Time next)

(Α) Χρονοπρογραμματισμός σε συστήματα δέσμης *(Με βάση τη σειρά άφιξης)*

- Η εξυπηρέτηση με βάση τη σειρά άφιξης δημιουργεί μία μοναδική ουρά FIFO (First-In-First-Out).
- Οι νέες διεργασίες τοποθετούνται στο τέλος της λίστας.
- Εάν η εκτελούμενη εργασία μπλοκαριστεί, πηγαίνει στο τέλος της ουράς και ο αλγόριθμος συνεχίζει.
- Πολύ απλός αλγόριθμος. Η υλοποίηση μπορεί να γίνει με μία **συνδεδεμένη λίστα**.
- *Μειονέκτημα αλγορίθμου:*
  - Όταν υπάρχουν λίγες διεργασίες εξαρτημένες από τη CPU και πολλές εξαρτημένες από Ε/Ε γίνεται σπατάλη της CPU.

(Α) Χρονοπρογραμματισμός σε συστήματα δέσμης
(Εξυπηρέτηση με βάση τη διάρκεια)



Εικόνα 2-40. Παράδειγμα χρονοπρογραμματισμού με βάση τη μικρότερη διάρκεια (Shortest Job First).
(a) Εκτέλεση 4 εργασιών με βάση την αρχική σειρά.
(b) Εκτέλεση με βάση τη μικρότερη διάρκεια.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

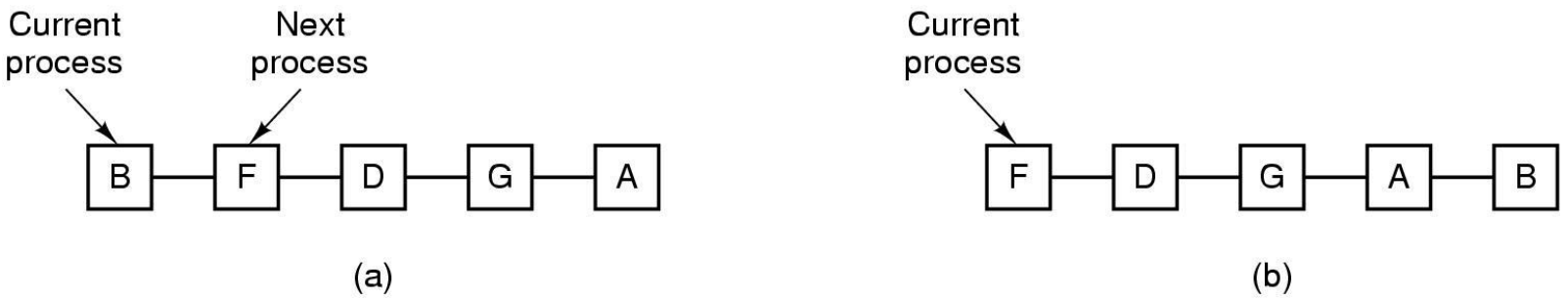
(Β) Χρονοπρογραμματισμός σε αλληλεπιδραστικά συστήματα

- **Χρονοπρογραμματισμός εκ περιτροπής** (Round-robin scheduling)
- **Χρονοπρογραμματισμός με βάση την προτεραιότητα** (Priority scheduling)
- **Πολλαπλές ουρές** (Multiple queues)
- **Εξυπηρέτηση με βάση τη μικρότερη διάρκεια** (Shortest process next)
- **Εγγυημένος χρονοπρογραμματισμός** (Guaranteed scheduling)
- **Χρονοπρογραμματισμός με λοταρία** (Lottery scheduling)
- **Χρονοπρογραμματισμός δίκαιης διανομής** (Fair-share scheduling)

(Β) Σε αλληλεπιδραστικά συστήματα ***(Round-Robin Scheduling)*** (1/2)

- Από τους παλαιότερους και δικαιότερους αλγορίθμους.
- Σε κάθε διεργασία εκχωρείται ένα ***κβάντο χρόνου*** (quantum time).
  - Εάν δεν έχει ολοκληρωθεί, μπλοκάρεται και πηγαίνει στο τέλος της λίστας.
- Ζητήματα:
  - *Πόσο διαρκεί το κβάντο χρόνου;*
  - *Πόσο διαρκεί η εναλλαγή μεταξύ των διεργασιών (process switch);*

(Β) Σε αλληλεπιδραστικά συστήματα ***(Round-Robin Scheduling)*** (1/2)

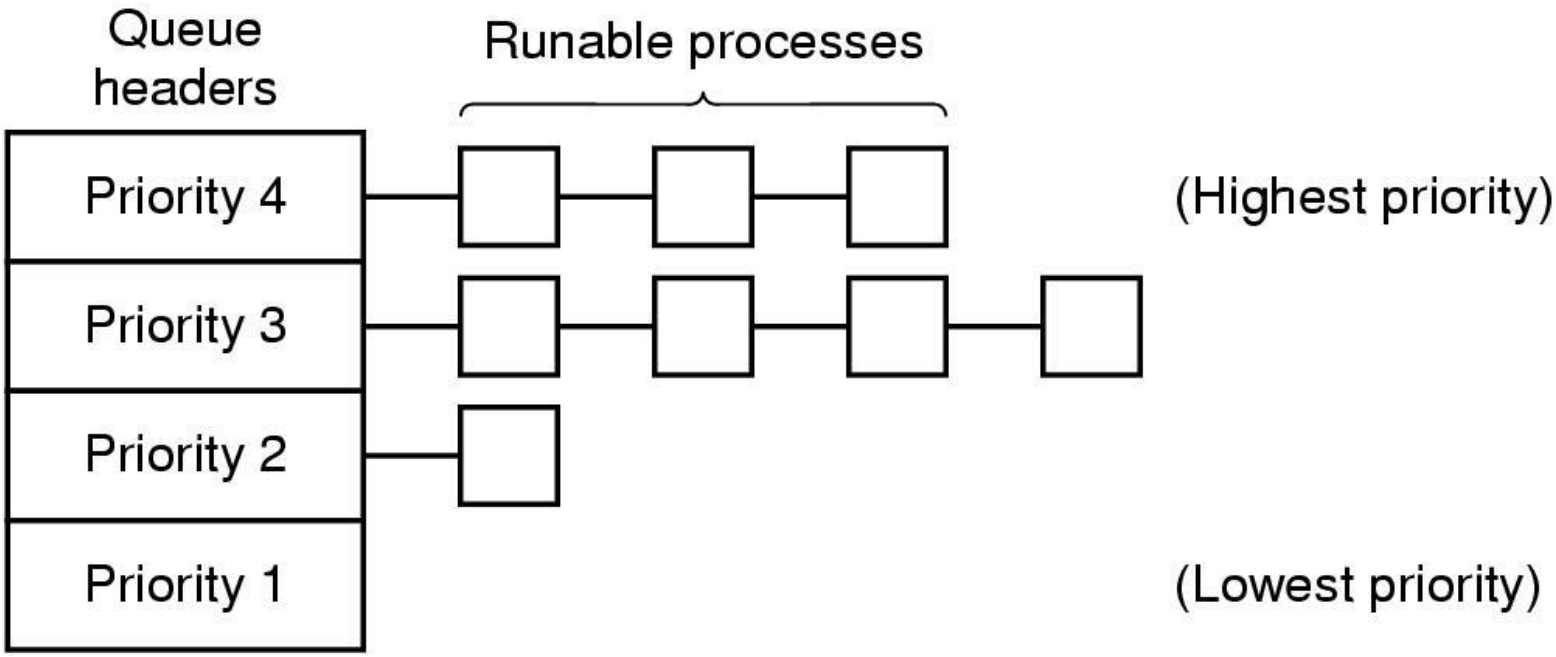


Εικόνα 2-41. Χρονοπρογραμματισμός εκ περιτροπής . (a) Η λίστα των εκτελέσιμων διαδικασιών. (b) Η λίστα των εκτελέσιμων διαδικασιών μετά από τη χρήση του κβάντου που αντιστοιχεί στη διεργασία Β.

(Β) Σε αλληλεπιδραστικά συστήματα
(Με βάση την προτεραιότητα) (1/2)

- Σε κάθε διεργασία αντιστοιχίζεται μία προτεραιότητα.
- Κάθε στιγμή εκτελείται μία διαθέσιμη (runable) διεργασία με την υψηλότερη προτεραιότητα.
- Η προτεραιότητα των διεργασιών μπορεί να μειώνεται μετά από κάθε διακοπή ρολογιού, ώστε να μπορέσουν να εκτελεστούν και οι υπόλοιπες.
- Εναλλακτικά, κάθε διεργασία μπορεί να έχει ένα μέγιστο κβάντο χρόνου.
- Οι προτεραιότητες μπορεί να ανατεθούν στατικά η δυναμικά.
  - Εντολή *nice* στο Unix.

(Β) Σε αλληλεπιδραστικά συστήματα
(Με βάση την προτεραιότητα) (2/2)



Εικόνα 2-42. Ένας αλγόριθμος χρονοπρογραμματισμού με τέσσερις κλάσεις προτεραιοτήτων.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

(Γ) Χρονοπρογραμματισμός σε συστήματα πραγματικού χρόνου

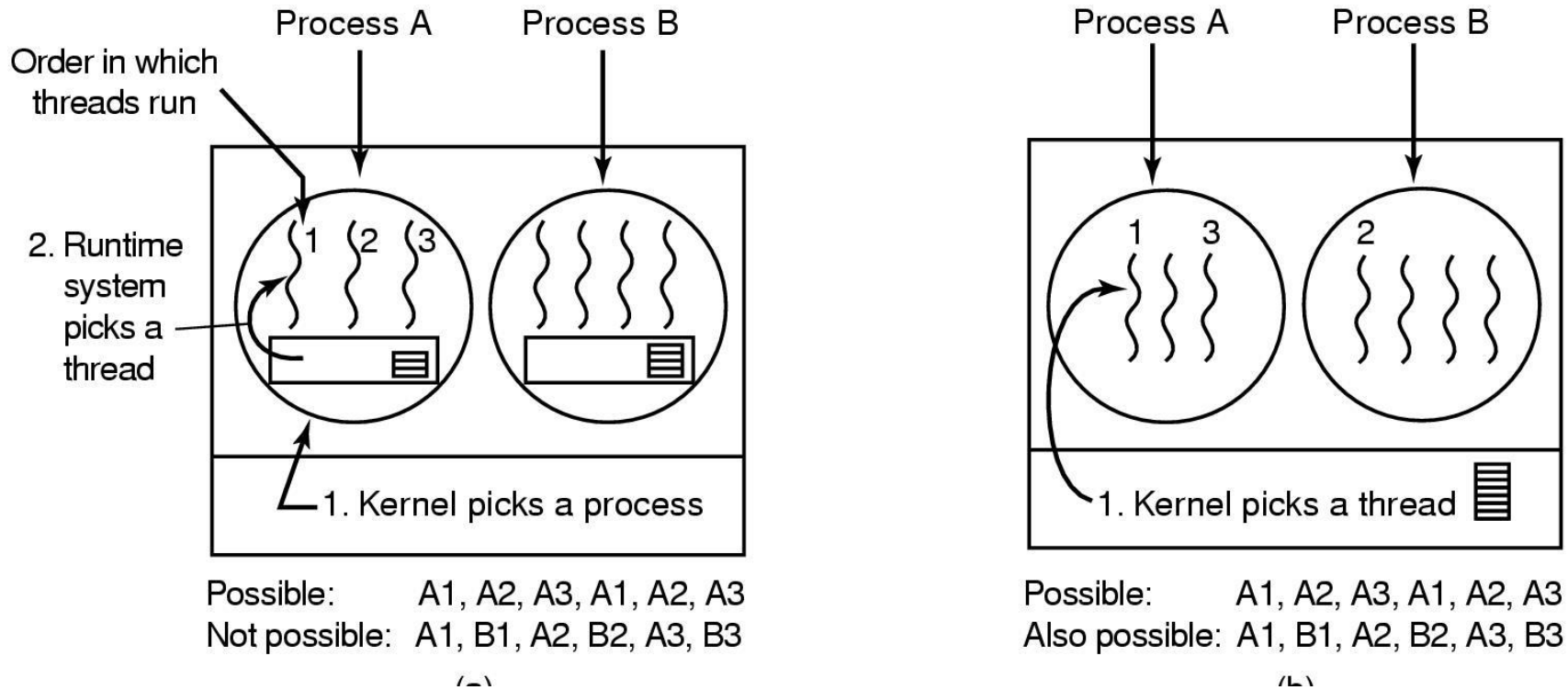
- Χρονοπρογραμματίσιμα συστήματα πραγματικού χρόνου (Schedulable real-time system)
- Έστω ότι ισχύει:
  - Συμβαίνουν ***m*** περιοδικά γεγονότα
  - Το γεγονός ***i*** συμβαίνει με περίοδο P_i και απαιτεί C_i δευτερόλεπτα
- Τότε το σύστημα είναι δυνατό να χειριστεί το συνολικό φορτίο εργασίας εάν ισχύει:

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

Πολιτική και μηχανισμός

- Διαχωρισμός μεταξύ τι επιτρέπεται να γίνει και πως γίνεται
  - Μία διεργασία γνωρίζει ποιες θυγατρικές διεργασίες της είναι σημαντικές και χρειάζονται προτεραιότητα
- Ο αλγόριθμος χρονοπρογραμματισμού είναι παραμετροποιήσιμος
  - Ο μηχανισμός βρίσκεται στον πυρήνα
- Οι παράμετροι συμπληρώνονται από τις διεργασίες επιπέδου χρήστη
  - Η πολιτική τίθεται από τις διεργασίες χρήστη

Χρονοπρογραμματισμός νημάτων

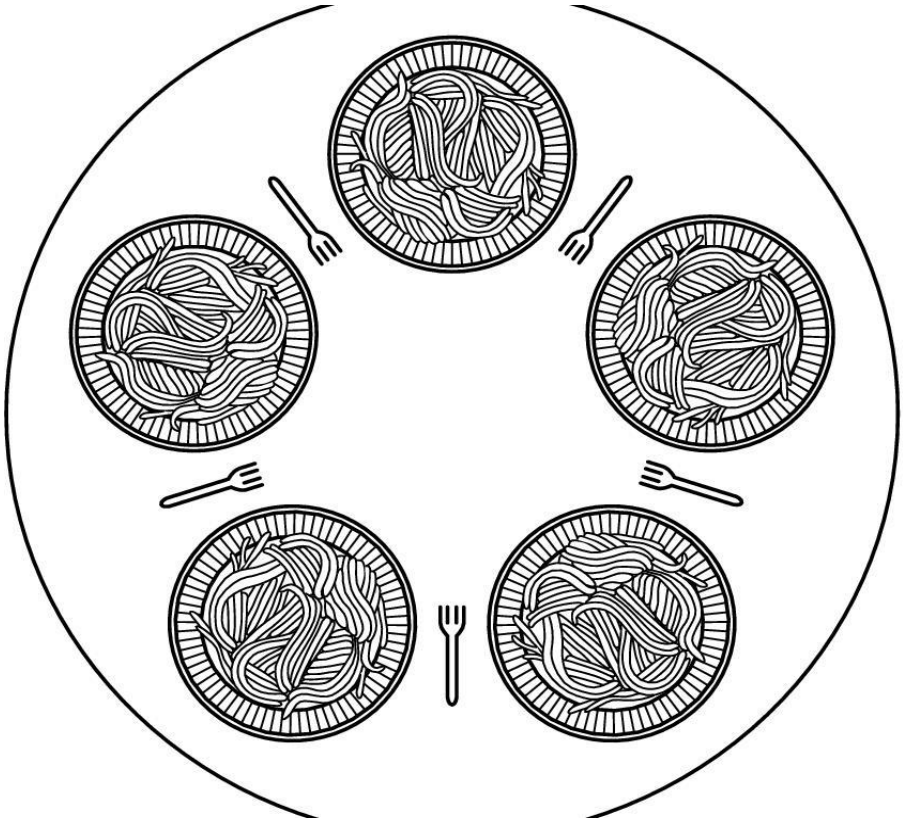


Εικόνα 2-43. (a) Πιθανός χρονοπρογραμματισμός νημάτων επιπέδου χρήστη και (b) επιπέδου πυρήνα

Κβάντο χρόνου διεργασιών 50 msec και νήματος 5 msec, κάθε φορά που τους εκχωρείται η CPU.

Διαδιεργα2.5. Κλασικά Προβλήματα σιακής Επικοινωνίας

Το πρόβλημα του δείπνου των φιλοσόφων (1)



Εικόνα 2-44. Ώρα φαγητού στο Τμήμα Φιλοσοφίας.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Το πρόβλημα του δείπνου των φιλοσόφων (2)

```
#define N 5                                     /* number of philosophers */

void philosopher(int i)                         /* i: philosopher number, from 0 to 4 */
{
    while (TRUE) {
        think( );                               /* philosopher is thinking */
        take_fork(i);                           /* take left fork */
        take_fork((i+1) % N);                   /* take right fork; % is modulo operator */
        eat( );                                 /* yum-yum, spaghetti */
        put_fork(i);                            /* put left fork back on the table */
        put_fork((i+1) % N);                    /* put right fork back on the table */
    }
}
```

Εικόνα 2-45. Μία μη λύση στο πρόβλημα του δείπνου των φιλοσόφων.

Το πρόβλημα του δείπνου των φιλοσόφων (3)

```
#define N                5                  /* number of philosophers */
#define LEFT             (i+N–1)%N          /* number of i's left neighbor */
#define RIGHT            (i+1)%N            /* number of i's right neighbor */
#define THINKING         0                  /* philosopher is thinking */
#define HUNGRY           1                  /* philosopher is trying to get forks */
#define EATING           2                  /* philosopher is eating */
typedef int semaphore;                      /* semaphores are a special kind of int */
int state[N];                               /* array to keep track of everyone's state */
semaphore mutex = 1;                        /* mutual exclusion for critical regions */
semaphore s[N];                             /* one semaphore per philosopher */

void philosopher(int i)                     /* i: philosopher number, from 0 to N–1 */
{
      while (TRUE) {                        /* repeat forever */
            think( );                       /* philosopher is thinking */
            take_forks(i);                  /* acquire two forks or block */
            eat( );                         /* yum-yum, spaghetti */
            put_forks(i);                   /* put both forks back on table */
      }
}
   . . .
```

Εικόνα 2-46. Μία ικανοποιητική λύση στο πρόβλημα.

Το πρόβλημα του δείπνου των φιλοσόφων (4)

```
. . .

void take_forks(int i)                  /* i: philosopher number, from 0 to N–1 */
{
        down(&mutex);                   /* enter critical region */
        state[i] = HUNGRY;              /* record fact that philosopher i is hungry */
        test(i);                        /* try to acquire 2 forks */
        up(&mutex);                     /* exit critical region */
        down(&s[i]);                    /* block if forks were not acquired */
}

. . .
```

Εικόνα 2-46. Μία ικανοποιητική λύση στο πρόβλημα.

Το πρόβλημα του δείπνου των φιλοσόφων (5)

```
. . .
void put_forks(i)                          /* i: philosopher number, from 0 to N–1 */
{
        down(&mutex);                      /* enter critical region */
        state[i] = THINKING;               /* philosopher has finished eating */
        test(LEFT);                        /* see if left neighbor can now eat */
        test(RIGHT);                       /* see if right neighbor can now eat */
        up(&mutex);                        /* exit critical region */
}

void test(i) /* i: philosopher number, from 0 to N–1 */
{
        if (state[i] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING) {
                state[i] = EATING;
                up(&s[i]);
        }
}
```

Εικόνα 2-46. Μία ικανοποιητική λύση στο πρόβλημα.

3η ενότητα

ΔΙΑΧΕΙΡΙΣΗ ΜΝΗΜΗΣ

Διαχείριση μνήμης

Περιεχόμενα

Η κύρια μνήμη (RAM)

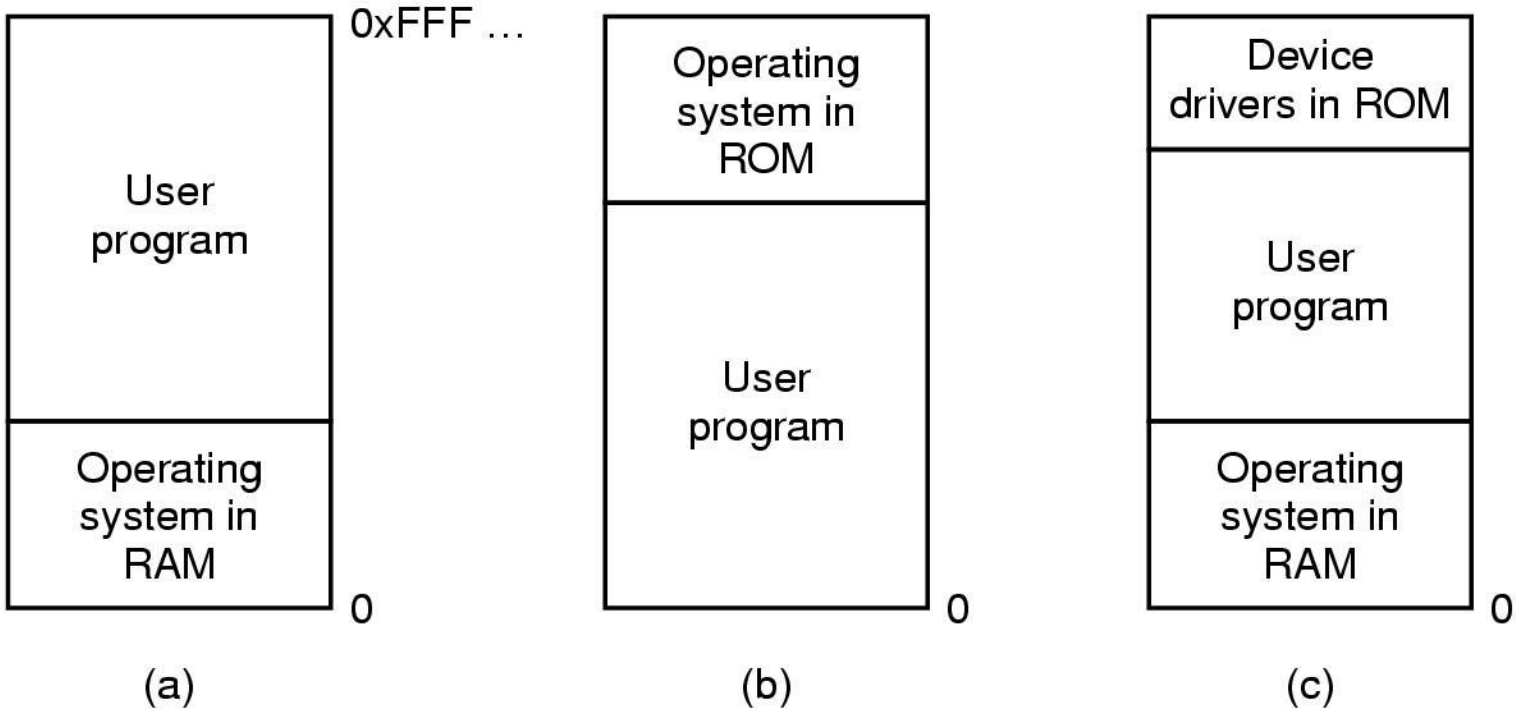
- Ιδεατά, ένας Η/Υ θα έπρεπε να έχει μία απείρως μεγάλη και πολύ γρήγορη μνήμη. Στην πράξη δεν είναι εφικτό και χρησιμοποιείται μία ιεραρχία μνήμης (registers, cache, RAM, δίσκος κτλ – βλέπε 1η ενότητα).
- Το Λ.Σ. είναι υπεύθυνο να *διαχειριστεί* αυτή την ιεραρχία μνήμης.
- Το τμήμα του Λ.Σ. που κάνει αυτή τη διαχείριση λέγεται ***διαχειριστής μνήμης (memory manager).***
  - Ποια τμήματα της μνήμης χρησιμοποιούνται;
  - Ποιες διεργασίες χρειάζονται μνήμη;
  - Ποιες διεργασίες δεν την χρειάζονται πλέον;
- Για τη διαχείριση της μνήμης χρησιμοποιούνται διάφορα ***αφαιρετικά μοντέλα*** (abstractions).

3.1 Διαχείριση μνήμης χωρίς αφαίρεση μνήμης (no memory abstraction)

Χωρίς αφαίρεση μνήμης (1)

- Η απλούστερη λύση η οποία εφαρμόστηκε στα πρώτα συστήματα. Κάθε πρόγραμμα βλέπει *όλη τη διαθέσιμη φυσική μνήμη*.
- Η εντολή **MOV REGISTER1, 1000** αντέγραφε τα περιεχόμενα του register στην πραγματική θέση 1000 της μνήμης.
- Δεν ήταν δυνατό *να βρίσκονται ταυτόχρονα 2 ή περισσότερα προγράμματα στη μνήμη!*
  - Χωρίς προστασία της μνήμης, κάθε πρόγραμμα θα μπορούσε να διαβάσει ή να γράψει σε μία φυσική θέση που χρησιμοποιεί κάποιο άλλο πρόγραμμα.

Χωρίς αφαίρεση μνήμης (2)

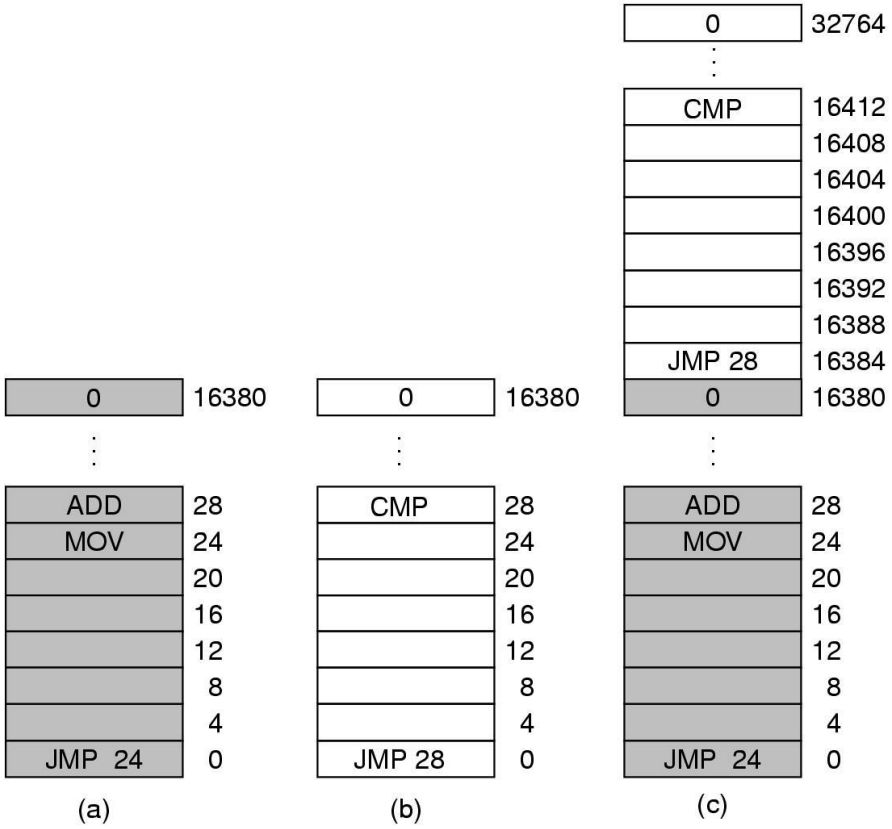


Εικόνα 3-1. Τρεις απλοί τρόποι οργάνωσης της μνήμης με ένα λειτουργικό σύστημα και μία διεργασία χρήστη. Υπάρχουν επίσης και άλλες πιθανές καταστάσεις.

Εκτέλεση πολλών προγραμμάτων χωρίς αφαίρεση μνήμης (1)

- Προσθήκη **ειδικού υλικού προστασίας** (h/w protection) της πρόσβασης στη μνήμη.
  - Η μνήμη διαιρείται σε **block** (των 2 KB).
  - Σε κάθε μπλοκ αντιστοιχίζεται ***ένα κλειδί προστασίας*** (4 bit) το οποίο αποθηκεύεται σε καταχωρητές της CPU.
  - Για 1 MB μνήμης απαιτούνται μόλις 256 byte αποθηκευτικού χώρου! (512 καταχωρητές × 4 bit)
  - Ο καταχωρητής PSW περιείχε επίσης ένα κλειδί των 4 bit.
  - Το υλικό προστασίας εμπόδιζε την πρόσβαση σε μία λέξη μνήμης, ***εάν το κλειδί του PSW ήταν διαφορετικό από το κλειδί προστασίας*** του μπλοκ μνήμης.
  - Μόνο το ΛΣ είχε δυνατότητα να αλλάξει τις τιμές των κλειδιών προστασίας.

Εκτέλεση πολλών προγραμμάτων χωρίς αφαίρεση μνήμης (2)



Εικόνα 3-2. Παρουσίαση του προβλήματος της επανατοποθέτησης. (a)Ένα πρόγραμμα των 16KB. (b) Ένα 2ο πρόγραμμα των 16KB. (c) Τα δύο προγράμματα φορτωμένα διαδοχικά στη μνήμη.

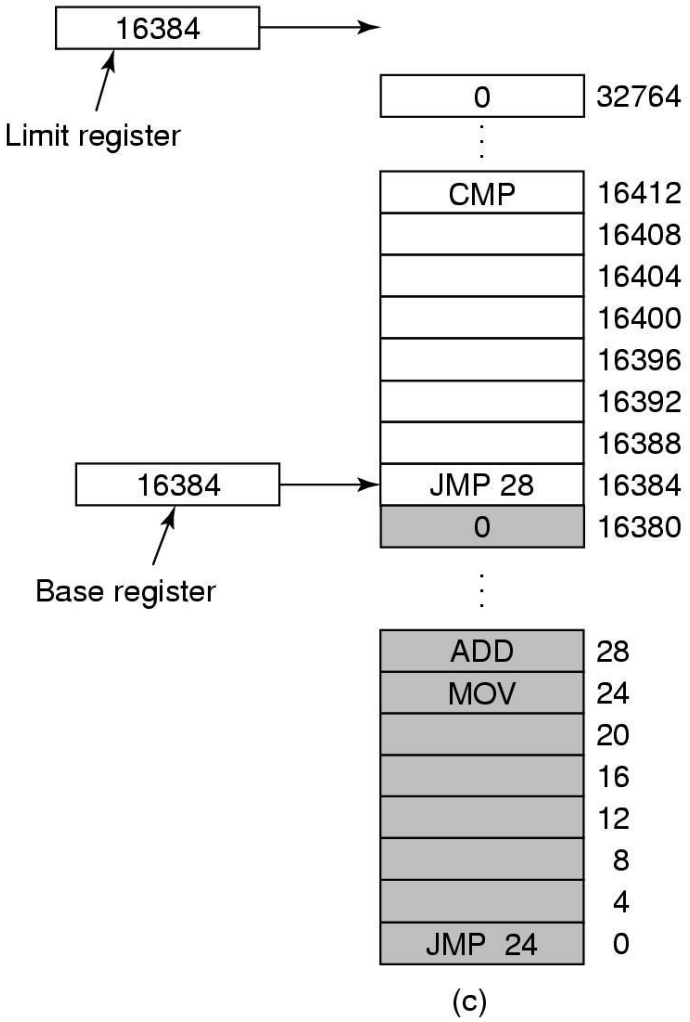
Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

3.2 Μία αφαίρεση μνήμης: Χώροι διευθύνσεων (Memory address spaces)

Χώροι διευθύνσεων (address space) (1)

- ***Χώρος διευθύνσεων (address space)***: ο χώρος που έχει στη μνήμη ένα πρόγραμμα.
  - Με τις διεργασίες το Λ.Σ. θεωρεί ότι κάθε πρόγραμμα έχει τη δική του CPU.
  - Με τους χώρους μνήμης, το Λ.Σ. θεωρεί ότι *κάθε πρόγραμμα έχει τη δική του RAM*, ανεξάρτητη από τη μνήμη των άλλων προγραμμάτων (εκτός από ειδικές περιπτώσεις).
- 1η λύση: ***Δυναμική επανατοποθέτηση (dynamic relocation)***
  - Χρήση δύο registers στη CPU.
  - Καταχωρητής βάσης (base register): από πού ξεκινάει το πρόγραμμα.
  - Καταχωρητής ορίου (limit register): το μήκος του προγράμματος.

