ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΣΥΣΤΗΜΑ ΑΡΧΕΙΩΝ

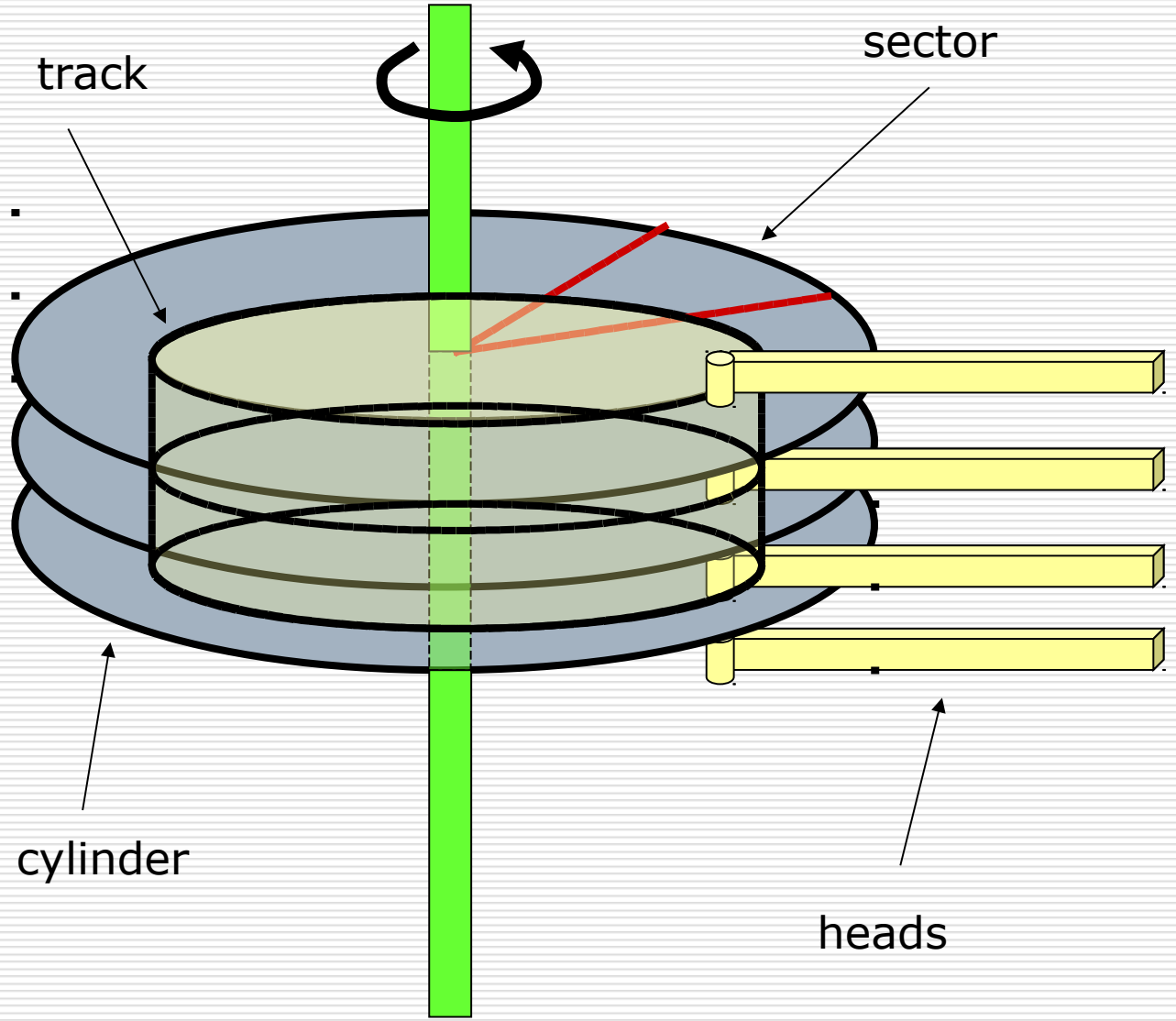
Αποθήκευση δεδομένων

- Μαγνητικοί δίσκοι
  - Σκληρός δίσκος/hard disk
  - Μαλακός δίσκος/floppy disk (δισκέττα)
- Οπτικοί δίσκοι
  - CD-ROM (WORM – Write Once Read Many)
  - CD-RW
- Δίσκοι «στερεάς κατάστασης» (πυριτίου)
  - Flash memory
  - Ramdisk (κύριας μνήμης)
- Network file system
  - File server
  - Network disk

Μαγνητικοί δίσκοι

Tracks = cylinders * heads

Capacity = sector_size *
Tracks * sectors_per_track



Timing parameters

Rotation speed: R = 3000 – 10000 rpm (rot./min)

- Rotation latency:
  $t_r = 1 / 2R$
- Seek time:
  $t_s = 1\text{-}15$ msec
  - Track-to-track time: t_{tt}
  - Average
- Access time:
  $t_a = t_s + t_r$

- Bandwidth:
  $B = BytesPerTrack / (1/R + t_{tt})$

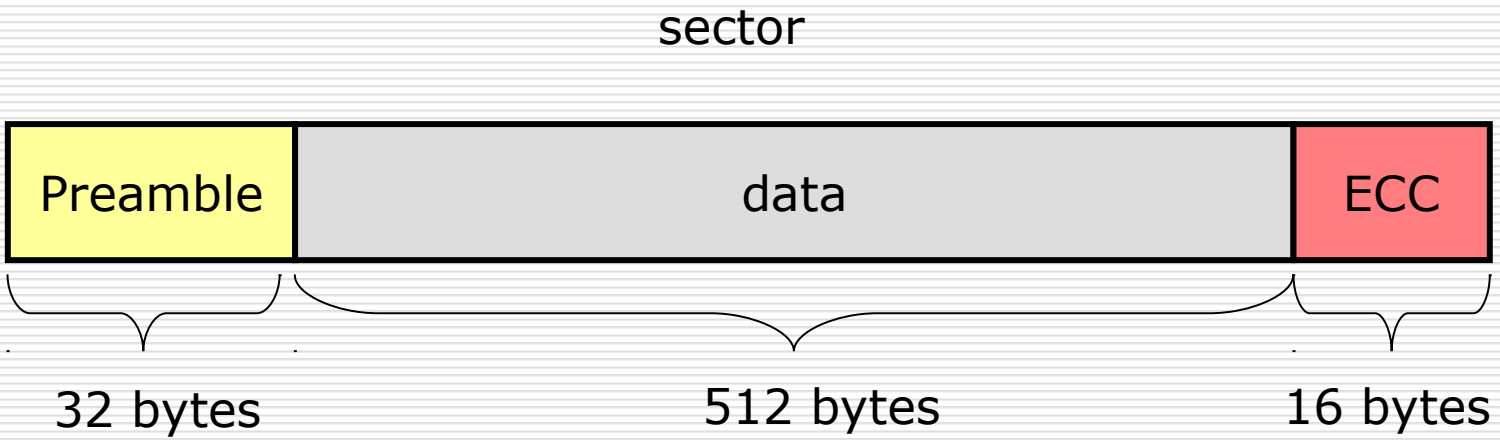
Παράδειγμα

Παράμετρος	**Western Digital WD 18300**
Cylinders	10601
Tracks/cylinder	12
Sectors/track (μέση τιμή)	281
Sectors (συνολικά)	35,742,000
Μέγεθος sector	512 byte
Χωρητικότητα	18.3 GB
Seek time (track-to-track)	0.8 msec
Seek time (average)	6.9 msec
Rotation time	8.33 msec (7,200 rpm)
Transfer rate (max)	32 MByte/sec
Χρόνος μεταφοράς 1 sector	17 μsec

Formatting

- Low-level / high level
  - Low-level: Διαμόρφωση και αρίθμηση sectors πάνω στο μαγνητικό υλικό
  - High-level: Διαμόρφωση partitions
- Sector formatting
- Cylinder skew
- Interleaving

Sector formatting



- Preamble: bit-pattern για να μαρκάρεται η αρχή του sector
- ECC: Error-Correcting Code

