



Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εργαστήριο Υπολογιστικών Συστημάτων

# Συστήματα Αρχείων σε Linux

Εργαστήριο Λειτουργικών Συστημάτων

3η εργαστηριακή άσκηση

Δεκέμβριος 2024

# 3<sup>η</sup> Εργαστηριακή Άσκηση – Linux Filesystems

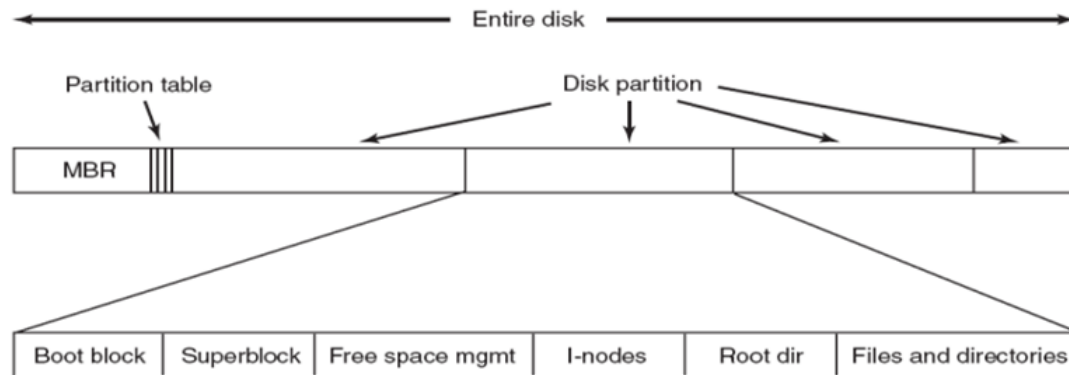
- Στόχος: εξοικίωση με τα συστήματα αρχείων στο Linux
- Δύο μέρη:
  1. Εξοικίωση με το σύστημα αρχείων ext2
    - Ανάλυση του πως οργανώνεται η πληροφορία σε έναν δίσκο με ext2 σύστημα αρχείων
    - Χρήση εργαλείων του userspace (hexdump, dmpre2fs, ...)
  2. Εξοικίωση με τη διεπαφή VFS του πυρήνα του Linux
    - Προσθήκη δικού σας συστήματος αρχείων
    - ext2-lite: μικρό υποσύνολο του ext2
- Χρονοδιάγραμμα:
  - Παρουσίαση: 5 Δεκ.
  - Εργαστήριο για απορίες: 12 Δεκ., 19 Δεκ.
  - Επίδειξη/Εξέταση: 9 Ιαν.
  - Υποβολή αναφοράς: Τέλος εξεταστικής

# Ερωτήσεις

- Τι είναι ένα Σύστημα Αρχείων (filesystem);
- Τι είναι το Αρχείο (file);
- Τι είναι ο Κατάλογος (directory);
- Τι ΣΑ χρησιμοποιεί ο υπολογιστής σας;
- Τι ΣΑ χρησιμοποιεί το Linux;

# Τι είναι ένα Σύστημα Αρχείων;

- **Σύστημα** οργάνωσης και διαχείρισης των δεδομένων σε ένα μέσο αποθήκευσης (HDD, SSD, CD-ROM, floppy, κλπ)
- Ο χρήστης αλληλεπιδρά με το ΣΑ μέσω της διεπαφής των **αρχείων**
- Διεπαφή συσκευών block (HDD, SSD, CD-ROM, ...)
  - Συνεχόμενη σειρά απο blocks σταθερού μεγέθους
  - Διάβασε το block N
  - Γράψε το block N



Source: Tanenbaum, Modern Operating Systems 3 e, (c) 2008 Prentice-Hall, Inc. 0-13-6006639

# Τι είναι ένα Σύστημα Αρχείων;

- Ερωτήσεις που προκύπτουν:
  - Πώς βρίσκω την πληροφορία (αρχείο) που θέλω μέσα στο μέσο;
  - Πώς μπορώ να βρω ποια blocks στο μέσο είναι διαθέσιμα;
  - Ποιο είναι το καταλληλότερο block (ή blocks) για την αποθήκευση νέας πληροφορίας;
  - Πώς μπορώ να προστατεύσω τα δεδομένα ενός χρήστη από έναν άλλον χρήστη;
  - ...
- Όλες αυτές τις ερωτήσεις τις απαντάει ένα ΣΑ