Καταχωρητές βάσης και ορίου

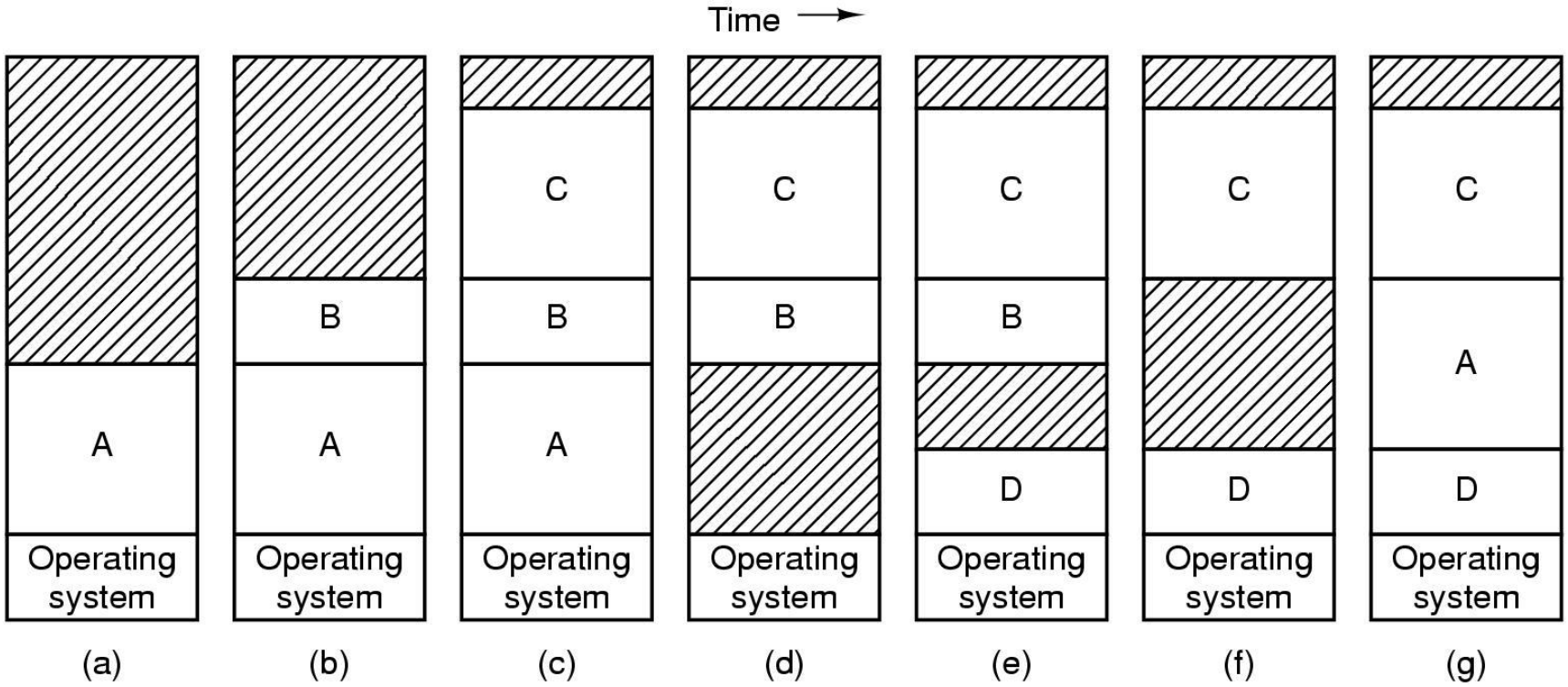


Εικόνα 3-3. Οι καταχωρητές βάσης και ορίου δίνουν σε κάθε διεργασία το δικό της χώρο διευθύνσεων

Εναλλαγή (swapping) (1)

- Εάν η διαθέσιμη μνήμη δεν επαρκεί για όλα τα προγράμματα, τότε η λύση των καταχωρητών base και limit δεν επαρκεί.
- 2η λύση: ***Εναλλαγή (swapping)***
  - Το Λ.Σ. μεταφέρει κάποιες διεργασίες στη μνήμη για να μπορούν να εκτελεστούν στη CPU.
  - Μετά από κάποιο χρονικό διάστημα, τις μεταφέρει στο δίσκο.
  - Οι αδρανείς διεργασίες θα πρέπει να βρίσκονται στο δίσκο ώστε να μην καταλαμβάνουν μνήμη.
  - Περιοδικά γίνεται έλεγχος ώστε να έχουν μεταφερθεί κάποια στιγμή, όλες οι διεργασίες στη μνήμη.
  - Κάθε διεργασία θα είναι είτε **ολόκληρη στη μνήμη** είτε **ολόκληρη στο δίσκο.**

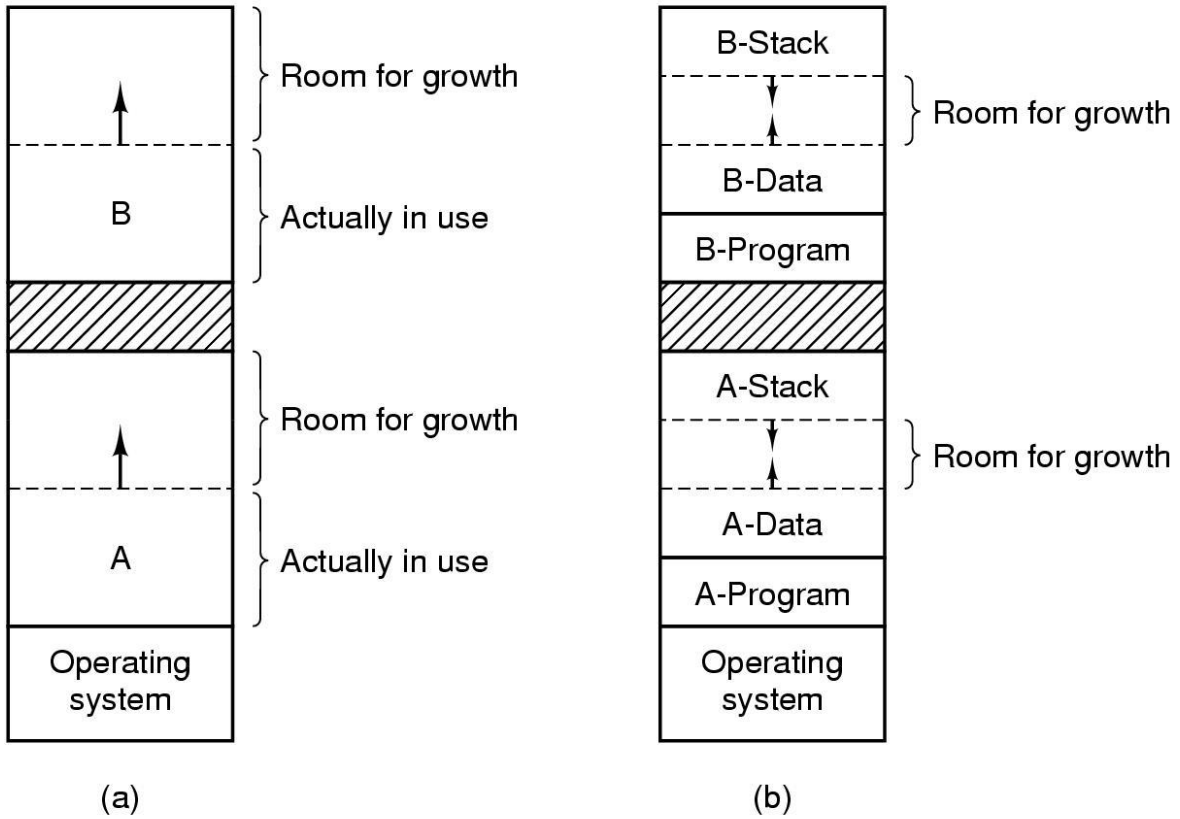
Εναλλαγή (swapping) (2)



Εικόνα 3-4. Αλλαγές στην κατανομή της μνήμης, καθώς οι διεργασίες μεταφέρονται στη μνήμη και αφαιρούνται από αυτή. Οι αλλαγές αυτές δημιουργούν κενούς χώρους μεταξύ των προγραμμάτων.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Εναλλαγή (swapping) (3)



Εικόνα 3-5. (a) Κατανομή μνήμης για ένα τμήμα δεδομένων που είναι πιθανό να επεκταθεί.

(b) Κατανομή μνήμης για μια στοίβα και ένα τμήμα δεδομένων που μπορεί να επεκταθούν.

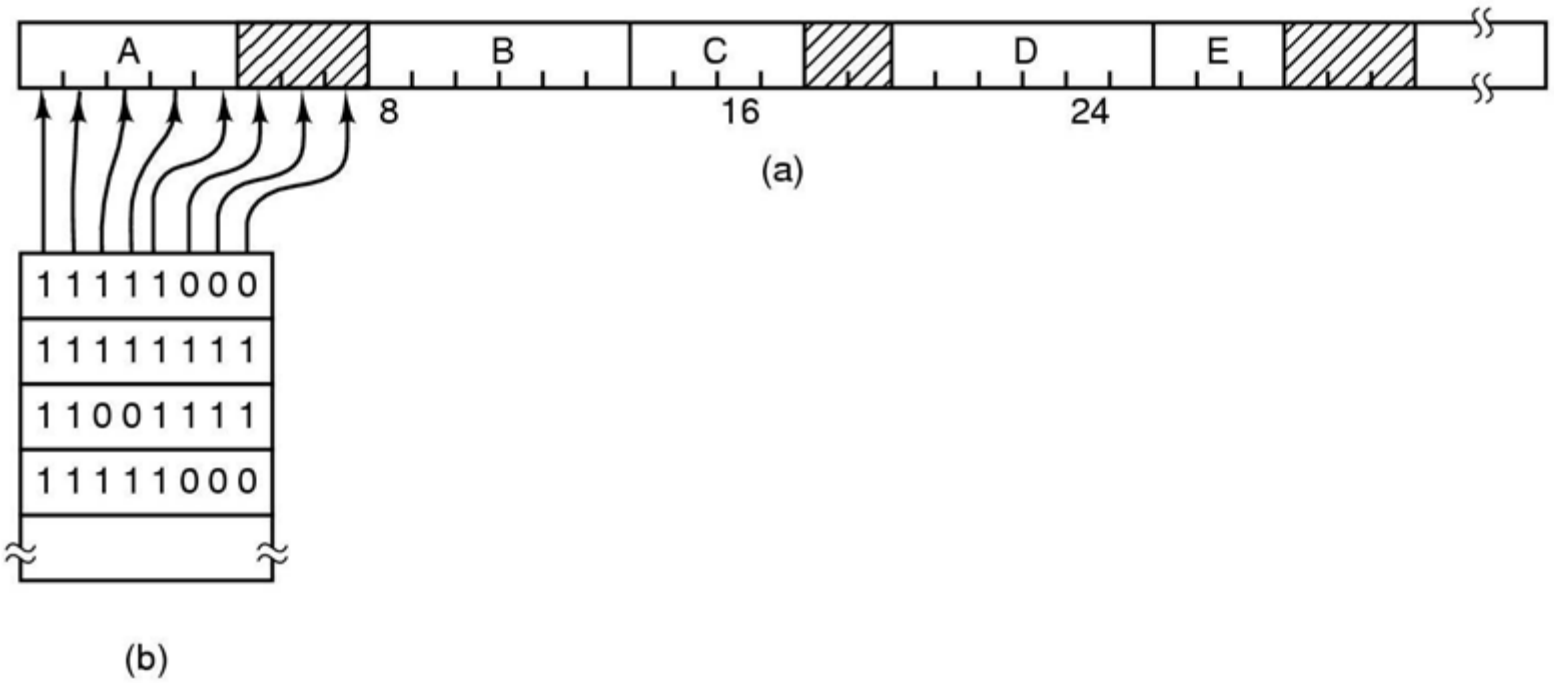
Διαχείριση ελεύθερης μνήμης

- Για τη δυναμική εκχώρηση της μνήμης, θα πρέπει το Λ.Σ. να έχει κάποιο μηχανισμό για τη διαχείρισή της.
- Τεχνικές διαχείρισης ελεύθερης μνήμης
  1. Με χάρτες bit (bitmaps)
  2. Με συνδεδεμένες λίστες (free lists)

Διαχείριση μνήμης με χάρτες bit (1)

- Η μνήμη διαιρείται σε μονάδες κατανομής (allocation units).
- Σε κάθε allocation unit αντιστοιχεί ένα bit στον χάρτη bit.
  - bit = 0: η μονάδα είναι ελεύθερη.
  - bit = 1: η μονάδα είναι κατειλημμένη.
- Μέγεθος της μονάδας κατανομής.
  - *Μικρό allocation unit*: μεγάλος χάρτης bit.
  - *Μεγάλο allocation unit*: σπατάλη μνήμης στο τελευταίο unit που χρησιμοποιεί κάθε διεργασία.

Διαχείριση μνήμης με χάρτες bit (2)



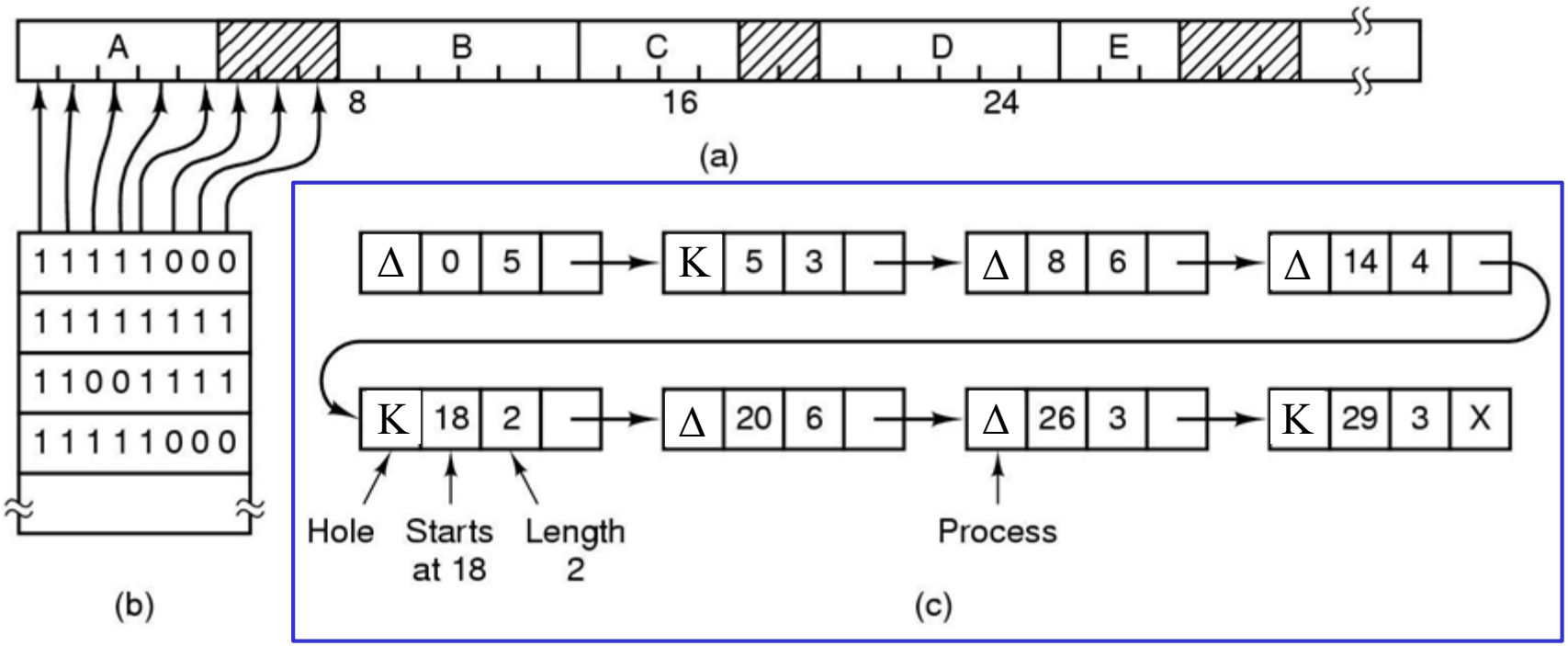
Εικόνα 3-6. (a) Ένα τμήμα της μνήμης με πέντε διεργασίες και τρεις οπές. Οι διαβαθμίσεις δείχνουν τις μονάδες κατανομής. Οι γραμμοσκιασμένες περιοχές (που συμβολίζονται με (0) στο χάρτη bit) είναι ελεύθερες. (b) Ο αντίστοιχος χάρτης bit.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Διαχείριση μνήμης με συνδεδεμένες λίστες (1)

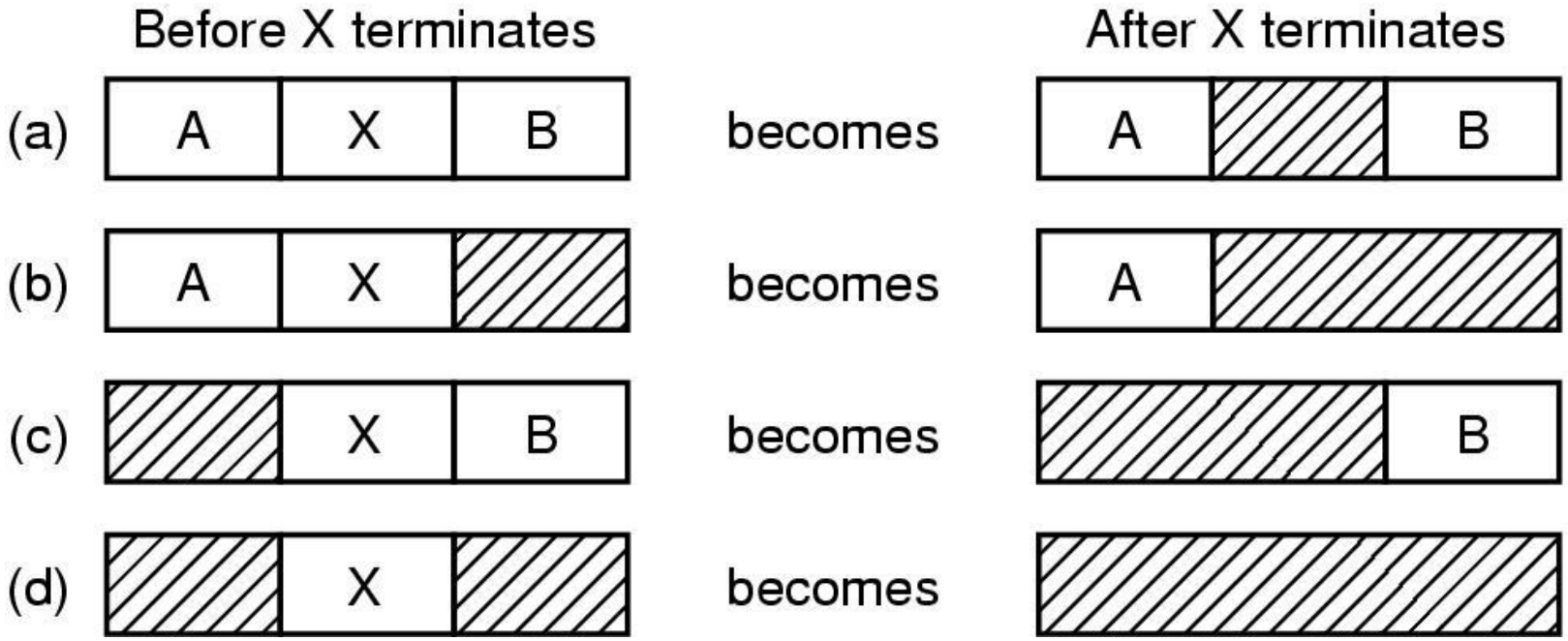
- Η συνδεδεμένη λίστα περιέχει πληροφορίες για το ***ποια*** και **πόσα** units είναι γεμάτα ή είναι ελεύθερα.
- Κάθε καταχώρηση στη λίστα έχει 4 πεδία:
  - Μία ένδειξη για το εάν το τμήμα της μνήμης περιέχει μία (Δ)ιεργασία ή ένα (Κ)ενό (είναι άδειο).
  - Τη διεύθυνση εκκίνησης.
  - Το μέγεθος του τμήματος.
  - Ένα δείκτη στην επόμενη καταχώρηση.
- Μόλις ολοκληρωθεί μία διεργασία, μπορεί η λίστα να ενημερωθεί πολύ εύκολα.

Διαχείριση μνήμης με συνδεδεμένες λίστες (2)



Εικόνα 3-6. (c) Οι ίδιες πληροφορίες με τη μορφή λίστας.

Διαχείριση μνήμης με συνδεδεμένες λίστες (3)



Εικόνα 3-7. Τέσσερις συνδυασμοί για τους γείτονες της διεργασίας X, η οποία ολοκληρώνεται.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

3.3 Εικονική μνήμη (virtual memory)

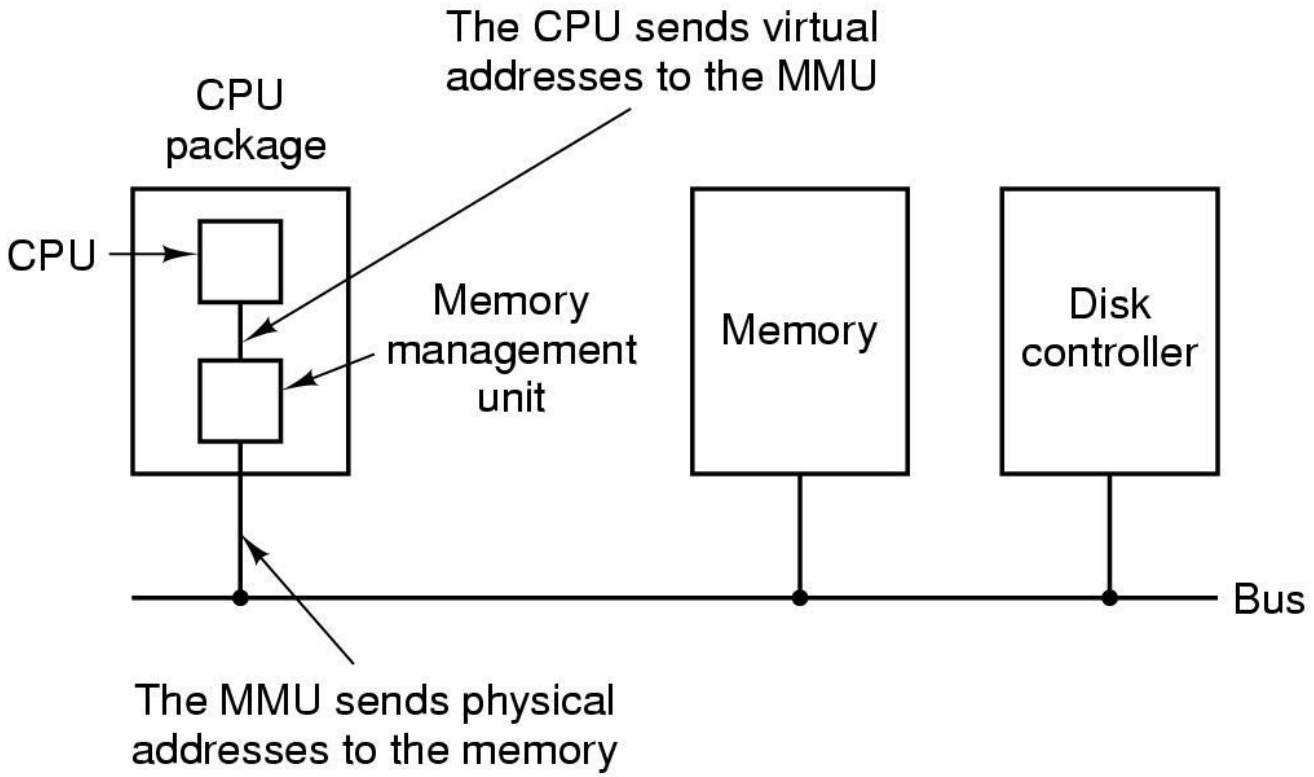
Εικονική Μνήμη (virtual memory)

- Ανάγκη εκτέλεσης μεγάλων προγραμμάτων (μεγαλύτερων από τη διαθέσιμη μνήμη).
- Η εναλλαγή (swapping) ολόκληρων των προγραμμάτων δεν είναι αποδοτική (λόγω της ταχύτητας του δίσκου).
- Η εικονική μνήμη (virtual memory) επιλύει το πρόβλημα, διαιρώντας τα προγράμματα σε **σελίδες μνήμης (pages)**.
- Δεν χρειάζεται να βρίσκονται ταυτόχρονα όλες οι σελίδες μνήμης ενός προγράμματος στη φυσική μνήμη.
- Όταν χρειάζεται πρόσβαση σε ένα τμήμα του προγράμματος που δεν βρίσκεται στη μνήμη, **θα φορτωθεί δυναμικά η αντίστοιχη σελίδα**.

Σελιδοποίηση (paging) (1)

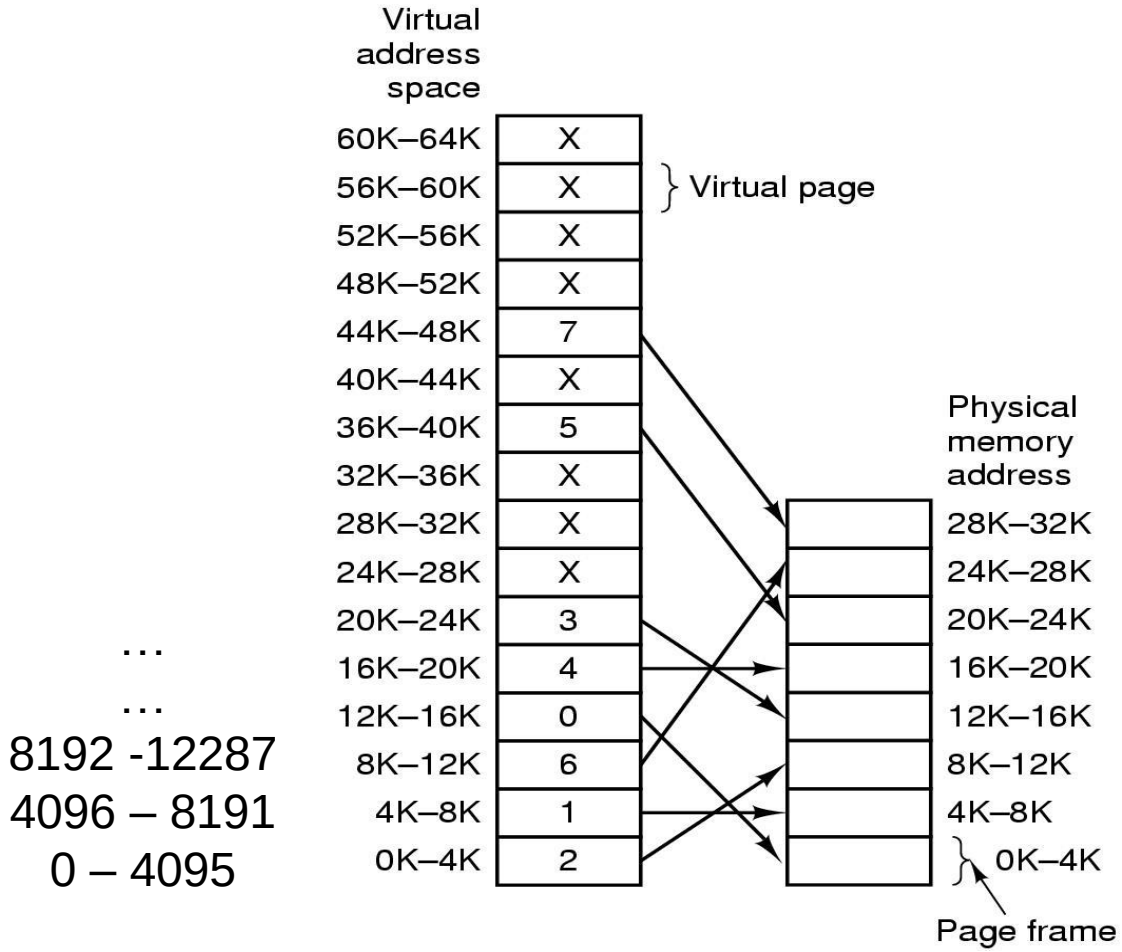
- Οι διευθύνσεις που δημιουργούνται από τα προγράμματα είναι ***εικονικές διευθύνσεις*** (virtual addresses).
- Άθροισμα όλων των εικονικών διευθύνσεων = **χώρος εικονικών διευθύνσεων** (virtual address space).
- Οι εικονικές διευθύνσεις μεταφέρονται από το δίσκο στη μνήμη μέσω της **Μονάδας Διαχείρισης Μνήμης** (Memory Management Unit – MMU).

Σελιδοποίηση (2)



Εικόνα 3-8. Η θέση και η λειτουργία της μονάδας διαχείρισης μνήμης (memory management unit – MMU). Η MMU αποτελεί τμήμα του τσιπ της CPU. Θα μπορούσε να βρίσκεται σε ξεχωριστό τσιπ (συνέβαινε παλαιότερα).

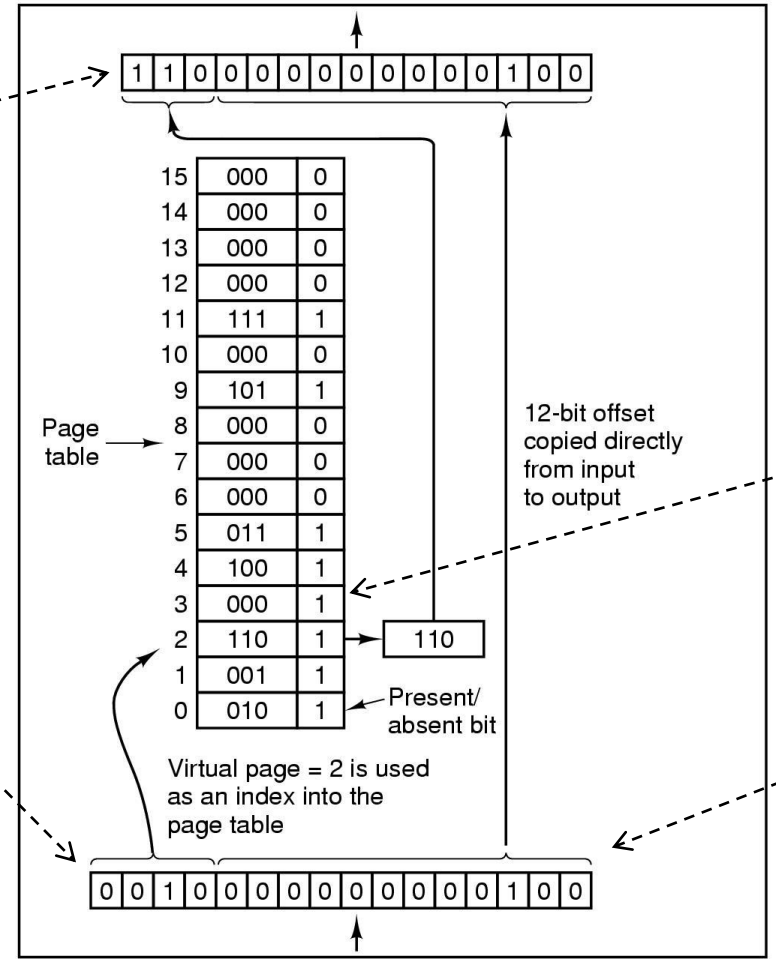
Σελιδοποίηση (3)



Εικόνα 3-9. Η σχέση μεταξύ των εικονικών διευθύνσεων και των διευθύνσεων της φυσικής μνήμης δίνεται από **τον πίνακα σελίδων**.