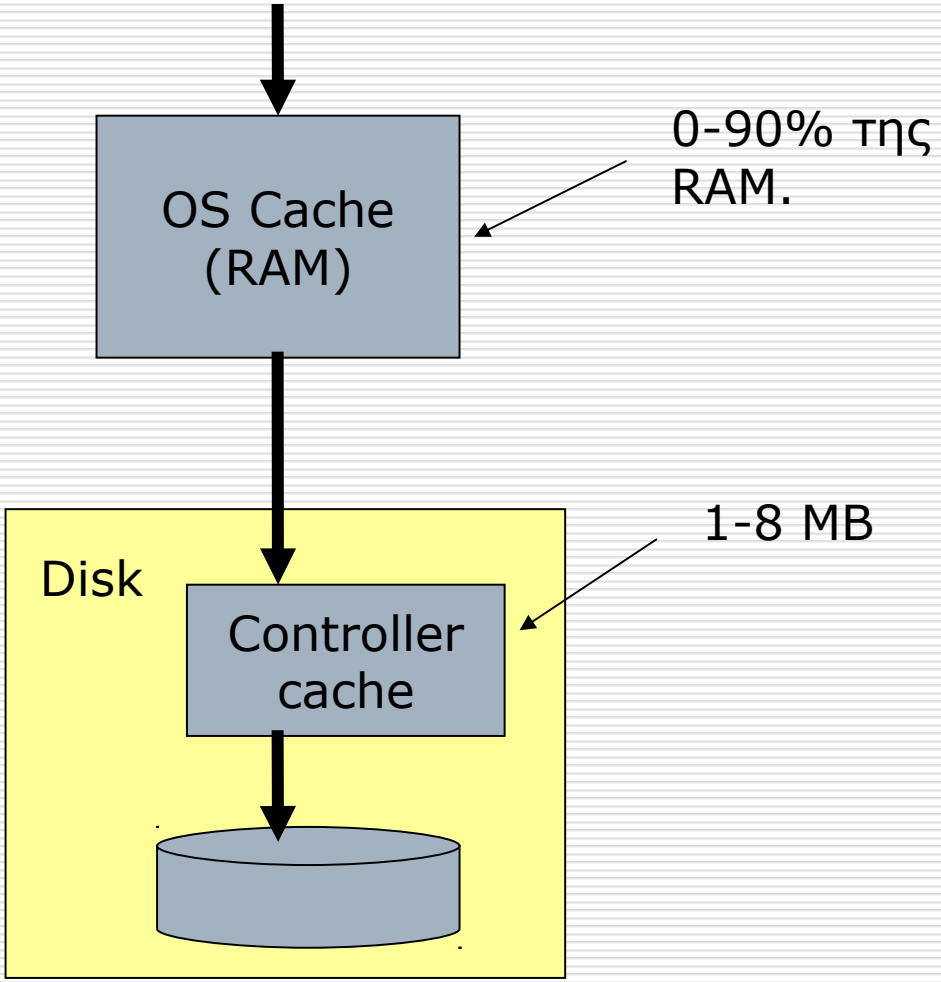
Sector numbering

- Disk geometry
  - (cylinders, heads, sectors)
  - Zones
  - LBA (Logical Block Addressing)

- Cylinder skew
  - Το «sector 0» κάθε track μετατοπίζεται κατά μια γωνία θ σε διαδοχικά tracks.
  - Εξασφαλίζει ταχύτερη ανάγνωση διαδοχικών tracks
- Interleaving
  - Για ομαλή ροή μεταφοράς.

Caching/prefetching

- Caching: γίνεται σε δύο μέρη
  - Disk controller (από το firmware του δίσκου)
  - Στη RAM (από το ΛΣ)
- OS Cache
  - Προηγούμενες προσπελάσεις, που μπορεί να ξαναζητηθούν (temporal locality)
- Controller cache
  - Prefetching: όταν διαβάζεται ένας sector, προ-διαβάζονται οι γειτονικοί του.
  - Spatial locality

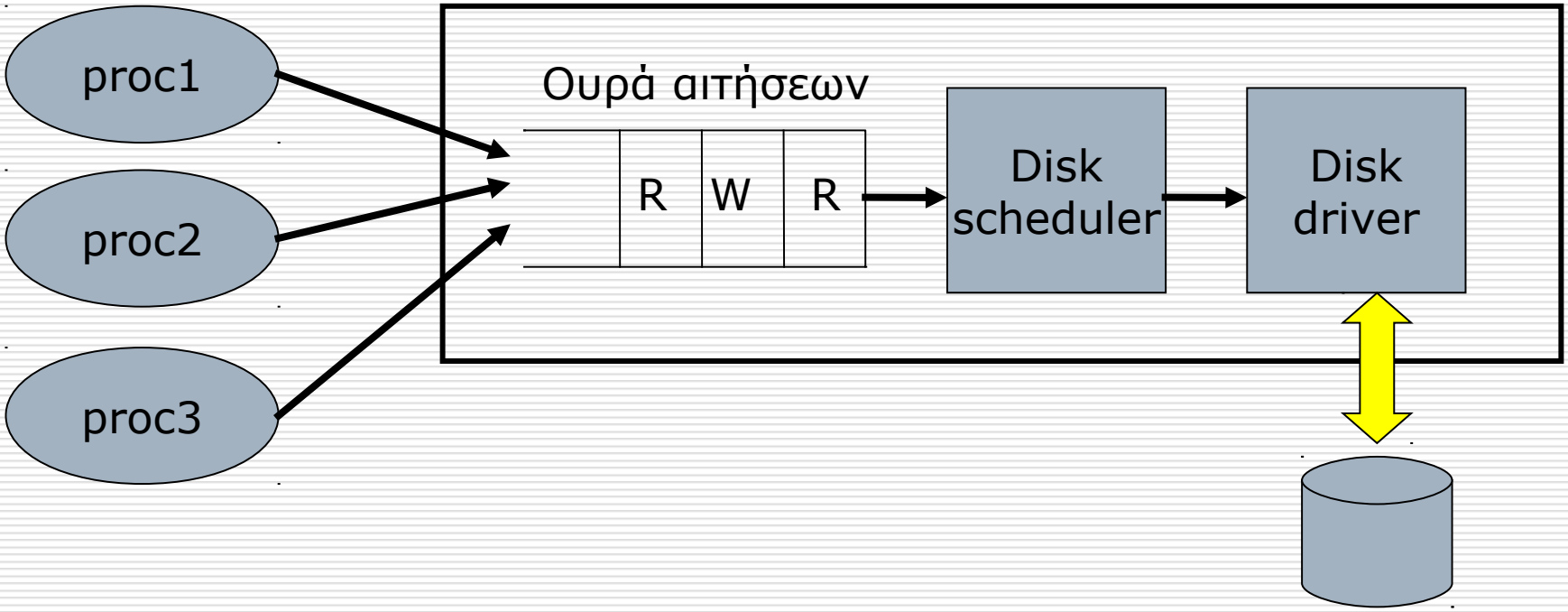


Scheduling

- Διεργασίες -> αιτήσεις I/O
- Οδήγηση δίσκου

```
read_sector(sid, buffer)
write_sector(sid, buffer)
```

- Scheduling = επιλογή επόμενης αίτησης



Αλγόριθμοι

- First Come First Served (FCFS)
  - Πλεονέκτημα: δίκαιη εξυπηρέτηση.
  - Μειονέκτημα: κακή απόδοση.
- Shortest Seek First (SSF)
  - Άπληστη λύση.
  - Πλεονέκτημα: πολύ καλή απόδοση.
  - Μειονέκτημα: άδικη συμπεριφορά προς κάποιες διεργασίες.
- Elevator
  - Δίκαιος και με καλή απόδοση.

Αλγόριθμος ασανσέρ (elevator)

- Πρόβλημα
  Σε ένα ψηλό κτίριο, ένας αριθμός από ορόφους έχει καλέσει το ασανσέρ. Ποιος θα εξυπηρετηθεί μετά?
- Αλγόριθμος: το ασανσέρ θα συνεχίσει να κινείται προς την ίδια διεύθυνση, όσο το δυνατόν περισσότερο.