# Τι είναι το Αρχείο;

- Οι χρήστες δεν είναι βολικό να αναφέρονται σε blocks πληροφορίας
- Η διεπαφή του **αρχείου** είναι πιο εύκολη για τον χρήστη
- Ένα αρχείο είναι ένας μόνιμος, συνεχής, λογικός χώρος διευθύνσεων
  - **Μόνιμος:** Παραμένει προσβάσιμο και μετά τον τερματισμό του προγράμματος ή το κλείσιμο του υπολογιστή
  - **Συνεχής:** Χωρίς κενά
  - **Λογικός:** Ξεχωριστό από την φυσική απεικόνισή στην συσκευή αποθήκευσης
  - **Χώρος διευθύνσεων:** Διευθυνσιοδότηση δεδομένων σε επίπεδο byte

# Τι είναι το Αρχείο;

- Ιδιότητες αρχείου
  - Όνομα (π.χ., `transactions.txt`)
  - Τύπος\* (π.χ., regular file, directory, symbolic link, special file)
  - Μέγεθος
  - Δικαιώματα (προστασία)
  - Ώρα/Ημερομηνία πρόσβασης/τροποποίησης/δημιουργίας
  - Δεδομένα
  - ...

\* **“Everything is a file”** – ρητό του Unix

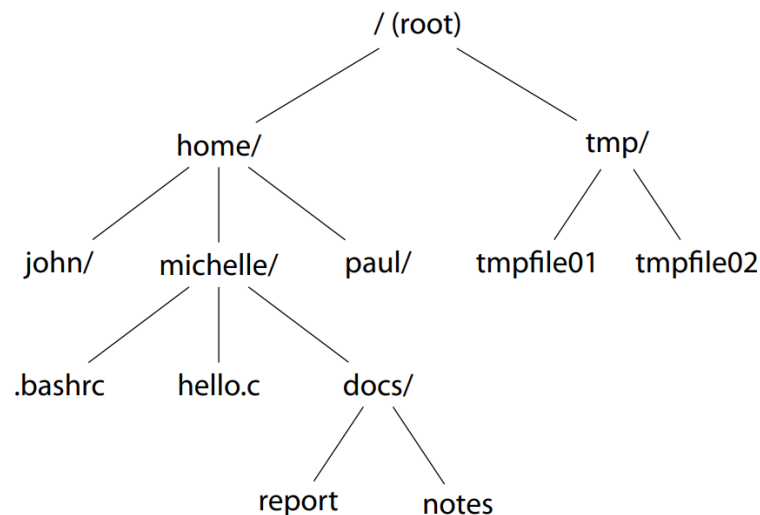
# Ποιες λειτουργίες υποστηρίζονται στα αρχεία;

- **Δημιουργία (create):** `creat(2)`, `open(2)`
- **Διαγραφή (delete):** `unlink(2)`
- **Μετονομασία (rename):** `rename(2)`
- **Άνοιγμα (open):** `open(2)`
- **Κλείσιμο (close):** `close(2)`
- **Ανάγνωση (read):** `read(2)`
- **Εγγραφή (write):** `write(2)`
- **Επανατοποθέτηση (seek):** `lseek(2)`
- **Αποκοπή (truncate):** `truncate(2)`
- **Εκτέλεση (execute):** `execve(2)`
- **Απεικόνιση στη μνήμη (mmap):** `mmap(2)`



# Τι είναι ο Κατάλογος;

- Οι κατάλογοι (directories, αλλιώς φάκελοι) βοηθούν στην οργάνωση των αρχείων στο δίσκο.
- Διαφορετικά ΣΑ υποστηρίζουν:
  - Κατάλογο ενός επιπέδου
  - Κατάλογο δύο επιπέδων
  - Καταλόγους δενδρικής δομής



# Ποιες λειτουργίες υποστηρίζονται στους καταλόγους;

- **Δημιουργία:** `mkdir(2)`
- **Διαγραφή:** `rmdir(2)`
- **Αναζήτηση αρχείου (με βάση το όνομα):** `open(2)`, `access(2)`
- **Δημιουργία αρχείου:** `creat(2)`, `open(2)`
- **Διαγραφή αρχείου:** `unlink(2)`
- **Μετονομασία αρχείου:** `rename(2)`
- **Λίστα καταλόγου:** `getdents64(2)`, `readdir(2)`

# Κατάλογοι δενδρικής δομής

(σε Σ.Α. τύπου Unix)