Πίνακες σελίδων

(4) Αντιγραφή του αριθμού πλαισίου που περιέχεται στην θέση 2 του πίνακα σελίδων



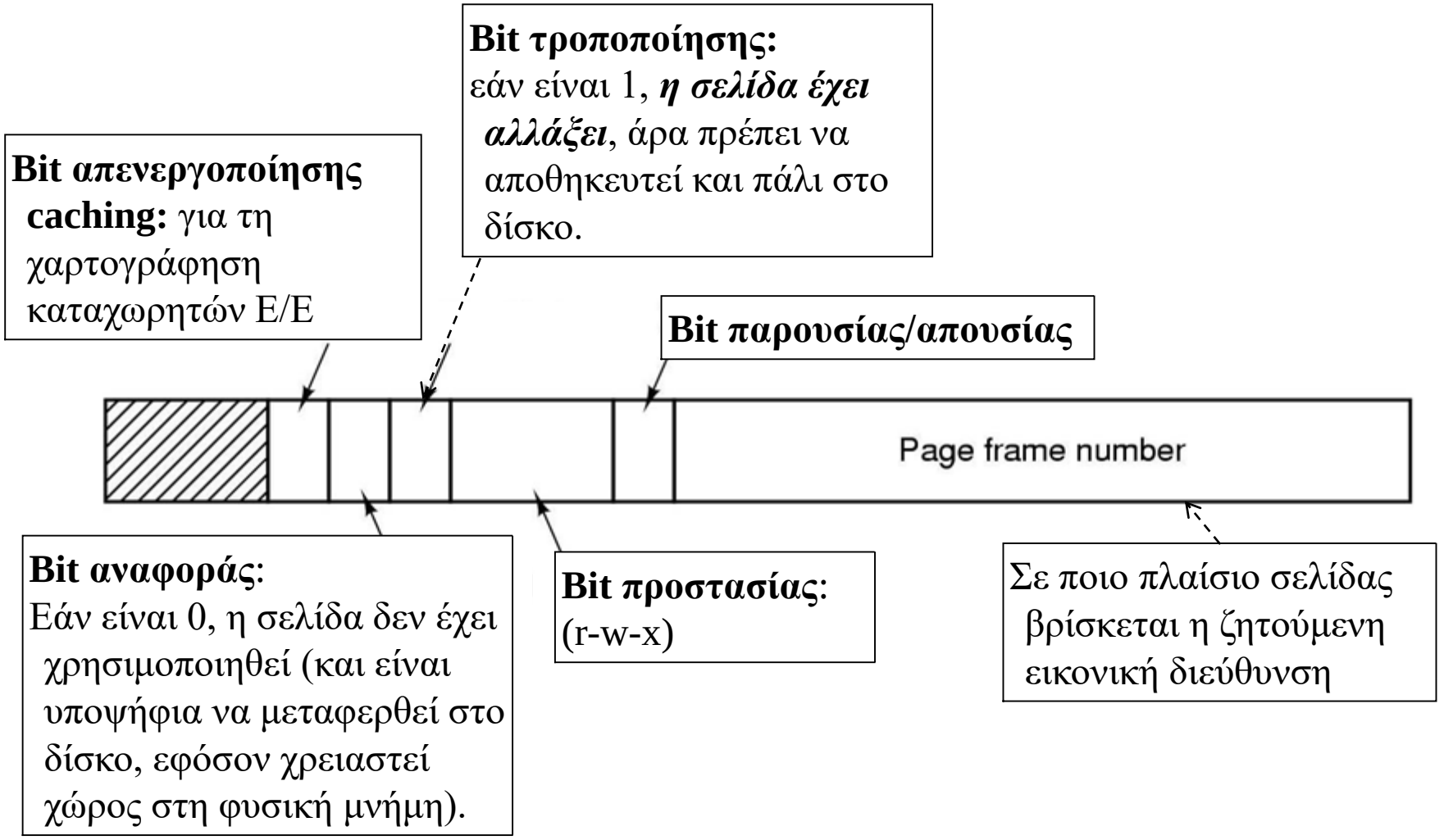
(3) bit παρουσίας:
1 = η σελίδα είναι στη φυσική μνήμη.
0 = δεν είναι (page fault).

(1) Τα 4 πρώτα bit της εικονικής διεύθυνσης είναι ***δείκτης στον πίνακα σελίδων.***

(2) Τα 12 τελευταία bit είναι η ***σχετική διεύθυνση***.

Εικόνα 3-10. Η εσωτερική λειτουργία της MMU όταν υπάρχουν 16 σελίδες των 4KB.

Δομή μιας καταχώρισης πίνακα σελίδων



Εικόνα 3-11. Μια τυπική καταχώριση του πίνακα σελίδων.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Επιτάχυνση της σελιδοποίησης (1)

Ζητήματα υλοποίησης που αφορούν τη σελιδοποίηση:

1. Η χαρτογράφηση από την εικονική διεύθυνση στη φυσική πρέπει να είναι γρήγορη.
   - Διαφορετικά η χαρτογράφηση της μνήμης θα αποτελέσει σημείο συμφόρησης του συστήματος.
2. Αν ο χώρος των εικονικών διευθύνσεων είναι μεγάλος, ο πίνακας σελίδων μπορεί να είναι μεγάλος.
   - Με σελίδες των 4 KB και σελίδες των 32 bit, περίπου 1.000.000 εικονικές διευθύνσεις – άρα και 1.000.000 εγγραφές στον πίνακα σελίδων.

Επιτάχυνση της σελιδοποίησης (2)

1η λύση: Χρήση γρήγορων καταχωρητών υλικού. Σε αυτούς τους καταχωρητές φορτώνεται μία φορά ο πίνακας σελίδων κάθε εκτελούμενης διεργασίας.

- Γρήγορη εκτέλεση, αργή φόρτωση του πίνακα.

2η λύση: Τοποθέτηση του πίνακα σελίδων στη RAM. Χρειάζεται μόνο ένας καταχωρητής για να δείχνει στην αρχή του πίνακα σελίδων.

- Απαιτούνται μία ή περισσότερες αναφορές στη μνήμη για την εκτέλεση κάθε εντολής.

Κρυφή μνήμη αναζήτησης μετάφρασης (Translation Lookaside Buffers – TLB)

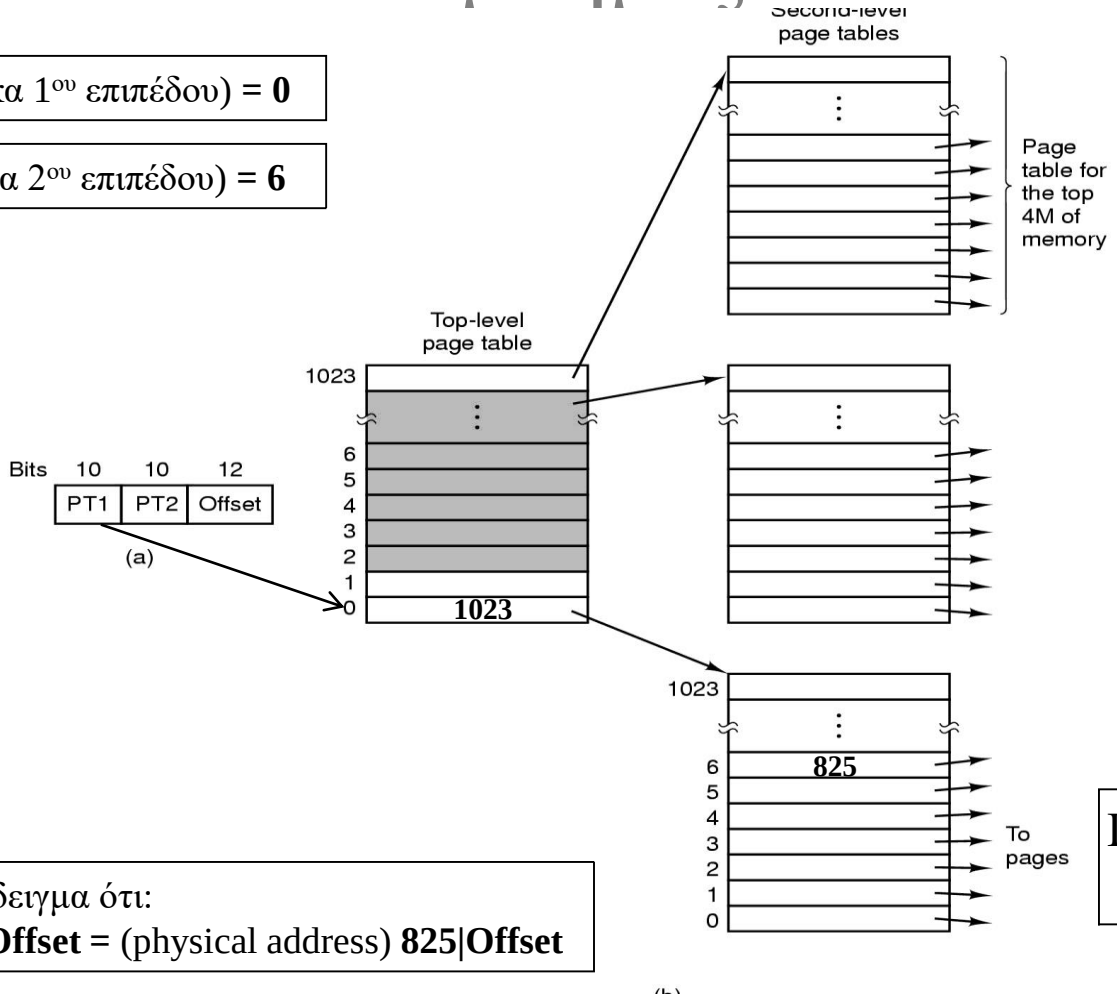
Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Εικόνα 3-12. Μια συσκευή TLB για την επιτάχυνση της σελιδοποίησης.

Πολυεπίπεδοι πίνακες σελίδων για μεγάλες μνήμες

PT1: (δείκτης πίνακα 1ου επιπέδου) = **0**

PT2: (δείκτης πίνακα 2ου επιπέδου) = **6**



Πίνακας στοίβας

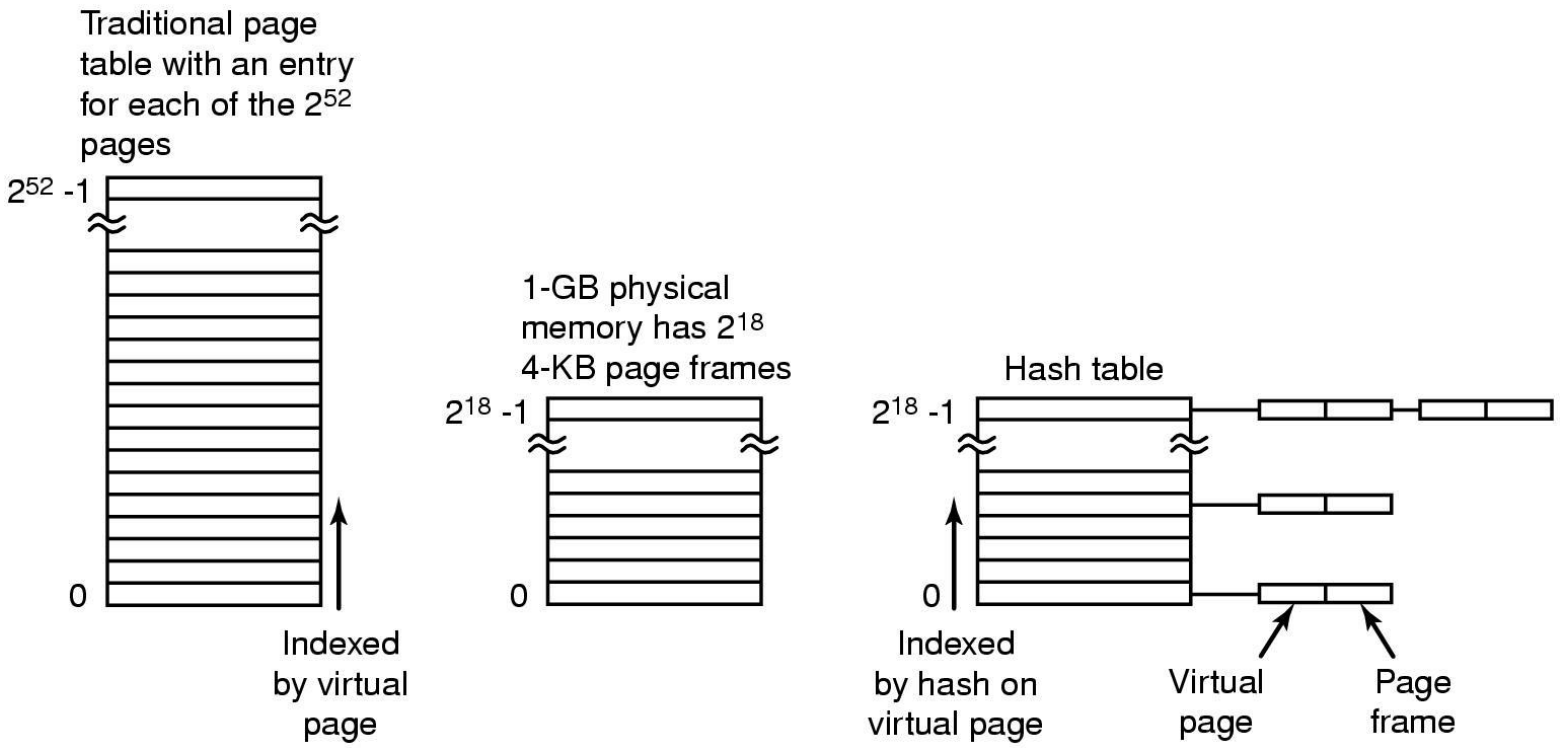
Πίνακας δεδομένων

Πίνακας κώδικα προγράμματος

Άρα ισχύει στο παράδειγμα ότι:
(virtual address) **0|6|Offset** = (physical address) **825|Offset**

Εικόνα 3-13. (a) Μια διεύθυνση των 32-bit με δύο πεδία του πίνακα σελίδων. (b) Πίνακες σελίδων δύο επιπέδων.

Ανεστραμμένοι πίνακες σελίδων



Εικόνα 3-14. Σύγκριση συμβατικού πίνακα σελίδων με ανεστραμμένο πίνακα σελίδων.

3.4 Αλγόριθμοι αντικατάστασης σελίδων

Αλγόριθμοι αντικατάστασης σελίδων

1. Ποιες σελίδες πρέπει να αφαιρεθούν από τη φυσική μνήμη όταν υπάρχει ανάγκη να μεταφερθεί μία σελίδα;
   - Ζητήθηκε πρόσφατα η σελίδα;

2. Η σελίδα που αφαιρέθηκε από τη φυσική μνήμη, θα πρέπει να ξαναγραφτεί στο δίσκο;
   - Τροποποιήθηκε η σελίδα όσο ήταν στη φυσική μνήμη;

Αλγόριθμοι αντικατάστασης σελίδων

- Βέλτιστος αλγόριθμος αντικατάστασης σελίδας
- Αλγόριθμος αντικατάστασης σελίδας NRU
- Αλγόριθμος αντικατάστασης σελίδας FIFO
- Αλγόριθμος αντικατάστασης σελίδας της δεύτερης ευκαιρίας
- Αλγόριθμος αντικατάστασης σελίδας του ρολογιού
- Αλγόριθμος αντικατάστασης σελίδας LRU
- Αλγόριθμος αντικατάστασης σελίδας του συνόλου εργασίας
- Αλγόριθμος αντικατάστασης σελίδας WSClock

Βέλτιστος αλγόριθμος αντικατάστασης σελίδας

- Σε κάθε σελίδα αντιστοιχίζεται μία ετικέτα που δείχνει πόσες εντολές θα εκτελεστούν, μέχρι να γίνει αναφορά σε αυτή τη σελίδα.
- Όταν πρέπει να αφαιρεθεί μία σελίδα από τη φυσική μνήμη, *αφαιρείται εκείνη με το μεγαλύτερο αριθμό ετικέτας.*
- ***Δεν είναι υλοποιήσιμος.***
  - Το Λ.Σ. *δεν μπορεί να γνωρίζει εκ' των προτέρων*, μετά από πόσες εντολές θα γίνει αναφορά σε κάθε σελίδα.
- Χρήσιμος μόνο ως μέτρο σύγκρισης, για την αξιολόγηση των πραγματικών αλγορίθμων αντικατάστασης.

Αλγόριθμος αντικατάστασης σελίδας NRU (Not Recently Used)

- Χρησιμοποιεί τα bit Αναφοράς (Α) και Τροποποίησης (Τ).
- Δημιουργούνται 4 κατηγορίες κατάταξης με βάση τα bit.
  1. Κατηγορία 0: Α=0, Τ=0.
  2. Κατηγορία 1: Α=0, Τ=1.
  3. Κατηγορία 2: Α=1, Τ=0.
  4. Κατηγορία 3: Α=1, Τ=1.
- Ο αλγόριθμος αφαιρεί μία σελίδα στην τύχη***, από την μικρότερη διαθέσιμη κατηγορία***.

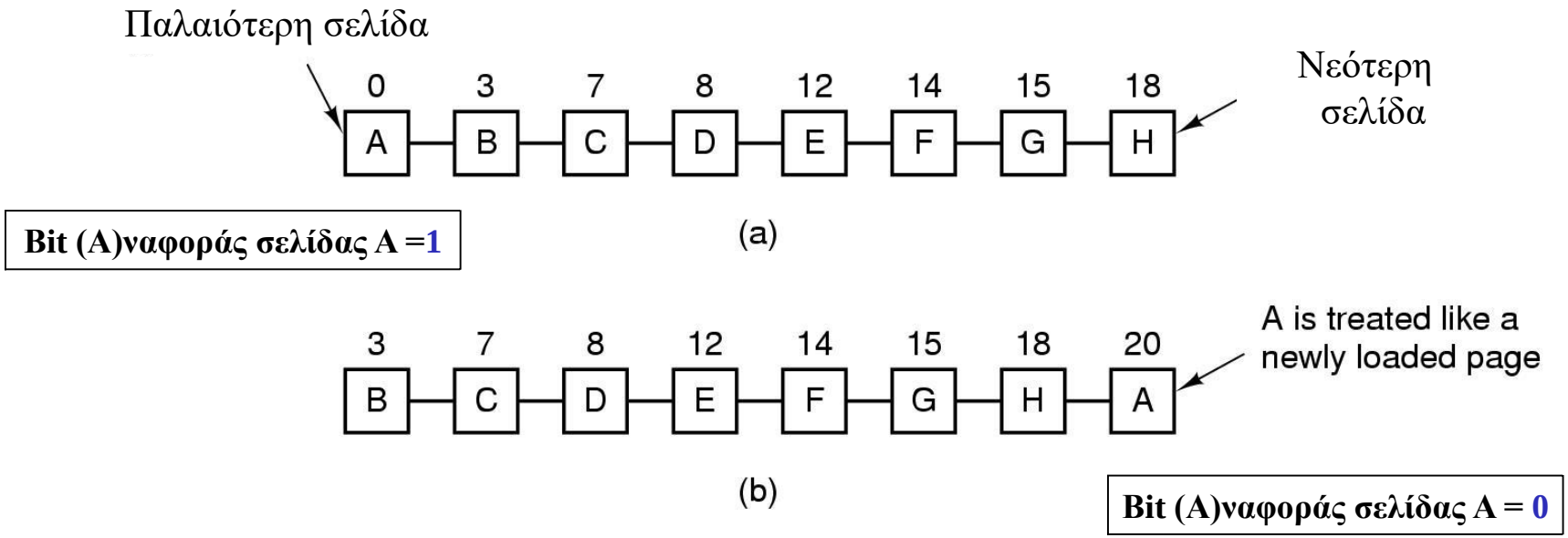
Αλγόριθμος αντικατάστασης σελίδας FIFO (First In First Out)

- Διατηρεί μία συνδεδεμένη λίστα όλων των σελίδων από την παλαιότερη προς τη νεότερη.
- Όταν χρειαστεί χώρος στη φυσική μνήμη, θα αφαιρεθεί ***η παλαιότερη σελίδα, ανεξάρτητα από τη χρήση της*****.**
- Μειονέκτημα:
  - Θα μπορούσε να είναι κάποια σελίδα που χρησιμοποιείται συχνά (π.χ. καθολική μεταβλητή).

Αλγόριθμος αντικατάστασης σελίδας της 2ης ευκαιρίας (1)

- Παραλλαγή του FIFO.
- Για την πιο παλιά σελίδα, εξετάζεται το Bit αναφοράς. Εάν A = 1, η σελίδα έχει χρησιμοποιηθεί, οπότε της δίδεται μία 2η ευκαιρία.
  - Παραμένει αλλά το bit αναφοράς αλλάζει (A =0).
  - … και η σελίδα μεταφέρεται στο τέλος της λίστας.

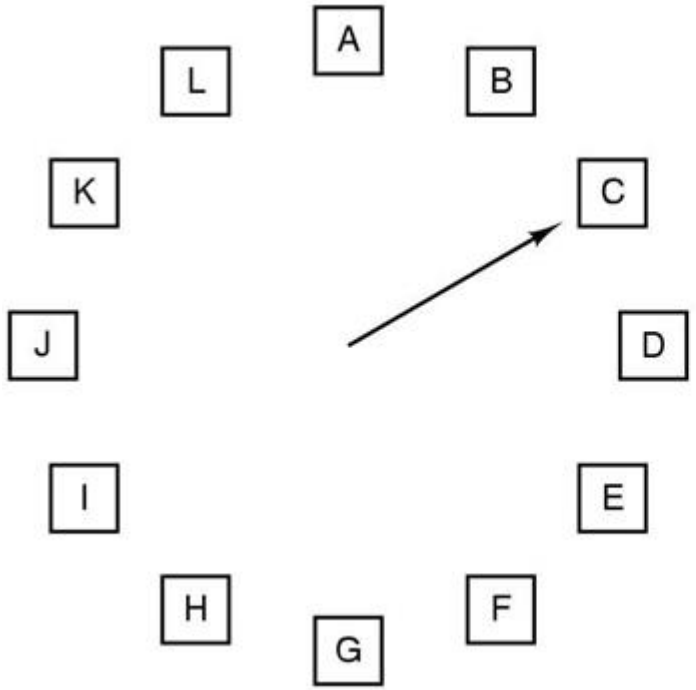
Αλγόριθμος αντικατάστασης σελίδας της δεύτερης ευκαιρίας (2)



Εικόνα 3-15. Παράδειγμα αλγορίθμου της δεύτερης ευκαιρίας.
(a) Ταξινόμηση σελίδων με τη σειρά FIFO (πάνω οι χρόνοι άφιξης).
(b) Η λίστα σελίδων μετά από σφάλμα σελίδας που συνέβη τη χρονική στιγμή 20 και ενώ η σελίδα *A* είχε bit αναφοράς *(A)* με τιμή 1.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Ο αλγόριθμος αντικατάστασης σελίδας του ρολογιού



- Κυκλική λίστα των σελίδων.
- Ο δείκτης δείχνει την παλαιότερη σελίδα.
- Συμβαίνει σφάλμα σελίδας.
  - Είναι το bit A = 0; Η σελίδα αφαιρείται.
  - bit A = 1; Αλλάζει σε 0 και ο δείκτης προχωρά στην επόμενη σελίδα.

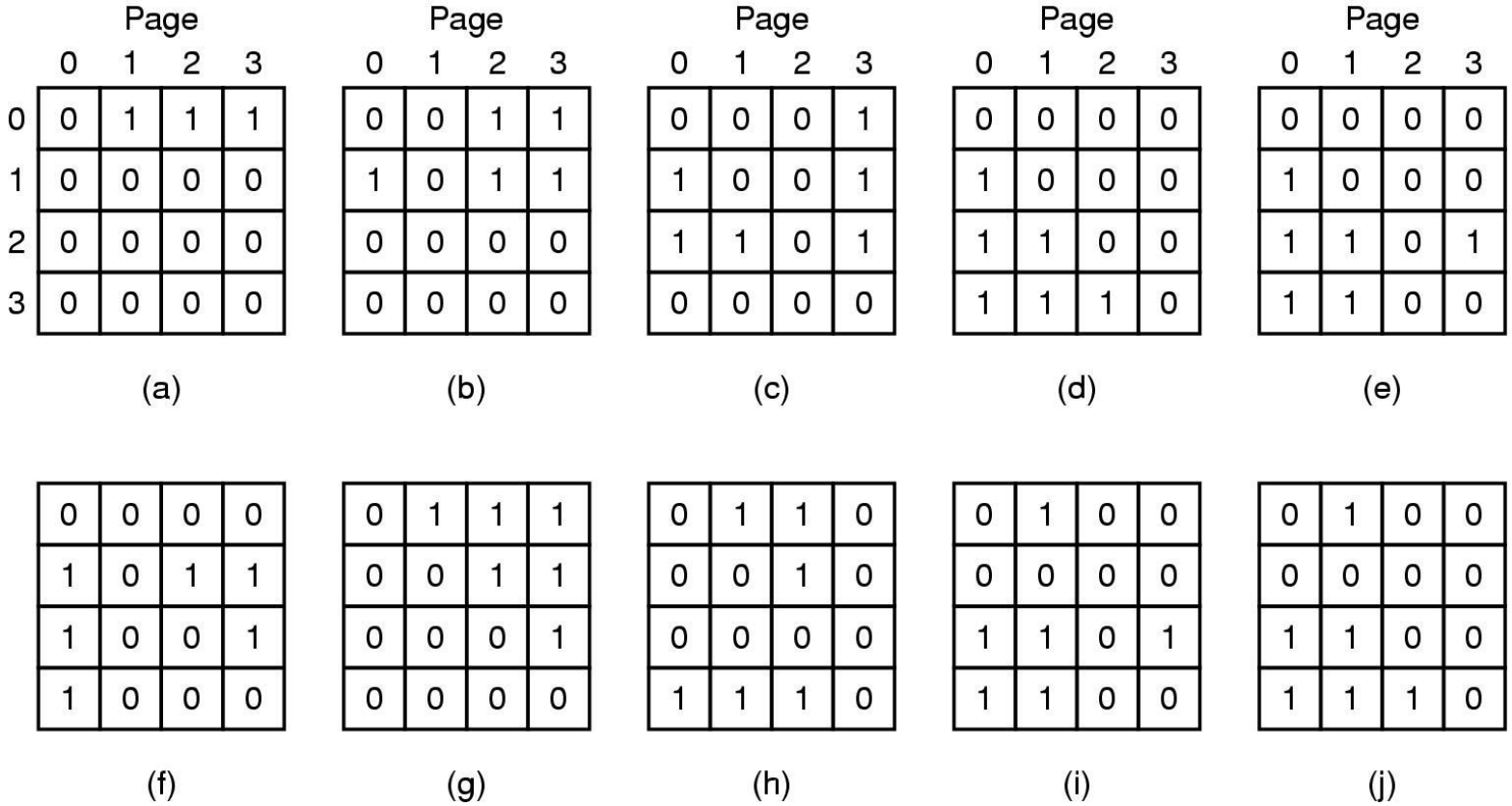
Εικόνα 3-16. Ο αλγόριθμος του ρολογιού για την αντικατάσταση σελίδων

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Ο αλγόριθμος αντικατάστασης σελίδας LRU (Least Recently Used)

- Οι σελίδες που χρησιμοποιήθηκαν πολύ στο παρελθόν, είναι πιθανό να χρησιμοποιηθούν και στο μέλλον.
- Χρήση ενός μεγάλου μετρητή M (64 bit) για τη μέτρηση του συνόλου των εντολών (αυξάνει κατά 1 μετά από κάθε εντολή).
- Κάθε καταχώρηση στον πίνακα σελίδων, διατηρεί και την τιμή του M. Μετά από μία αναφορά σε μία σελίδα (bit A=1), ανανεώνεται στην καταχώρηση της σελίδας η νέα τιμή του M.
- Όταν σημειωθεί σφάλμα σελίδας, θα αφαιρεθεί η σελίδα με τη μικρότερη τιμή του M (είναι εκείνη που χρησιμοποιήθηκε λιγότερο πρόσφατα).
- Μειονέκτημα: ο αλγόριθμος δεν «ξεχνάει».

Ο αλγόριθμος αντικατάστασης της LRU – Υλοποίηση με υλικό



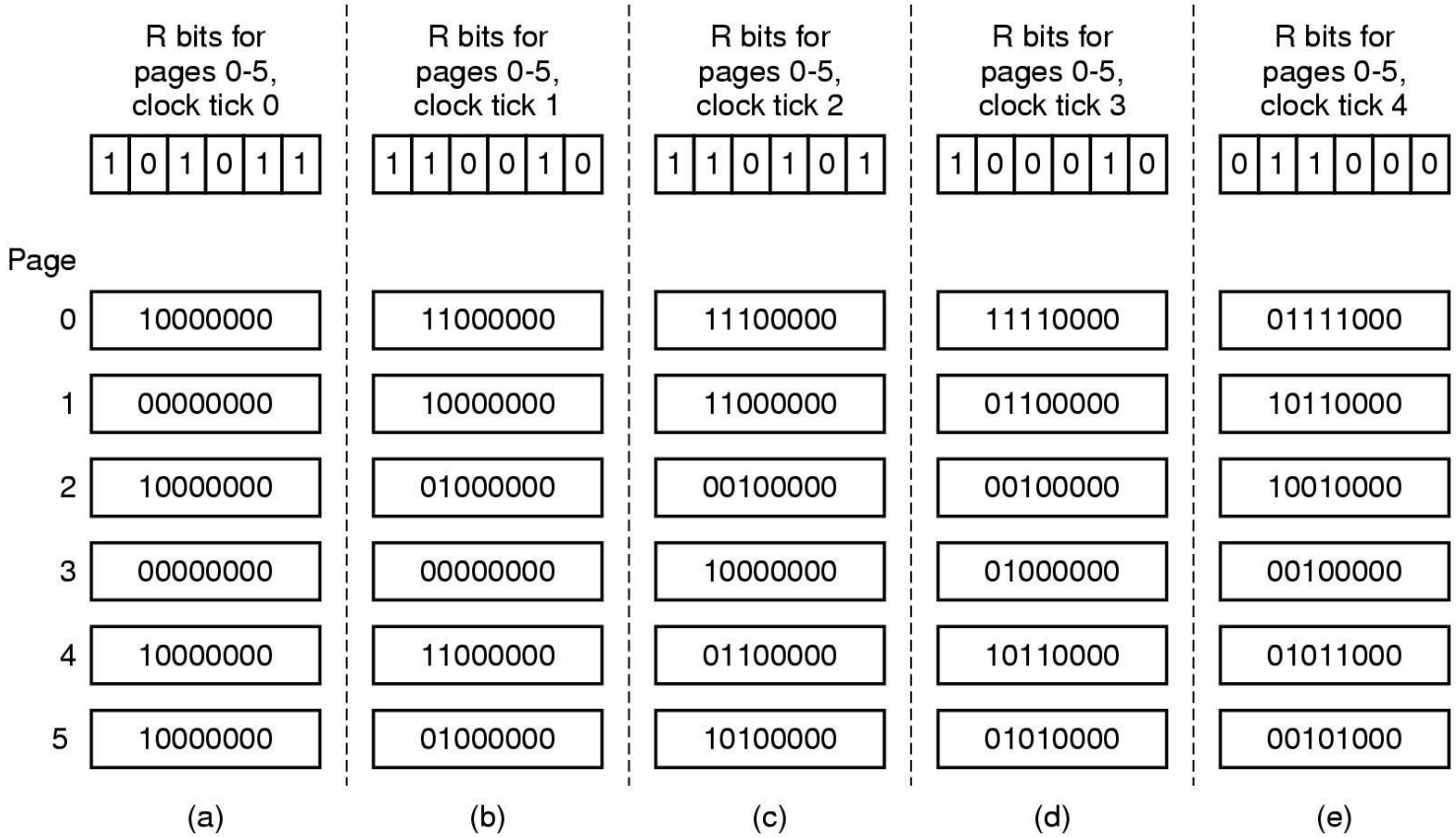
Εικόνα 3-17. Ο αλγόριθμος LRU με τη χρήση πίνακα, όταν οι αναφορές στις σελίδες γίνονται με τη σειρά: 0, 1, 2, 3, 2, 1, 0, 3, 2, 3.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Ο αλγόριθμος Not Frequently Used (NFU)

- Κάθε σελίδα έχει και στον αλγόριθμο αυτό, ένα μετρητή Μ με αρχική τιμή 0 για κάθε καταχώριση του πίνακα σελίδων.
- Σε κάθε διακοπή του ρολογιού, το ΛΣ σαρώνει όλες τις σελίδες στη μνήμη.
  - Η τιμή του bit Α κάθε σελίδας (0 ή 1), προστίθεται στον μετρητή Μ της συγκεκριμένης σελίδας στον πίνακα σελίδων.
- Όταν σημειωθεί σφάλμα σελίδας, θα αφαιρεθεί η σελίδα με τη μικρότερη τιμή του Μ (είναι εκείνη που χρησιμοποιήθηκε λιγότερο συχνά).
- Μειονέκτημα: ο αλγόριθμος δεν «ξεχνάει».