- Πρόβλημα:
  Σε ένα δίσκο, ένας αριθμός από αιτήσεις απαιτούν να κινηθεί η κεφαλή του δίσκου σε διάφορα tracks. Ποια αίτηση θα εξυπηρετηθεί μετά?
- Αλγόριθμος: η κεφαλή θα συνεχίσει να κινείται προς την ίδια διεύθυνση όσο το δυνατόν περισσότερο.

RAID

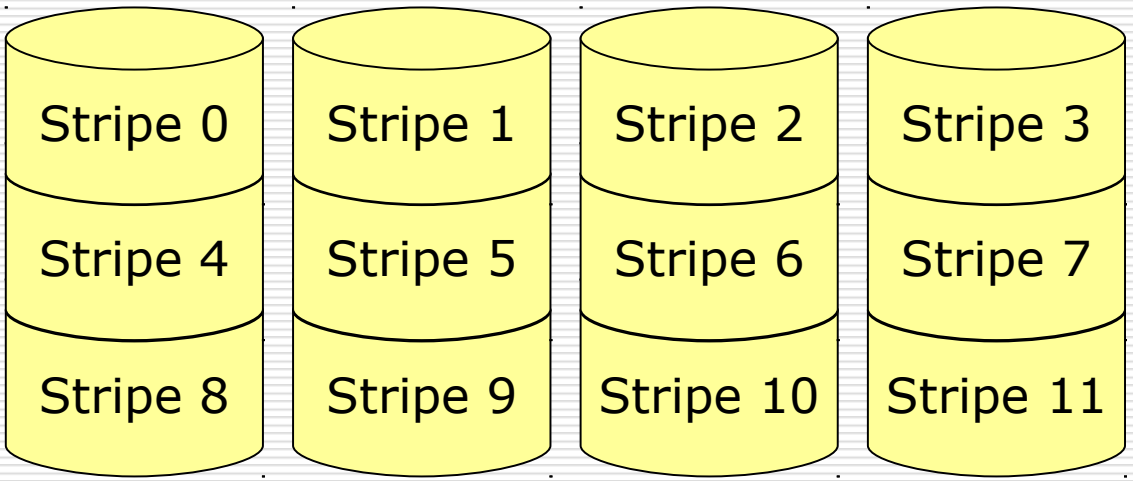
- Redundant Array of Inexpensive Disks
- Patterson et al. 1988
- Ιδέα: πολλοί μικροί δίσκοι δουλεύουν παράλληλα
  - Αυξημένη ταχύτητα
  - ?Αυξημένη αξιοπιστία?

- Αξιοπιστία:
  - MTBF: Mean Time Between Failures
  - Ενδεικτικά για δίσκους: 10,000-50,000 ώρες
  - Για M δίσκους, MTBF(M) = MTBF(1) / M !!!

RAID level 0: Striping

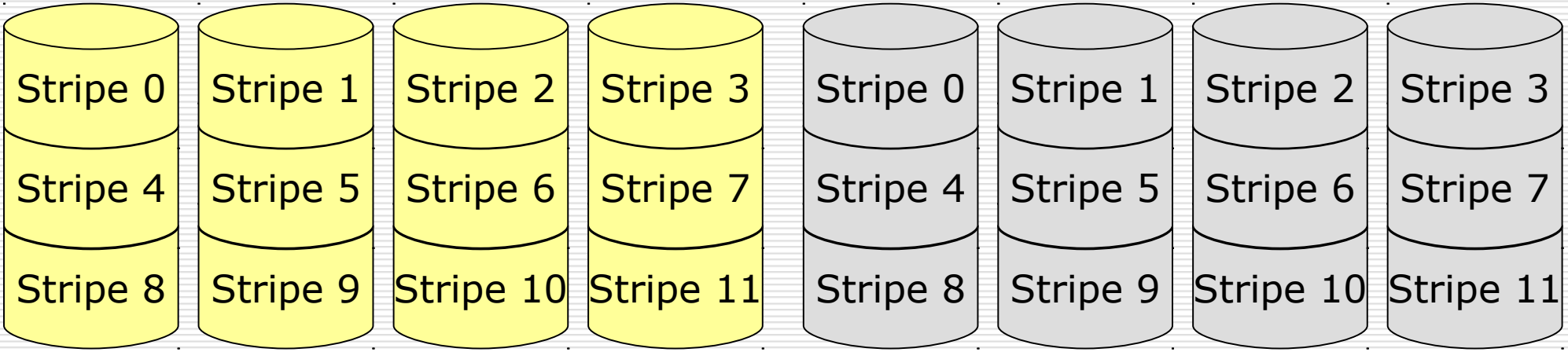
- 1 stripe = k sectors (k>=1)
- Stripe i = sectors i*k … i*k + (k-1)
- Απόδοση: καλή (για μεγάλες προσπελάσεις, εξαιρετική)
- Αξιοπιστία: κακή

4 δίσκοι



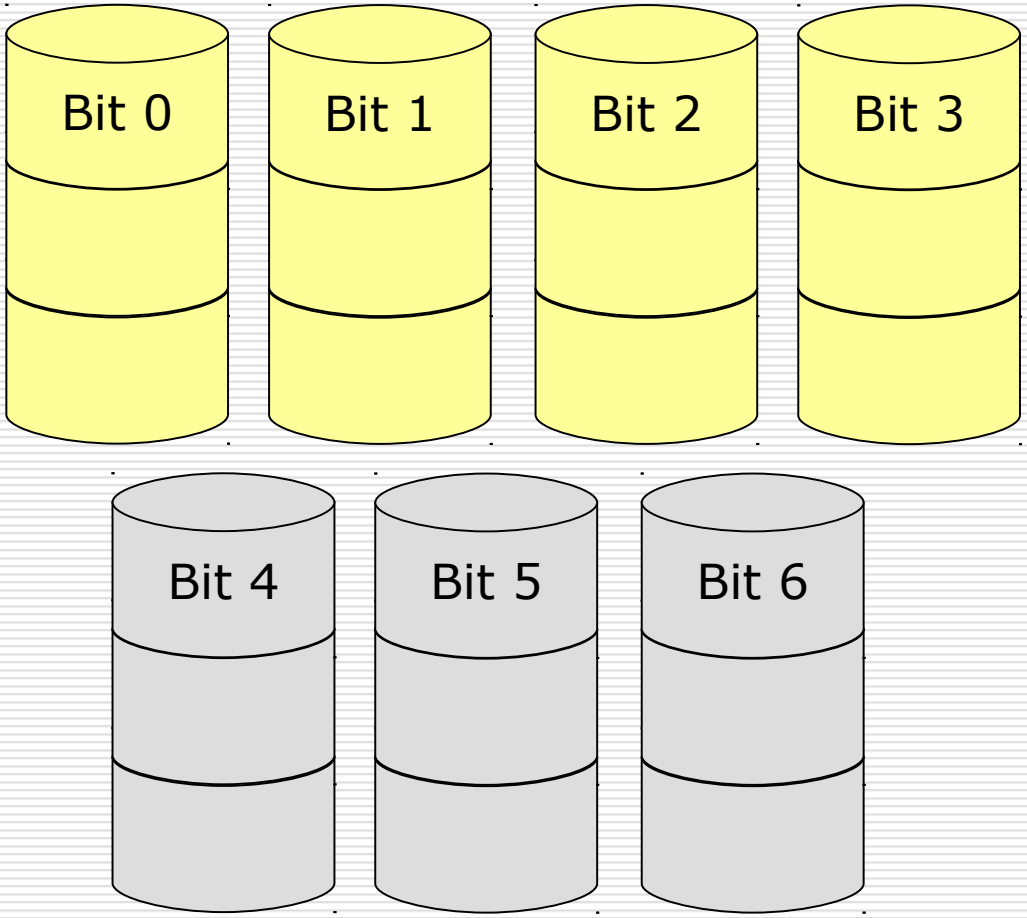
RAID level 1: Mirroring

- Mirroring: διπλάσιοι δίσκοι, ο κάθε δίσκος έχει αντίγραφο.
- Απόδοση: όπως για RAID-0
- Αξιοπιστία: άριστη