Ο αλγόριθμος γήρανσης (aging algorithm) (Προσομοίωση του LRU με λογισμικό)



Εικόνα 3-18. Ο αλγόριθμος γήρανσης προσομοιώνει τον αλγόριθμο LRU με λογισμικό. Παρουσιάζονται έξι σελίδες για πέντε χτύπους του ρολογιού. Οι πέντε χτύποι του ρολογιού αναπαρίστανται από τα σχήματα (a) έως (e).

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

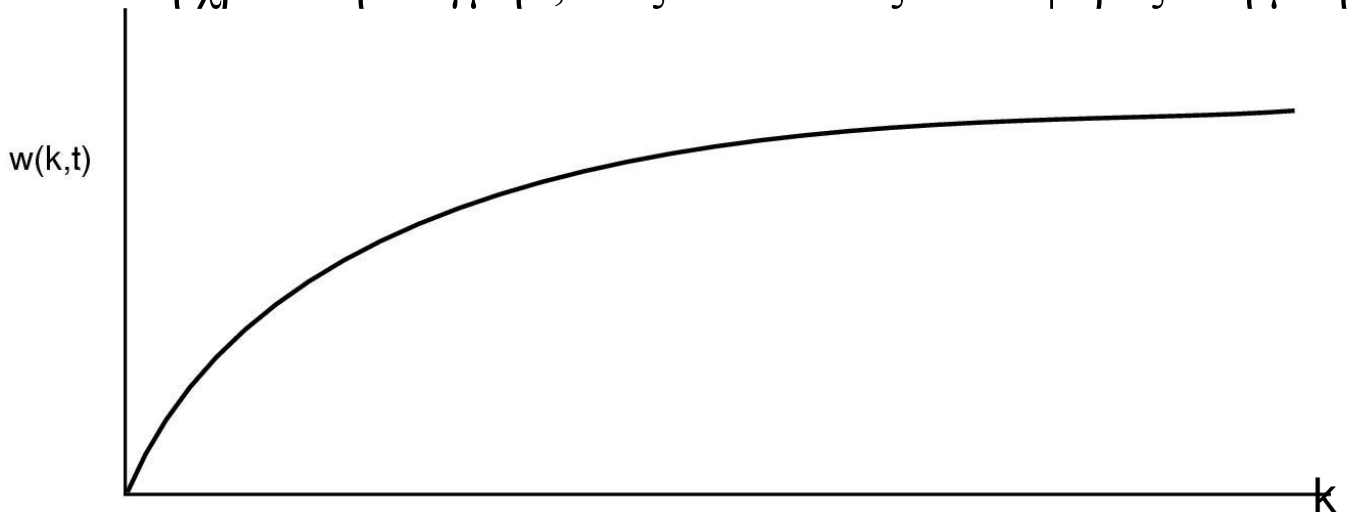
Ο αλγόριθμος αντικατάστασης σελίδας του συνόλου εργασίας (1)

- **Τοπικότητα αναφορών** (locality of reference)
  - Σε κάθε φάση της εκτέλεσης του προγράμματος, η διεργασία αναφέρεται σε ένα σχετικά περιορισμένο υποσύνολο των σελίδων.
- **Σύνολο εργασίας** (working set): Το σύνολο των σελίδων που χρησιμοποιεί μία διεργασία, μία δεδομένη χρονική στιγμή.
  - Ιδανικά, ***εάν φορτωθεί όλο το σύνολο εργασίας*** μίας διεργασίας φορτωθεί στη μνήμη τη στιγμή που πρέπει, ***θα ελαχιστοποιηθούν τα σφάλματα σελίδας***.
  - Η στρατηγική αυτή λέγεται προσελιδοποίηση (prepaging)
  - Το σύνολο εργασίας αλλάζει διαρκώς.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Ο αλγόριθμος αντικατάστασης σελίδας του συνόλου εργασίας (2)

Σύνολο εργασίας w(k,t): το πλήθος των σελίδων που χρησιμοποιήθηκαν τη χρονική στιγμή t, στις τελευταίες k αναφορές στη μνήμη



Τις τελευταίες k=1 αναφορές στη μνήμη, χρησιμοποιήθηκαν w(k,t) =1 σελίδες.

k=1.000	w(k,t) = 1.000.000
k=100.000	w(k,t) = 2.000.000
k=1.000.000	w(k,t) ~= 2.000.000

Εικόνα 3-19. Το σύνολο εργασίας είναι το σύνολο των σελίδων που χρησιμοποιήθηκαν κατά τις *k* πιο πρόσφατες αναφορές στη μνήμη.

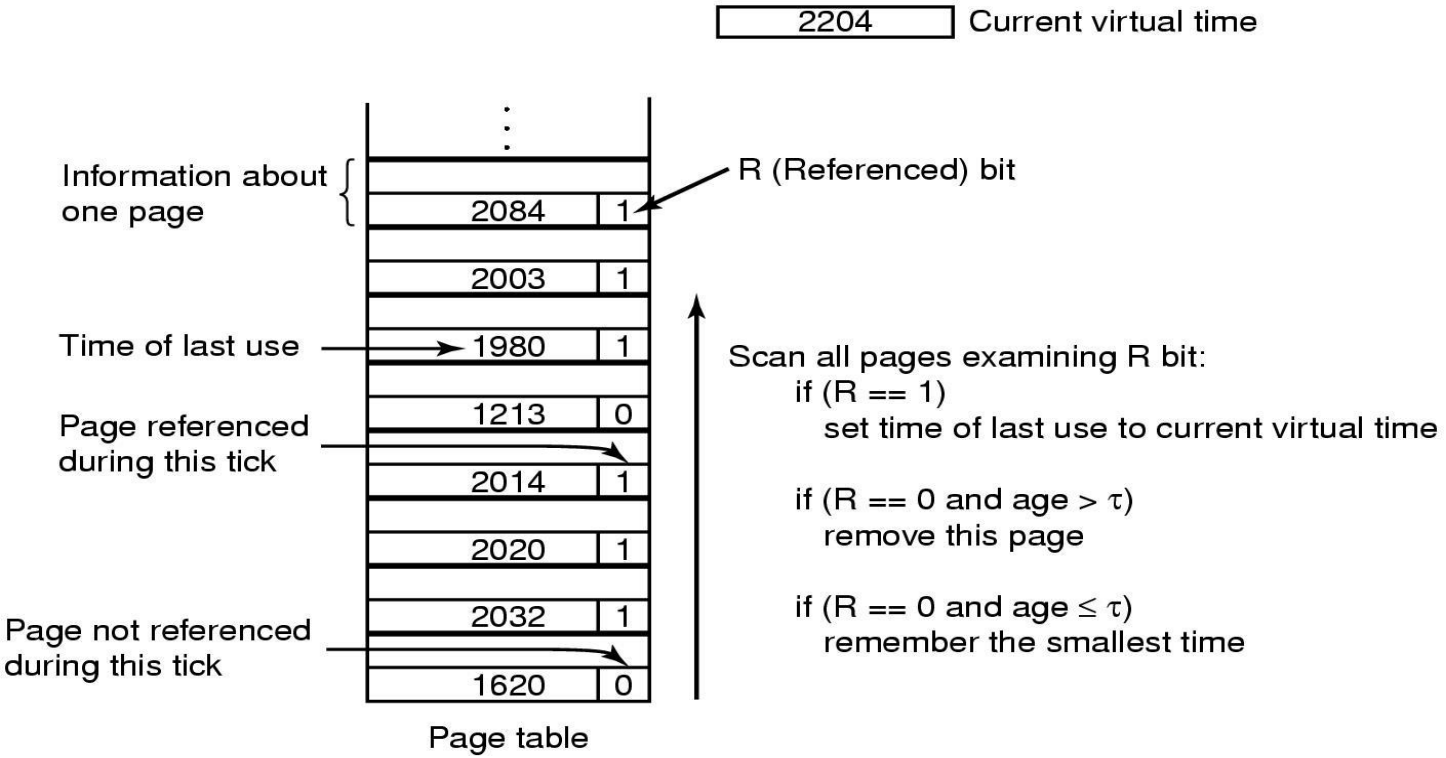
Ο αλγόριθμος αντικατάστασης σελίδας του συνόλου εργασίας (3)

- Υλοποίηση του αλγορίθμου του συνόλου εργασίας:
  - Το ΛΣ παρακολουθεί *ποιες σελίδες ανήκουν στο σύνολο εργασίας* κάθε διεργασίας (*για κάποιο k*).
  - Μετά από κάθε αναφορά στη μνήμη, το σύνολο εργασίας (δηλ. το σύνολο των σελίδων που χρησιμοποιήθηκαν από τις προηγούμενες *k* αναφορές στη μνήμη) προσδιορίζεται μονοσήμαντα.
  - Όταν προκύψει σφάλμα σελίδας, κάποια από τις σελίδες που *δεν ανήκει* στο σύνολο εργασίας *θα αφαιρεθεί* από τη μνήμη.

Ο αλγόριθμος αντικατάστασης σελίδας του συνόλου εργασίας (4)

- Πρακτικά χρησιμοποιείται **ο χρόνος εκτέλεσης**.
  - Δηλαδή ορίσουμε το σύνολο εργασίας ως *τις σελίδες που χρησιμοποιήθηκαν κατά τη διάρκεια των τελευταίων **T** δευτερολέπτων του χρόνου εκτέλεσης*.
  - Για κάθε διεργασία, *μετράει μόνο ο δικός της χρόνος εκτέλεσης*. Αν η διεργασία ξεκινά τη χρονική στιγμή ***T*** και τη στιγμή ***T***+100 msec έχει πάρει 40 msec της CPU, τότε θα βρίσκεται στη χρονική στιγμή **40 msec.**
  - Η ποσότητα του χρόνου που μία διεργασία έχει πρακτικά χρησιμοποιήσει ονομάζεται ***τρέχων εικονικός χρόνος** της διεργασίας* (current virtual time).

Ο αλγόριθμος αντικατάστασης σελίδας του συνόλου εργασίας (5)



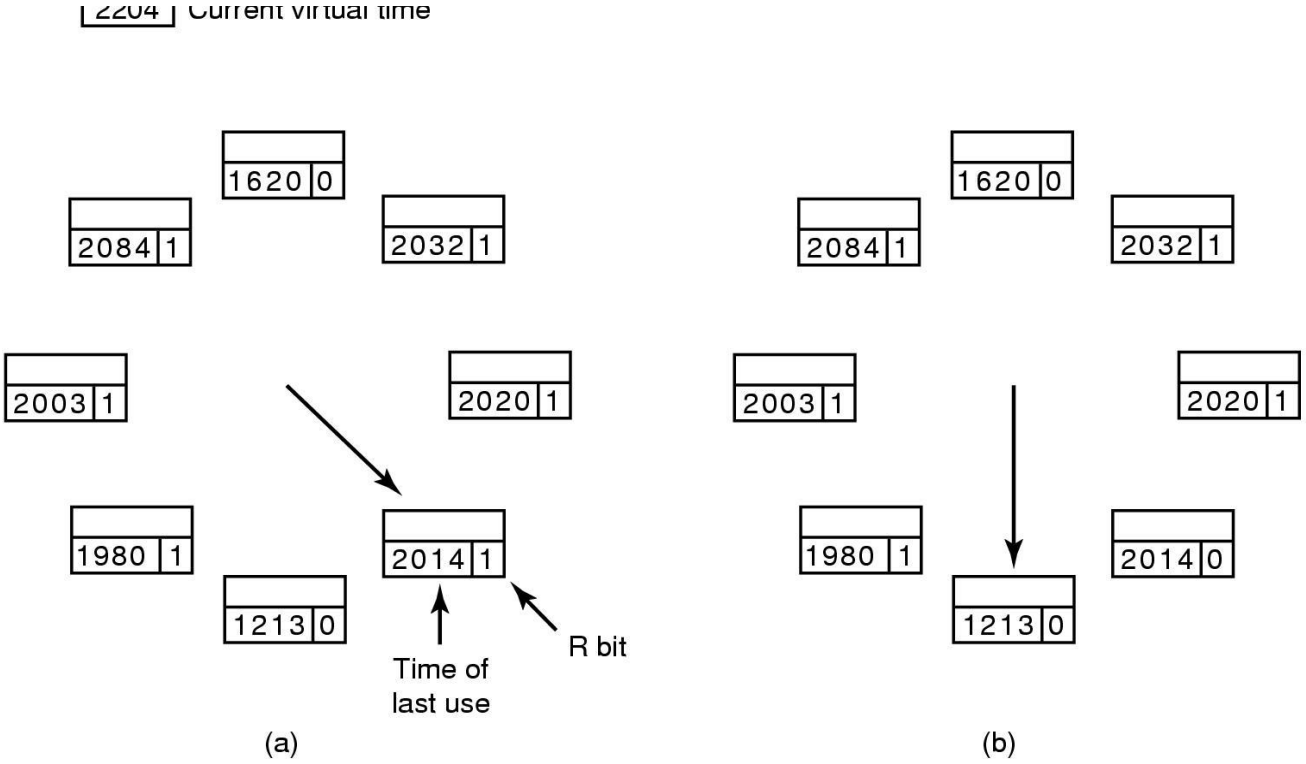
Εικόνα 3-20. Ο αλγόριθμος του συνόλου εργασίας.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Ο αλγόριθμος αντικατάστασης σελίδας WSClock (1)

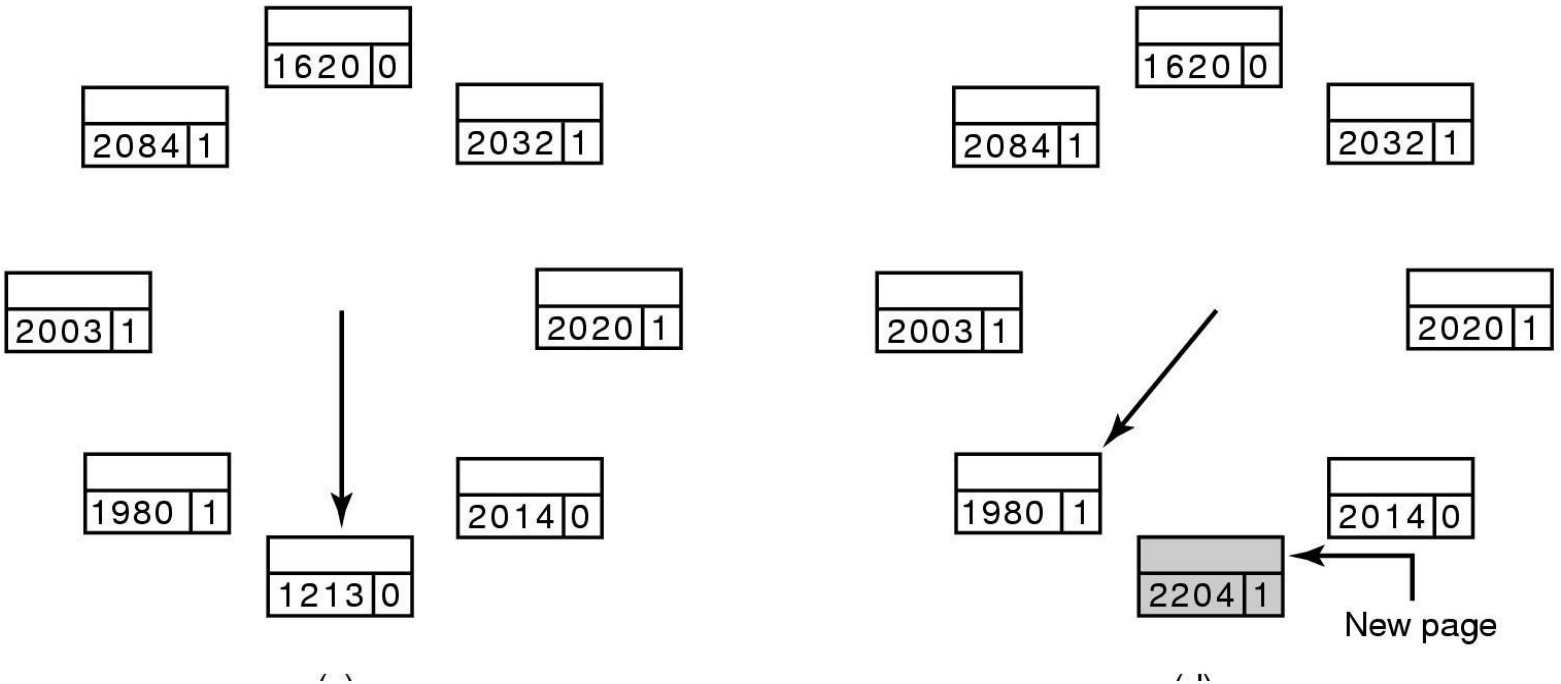
- Πολύ πρακτικός και αποδοτικός αλγόριθμος που συνδυάζει τον αλγόριθμο του ρολογιού, και το σύνολο εργασίας.
- Χρησιμοποιεί μία κυκλική λίστα, όπως και στον αλγόριθμο του ρολογιού,
- Το χρόνο τελευταίας χρήσης (από τον αλγόριθμο του συνόλου εργασίας).
- Τα bit αναφοράς A (reference bit R) και τροποποίησης Τ (Modification bit M).

Ο αλγόριθμος αντικατάστασης σελίδας WSClock (2)



Εικόνα 3-21. Η λειτουργία του αλγόριθμου WSClock. Τα (a) και (b) δίνουν ένα παράδειγμα των ενεργειών που γίνονται όταν το Reference bit είναι =1.

Ο αλγόριθμος αντικατάστασης σελίδας WSClock (3)



Εικόνα 3-21. Η λειτουργία του αλγόριθμου WSClock. Τα (c) και (d) δίνουν ένα παράδειγμα για την περίπτωση R=0.

Ο αλγόριθμος αντικατάστασης σελίδας WSClock (4)

- Τι θα συμβεί εάν ο δείκτης κάνει ένα πλήρη κύκλο και επιστρέψει στην αφετηρία;
- Υπάρχουν δύο διακριτές περιπτώσεις:
  - Έχει χρονοπρογραμματιστεί τουλάχιστον μία εγγραφή στο δίσκο.
    - Ο δείκτης συνεχίζει την πορεία του, ψάχνοντας μία καθαρή σελίδα.
  - Δεν έχει χρονοπρογραμματιστεί καμία εγγραφή στο δίσκο.
    - Διαλέγεται μία σελίδα στην τύχη.

Σύνοψη των αλγόριθμων αντικατάστασης σελίδας

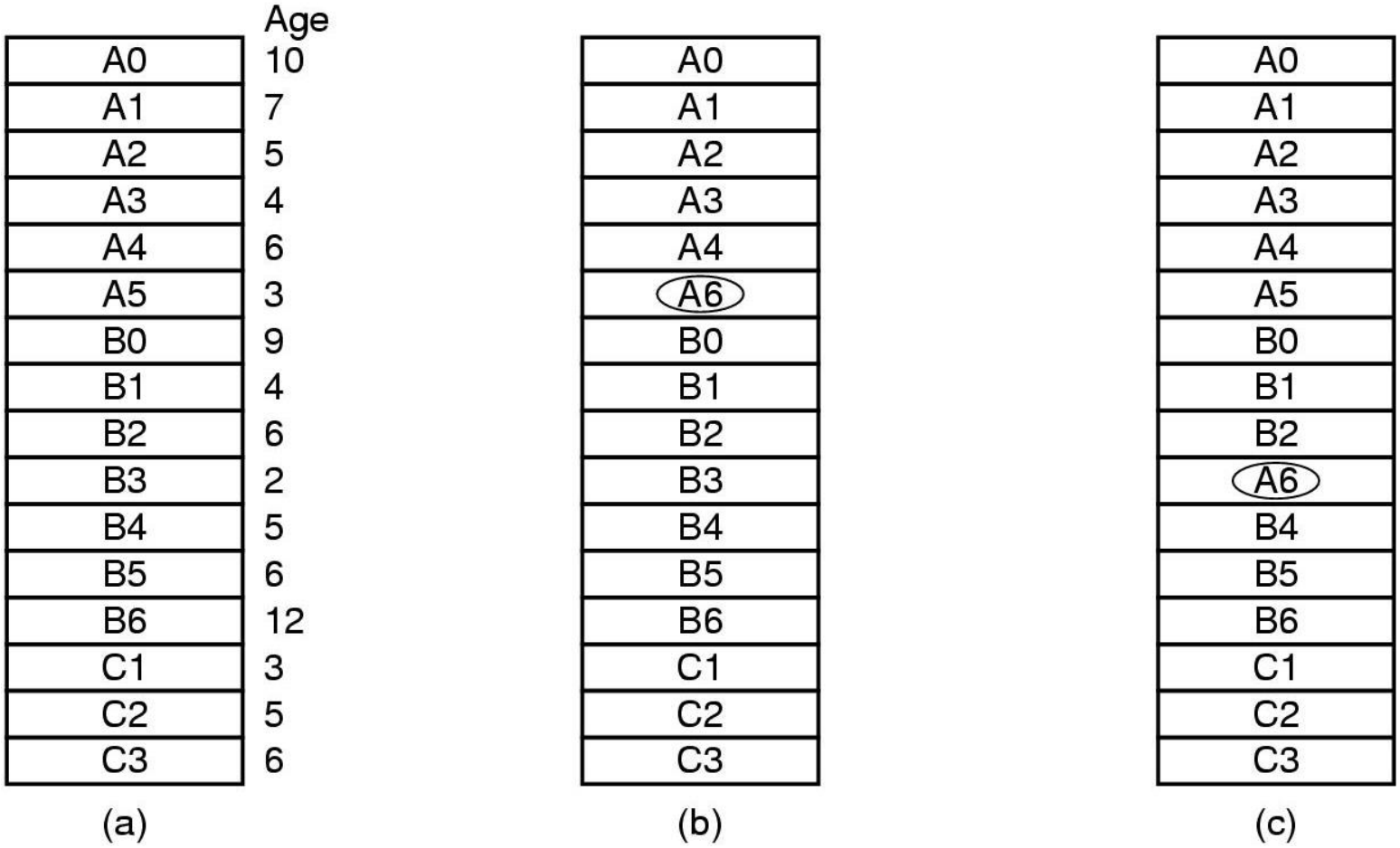
Αλγόριθμος	Περιγραφή
Βέλτιστος	Δεν είναι δυνατόν να υλοποιηθεί, αλλά χρησιμοποιείται ως μέτρο σύγκρισης
NRU (Not Recently Used)	Πολύ χονδροειδής
FIFO (First–In, First–Out)	Υπάρχει περίπτωση να αφαιρέσει σημαντικές σελίδες
Δεύτερης ευκαιρίας	Πολύ βελτιωμένος σε σχέση με τον FIFO
Ρολογιού	Ρεαλιστικός
LRU (Least Recently Used)	Εξαιρετικός, αλλά η ακριβής υλοποίησή του είναι δύσκολη
NFU (Not FrequentlyUsed)	Μάλλον χονδροειδής προσέγγιση του LRU
Γήρανσης	Αποδοτικός αλγόριθμος, καλή προσέγγιση του LRU
Συνόλου εργασίας	Κάπως ακριβή υλοποίηση
WSClock	Καλός και αποδοτικός αλγόριθμος

3.5 Θέματα σχεδιασμού για τα συστήματα σελιδοποίησης

Τοπικές και καθολικές πολιτικές κατανομής (1)

- Οι αλγόριθμοι αντικατάστασης σελίδας, εξαρτώνται επίσης από τον ***τρόπο κατανομής της μνήμης*** *μεταξύ των εκτελούμενων διεργασιών*.
  1. **Τοπικός αλγόριθμος αντικατάστασης σελίδας** (Local page replacement). Λαμβάνει υπόψη μόνο τις σελίδες της ίδιας διεργασίας.
  2. **Καθολικός αλγόριθμος αντικατάστασης σελίδας** (Global page replacement algorithm). Λαμβάνει υπόψη δυναμικά όλες τις σελίδες των διεργασιών που εκτελούνται.

Τοπικές και καθολικές πολιτικές κατανομής (2)



Εικόνα 3-23. Τοπική και καθολική αντικατάσταση σελίδας. (a) Αρχική κατάσταση. (b) Τοπική αντικατάσταση σελίδας. (c) Καθολική αντικατάσταση σελίδας.

Τοπικές και καθολικές πολιτικές κατανομής (3)

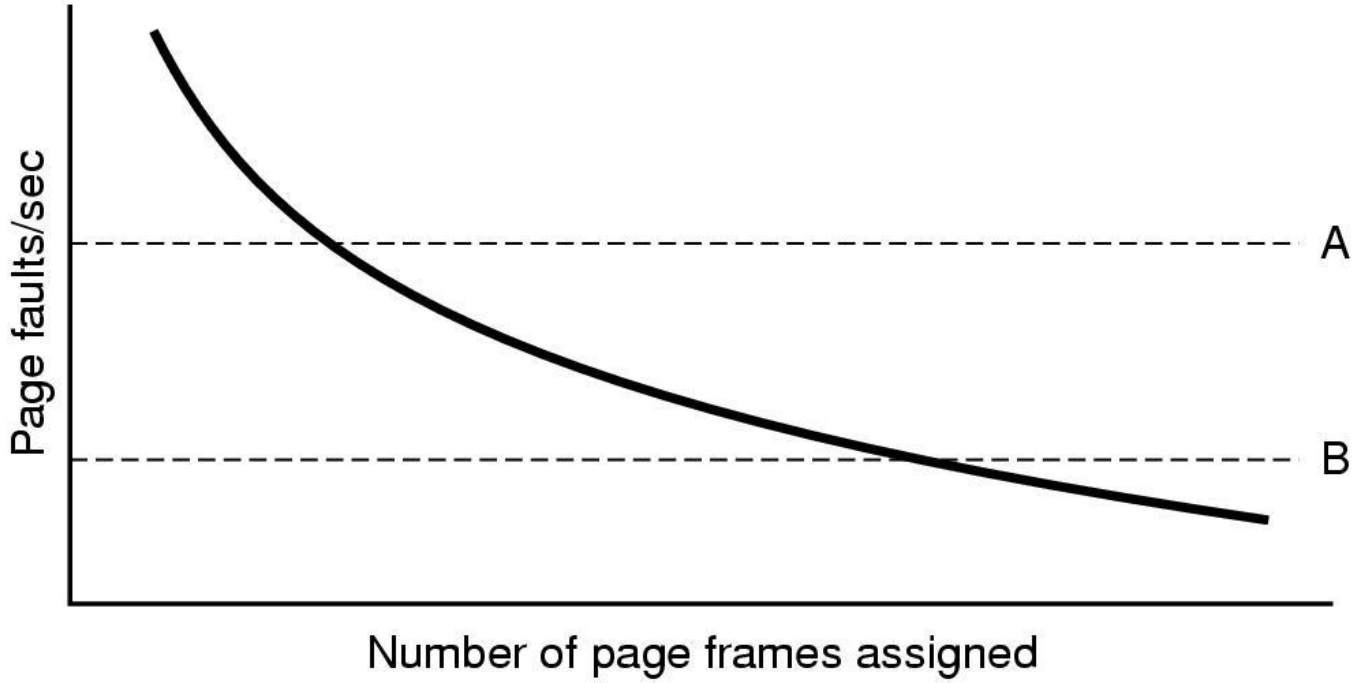
- Οι καθολικές πολιτικές είναι πιο ευέλικτες και αποδοτικές.
- Προσεγγίσεις καθολικών πολιτικών:
  1. Παρακολούθηση του μεγέθους του συνόλου εργασίας κάθε διεργασίας.
  2. Υπολογίζεται περιοδικά ο αριθμός v των διεργασιών που μπορούν να εκτελεστούν στη μνήμη ταυτόχρονα, και να μοιράζονται εξίσου σε αυτές ο συνολικός αριθμός των πλαισίων μνήμης π: π/v
  3. Ένας ακόμα καλύτερος τρόπος είναι να μοιράζεται η μνήμη όχι εξίσου, αλλά ανάλογα με το μέγεθος μ_i κάθε διεργασίας:

$$\frac{(\mu_i / \sum \mu_i)\,\pi}{v}$$

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Τοπικές και καθολικές πολιτικές κατανομής (4)

Page Fault Frequency: Η διεργασία ξεκινάει με έναν αριθμό σελίδων ανάλογα με το μέγεθός της. Ο αριθμός αλλάζει δυναμικά κατά την εκτέλεση, ανάλογα με τη συχνότητα σφαλμάτων σελίδας που προκαλεί.



Εικόνα 3-24. Ο ρυθμός των σφαλμάτων σελίδας ως συνάρτηση του αριθμού των πλαισίων σελίδας που έχουν εκχωρηθεί.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Μέγεθος σελίδας (1)

- Εάν το υλικό υποστηρίζει μέγεθος σελίδας π.χ. 512 Byte, το ΛΣ υποστηρίζει μέγεθος σελίδας $(\kappa \times 512)$ Byte.
  - Συνενώνει όσες φυσικές σελίδες θέλει, σε μία μεγαλύτερη λογική σελίδα.

- Μεγάλο μέγεθος σελίδας (*μειονεκτήματα*)
  - *Εσωτερική κατάτμηση* (internal fragmentation): Χάνεται κατά μ.ο. το ½ της τελευταίας σελίδας.
  - Για $\boldsymbol{\nu}$ διεργασίες και μέγεθος σελίδας $\boldsymbol{\sigma}$, χάνονται $\boldsymbol{\nu \times \sigma / 2}$ Byte.

Μέγεθος σελίδας (2)

- Μικρό μέγεθος σελίδας (*μειονεκτήματα*)
  - *Μεγάλος πίνακας σελίδων.*
  - Περισσότερες σελίδες ανά πρόγραμμα, άρα *μεγαλύτερο κόστος μεταφοράς σελίδων* από και προς τον δίσκο.
  - Πρόγραμμα 32 KB: 64 σελίδες×512 Byte ή 4 σελίδες×8 KB
    - Κόστος μεταφοράς 1ης περίπτωσης: 64× 10 msec = 640 msec
    - Κόστος μεταφοράς 2ης περίπτωσης: 4× 12 msec = 48 msec.