RAID level 2

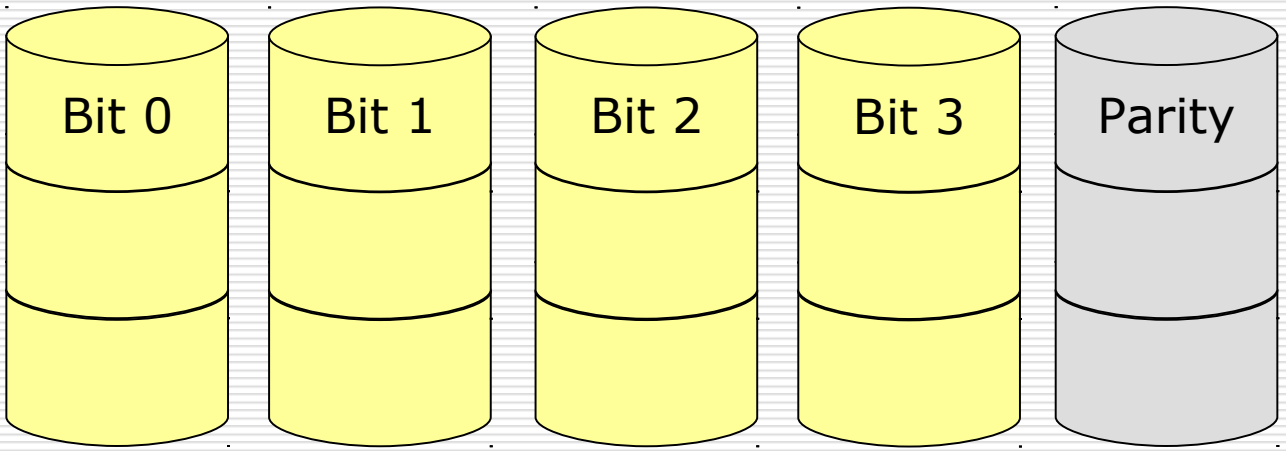
- Nibble-level splitting
- Hamming coding:
  Hamming distance = 3



0000	000
0001	011
0010	101
0011	110
0100	110
0101	101
0110	011
0111	000
1000	111
1001	100
1010	010
1011	001
1100	001
1101	010
1110	100
1111	111

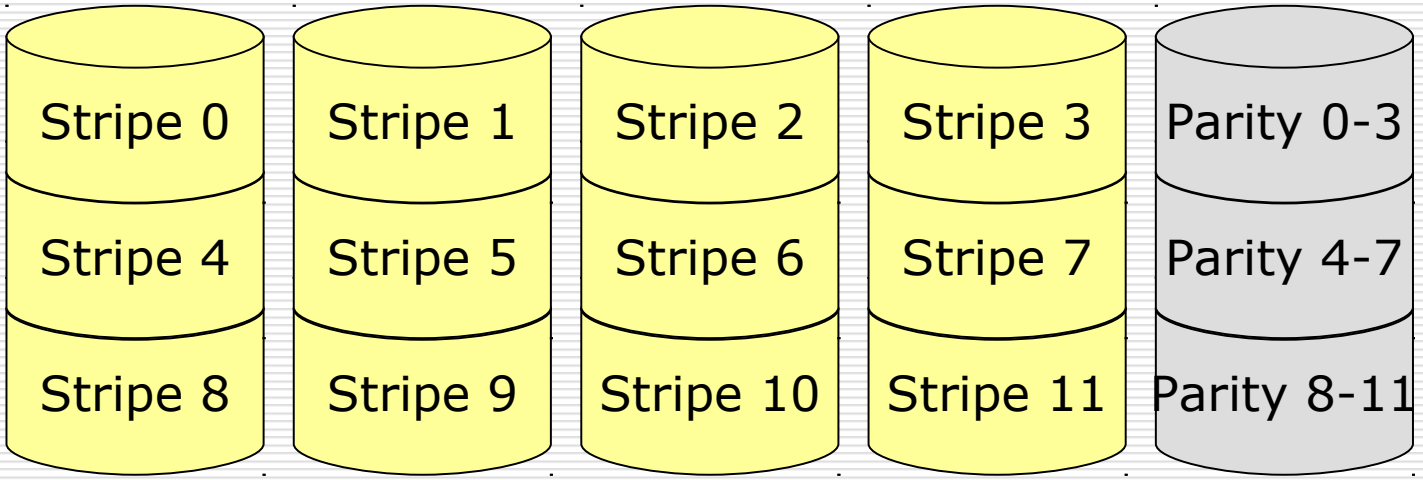
RAID level 3

- Nibble-level splitting
- Parity



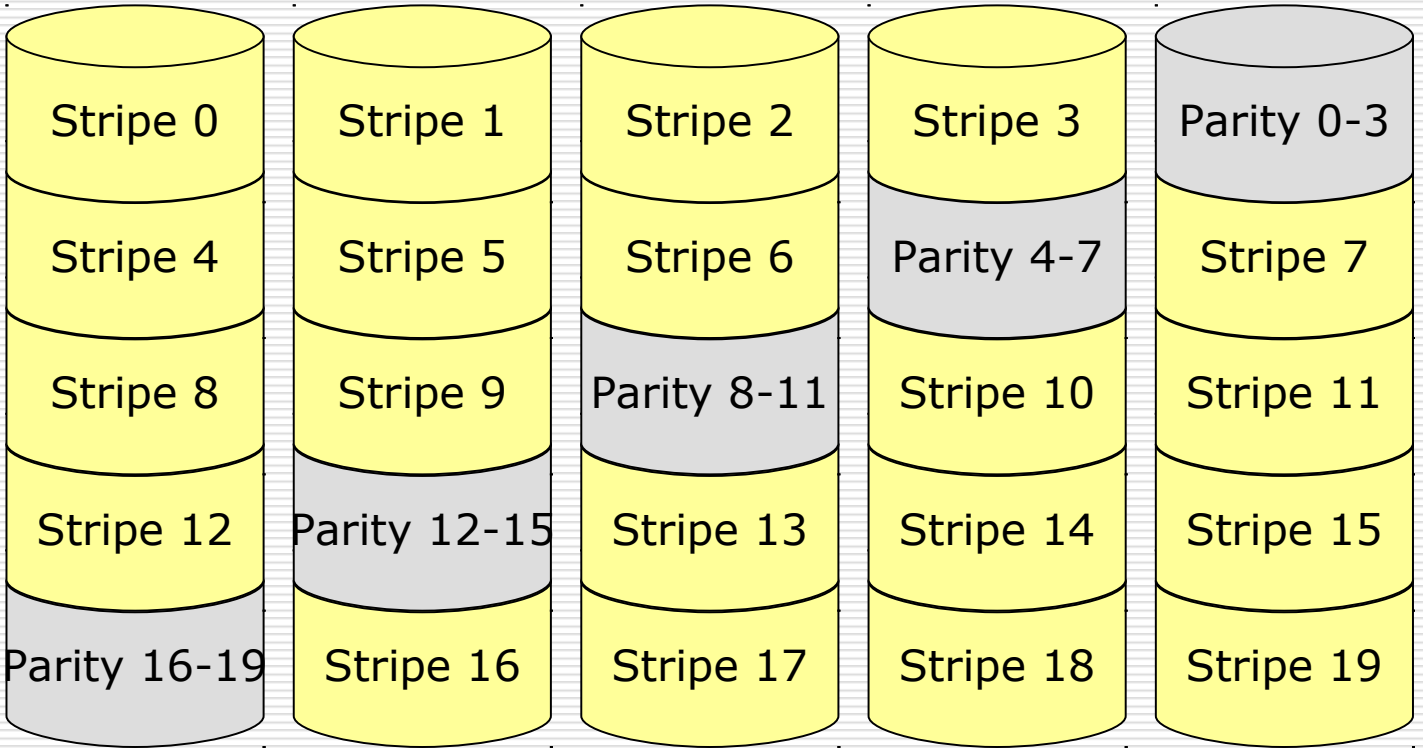
RAID level 4

- Striping with parity
- Parity disk is a bottleneck!



RAID level 5

- No bottleneck!



Βασική οργάνωση δίσκων

- Sector addressing: linear
- Partitions
  - Σταθερού μεγέθους
  - Σταθερής θέσης
  - Περιορίζονται σε ένα δίσκο
- Volumes
  - Μεταβλητού μεγέθους
  - Μεταβλητής θέσης
  - Εκτείνονται σε πιθανόν περισσότερους δίσκους
- Ένα filesystem ανά partition/volume.

Booting (Wintel hardware)

- Master Boot Record = Sector 0 του δίσκου.
- Περιέχει
  1. Κώδικα για boot
  2. Partition table
- Εκτελείται κατά την εκκίνηση
  - Εντοπίζει το active partition
  - Φορτώνει και εκτελεί το block 0 (boot block).

Blocks

- Sectors = η μικρότερη μονάδα μνήμης του δίσκου.
- Blocks = η μικρότερη μονάδα μνήμης των αρχείων ενός filesystem.
- Block size = k * Sector size (επιθυμητό: $k=2^m$)
- Μικρά blocks
  - Καλύτερη αξιοποίηση χώρου (internal fragmentation)
  - Μεσαίο μέγεθος αρχείων ¼ 2 KB !!!
- Μεγάλα blocks
  - Καλύτερος ρυθμός μεταφοράς σε random accesses.

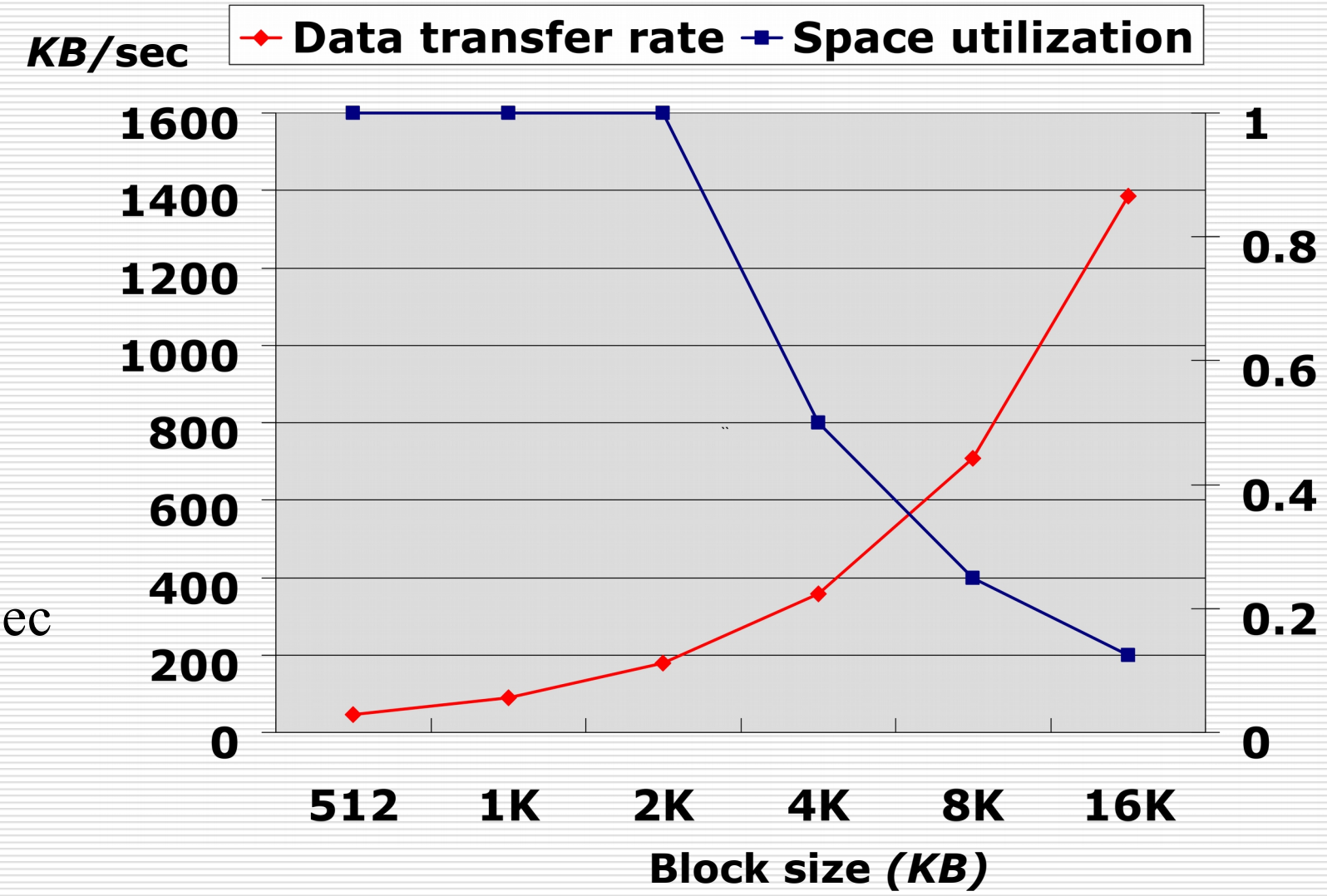
$$\text{transfer rate} = \frac{\text{block size}}{\text{access time} + \text{transfer time}}$$