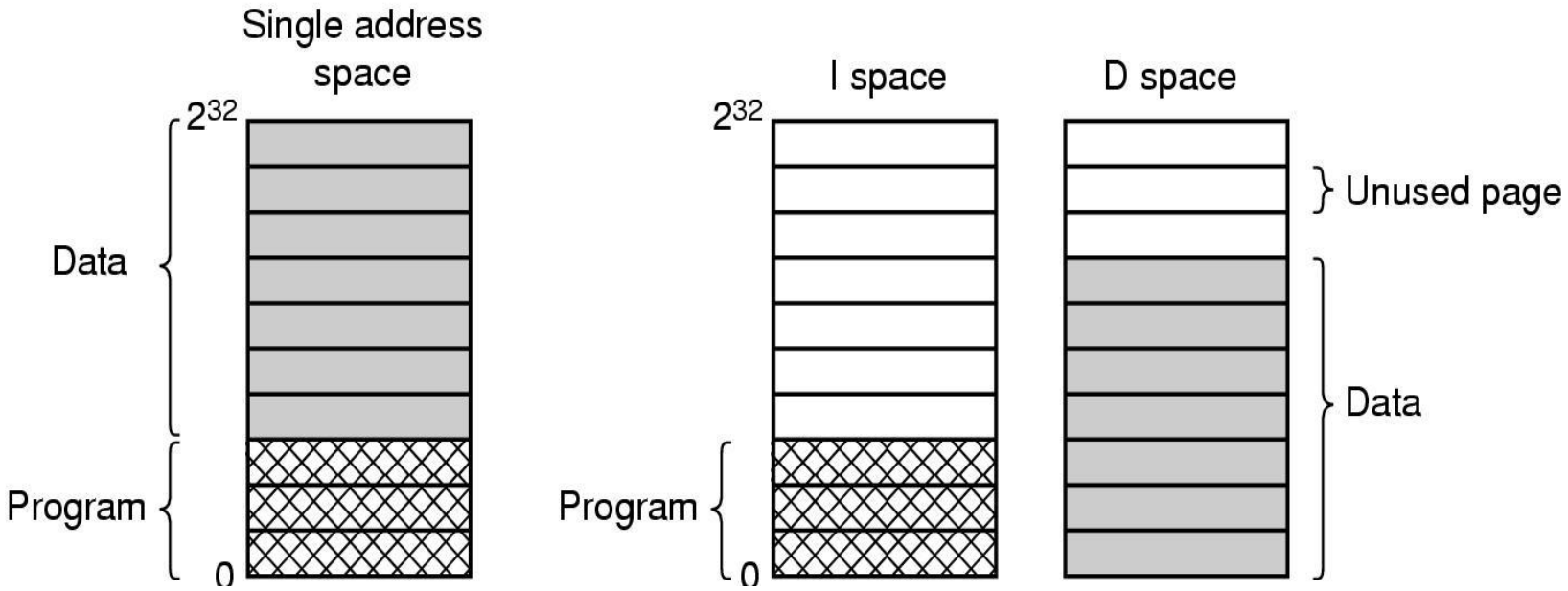
Μέγεθος σελίδας (3)

- Έστω ***μ*** το μέσο μέγεθος των διεργασιών, και ***σ*** το μέγεθος της σελίδας. Κάθε καταχώρηση σελίδας απαιτεί ***κ byte***.
- Μέσος αρ. σελίδων ανά διεργασία: ***μ/σ***
- Χώρος πίνακα σελίδων: ***μκ/σ byte***.
- Χαμένος χώρος (εσωτερικής κατάτμησης): ***σ/2***
- ***Συνολική επιβάρυνση $f(\sigma) = \mu\kappa/\sigma + \sigma/2$***
- Εύρεση βέλτιστης τιμής του ***σ:***

$$f'(\sigma) = 0 \Rightarrow -\mu\kappa / \sigma^2 + 1/2 = 0 \Rightarrow \sigma = \sqrt{2\mu\kappa}$$

- **Τυπικές τιμές *σ: 4 KB, 8 KB***

Διαχωρισμός των χώρων εντολών και δεδομένων



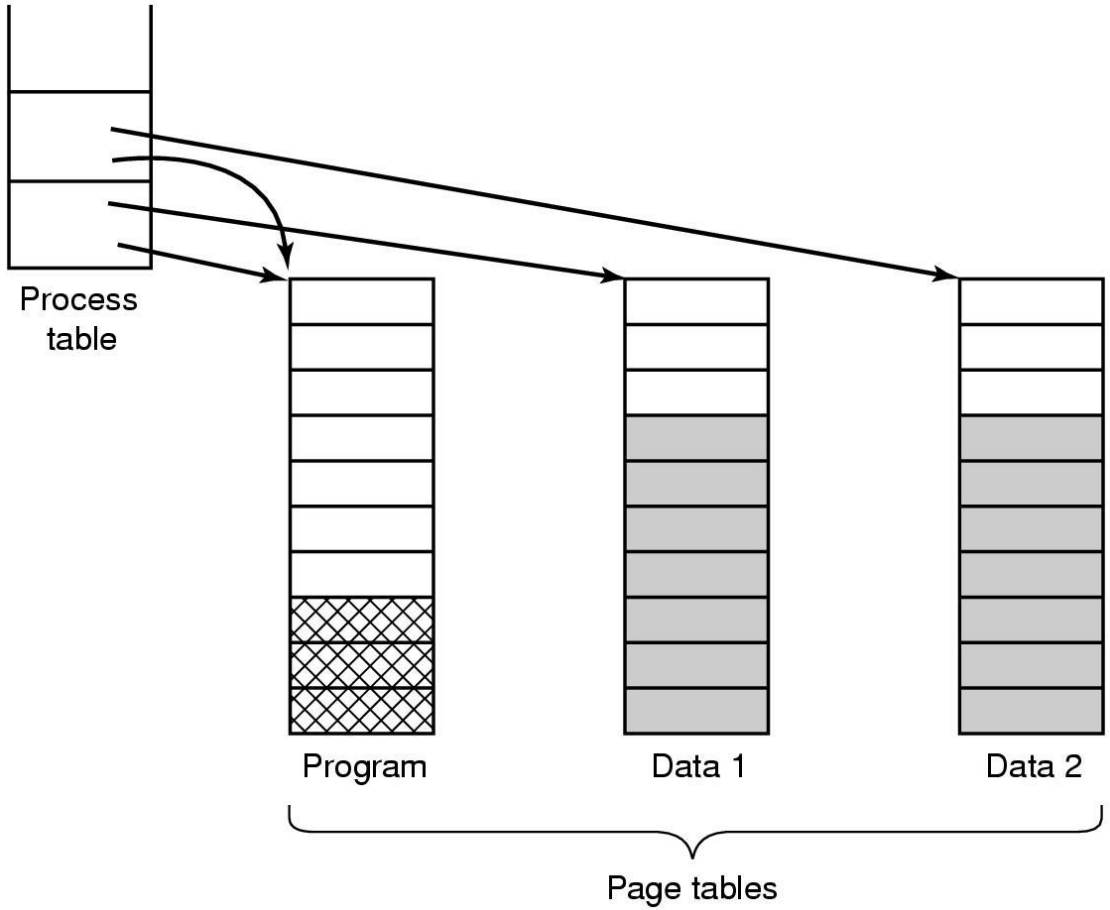
Εικόνα 3-25. (a) Ενιαίος χώρος διευθύνσεων. (b) Διαχωρισμός των χώρων Εντολών (Instructions) και Δεδομένων (Data).

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Κοινόχρηστες σελίδες (1)

- Εάν 2 χρήστες εκτελούν το ίδιο πρόγραμμα, είναι δυνατή η κοινή χρήση των σελίδων του προγράμματος.
- Μόνο οι read-only σελίδες μπορούν να διαμοιραστούν (άρα είναι εύκολο για το χώρο Εντολών).
- Είναι ευκολότερο με ξεχωριστούς χώρους Ε και χώρους Δ.
- Πρόβλημα: αν αφαιρεθεί η μία διεργασία από τη μνήμη, θα προκληθούν πολλά σφάλματα σελίδας στην άλλη διεργασία!

Κοινόχρηστες σελίδες (2)



Εικόνα 3-26. Δύο διεργασίες που μοιράζονται το ίδιο πρόγραμμα μοιράζονται τον πίνακα σελίδων του.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Κοινόχρηστες σελίδες (3)

- Η κοινή χρήση δεδομένων είναι δυσκολότερη.
  - Κλήση *fork*( ): η γονική και θυγατρική διεργασία μοιράζονται κώδικα και δεδομένα.
  - Κάθε διεργασία έχει ξεχωριστό πίνακα σελίδων, αλλά και οι 2 χρησιμοποιούν *δείκτες προς το ίδιο σύνολο σελίδων*.
  - Οι κοινόχρηστες σελίδες είναι read-only.
- Μόλις μία διεργασία τροποποιήσει μία κοινόχρηστη σελίδα:
  - Η παραβίαση της προστασίας read-only προκαλεί παγίδευση.
  - Δημιουργείται αντίγραφο της σελίδας (1 για κάθε διεργασία).
  - Και τα δύο αντίγραφα γίνονται read-write.
  - Συνεπώς, μόνο οι σελίδες που τροποποιούνται αντιγράφονται (***αντιγραφή κατά την εγγραφή*** – *copy on write*).

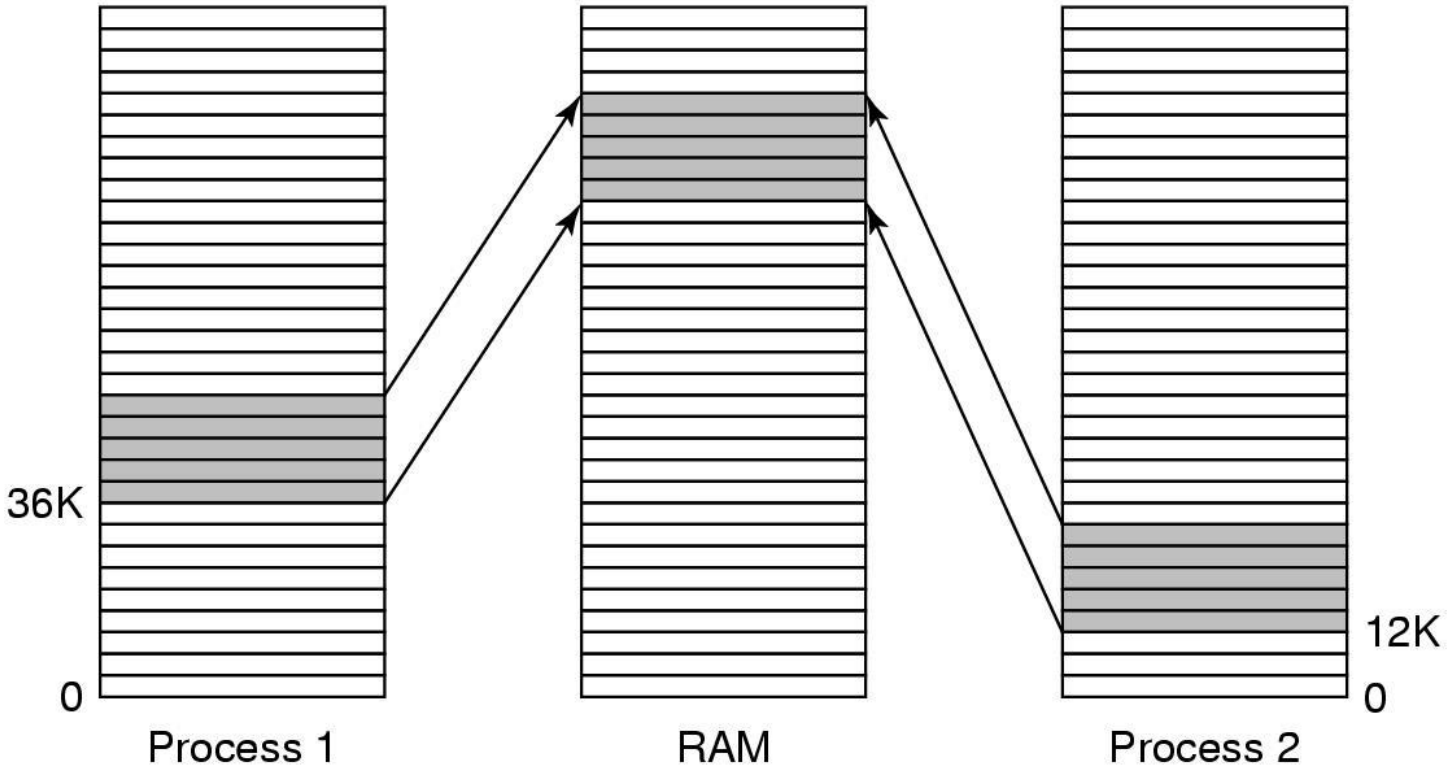
Κοινόχρηστες βιβλιοθήκες (1)

- Στα ΛΣ υπάρχουν μεγάλες κοινόχρηστες βιβλιοθήκες.
- Αν δεσμευόντουσαν στατικά ολόκληρες οι βιβλιοθήκες για κάθε εκτελέσιμο πρόγραμμα θα υπήρχε τεράστια διόγκωση.
- Δυναμική Δέσμευση Βιβλιοθηκών.
  - Περίπτωση Windows: DLL – Dynamic Link Libraries
  - Ο linker *ψάχνει δυναμικά* τα object files που χρειάζεται το πρόγραμμα και τις συναρτήσεις που αυτά καλούν.
  - Π.χ. το πρόγραμμα καλεί την *printf*( ).
  - Η printf( ) χρησιμοποιεί την *write*( ).
  - Ο linker βρίσκει τις *printf* και *write* και δημιουργεί ένα binary file που τις περιέχει.
  - Οι συναρτήσεις που υπάρχουν στη βιβλιοθήκη αλλά δεν καλούνται, *δεν συμπεριλαμβάνονται*.

Κοινόχρηστες βιβλιοθήκες (2)

- Φορτώνονται στη μνήμη των κοινόχρηστων βιβλιοθηκών:
  - Όταν το πρόγραμμα γίνεται link με κάποια DLL δεν περιλαμβάνεται η κανονική συνάρτηση, αλλά *μία μικρή ρουτίνα η οποία δεσμεύεται κατά το χρόνο εκτέλεσης*.
  - Οι κοινόχρηστες βιβλιοθήκες φορτώνονται όταν καλούνται για 1η φορά κάποιες συναρτήσεις τους.
  - Εάν ένα άλλο πρόγραμμα ζητήσει μία συνάρτηση που *έχει ήδη χρησιμοποιηθεί*, οι αντίστοιχες σελίδες *θα βρίσκονται ήδη στη μνήμη*.

Κοινόχρηστες βιβλιοθήκες (3)



Εικόνα 3-27. Μια κοινόχρηστη βιβλιοθήκη που χρησιμοποιείται από δύο διεργασίες.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Κοινόχρηστες βιβλιοθήκες (4)

- Οι διεργασίες μοιράζονται την βιβλιοθήκη η οποία βρίσκεται σε *διαφορετική θέση* για κάθε διεργασία.
  - Στη διεργασία 1, η βιβλιοθήκη ξεκινά από 36K
  - Στη διεργασία 2, η βιβλιοθήκη ξεκινά από 12K
- Εάν γίνει μία απόλυτη αναφορά από την 1η διεργασία στη θέση διεύθυνση 16 της βιβλιοθήκης, θα πρέπει να γίνει επανατοποθέτηση.
- Λύση: **Position-independent code.**
  - Οι κοινόχρηστες βιβλιοθήκες μεταγλωττίζονται με μία σημαία η οποία απαγορεύει τη δημιουργία εντολών που χρησιμοποιούν απόλυτες διευθύνσεις.
  - ***Επιτρέπονται μόνο σχετικές διευθύνσεις***.
  - Π.χ: άλμα *n* bytes πίσω αντί για άλμα στη διεύθυνση X.

4η ενότητα

ΣΥΣΤΗΜΑΤΑ ΑΡΧΕΙΩΝ

Συστήματα αρχείων

Περιεχόμενα

Εισαγωγικές έννοιες (1)

- Όλες σχεδόν οι εφαρμογές διαβάζουν, γράφουν ή τροποποιούν κάποια δεδομένα.
- Τα δεδομένα αυτά θα πρέπει να είναι διαθέσιμα για μακρόχρονη χρήση (πιθανώς για πάντα).
- Βασικές απαιτήσεις για την μακρόχρονη αποθήκευση δεδομένων:
  1. Όγκος δεδομένων: Θα πρέπει να είναι δυνατή η αποθήκευση μεγάλης ποσότητας δεδομένων (>> RAM).
  2. Διατήρηση δεδομένων: οι πληροφορίες πρέπει να εξακολουθούν να υπάρχουν μετά τον τερματισμό της διεργασίας που τα δημιούργησε.
  3. Παράλληλη χρήση δεδομένων: πολλές διεργασίες θα πρέπει να έχουν τη δυνατότητα ταυτόχρονης χρήσης.

Εισαγωγικές έννοιες (2)

- Μέσα για την μακροπρόθεσμη αποθήκευση δεδομένων:
  - δίσκοι, μαγνητικές ταινίες, CD, DVD κτλ.
- Θεωρούμε τον δίσκο ως μία *γραμμική ακολουθία από μπλοκ* σταθερού μεγέθους, με 2 μόνο λειτουργίες:
  1. Ανάγνωση του μπλοκ k.
  2. Εγγραφή στο μπλοκ k.

- Βασικά θέματα που προκύπτουν είναι:
  1. Πως θα αναζητούνται τα δεδομένα;
  2. Πως προστατεύονται τα δεδομένα ενός χρήστη από κάποιον άλλο χρήστη?
  3. Πως βρίσκονται τα ελεύθερα μπλοκ;

4.1 Αρχεία

Αρχεία (files)

- Αρχεία: μία νέα αφαίρεση του ΛΣ για την διαχείριση των δεδομένων ενός συστήματος.
- Λογικές μονάδες δεδομένων, τις οποίες δημιουργούν, και διαχειρίζονται οι διεργασίες.
- Αποτελεί το χώρο διευθύνσεων των δεδομένων στο δίσκο
  - Ότι είναι ο χώρος διευθύνσεων (address space) για τη RAM, είναι το αρχείο για το δίσκο.
- Η διαχείριση των αρχείων γίνεται από το ΛΣ.
  - Δημιουργία, τροποποίηση, καταστροφή.
  - Ονομασία, δόμηση, υλοποίηση.
  - Προσπέλαση, προστασία, χρήση.

Ονομασία αρχείων (1)

- Μέγεθος ονόματος:
  - Στο FAT-16, FAT-32 (MS-DOS μέχρι Win98), μέχρι 8 χαρακτήρες.
  - Στο NTFS (WIN NT και μετά) μέχρι 255 χαρακτήρες
- Διάκριση πεζών-κεφαλαίων:
  - Στα windows δεν γίνεται διάκριση.
  - Στο UNIX γίνεται διάκριση.
- Προεκτάσεις αρχείων (file extensions):
  - Στα windows έχουν νόημα για το ΛΣ.
  - Στο UNIX είναι συμβάσεις των χρηστών.

Ονομασία αρχείων (2)

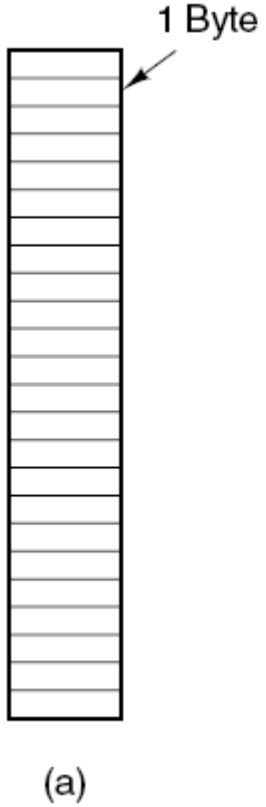
Extension	Meaning
file.bak	Backup file
file.c	C source program
file.gif	Compuserve Graphical Interchange Format image
file.hlp	Help file
file.html	World Wide Web HyperText Markup Language document
file.jpg	Still picture encoded with the JPEG standard
file.mp3	Music encoded in MPEG layer 3 audio format
file.mpg	Movie encoded with the MPEG standard
file.o	Object file (compiler output, not yet linked)
file.pdf	Portable Document Format file
file.ps	PostScript file
file.tex	Input for the TEX formatting program
file.txt	General text file
file.zip	Compressed archive

Εικόνα 4-1. Τυπικές προεκτάσεις ονομάτων αρχείων.

Δομή των αρχείων (1)

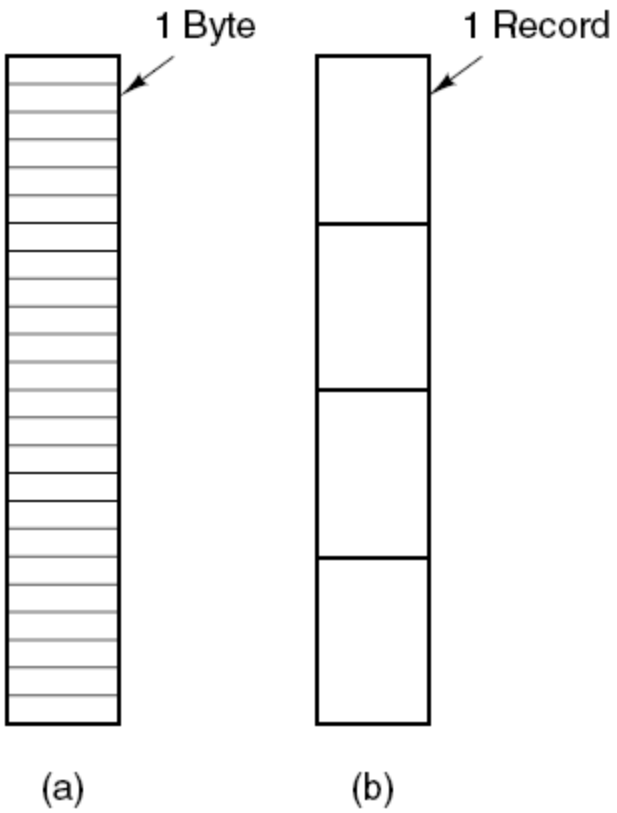
(1) Ακολουθία Byte

- Καμία δόμηση. Μονάδα του αρχείου είναι το byte.
- Το νόημα που θα δοθεί στα δεδομένα του αρχείου είναι δουλειά του προγράμματος χρήστη και δεν αφορά το ΛΣ.
- Το ΛΣ δεν απαγορεύει, ούτε επιβάλει κάποια ειδική δομή).
- Μέγιστη ευελιξία.
- Χρησιμοποιείται στο UNIX, windows, κτλ.



Εικόνα 4-2. Τρία είδη δόμηση αρχείων. (a) ακολουθία Byte.

Δομή των αρχείων (2)

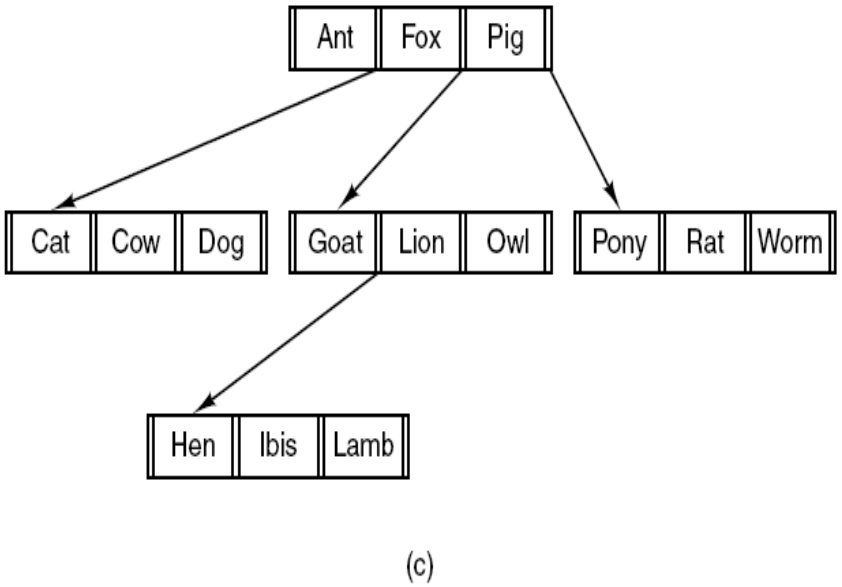


(2) Ακολουθία εγγραφών (record)

- Μονάδα του αρχείου είναι μία εγγραφή με συγκεκριμένη μορφή.
- Το ΛΣ απαγορεύει τη δημιουργία αρχείων με διαφορετική δομή.
- Η ανάγνωση γίνεται επίσης ανά εγγραφή.
- Δεν χρησιμοποιείται πλέον.

Εικόνα 4-2. Τρία είδη δόμηση αρχείων. (b) ακολουθία εγγραφών.

Δομή των αρχείων (3)



(c)

(3) Δένδρο (tree)

- Μονάδα του αρχείου είναι μία εγγραφή, αλλά χωρίς συγκεκριμένη μορφή.
- Κάθε εγγραφή έχει όμως υποχρεωτικά ένα κλειδί (key).
- Η αναζήτηση ή τοποθέτηση ενός αρχείου στο δένδρο, γίνεται με βάση το κλειδί.
- Χρησιμοποιείται σε mainframes για επεξεργασία εμπορικών δεδομένων.

Εικόνα 4-2. Τρία είδη δόμηση αρχείων. (c) Δένδρο.

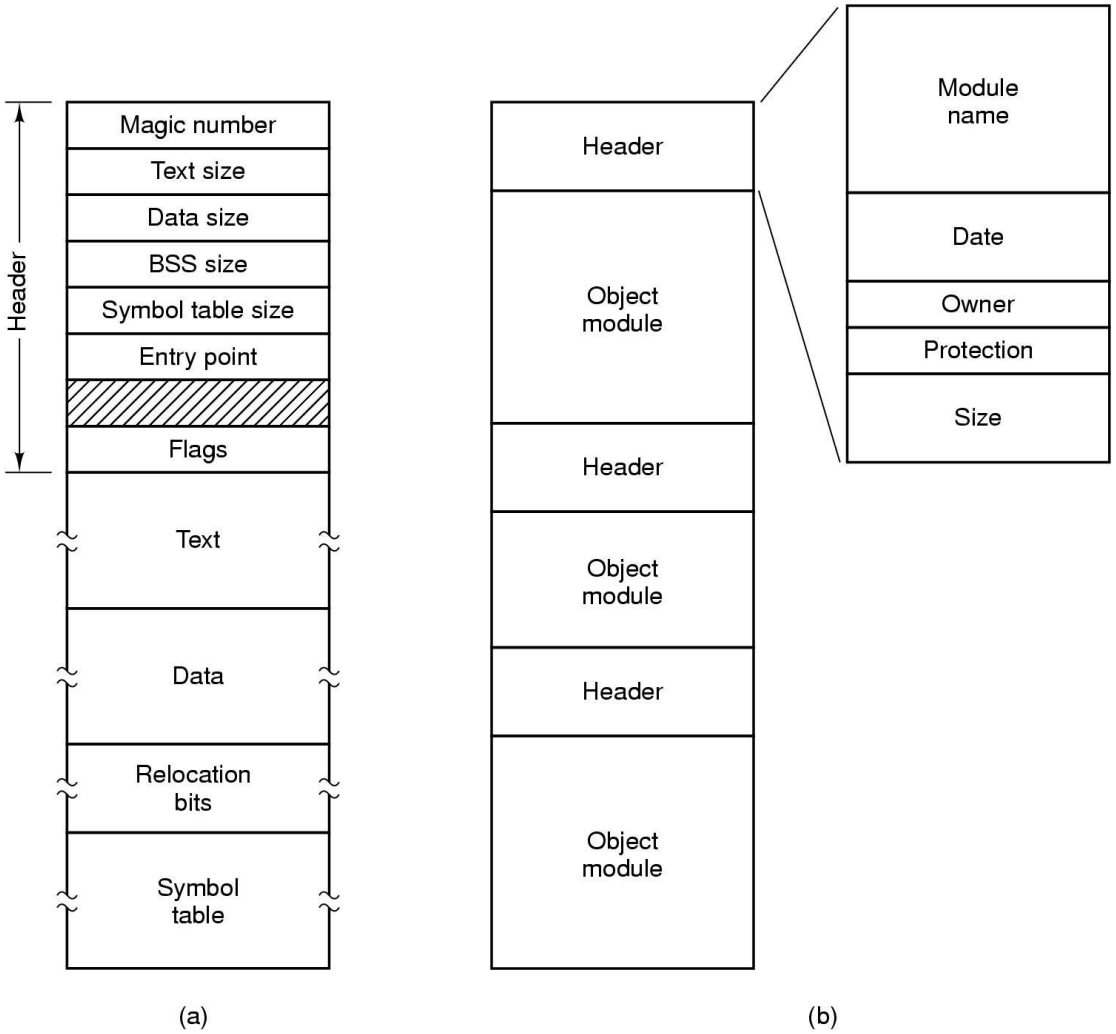
Τύποι αρχείων (1)

- UNIX:
  - Κανονικά αρχεία
  - Κατάλογοι
  - Ειδικά αρχεία χαρακτήρων (χρησιμοποιούνται για μοντελοποίηση σειριακών συσκευών Ε/Ε)
  - Ειδικά αρχεία μπλοκ (χρησιμοποιούνται για μοντελοποίηση δίσκων)
- Windows:
  - Μόνο κανονικά αρχεία και κατάλογοι

Τύποι αρχείων (2)

- Τα κανονικά αρχεία χωρίζονται σε ASCII και binary files
- ASCII αρχεία:
  - Περιλαμβάνουν κείμενο και ειδικούς χαρακτήρες (line feed, carrige return).
  - Μπορούν να επεξεργαστούν, τυπωθούν, χρησιμοποιηθούν για Ε/Ε, επικοινωνία διεργασιών κτλ.
- Binary (δυαδικά) αρχεία:
  - Ότι δεν είναι ascii.
  - Εσωτερική δομή, γνωστή στον προγραμματιστή.
  - Δεν επεξεργάζονται ούτε τυπώνονται.

Τύποι αρχείων (3)



Εικόνα 4-3. Δύο δυαδικά αρχεία: (a) Εκτελέσιμο. (b) αρχειοθήκη.

Πρόσβαση στα αρχεία

- Αρχεία σειριακής πρόσβασης (sequential files)
  - Ανάγνωση με τη σειρά, όλων των byte.
  - Μόνο δυνατότητα rewind.
- Αρχεία τυχαίας προσπέλασης (random access files)
  - Ανάγνωση σε οποιαδήποτε θέση.
  - Δυνατότητα αναζήτησης θέσης μέσα στο αρχείο (seek).

Χαρακτηριστικά (attributes) των αρχείων

Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file was last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

Εικόνα 4-4. Μερικά χαρακτηριστικά αρχείων.