Block size tradeoff

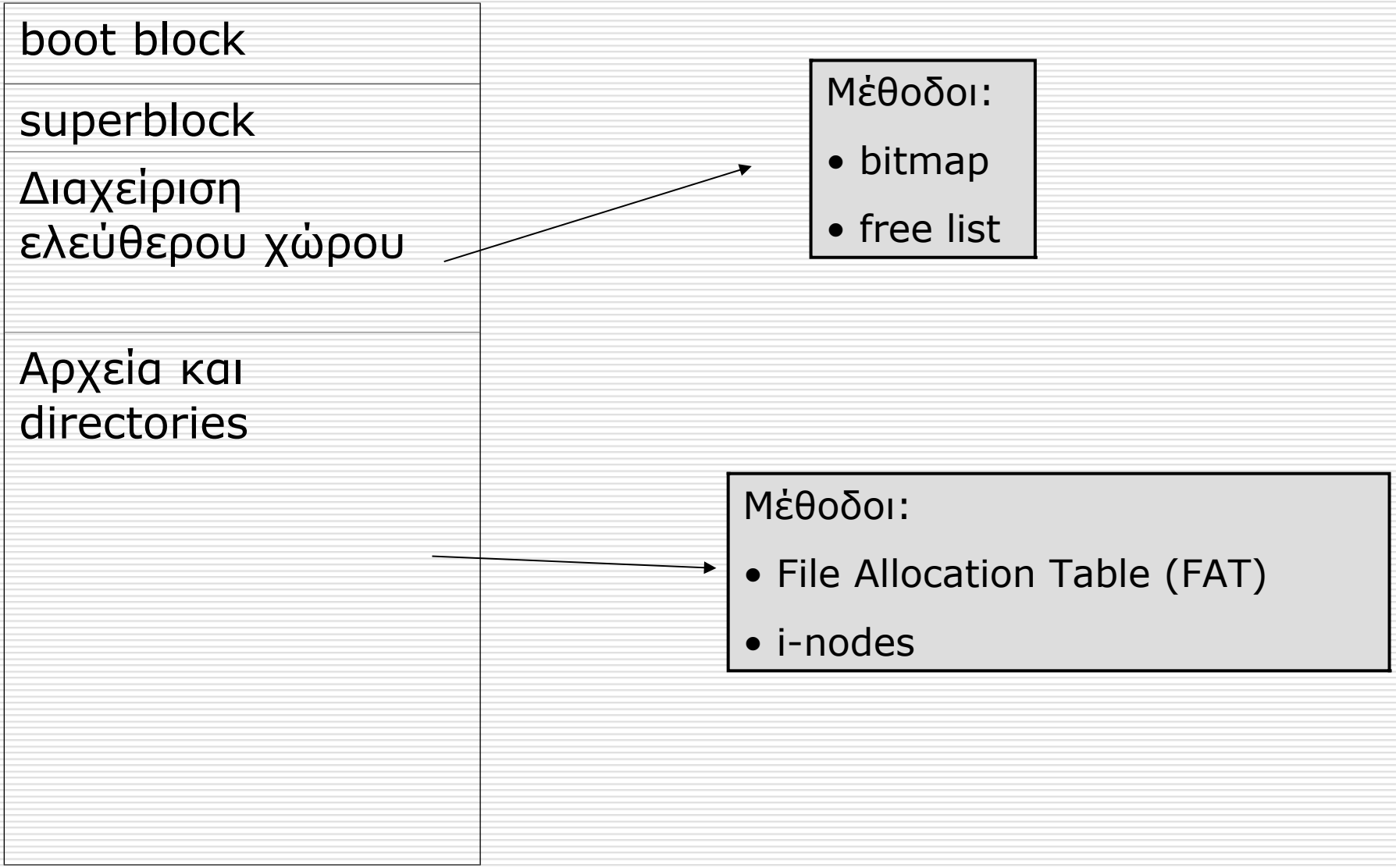
$$\frac{b}{t_s + b/B}$$

$t_s = 11 m\sec$

$B = 16\, MB/\sec$

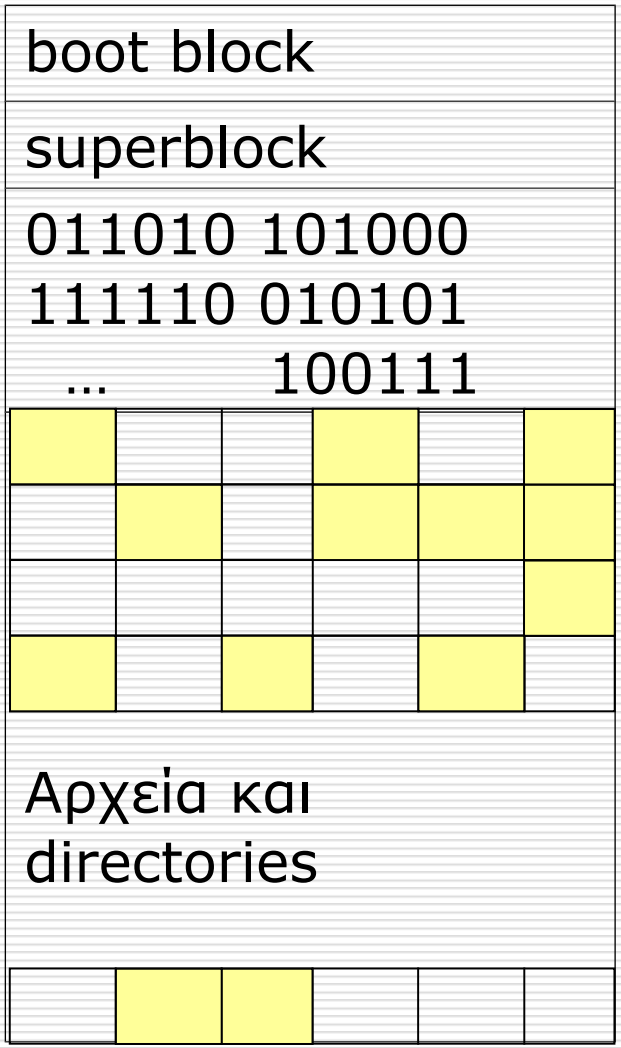


Μορφοποίηση filesystem

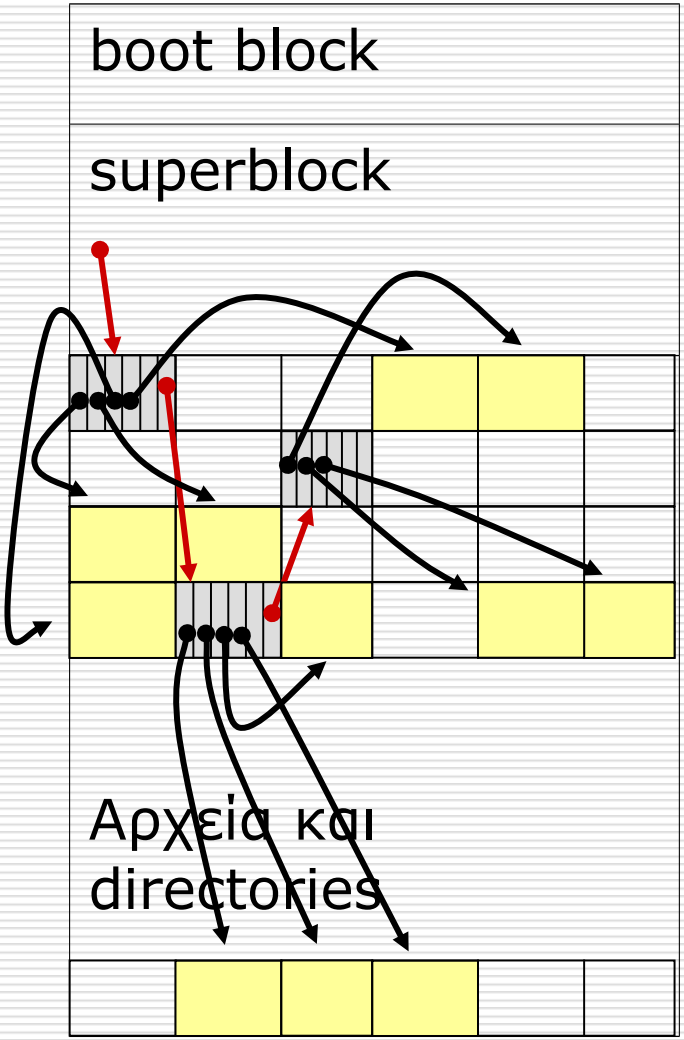


Διαχείριση ελεύθερου χώρου

bitmap



freelist

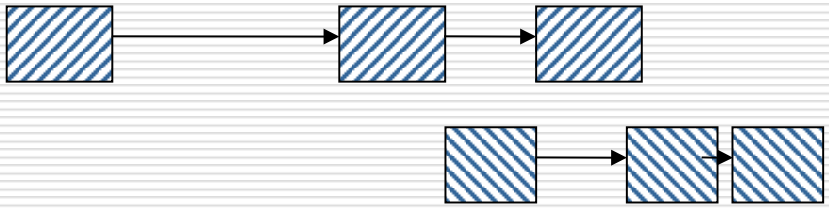


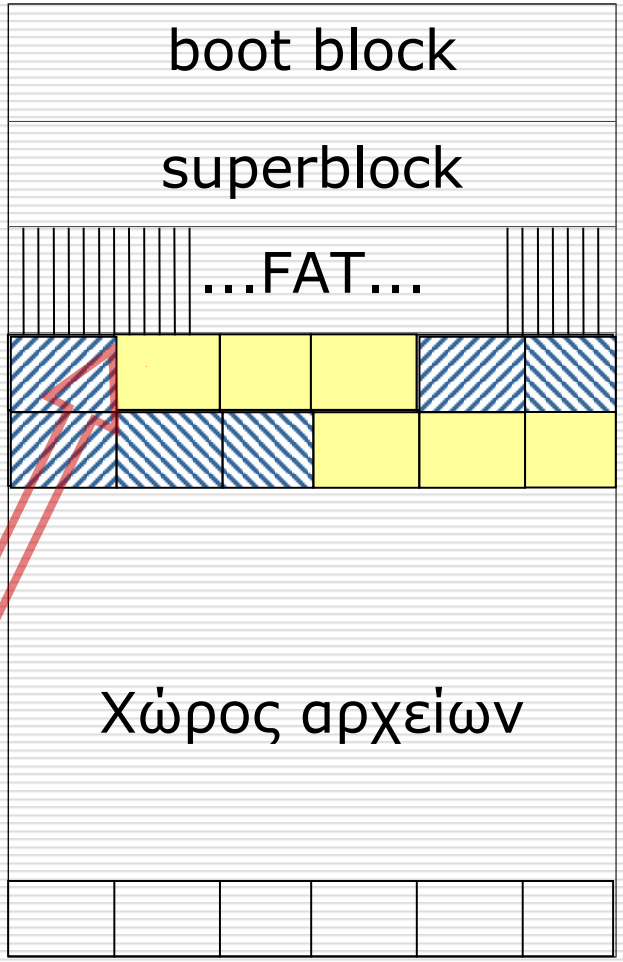
Αναπαράσταση αρχείων και directories

- Αρχείο = metadata + data
  - Metadata:
    - User and group ownership
    - Permissions
    - Creation, update, last access times
    - Flags (hidden, archive, …)
    - Type (Text/Binary/PDF/C/…)
    - Άλλα (πχ. σχετικά με GUI κλπ)