Λειτουργίες αρχείων

Οι βασικές κλήσεις συστήματος που για τα αρχεία:

- Create
- Delete
- Open
- Close
- Read
- Write
- Append
- Seek
- Get Attributes
- Set Attributes
- Rename

Παράδειγμα προγράμματος που χρησιμοποιεί κλήσεις συστήματος αρχείων (1)

```
/* File copy program. Error checking and reporting is minimal. */

#include <sys/types.h>                  /* include necessary header files */
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]);       /* ANSI prototype */

#define BUF_SIZE 4096                   /* use a buffer size of 4096 bytes */
#define OUTPUT_MODE 0700                /* protection bits for output file */

int main(int argc, char *argv[])
{
    int in_fd, out_fd, rd_count, wt_count;
    char buffer[BUF_SIZE];

    if (argc != 3) exit(1);             /* syntax error if argc is not 3 */

    /* Open the input file and create the output file */
    in_fd = open(argv[1], O_RDONLY);    /* open the source file */
    if (in_fd < 0) exit(2);             /* if it cannot be opened, exit */
    out_fd = creat(argv[2], OUTPUT_MODE); /* create the destination file */
    if (out_fd < 0) exit(3);            /* if it cannot be created, exit */
        • • •
```

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

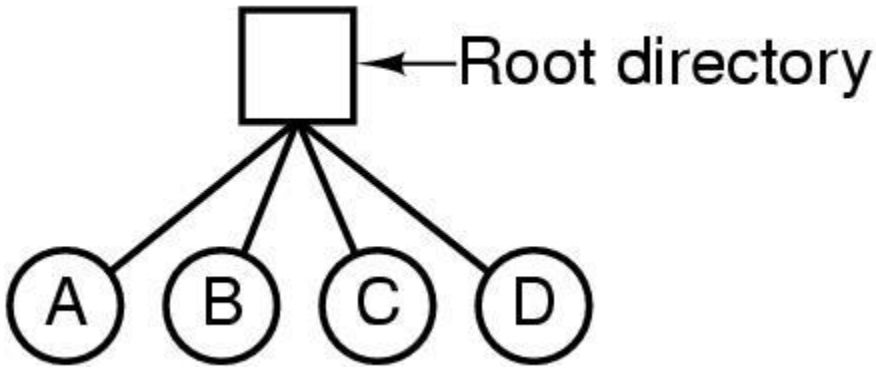
Παράδειγμα προγράμματος που χρησιμοποιεί κλήσεις συστήματος αρχείων (2)

```
        /* Copy loop */
        while (TRUE) {
                rd_count = read(in_fd, buffer, BUF_SIZE); /* read a block of data */
        if (rd_count <= 0) break;                       /* if end of file or error, exit loop */
                wt_count = write(out_fd, buffer, rd_count); /* write data */
                if (wt_count <= 0) exit(4);             /* wt_count <= 0 is an error */
        }

        /* Close the files */
        close(in_fd);
        close(out_fd);
        if (rd_count == 0)                              /* no error on last read */
                exit(0);
        else
                exit(5);                                /* error on last read */
}
```

Εικόνα 4-5. Ένα απλό πρόγραμμα αντιγραφής αρχείων.

4.2 Κατάλογοι

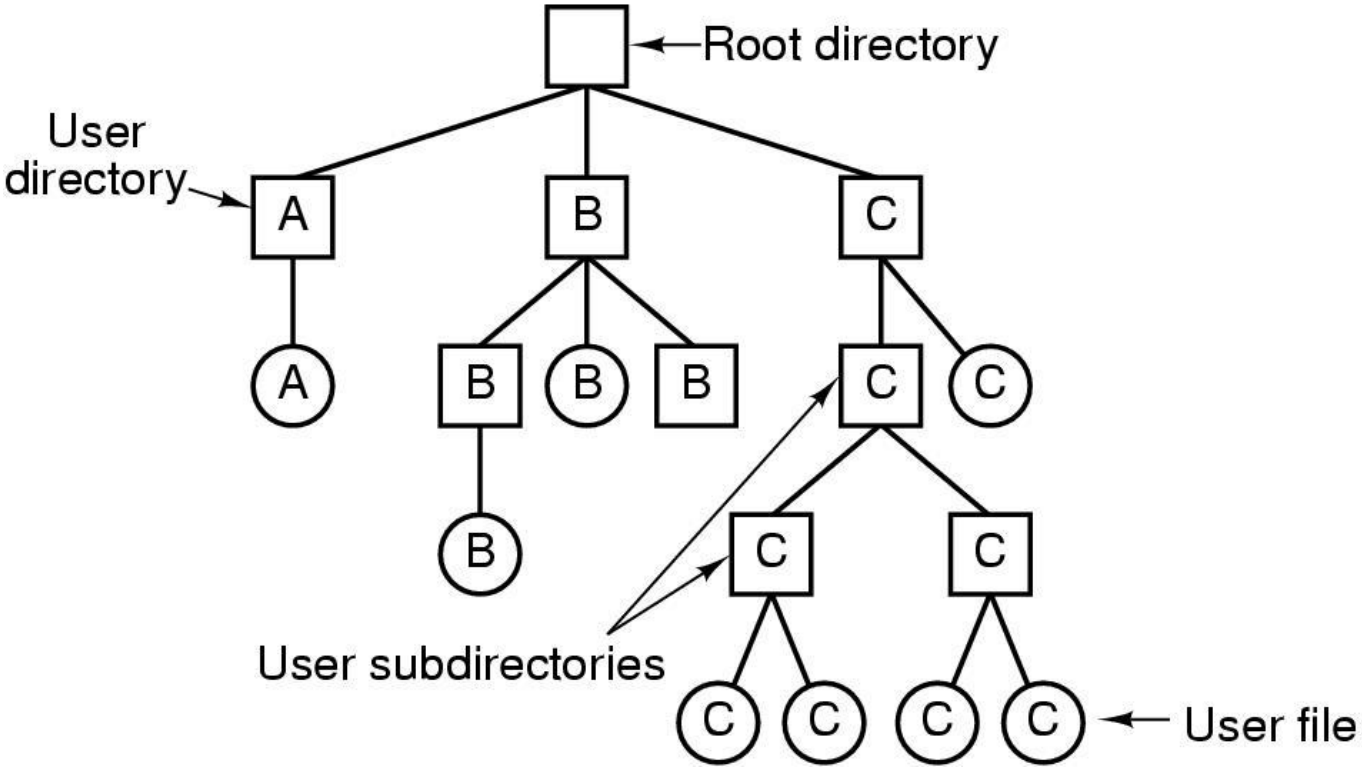
Συστήματα καταλόγων ενός επιπέδου



Εικόνα 4-6. Ένα μονοεπίπεδο σύστημα καταλόγου με 4 αρχεία.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

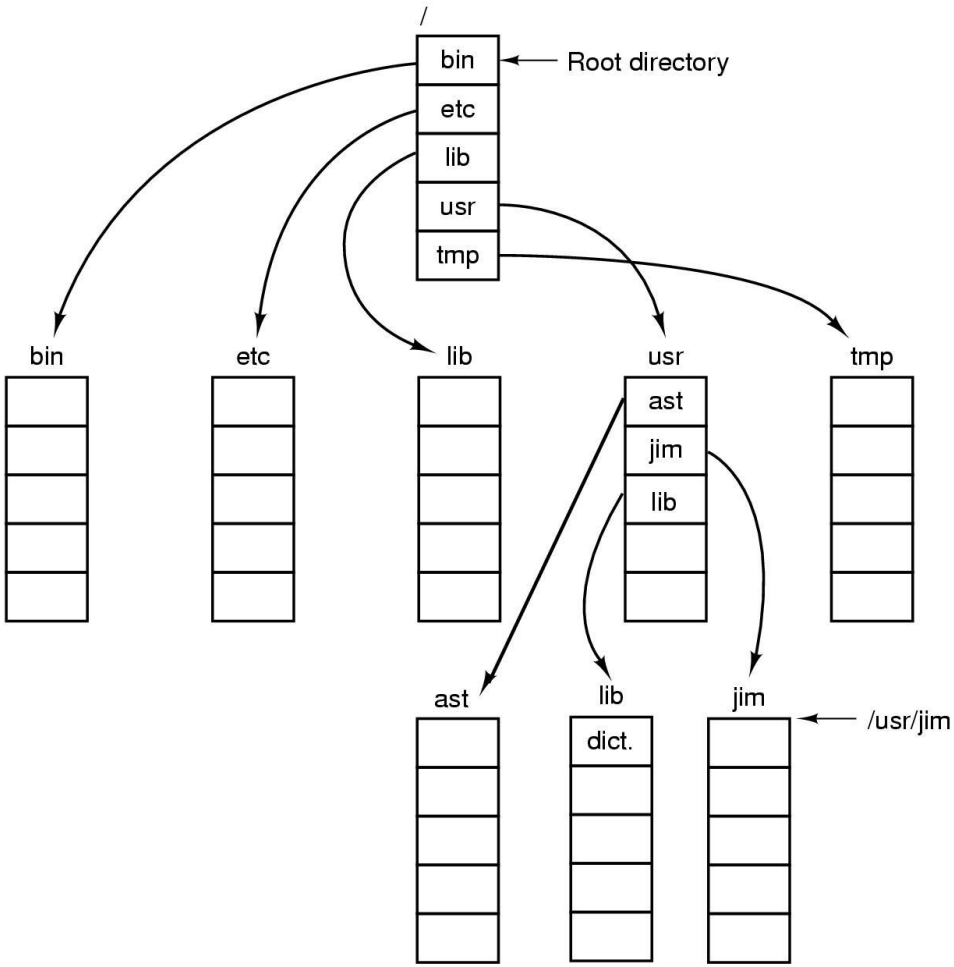
Ιεραρχικά συστήματα καταλόγων



Εικόνα 4-7. Ένα ιεραρχικό σύστημα καταλόγων.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Ονόματα διαδρομών



Εικόνα 4-8. Ένα δένδρο καταλόγων του UNIX.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Λειτουργίες καταλόγων

Κλήσεις συστήματος για τη διαχείριση καταλόγων:

- Create
- Delete
- Opendir
- Closedir
- Readdir
- Rename
- Link
- Unlink

4.3 Υλοποίηση συστήματος αρχείων

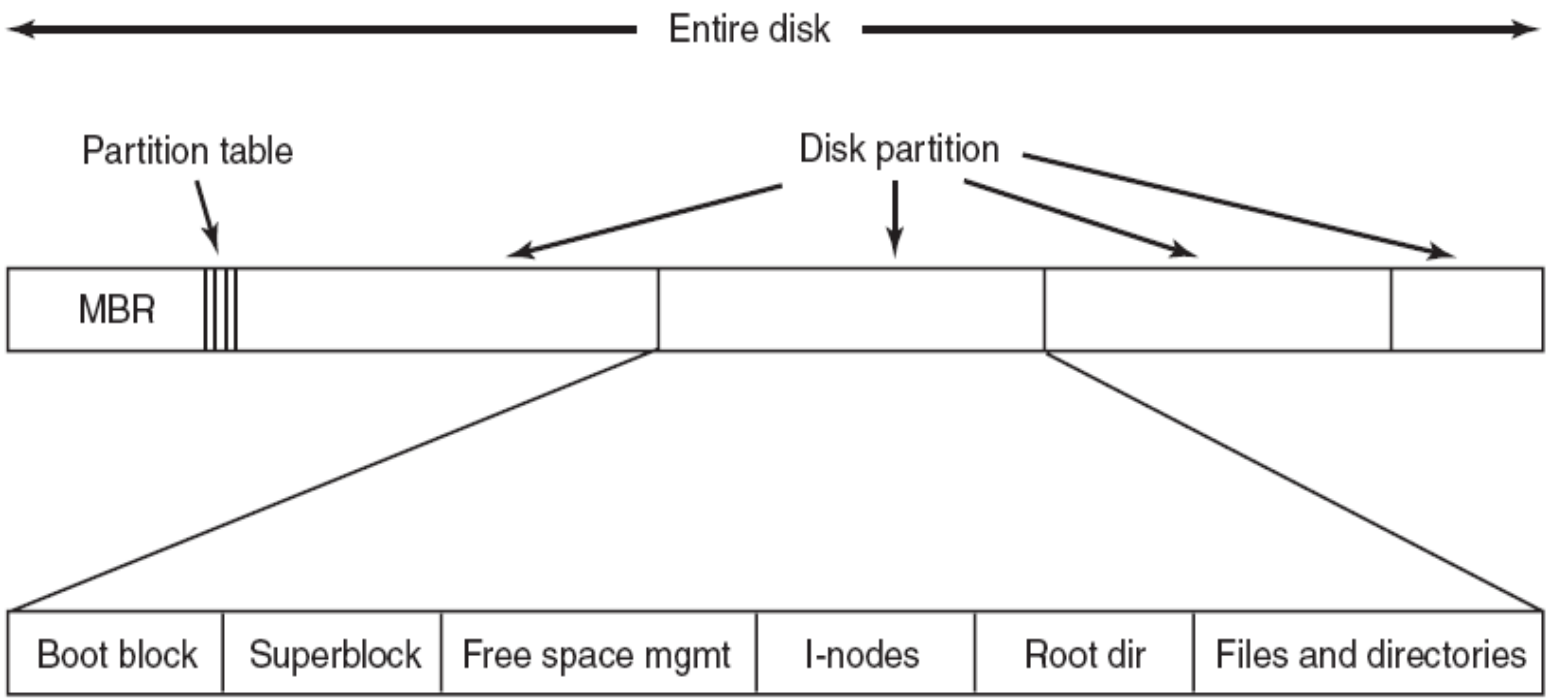
Διαφορετικές όψεις χρήσης και υλοποίησης συστήματος αρχείων

- Άποψη του χρήστη
  - Ονομασία αρχείων
  - Τρόπος προσπέλασης αρχείων και καταλόγων.
  - Δομή αποθήκευσης (δένδρο αρχείων).
  - Θέματα διασύνδεσης.
- Άποψη του προγραμματιστή
  - Αποθήκευση των αρχείων και καταλόγων.
  - Διαχείριση χώρου στο δίσκο.
  - Θέματα απόδοσης και αξιοπιστίας.

Διάταξη του συστήματος αρχείων (1)

- Διαμέρισμα δίσκου (disk partition)
  - Κάθε διαμέρισμα ενός δίσκου μπορεί να έχει διαφορετικό σύστημα αρχείων.
- Βασική Εγγραφή Εκκίνησης (Master Boot Record) – MBR: Χρησιμοποιείται για την εκκίνηση του υπολογιστή. Είναι ο τομέας 0 του δίσκου. Περιλαμβάνει:
  - Τον πίνακα διαμερισμάτων (partition table): Βρίσκεται στο τέλος του MBR. Το BIOS φορτώνει στη μνήμη το MBR και αναζητεί το ενεργό διαμέρισμα.
  - Το μπλοκ εκκίνησης (boot block): Είναι το 1ο μπλοκ στο MBR το οποίο φορτώνει το ενεργό διαμέρισμα.

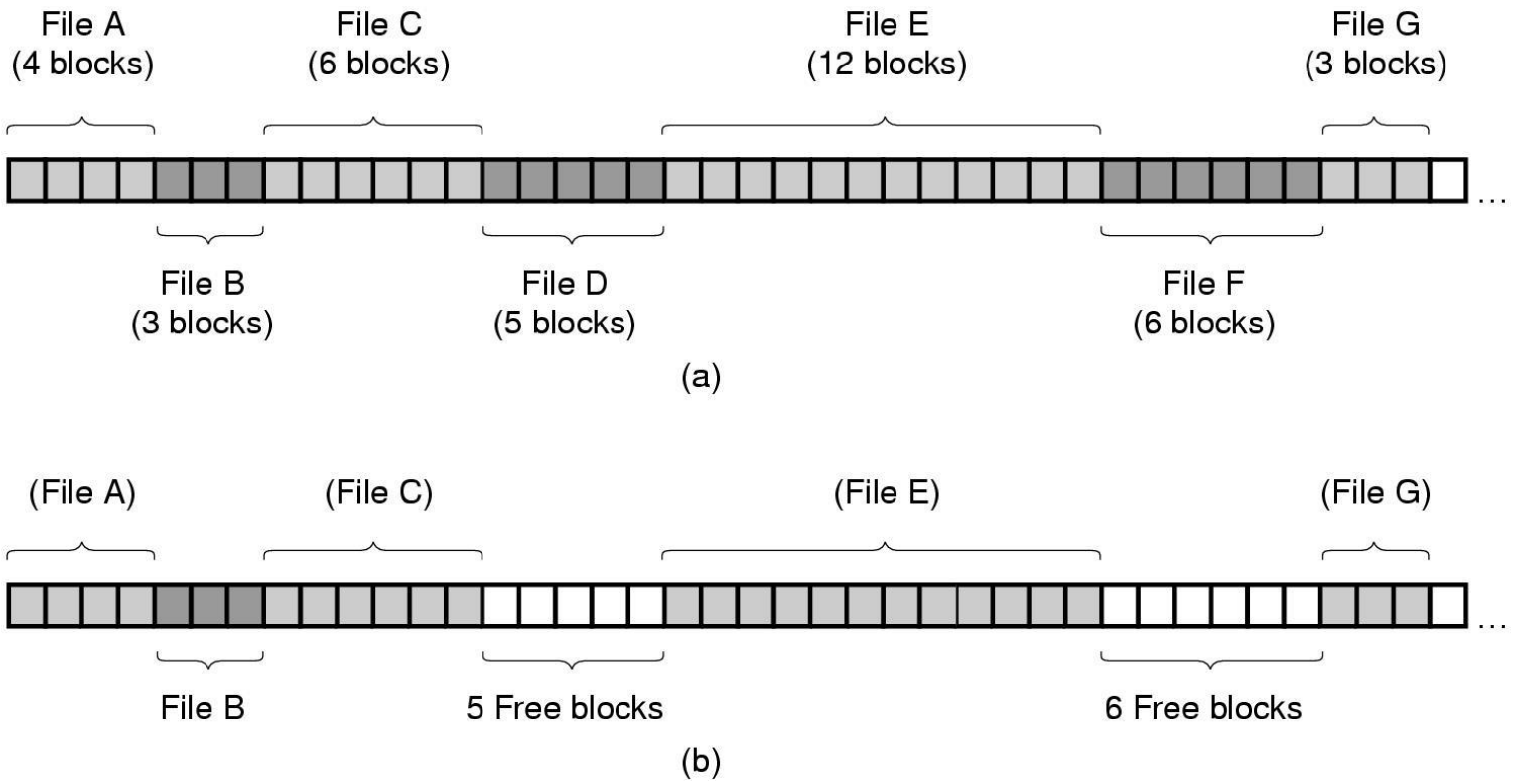
Διάταξη του συστήματος αρχείων (2)



Εικόνα 4-9. Πιθανή διάταξη του συστήματος αρχείων.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

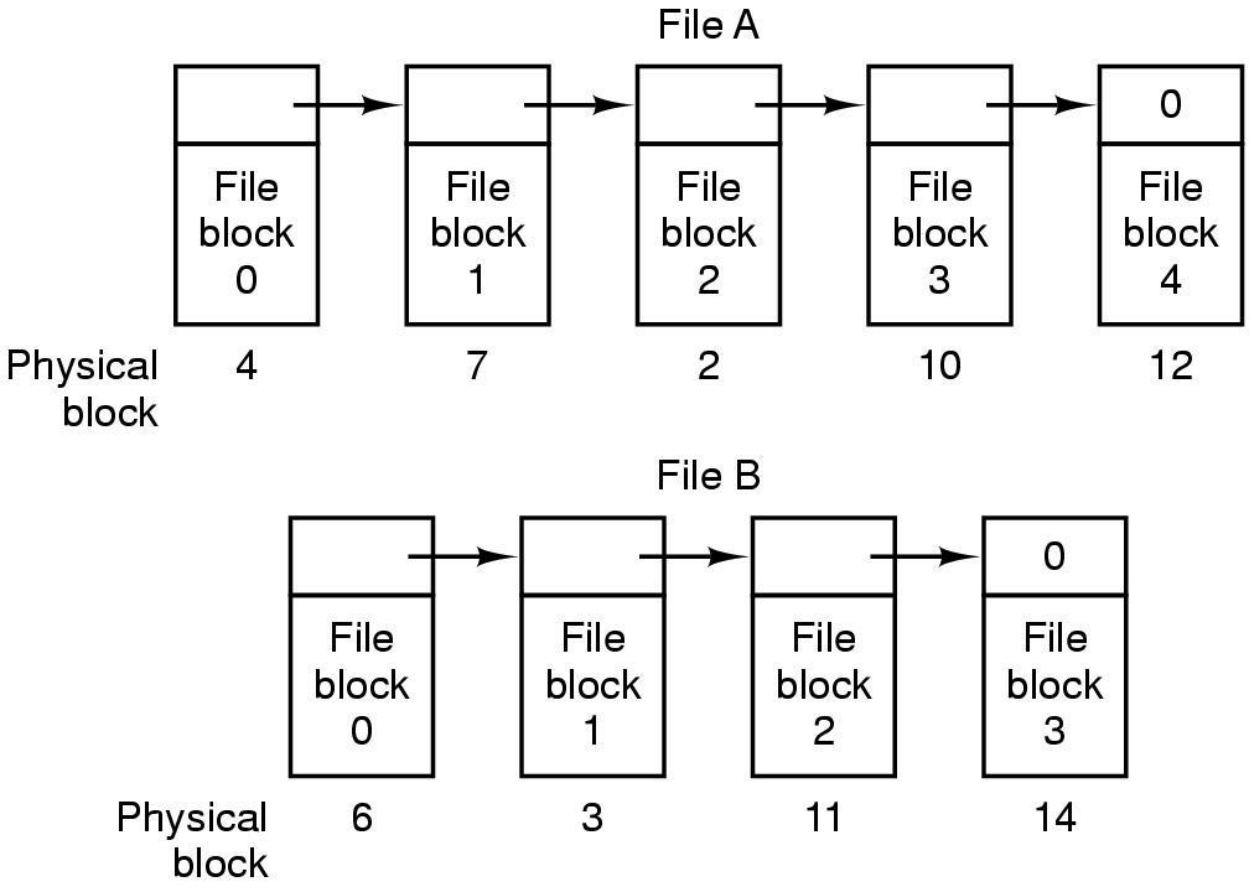
Υλοποίηση των αρχείων (συνεχής κατανομή)



Εικόνα 4-10. (a) Συνεχής κατανομή του χώρου του δίσκου σε 7 αρχεία. (b) Η κατάσταση του δίσκου μετά από τη διαγραφή των αρχείων D και F.

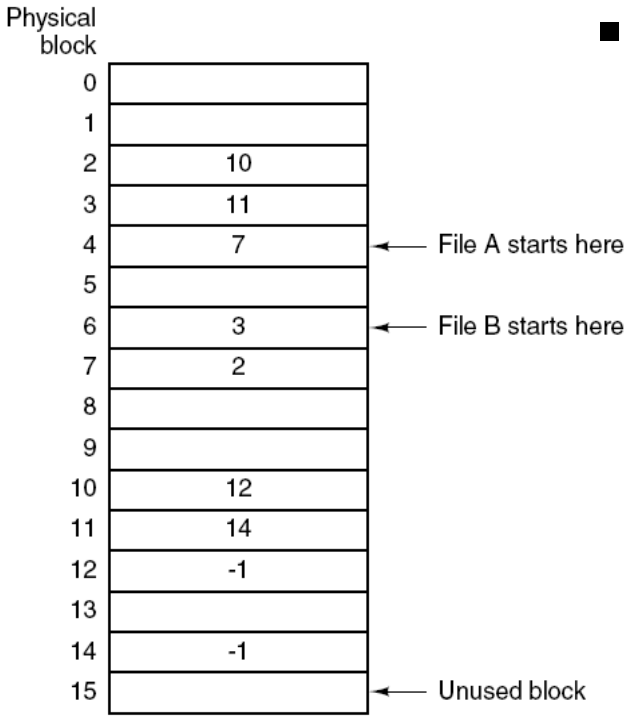
Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Υλοποίηση των αρχείων (κατανομή συνδεδεμένης λίστας)



Εικόνα 4-11. Αποθήκευση αρχείου σαν μία συνδεδεμένη λίστα από μπλοκ δίσκου.

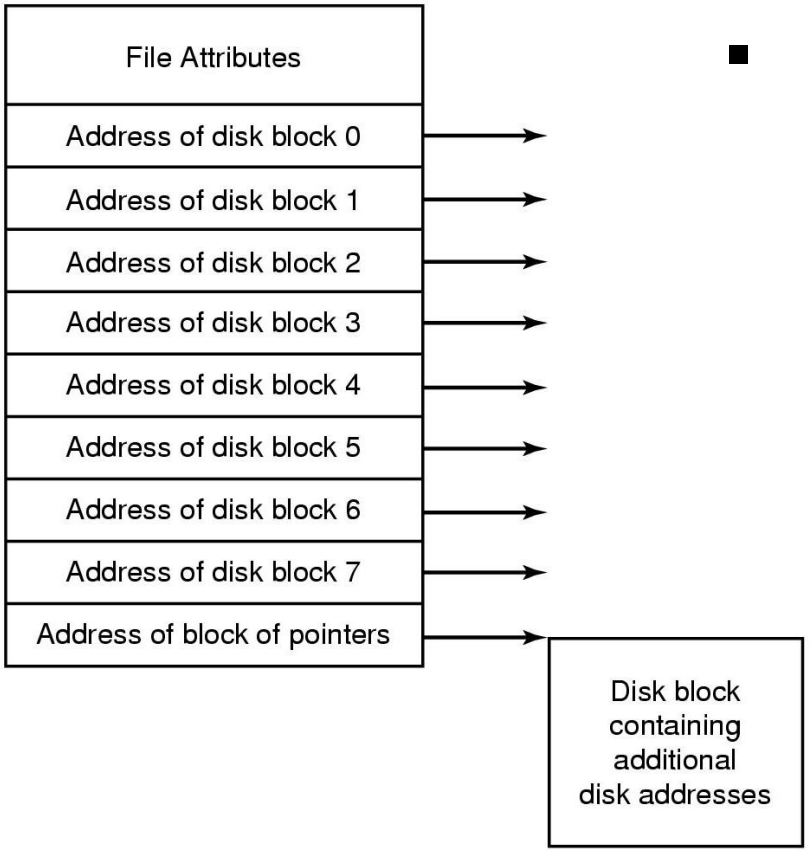
Υλοποίηση των αρχείων
(κατανομή συνδεδεμένης λίστας με πίνακα στη μνήμη)



- Τοποθέτηση σε ένα πίνακα, όλων των δεικτών από κάθε μπλοκ, και φόρτωση του πίνακα στη μνήμη.
  - Κάθε εγγραφή στη λίστα δείχνει ποιο (λογικό) μπλοκ του αρχείου αντιστοιχεί σε ποιο (φυσικό) μπλοκ του δίσκου.
  - Επίσης έχει ένα δείκτη προς το επόμενο μπλοκ.
  - Παράδειγμα: FAT-16, FAT-32

Εικόνα 4-12. Κατανομή συνδεδεμένης λίστας με χρήση πίνακα κατανομής αρχείων **(File Allocation Table).**

Υλοποίηση με κόμβους-i (i-nodes)

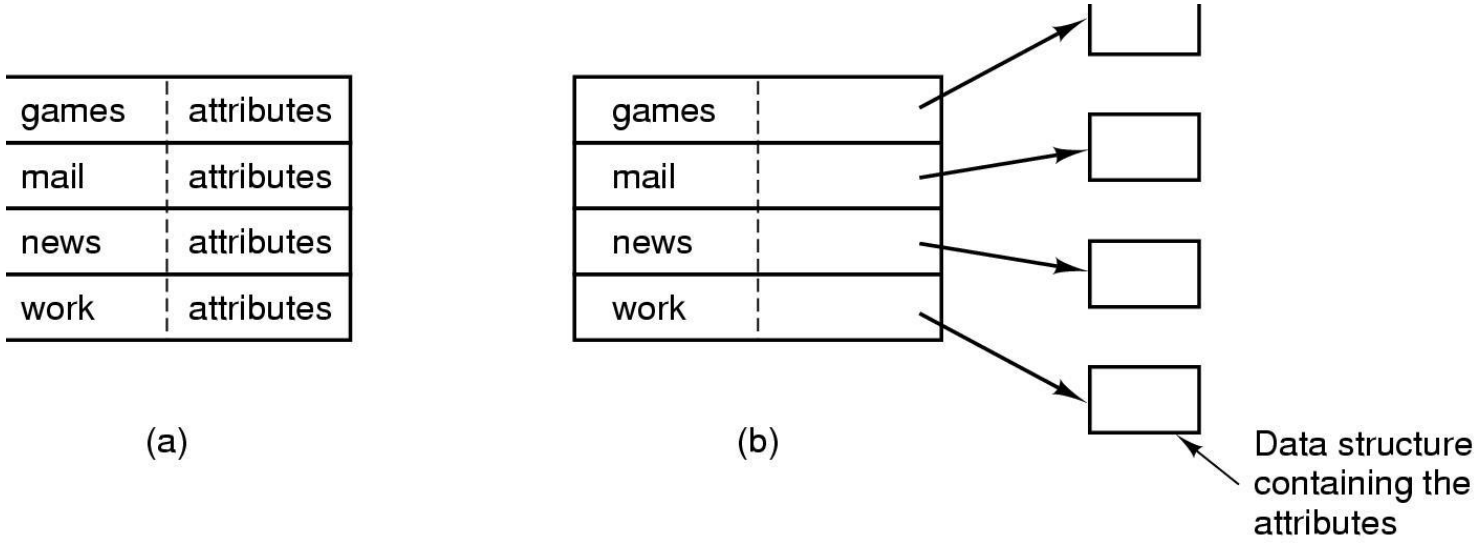


- Σε κάθε αρχείο αντιστοιχεί μία δομή δεδομένων, ο κόμβος-i (i-node), οποίος περιλαμβάνει:
  - Τα χαρακτηριστικά του αρχείου, και
  - Μία εγγραφή για κάθε λογικό μπλοκ του αρχείου, η οποία περιέχει έναν αριθμοδείκτη (index) προς τη διεύθυνση του φυσικού μπλοκ στο δίσκο που έχει αποθηκευτεί.

Εικόνα 4-13. Παράδειγμα ενός κόμβου-i (**index node**)

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Υλοποίηση καταλόγων (1)



Εικόνα 4-14. (a) Κατάλογος ο οποίος περιλαμβάνει εγγραφές σταθερού μεγέθους, με τις διευθύνσεις δίσκου και τα χαρακτηριστικά (Windows).