  - Data = ακολουθία από blocks
- Directory: ένα αρχείο
  - Metadata: ίσως διαφορετικά από των απλών αρχείων
  - Data = το mapping: όνομα → αντικείμενο

File Allocation Table

- Ακολουθίες των data blocks των αρχείων → λίστες.
- Το FAT φορτώνεται στη RAM.
- Metadata: αποθηκεύονται μέσα στα directories.
- Δεν απαιτεί χωριστή διαχείριση του ελεύθερου χώρου.
- Χρήση στο MS-DOS, δεν χρησιμοποιείται πια.

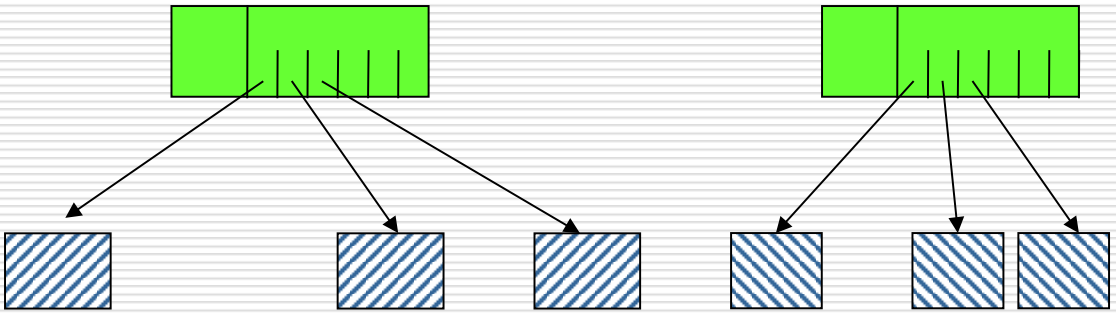


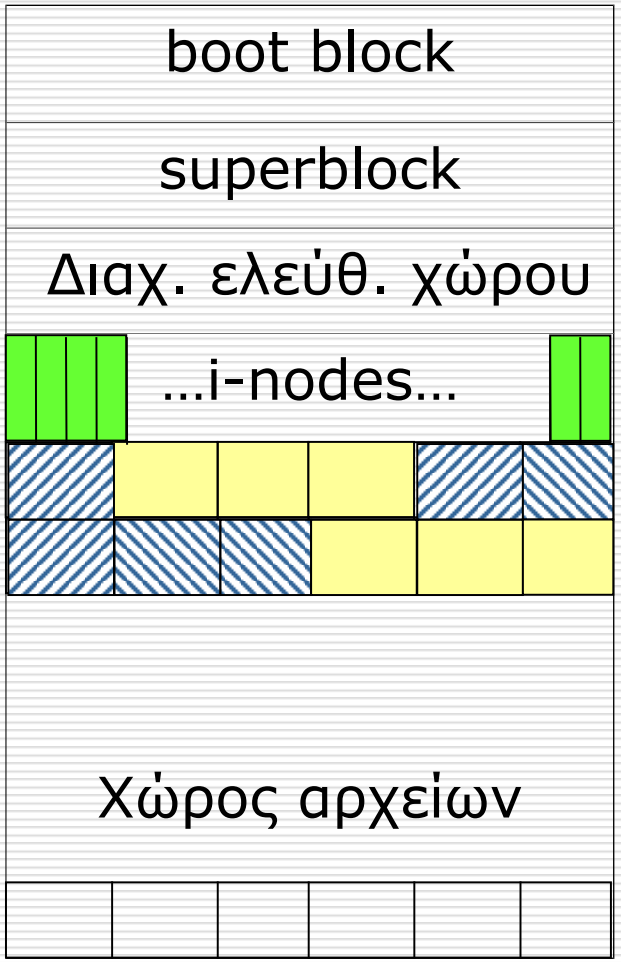


4	-1	-1	-1	6	7	-	8	-	-1	-1	-1						...

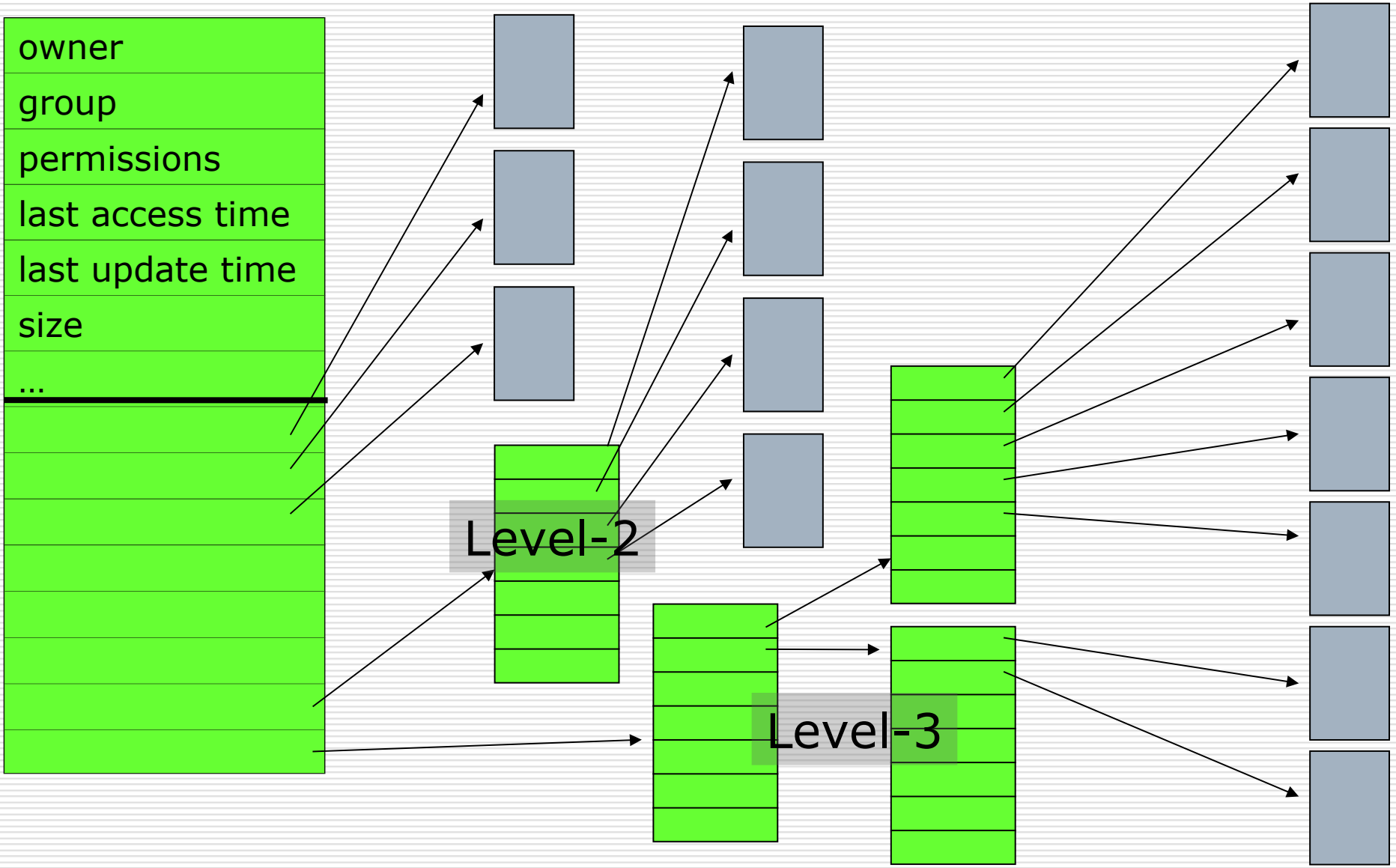
i-nodes

- Για κάθε αρχείο, ένας i-node (*index* node).
- Data blocks των αρχείων → δέντρα.





Δομή i-node


owner
group
permissions
last access time
last update time
size
...
Level-2
Level-3

Υλοποίηση Unix filesystem

- Βασισμένο σε i-nodes.
- Διαχείριση ελεύθερου χώρου με free-lists.
- Τα i-nodes είναι «σκορπισμένα» σε όλο το δίσκο.
- i-node παράμετροι
  - ????? level-1 blocks
  - ????? level-2/3 ????
- Directories: mappings όνομα → i-node
- i-nodes για μη-αρχεία
  - Περιέχουν τα αντίστοιχα metadata.

Metadata (μεταδεδομένα)

- Ορισμός: είναι δεδομένα που περιγράφουν άλλα δεδομένα (data about the data!)
- Εξίσου σημαντικά (ή περισσότερο !) με τα data.
- Σημαντική ποσότητα μνήμης.
- Δύο προβλήματα
  1. Integrity (ακεραιότητα): σε περίπτωση crash, θα πρέπει τα δεδομένα που είναι γραμμένα στο δίσκο να είναι «συνεπή» (consistent).
  2. Απόδοση: κάθε πράξη στο αρχείο (προσπέλαση, εγγραφή ...) πρέπει να ενημερώνει τα metadata (εγγραφή στο i-node).
     - 1 file write → 2 disk writes !!!!!

Integrity

- Εξασφαλίζεται από την ατομικότητα (atomicity).
- Μπορεί να παραβιαστεί από
  - Crash του συστήματος
  - Bugs! (πχ. Race conditions)
- Παραδείγματα προβλημάτων
  - Inode που δεν απεικονίζεται από direntry.
  - Inode ορίζει μεγαλύτερο μέγεθος αρχείου από τα blocks.
  - Block και στη freelist και σε κάποιο αρχείο
  - Block ούτε στη freelist ούτε σε κάποιο αρχείο.
  - ...
- Λύσεις
  - Off-line: το «καθάρισμα» του file system κατά την εκκίνηση.
  - On-line: journalling.

Καθάρισμα του file system

- Integrity
  - Εντολή fsck
  - Πρέπει να μπορεί να υπολογιστεί η θέση όλων των i-nodes.
  - Αντίγραφα του superblock.
  - Έλεγχος συνέπειας των i-nodes.
  - Blocks που μπορεί να περιέχουν χρήσιμα δεδομένα επιστρέφονται στο χρήστη
    - Πχ. ούτε στη freelist ούτε σε inode.
    - Φάκελος `lost+found`
- Defragmentation: αναδιάταξη των blocks ώστε να έχουμε ακολουθιακά αποθηκευμένα αρχεία.
- *Πρόβλημα*: πολύ αργή εκκκίνηση μετά από crash.

Journalling

- Βασική ιδέα: τηρούμε journal (ημερολόγιο) για όλες τις πράξεις που αφορούν metadata, δηλ.
  - Creat/unlink/mkdir/rmdir/…
  - Open/close/read/write/truncate
  - Chown/chmod/…
- Σε περίπτωση crash, διαβάζουμε το journal για να επαναφέρουμε το integrity.
- Τήρηση journal: Write-Ahead Log (WAL).
  - Οι εγγραφές του journal σώζονται πριν (ahead) το αντίστοιχο metadata update.
- Απόδοση: ελάχιστη επιβάρυνση.
- Πλεονέκτημα: πολύ γρήγορη εκκίνηση μετά από crash.

Δομή journal

- Σειριακό αρχείο (συστήματος).
  - Ακολουθία από εγγραφές
  - Πεδία εγγραφής:
    - Αύξων αριθμός, χρονοσφραγίδα (timestamp)
    - Περιγραφή πράξης: είδος, αρχείο, θέση κλπ.
- Εγγύηση:
  1. Κάθε ανανέωση metadata στο δίσκο πρέπει ήδη να αναφέρεται στο journal πριν εκτελεσθεί.
  2. Αφού εκτελεσθεί η ανανέωση, πρέπει το journal να περιέχει την επιβεβαίωση της εκτέλεσης.
- Ανάνηψη από crash: όσες πράξεις αναφέρονται αλλά δεν επιβεβαιώνονται θεωρούνται ως μη γενόμενες.
  - Αν χρειάζεται, *αναιρούνται*.

Metadata και performance

- Παραδείγματα προβλήματος.
  1. Αρχεία database: πολύ συχνή ενημέρωση του ίδιου αρχείου.
     - Μεγάλο κόστος ενημέρωσης, αλλά τα metadata δεν χρησιμοποιούνται!
  2. Πολύ μικρά αρχεία
     - Ένα seek επιπλέον για να ενημερωθεί ο i-node.
- Λύσεις
  - Ειδικές κλήσεις συστήματος, ρυθμίζουν τον τρόπο ενημέρωσης metadata.
    - Fsync/fdatasync/msync
    - Δε βοηθούν για μικρά αρχεία.
  - Log-structured file systems

Log-structured Filesystems (LFS)

- Berkeley, 1993(?)…
- Εφαρμογή: file servers όπου γράφονται συχνά πολλά μικρά αρχεία.
- Βασική ιδέα: επέκταση του journalling και στο γράψιμο μικρών αρχείων.
  1. Καθυστερεί το γράψιμο των μικρών αρχείων. Τα περιεχόμενά τους συλλέγονται και διατηρούνται στη μνήμη.
  2. Όταν μαζευτούν κάμποσα μικρά αρχεία, γράφονται όλα μαζί, i-node και δεδομένα, ακολουθιακά.
  - Εκτεταμένη χρήση journalling για να μειωθεί το overhead.

Backups

- Περιοδική δημιουργία αντιγράφων ασφαλείας για την αντιμετώπιση καταστροφικής απώλειας δεδομένων.
- Restore (επαναφορά): η ανάκτηση μιας συνεπούς κατάστασης του filesystem από αντίγραφα backup.
- Global (γενικό) backup. Αποθηκεύει ένα πλήρες αντίγραφο (στιγμιότυπο) της κατάστασης του filesystem σε μια δεδομένη στιγμή.
- Differential backup (με βάση κάποιο στιγμιότυπο). Αποθηκεύει τα αρχεία που μεταβλήθηκαν από τη λήψη του στιγμιοτύπου.
- Incremental backup (με βάση κάποιο στιγμιότυπο). Αποθηκεύει τα αρχεία που μεταβλήθηκαν από τη λήψη του τελευταίου global ή incremental backup.

Archive bit

- Αντιστοιχεί ένα bit Α σε κάθε αρχείο.
- Όταν το αρχείο μεταβάλλεται, Α=1.
- Global backup: Αντιγράφει όλα τα αρχεία. Θέτει Α=0.
- Differential backup: Αντιγράφει τα αρχεία με Α=1. ΔΕΝ αλλάζει το Α.
- Incremental backup: Αντιγράφει τα αρχεία με Α=1. Θέτει Α=0.
- Στην πράξη, χρησιμοποιούνται και οι 3 τρόποι:
  - Global: Μια φορά το μήνα.
  - Incremental: Μια φορά την εβδομάδα.
  - Differential: Κάθε μέρα.