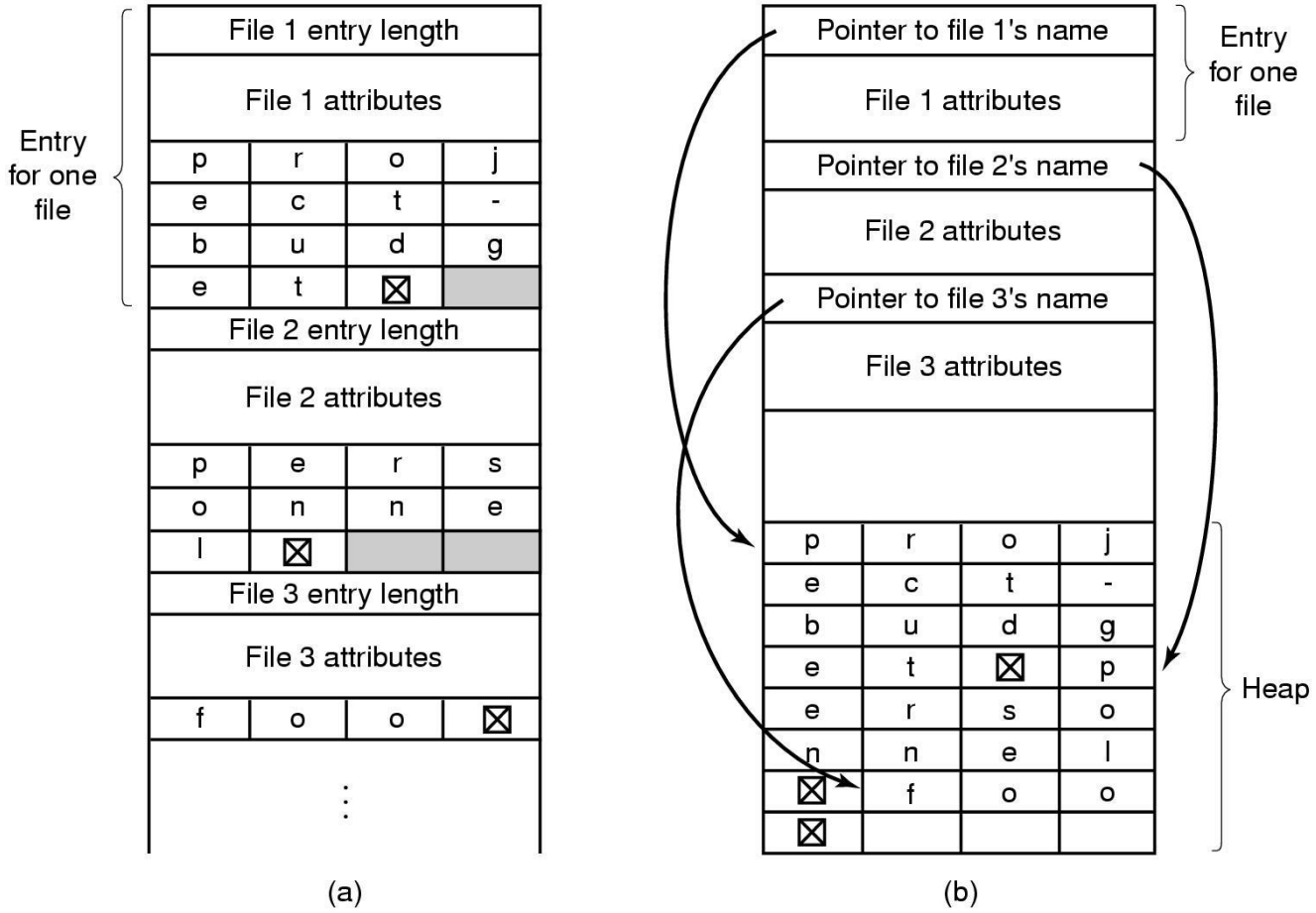
(b) Ένας κατάλογος στον οποίο κάθε καταχώρηση αναφέρεται απλώς σε έναν i-node (UNIX).

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Υλοποίηση καταλόγων (2)
Χρήση μεγάλων ονομάτων αρχείων

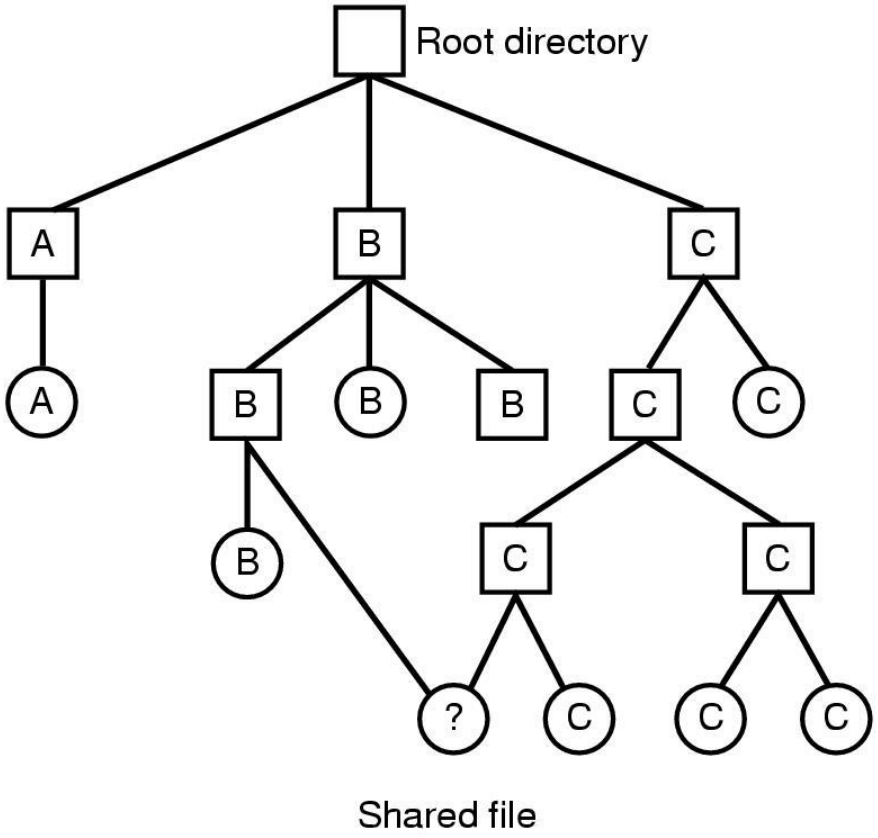
- Κάθε εγγραφή ενός αρχείου στον κατάλογο, έχει μία εγγραφή (π.χ. 255 χαρακτήρες) για το όνομα του αρχείου.
  - Όμως γίνεται σπατάλη σε περίπτωση χρήσης μικρών ονομάτων.
- Λύση: χρήση μεταβλητού μήκους για κάθε εγγραφή στον κατάλογο.

Υλοποίηση καταλόγων (3)
Χρήση μεγάλων ονομάτων αρχείων



Εικόνα 4-15. Χειρισμός μεγάλων ονομάτων αρχείων στου καταλόγους. (a) Εσωτερικά (In-line). (b) Με τη χρήση σωρού (heap).

Κοινόχρηστα αρχεία (1)



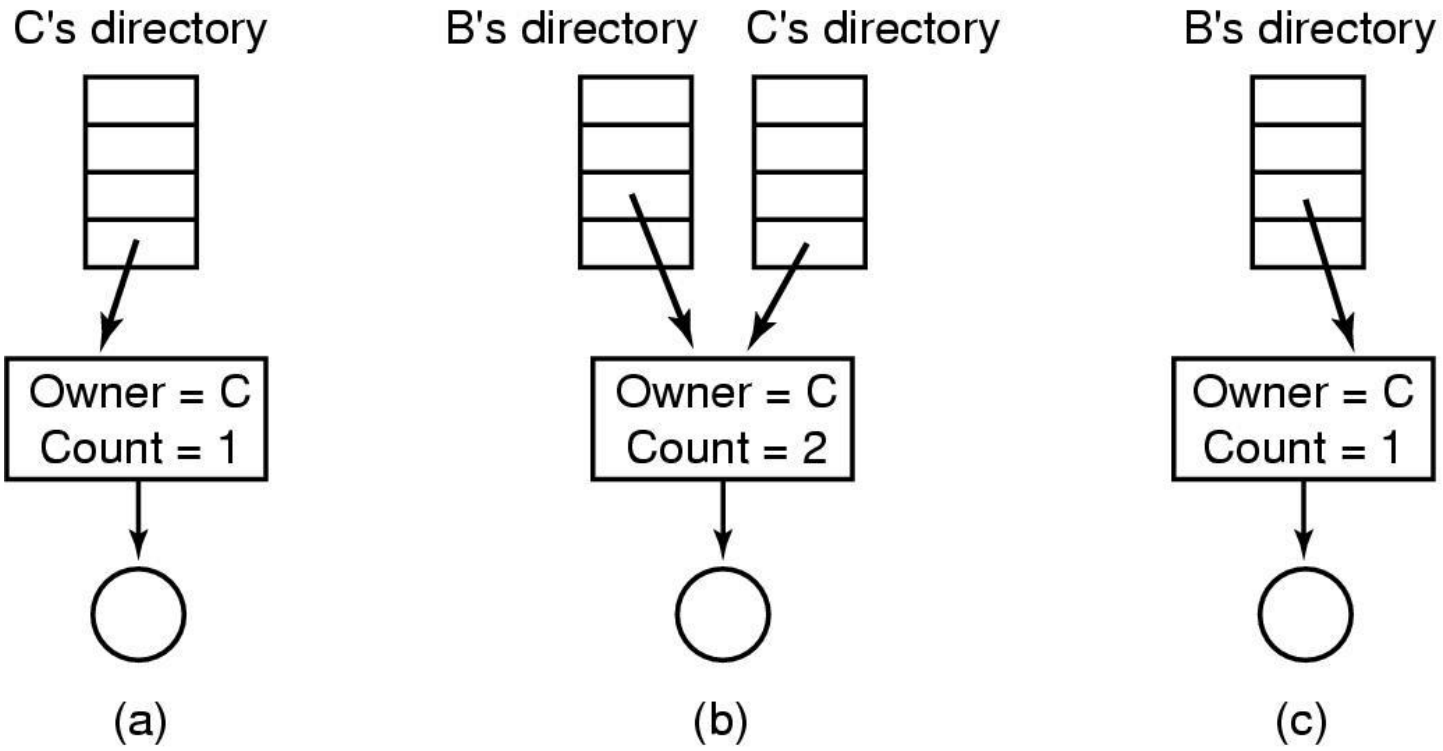
- Η σύνδεση για την κοινή χρήση ενός αρχείου λέγεται σύνδεσμος (link).
- Πλέον δεν έχουμε δομή δένδρου αλλά Προσανατολισμένου Ακυκλικού Γράφου.

Εικόνα 4-16. Σύστημα αρχείων που περιέχει κοινόχρηστο αρχείο.

Κοινόχρηστα αρχεία (2)

- Πραγματική σύνδεση (hard link).
  - Οι κατάλογοι δείχνουν σε i-nodes, έναν για κάθε αρχείο που περιλαμβάνουν.
  - Κάθε i-node περιλαμβάνει τα μπλοκ του δίσκου που αντιστοιχούν στο αρχείο.
  - Στο i-node κάθε αρχείου υπάρχει ένας μετρητής συνδέσεων, του ίδιου αρχείου σε έναν ή περισσότερους καταλόγους.
- Συμβολική σύνδεση (symbolic link)
  - Δημιουργία από το ΛΣ ενός νέου τύπου αρχείου (LINK) και τοποθέτηση αυτού του αρχείου στον κατάλογο Β.
  - Το LINK περιέχει μόνο το όνομα προς το πραγματικό αρχείο.

Κοινόχρηστα αρχεία (3)



Εικόνα 4-17. (a) Κατάσταση πριν τη σύνδεση.
(b) Μετά τη δημιουργία του συνδέσμου από τον B.
(c) Μετά την διαγραφή του αρχείου από τον αρχικό ιδιοκτήτη C.

Ημερολογιακά συστήματα αρχείων (Journaling File Systems) (1)

- Τήρηση ενός ημερολογίου (αρχείου καταγραφής) για την περιγραφή των ενεργειών που πρόκειται να κάνει το ΛΣ, πριν την πραγματοποίηση της ενέργειας.
- Σε περίπτωση κατάρρευσης του συστήματος, εξετάζεται το ημερολόγιο ώστε να μπορέσει να επαναληφθεί η αποτυχημένη λειτουργία.
- Παραδείγματα ημερολογιακών συστημάτων:
  - Windows: NTFS.
  - Linux: ext3, ReiserFS.

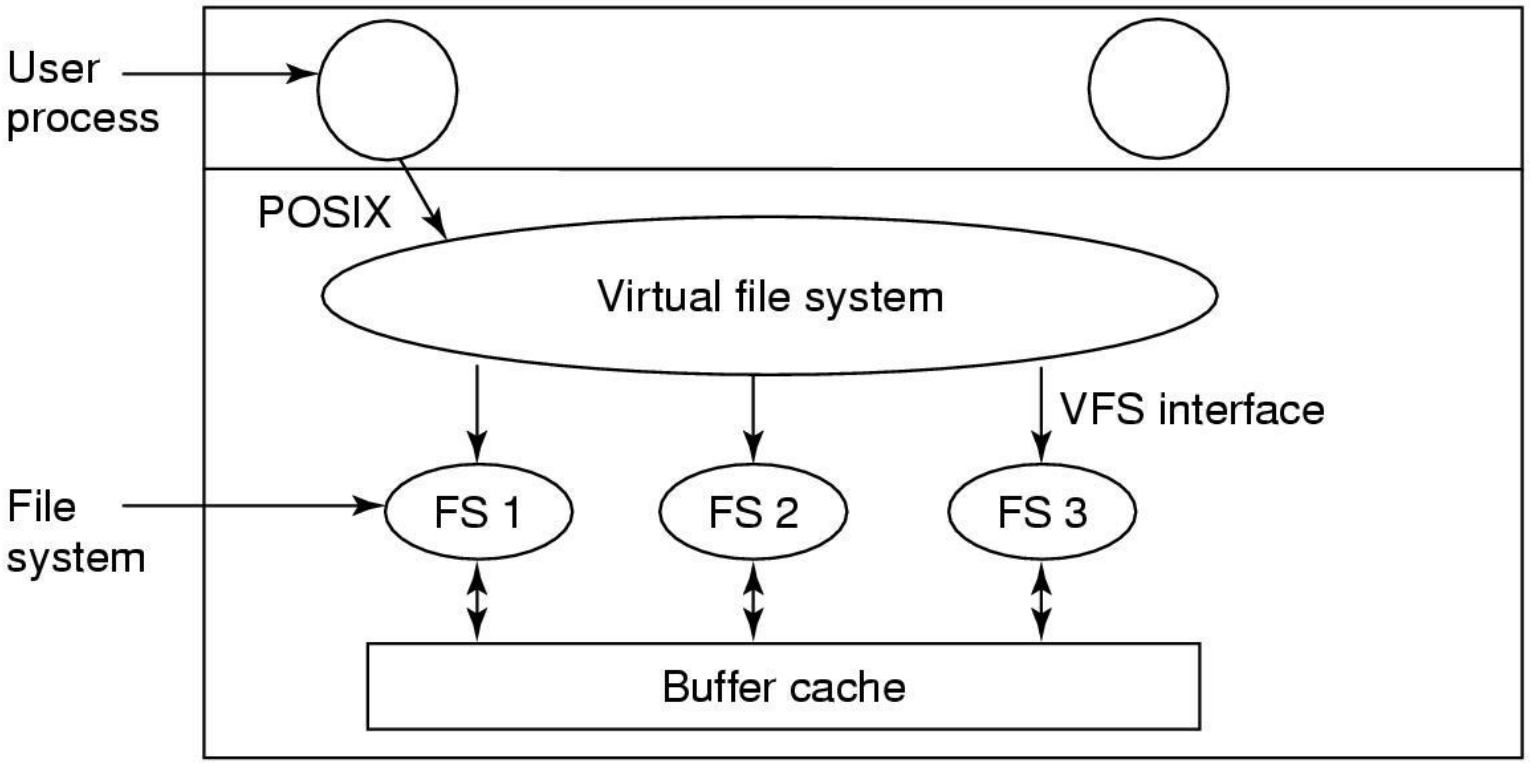
Ημερολογιακά συστήματα αρχείων (Journaling File Systems) (2)

- Παράδειγμα: ενέργειες για τη διαγραφή ενός αρχείου. :
  1. Αφαίρεση αρχείου από τον κατάλογό του.
  2. Αποδέσμευση του i-node που είχε δοθεί στο αρχείο, ώστε να μπορεί να επαναχρησιμοποιηθεί.
  3. Αποδέσμευση των μπλοκ του δίσκου που είχε το αρχείο, (μαρκάρισμα ως ελεύθερα).
- Σε περίπτωση κατάρρευσης του συστήματος :
  1. Το ΛΣ έχει ήδη καταχωρήσει στο ημερολόγιο, τις απαιτούμενες ενέργειες.
  2. Μετά από την εκτέλεση μίας ενέργειας, επαληθεύεται και διαγράφεται από το ημερολόγιο.
  3. Σε περίπτωση κατάρρευσης του ΛΣ, ελέγχεται το ημερολόγιο.

Εικονικά συστήματα αρχείων (Virtual File Systems) (1)

- Στα Windows μπορούν να υπάρχουν διαφορετικά συστήματα αρχείων (NTFS, FAT-16, FAT-32) χωρίς καμία ενοποίηση.
- Στο UNIX όλα τα συστήματα αρχείων (ext2, ext3, ReiserFS) ενοποιούνται σε ένα κατάλογο.
- VFS (Virtual File System - Εικονικό σύστημα αρχείων):
  - Αφαίρεση η οποία περιλαμβάνει όλες τις κοινές λειτουργίες των συστημάτων αρχείων, και τοποθέτηση του κώδικά του σε ένα υψηλότερο επίπεδο.
  - Το VFS καλεί στη συνέχεια τα πραγματικά συστήματα αρχείων από το χαμηλότερο επίπεδο.

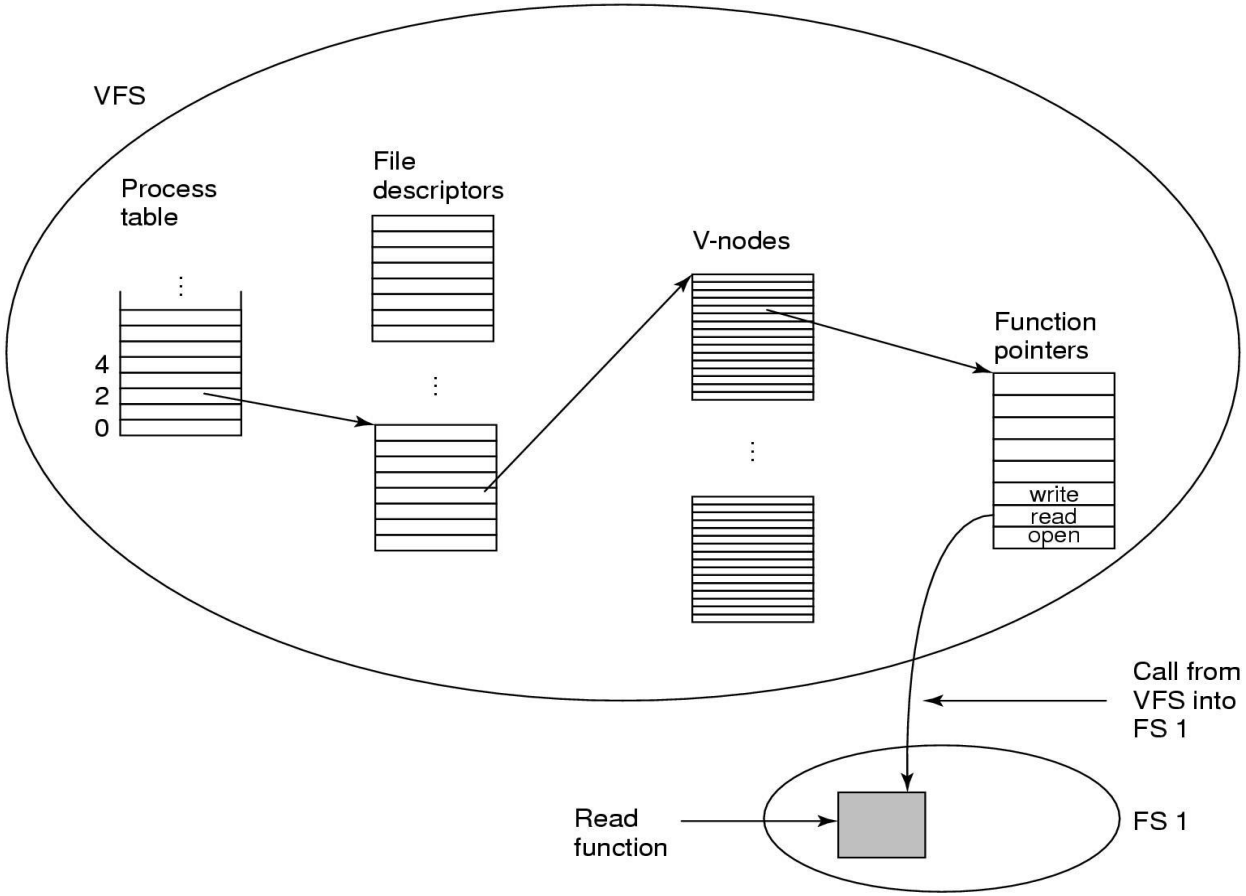
Εικονικά συστήματα αρχείων (Virtual File Systems) (2)



Εικόνα 4-18. Η θέση του Εικονικού Συστήματος Αρχείων.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Εικονικά συστήματα αρχείων (Virtual File Systems) (3)



Εικόνα 4-19. Απλουστευμένη μορφή των δομών δεδομένων που χρησιμοποιεί το VFS και το πραγματικό σύστημα αρχείων για μια λειτουργία ανάγνωσης read.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

4.4 Διαχείριση – βελτιστοποίηση συστήματος αρχείων

Μέγεθος μπλοκ (1)

- Τα αρχεία αποθηκεύονται σε μπλοκ σταθερού μεγέθους.
- Το μέγεθος του μπλοκ επηρεάζει την απόδοση του συστήματος αρχείων με δύο τρόπους:
  - Χωρικά (space): όσο μεγαλύτερο είναι το μπλοκ, τόσο μεγαλύτερη σπατάλη χώρου στο δίσκο.
    - Π.χ. αν μέγεθος μπλοκ = 64 KB, ένα αρχείο του 1KB θα καταλαμβάνει 64 KB στο δίσκο.
  - Χρονικά (time): όσο μικρότερο το μέγεθος του μπλοκ, τόσο περισσότερα μπλοκ ανά αρχείο, άρα και μεγαλύτερος χρόνος προσπέλασης.
    - Π.χ. αν μέγεθος μπλοκ = 1 KB, ένα αρχείο των 64KB θα χρειαστεί 64 προσπελάσεις.

Μέγεθος μπλοκ (2)

Length	VU 1984	VU 2005	Web
1	1.79	1.38	6.67
2	1.88	1.53	7.67
4	2.01	1.65	8.33
8	2.31	1.80	11.30
16	3.32	2.15	11.46
32	5.13	3.15	12.33
64	8.71	4.98	26.10
128	14.73	8.03	28.49
256	23.09	13.29	32.10
512	34.44	20.62	39.94
1 KB	48.05	30.91	47.82
2 KB	60.87	46.09	59.44
4 KB	75.31	59.13	70.64
8 KB	84.97	69.96	79.69

Length	VU 1984	VU 2005	Web
16 KB	92.53	78.92	86.79
32 KB	97.21	85.87	91.65
64 KB	99.18	90.84	94.80
128 KB	99.84	93.73	96.93
256 KB	99.96	96.12	98.48
512 KB	100.00	97.73	98.99
1 MB	100.00	98.87	99.62
2 MB	100.00	99.44	99.80
4 MB	100.00	99.71	99.87
8 MB	100.00	99.86	99.94
16 MB	100.00	99.94	99.97
32 MB	100.00	99.97	99.99
64 MB	100.00	99.99	99.99
128 MB	100.00	99.99	100.00

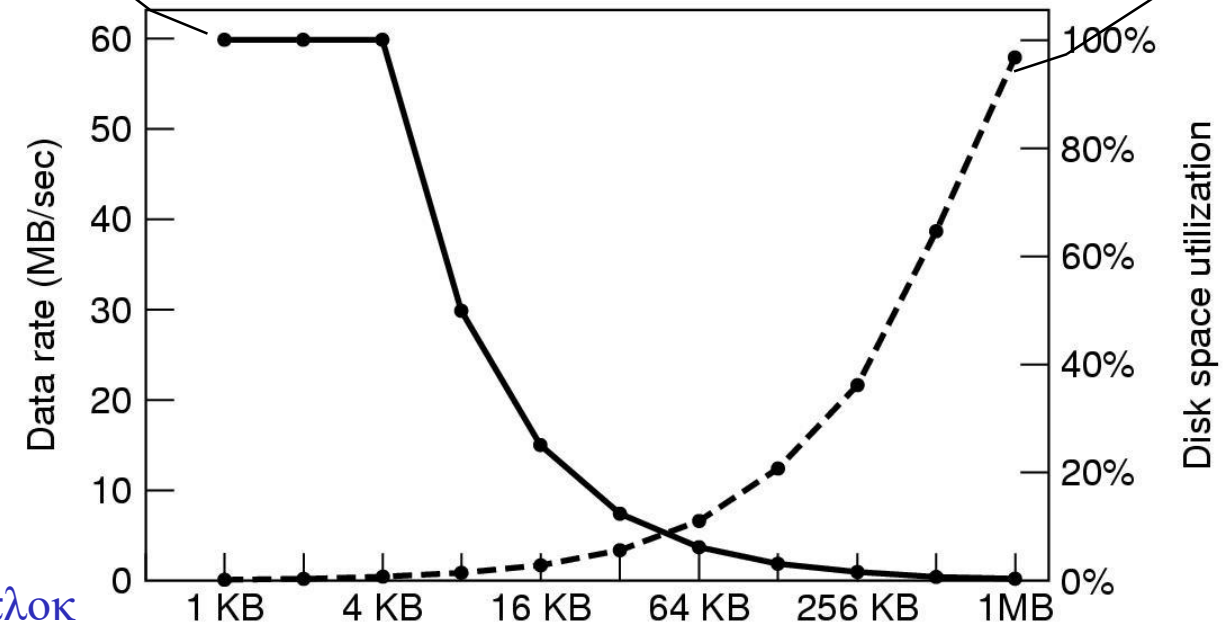
Εικόνα 4-20. Ποσοστό των αρχείων που είναι μικρότερα από ένα δεδομένο μέγεθος (σε byte).

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Μέγεθος μπλοκ (2)

Με block size = 4KB και μέγεθος αρχείων 1, 2 ή 4 KB, η αξιοποίηση του δίσκου είναι 100% (γιατί;)

Με block size = 1MB ο ρυθμός μεταφοράς δεδομένων είναι πολύ μεγάλος (60MB/sec) (γιατί;)

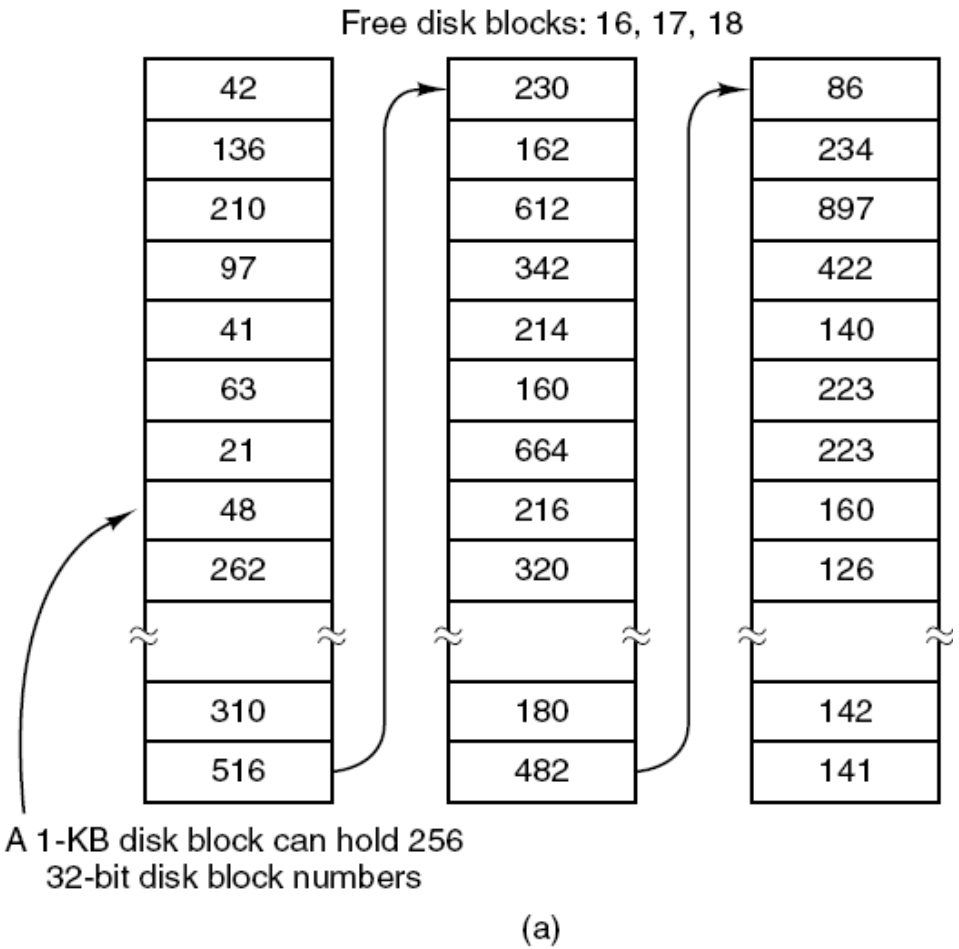


Εικόνα 4-21. Η **διακεκομμένη** καμπύλη (αριστερή κλίμακα) δίνει το ρυθμό μεταφοράς δεδομένων ενός δίσκου.

Η **συνεχής** καμπύλη (δεξιά κλίμακα) δίνει την αποδοτικότητα του χώρου δίσκου. Όλα τα αρχεία έχουν μέγεθος 4 KB.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Παρακολούθηση των ελεύθερων μπλοκ (1)



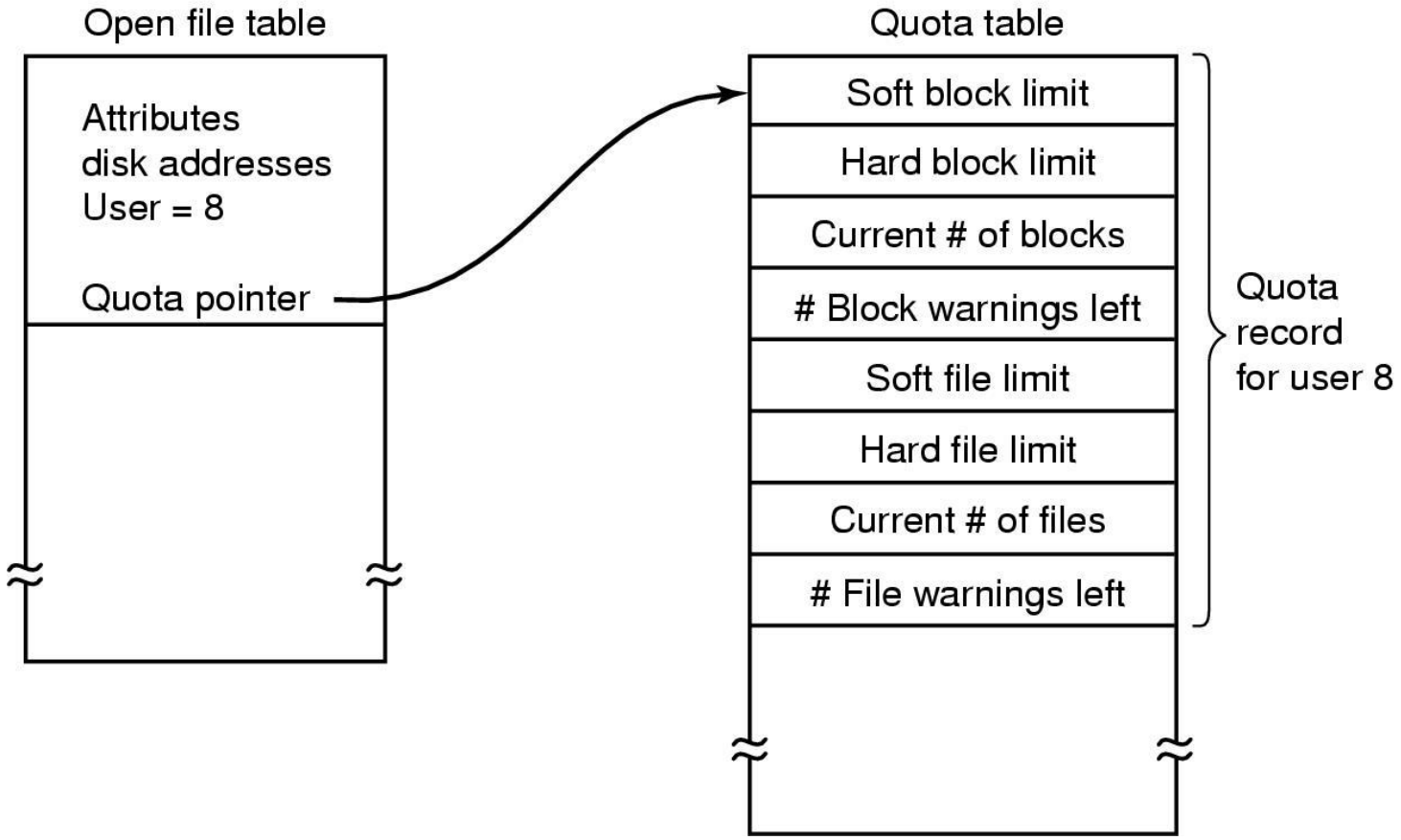
- Κάθε εγγραφή στη συνδεδεμένη λίστα, περιέχει:
  - *n* αριθμούς ελεύθερων μπλοκ δίσκου (όσους χωράει, ανάλογα με το μέγεθος της εγγραφής και του μεγέθους μπλοκ δίσκου).
  - Ένα δείκτη προς την επόμενη εγγραφή.
- Για την αποθήκευση της λίστας των ελεύθερων μπλοκ δίσκου, χρησιμοποιούνται τα ελεύθερα μπλοκ του δίσκου!)

Εικόνα 4-22. (a) Αποθήκευση της λίστας ελεύθερων μπλοκ σε συνδεδεμένη λίστα.

Ποσόστωση χρήσης δίσκου (Disk Quotas) (1)

- Ποσόστωση χρήσης δίσκου: ο διαχειριστής αναθέτει σε κάθε χρήστη ένα μέγιστο πλήθος αρχείων και μπλοκ.
  1. Ο χρήστης ανοίγει ένα αρχείο.
  2. Το ΛΣ βρίσκει τα χαρακτηριστικά του αρχείου και τα τοποθετεί σε έναν ***πίνακα ανοικτών αρχείων*** στη μνήμη.
  3. Κάθε αρχείο «χρεώνεται» στον ιδιοκτήτη του.
  4. Υπάρχει επίσης ένας πίνακας με τις ποσοστώσεις των χρηστών (σε αριθμό αρχείων ή μπλοκ δίσκου) για τον έλεγχο.

Ποσόστωση χρήσης δίσκου (Disk Quotas) (2)



Εικόνα 4-24. Οι ποσοστώσεις χρήσης δίσκου διατηρούνται ανά χρήστη σε έναν πίνακα ποσοστώσεων.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Αντίγραφα ασφαλείας συστήματος αρχείων (backups) (1)

- Με τα αντίγραφα ασφαλείας είναι δυνατή η ανάκαμψη των αρχείων μετά από καταστροφή (ή σφάλμα).

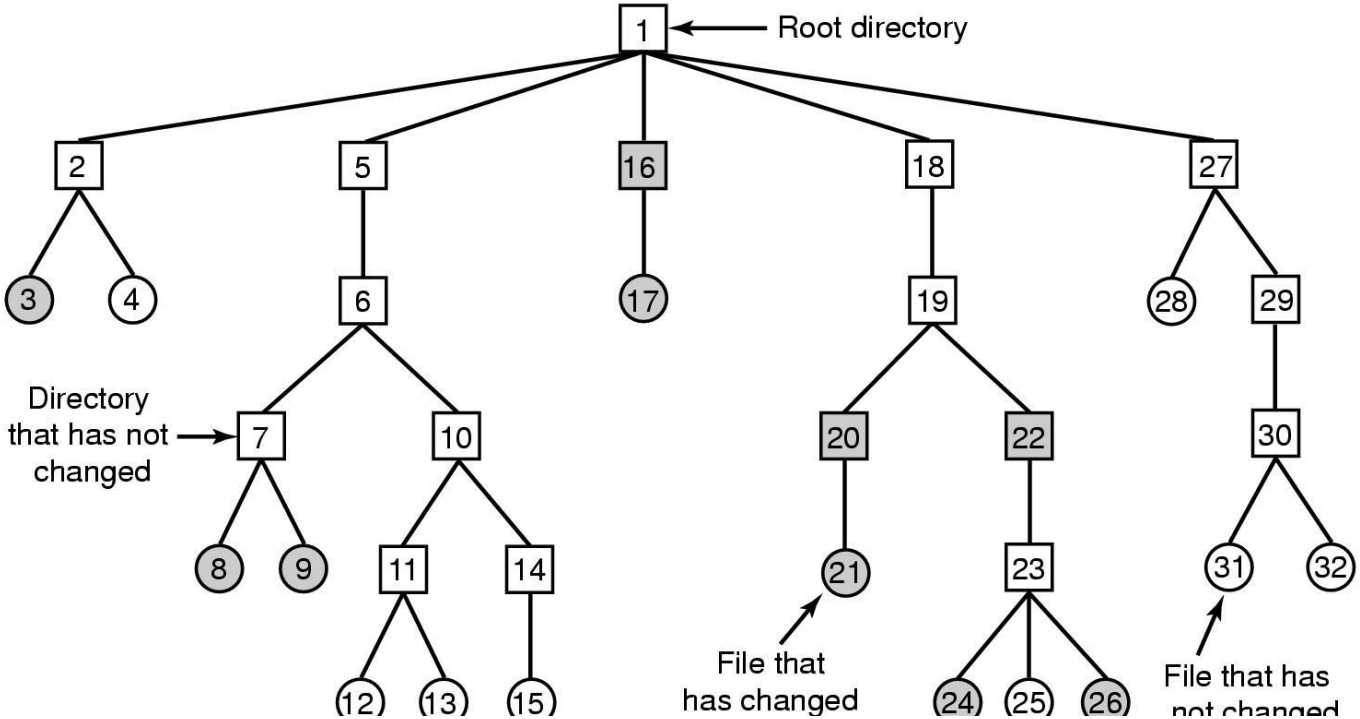
1. Φυσική αντιγραφή (physical dump)
   - Ξεκινά από το μπλοκ 0 μέχρι το τελευταίο μπλοκ του δίσκου.
   - Αν το πρόγραμμα αντιγραφής έχει πρόσβαση στη δομή παρακολούθησης των ελεύθερων μπλοκ, μπορεί να τα αποφύγει.
   - Επίσης θα πρέπει να καταγράψει την αντιστοιχία των αντιγραμμένων μπλοκ στα φυσικά μπλοκ.
   - Δεν μπορεί να γίνει επιλεκτική αντιγραφή αρχείων.

Αντίγραφα ασφαλείας συστήματος αρχείων (backups) (2)

- Με τα αντίγραφα ασφαλείας είναι δυνατή η ανάκαμψη των αρχείων μετά από καταστροφή (ή σφάλμα).

2. Λογική αντιγραφή (logical dump)
   - Εξετάζει ένα ή περισσότερους καταλόγους.
   - Αντιγράφει μόνο τα αρχεία που έχουν τροποποιηθεί.
   - Μπορεί να χρησιμοποιεί την ημερομηνία του τελευταίου πλήρες αντιγράφου.

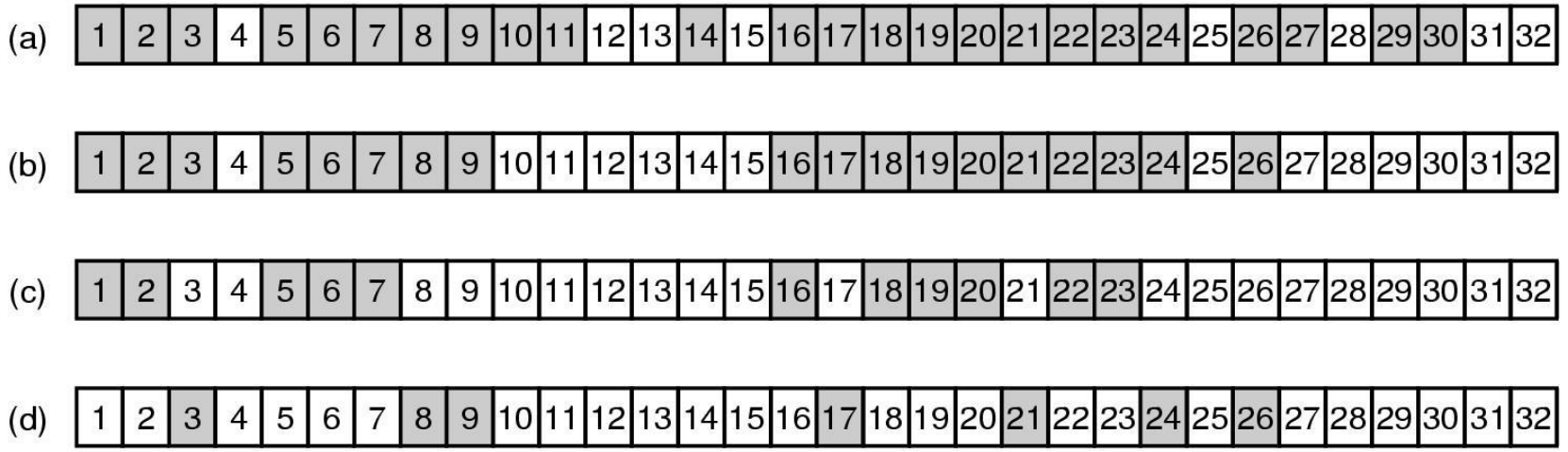
Αντίγραφα ασφαλείας συστήματος αρχείων (backups) (3)



Εικόνα 4-25. Τα τετράγωνα συμβολίζουν τους καταλόγους και οι κύκλοι τα αρχεία. Τα σκιασμένα στοιχεία έχουν τροποποιηθεί μετά από την τελευταία αντιγραφή. Κάθε κατάλογος και αρχείο έχει ως ετικέτα τον αριθμό του αντίστοιχου κόμβου-i.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Αντίγραφα ασφαλείας συστήματος αρχείων (backups) (4)



Εικόνα 4-26. Χάρτες bit για τον αλγόριθμο λογικής αντιγραφής.

(a) Καταγραφή των i-nodes όλων των αρχείων και καταλόγων.

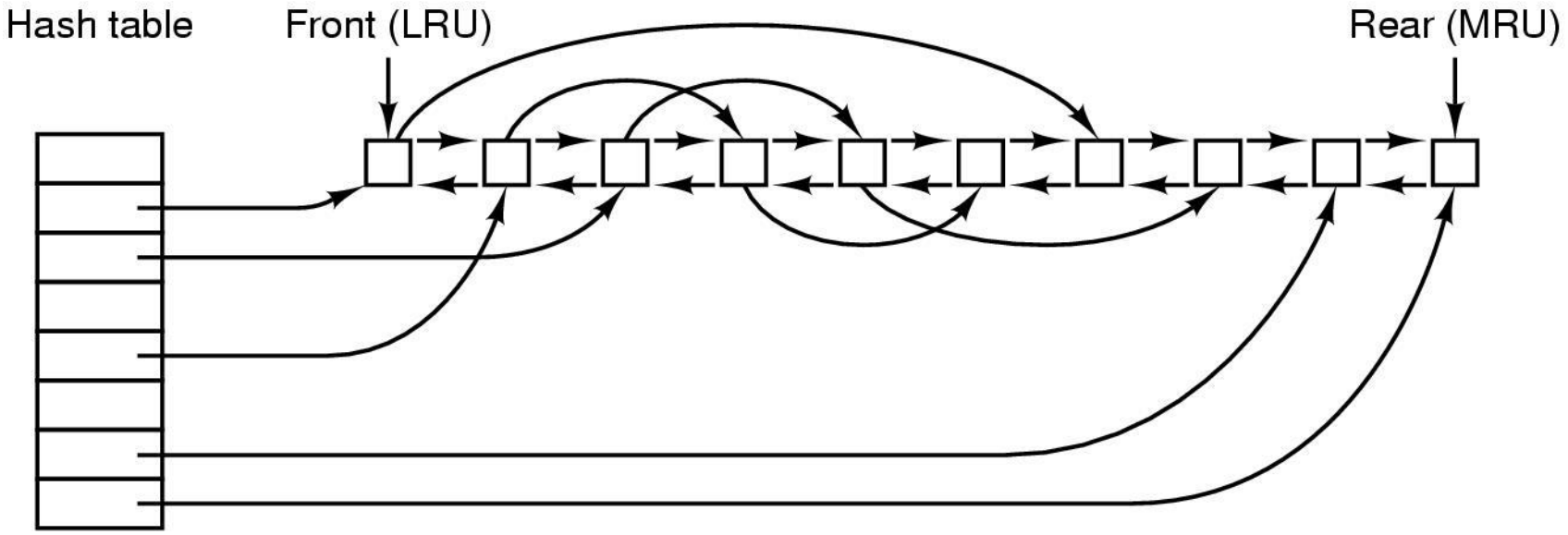
(b) «Σβήσιμο» των καταλόγων που δεν περιέχουν κανένα τροποποιημένο αρχείο ή υποκατάλογο.

(c) Σάρωση όλων των καταλόγων και αντιγραφή των μαρκαρισμένων.

(d) Σάρωση όλων των αρχείων και αντιγραφή των μαρκαρισμένων

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Βελτίωση επίδοσης με κρυφή μνήμη (Caching) (1)



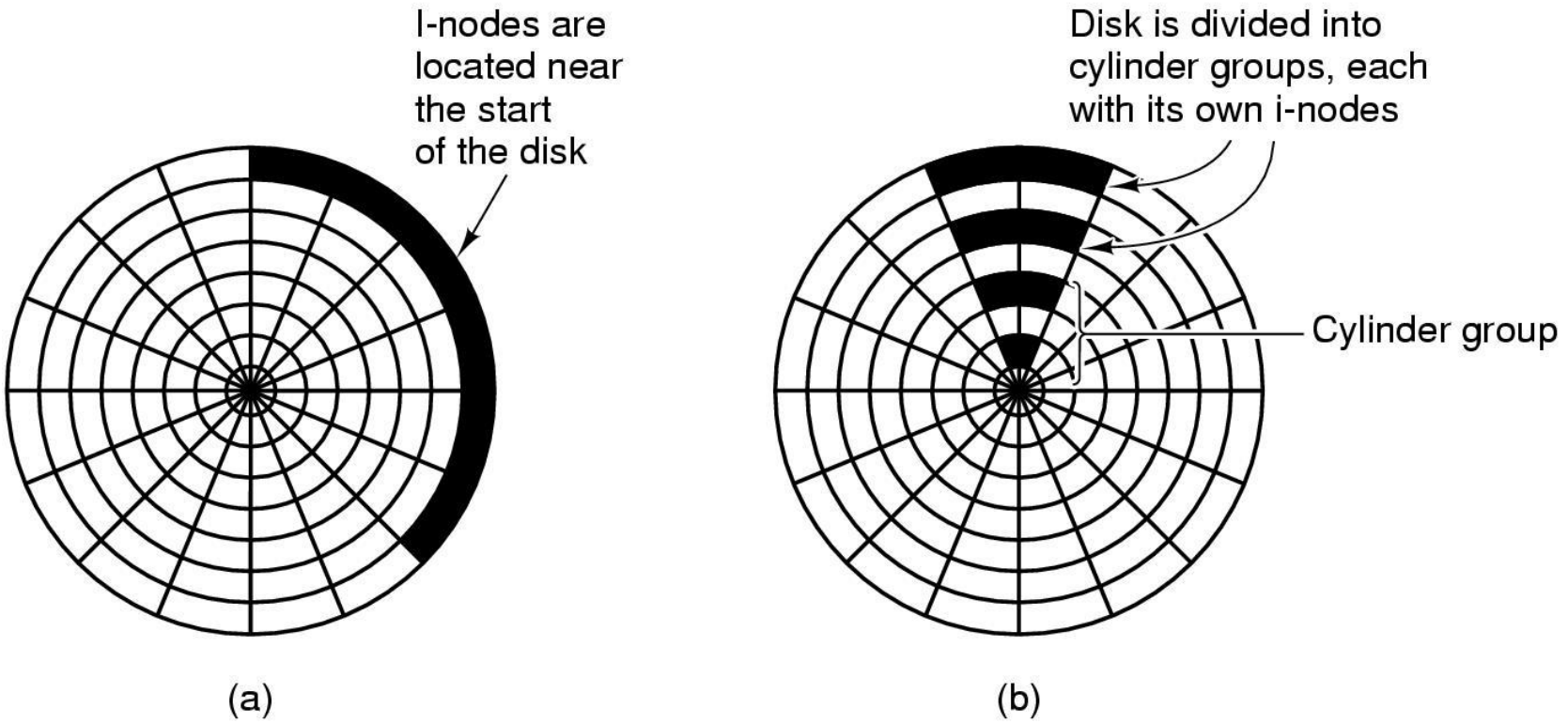
Εικόνα 4-28. Οι δομές δεδομένων της προσωρινής κρυφής μνήμης.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Κρυφή μνήμη (Caching) (2)

- Σε ορισμένα μπλοκ όπως τα μπλοκ των κόμβων-i, σπάνια γίνεται αναφορά δύο φορές σε μικρό χρονικό διάστημα .
- Θεωρούμε μια τροποποιημένη μέθοδο LRU, η οποία λαμβάνει υπόψη της δύο παράγοντες :
  - Είναι πιθανό να χρειαστούμε πάλι το μπλοκ σύντομα;
  - Είναι το μπλοκ απαραίτητο για τη συνέπεια του συστήματος;

Μείωση της κίνησης του βραχίονα του δίσκου



Εικόνα 4-29. (a) Οι κόμβοι-i τοποθετούνται κοντά στην αρχή του δίσκου.

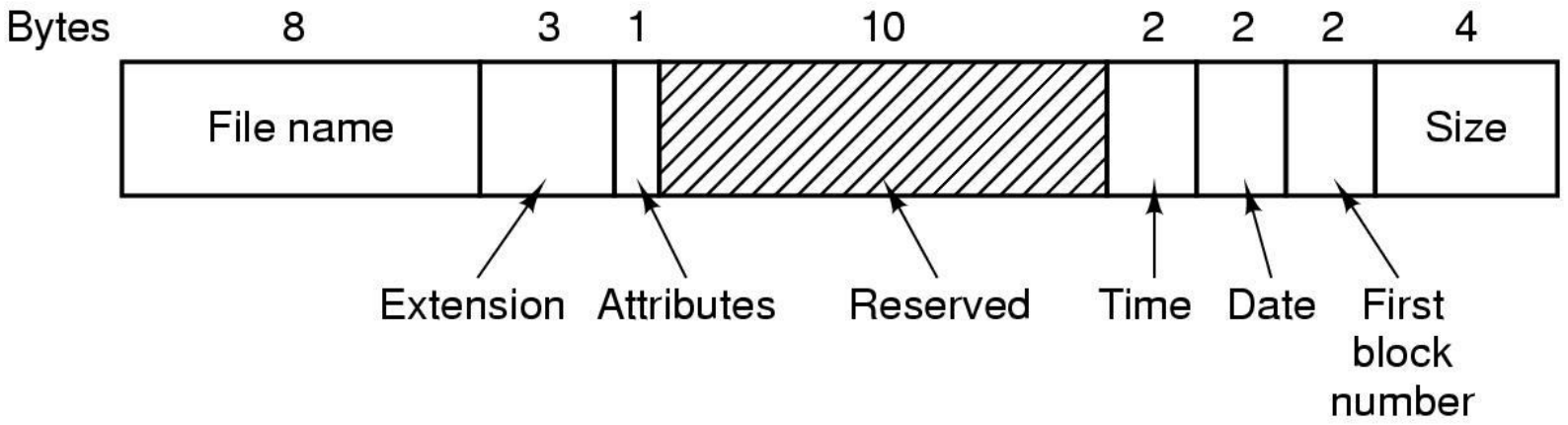
(b) Ο δίσκος διαιρείται σε ομάδες κυλίνδρων, κάθε μία από τις οποίες διαθέτει τα δικά της μπλοκ και τους δικούς της κόμβους i.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

4.5 Παραδείγματα συστημάτων αρχείων

Το σύστημα αρχείων MS-DOS (1)

- Για να διαβαστεί ένα αρχείο εκτελείται μία κλήση συστήματος ***open*** με την οποία αποκτά το χειριστήριο (handle) του αρχείου.
- Η κλήση *open* περιλαμβάνει τη διαδρομή προς τον κατάλογο που βρίσκεται το αρχείο.



Εικόνα 4-31. Μία καταχώρηση καταλόγου στο σύστημα αρχείων MS-DOS.

Το σύστημα αρχείων MS-DOS (2)

Block size	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

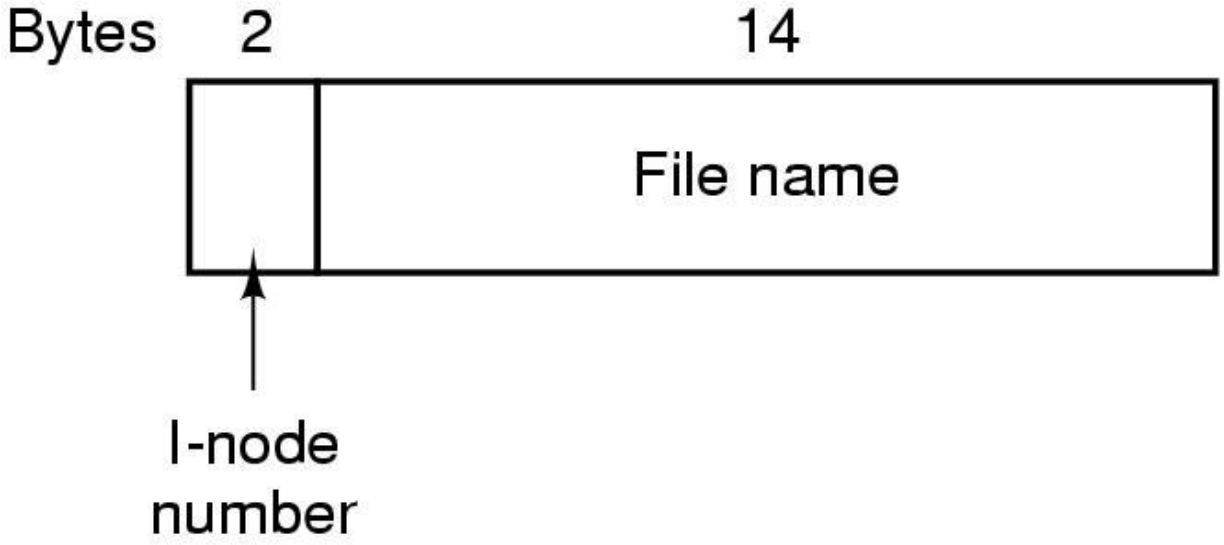
Εικόνα 4-32. Μέγιστο μέγεθος διαμερίσματος (partition) για διαφορετικά μεγέθη μπλοκ.

Τα κενά σημαίνουν μη-δυνατούς συνδυασμούς.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Το σύστημα αρχείων UNIX V7 (1)

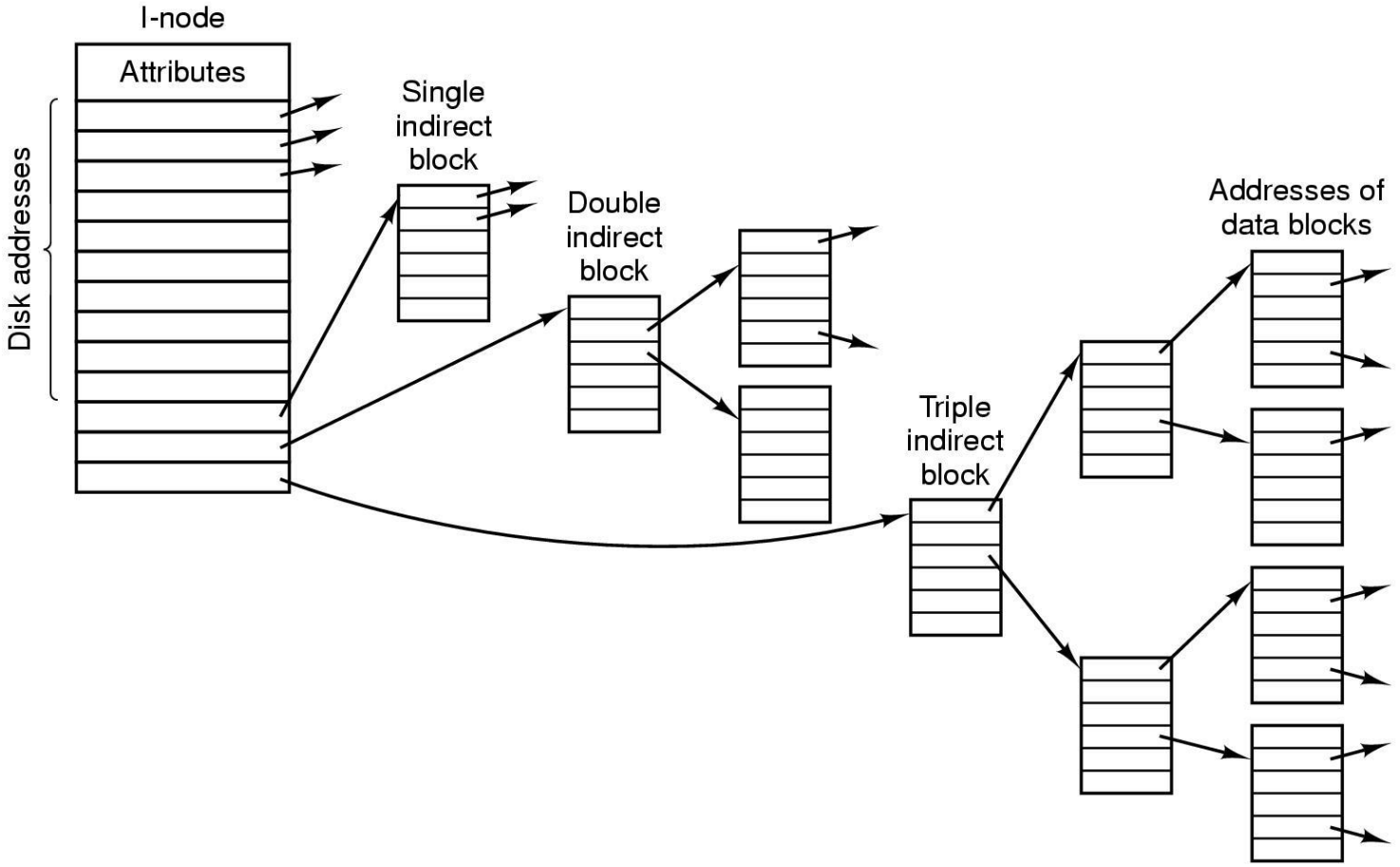
- Στο UNIX V7 περιέχεται μία καταχώρηση καταλόγου, για κάθε αρχείο που βρίσκεται στον συγκεκριμένο κατάλογο.
- 2 byte για τον κόμβο-i και 14 για το όνομα του αρχείου.
- Τα χαρακτηριστικά του αρχείου περιλαμβάνονται στον κόμβο-i.



Εικόνα 4-33. Μία καταχώρηση καταλόγου στο UNIX V7.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

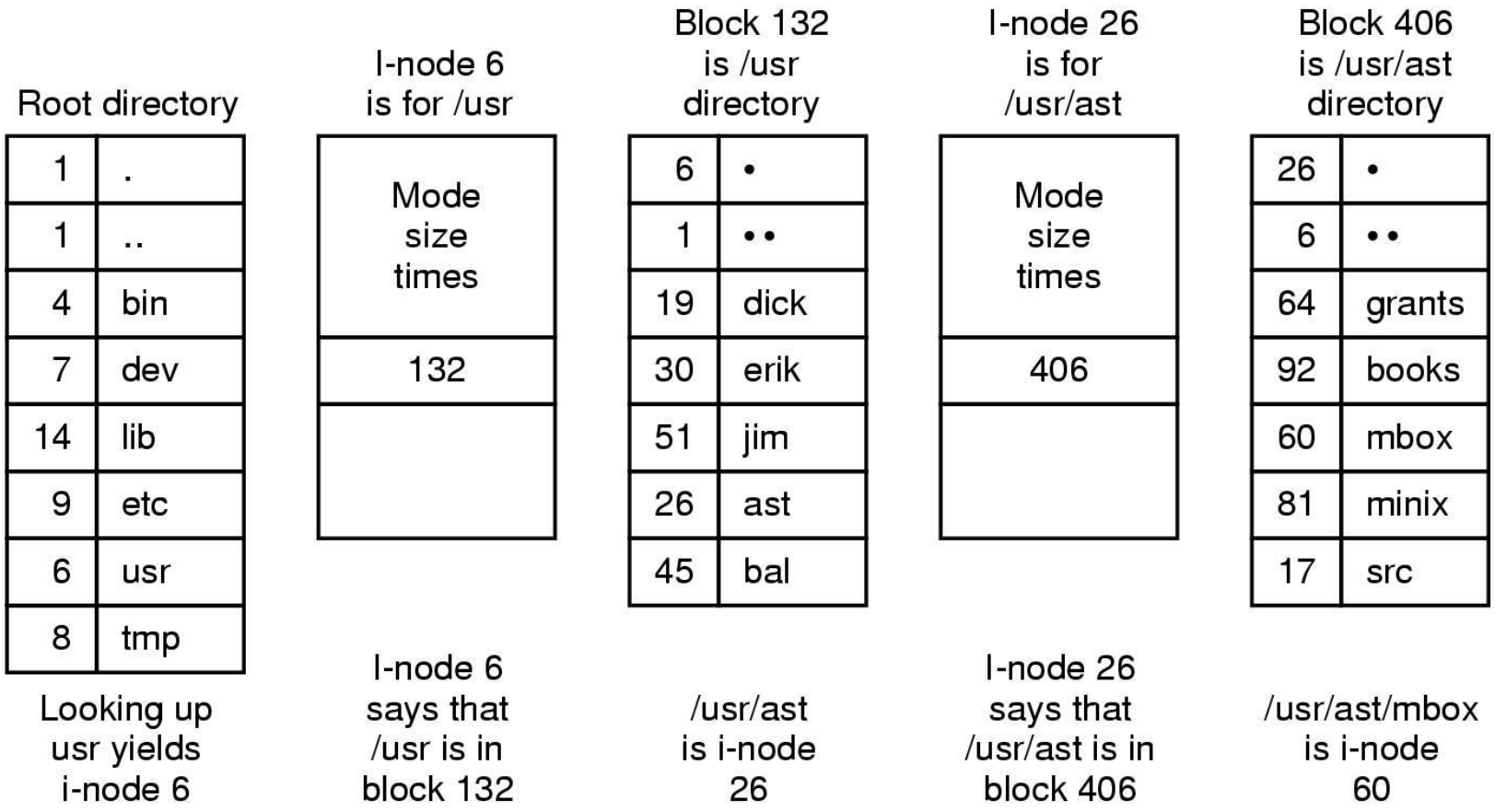
Το σύστημα αρχείων UNIX V7 (2)



Εικόνα 4-34. Ένας κόμβος-i στο UNIX.

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Το σύστημα αρχείων UNIX V7 (3)



Εικόνα 4-35. Τα βήματα για την αναζήτηση του αρχείου ***/usr/ast/mbox.***

Πηγή εικόνας: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc.

Βιβλιογραφικές Πηγές

1. *Σύγχρονα Λειτουργικά Συστήματα* (Modern Operating Systems) A.S.Tanenbaum, Prentice Hall 2008, 3nd Edition Ελληνική Μετάφραση, 2009, Εκδόσεις Κλειδάριθμος.
2. *Λειτουργικά Συστήματα* (Operating Systems), Silberschatz, Galvin, Gagne, Ελληνική μετάφραση, 2009, Εκδόσεις Ιών.
3. Advanced Bash-Scripting Guide, (http://www.tldp.org/LDP/abs/html)