## Μονοπάτι (path):

Συμβολοσειρά από αναγνωριστικά χωρισμένα από τον χαρακτήρα /

πχ: /this/is/a/path/name

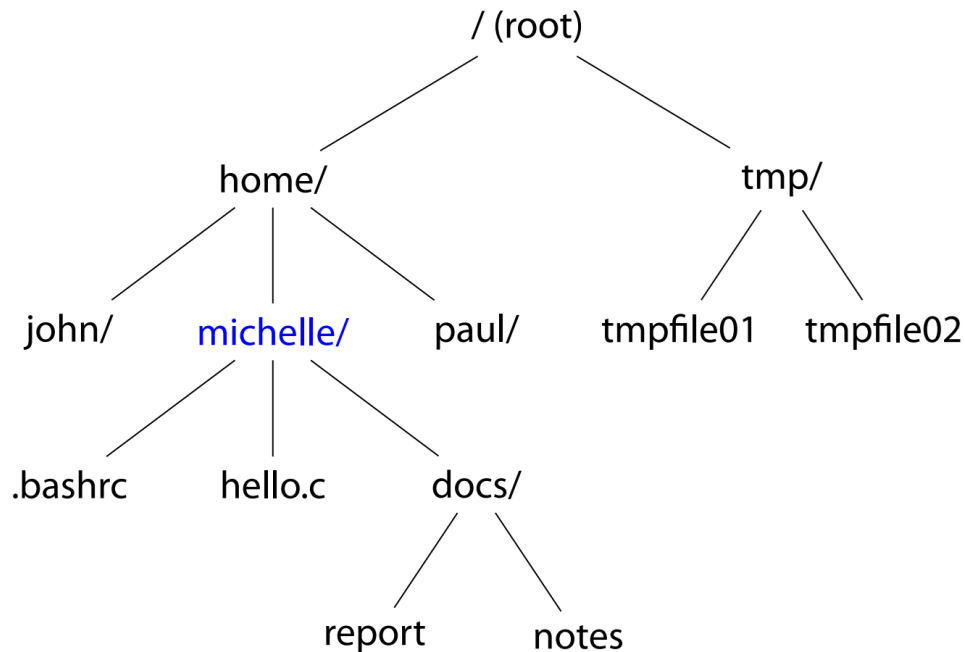
Κανόνες:

- Το μονοπάτι είναι
  1. **απόλυτο** αν ξεκινάει με / – αφετηρία είναι η αρχή της ιεραρχίας
  2. **σχετικό** (αν όχι) – αφετηρία είναι ο τρέχων κατάλογος (Current Working Directory - cwd)
- Το αναγνωριστικό:
  - σηματοδοτεί το cwd
  - • σηματοδοτεί τον πατέρα του cwd

# Παραδείγματα μονοπατιών

(σε Σ.Α. τύπου Unix)

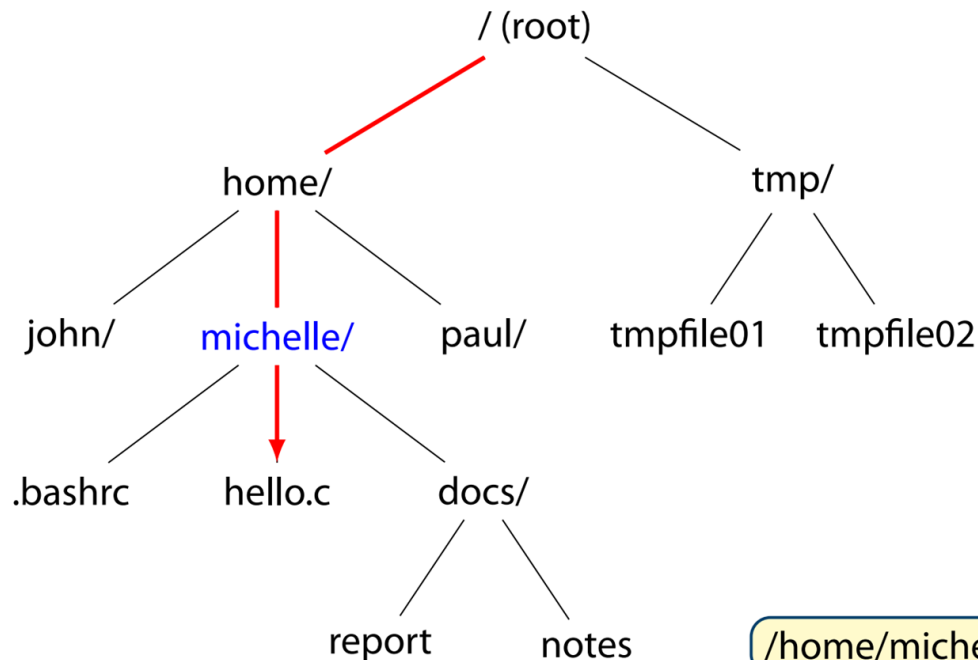
cwd: /home/michelle



# Παραδείγματα μονοπατιών

(σε Σ.Α. τύπου Unix)

cwd: /home/michelle

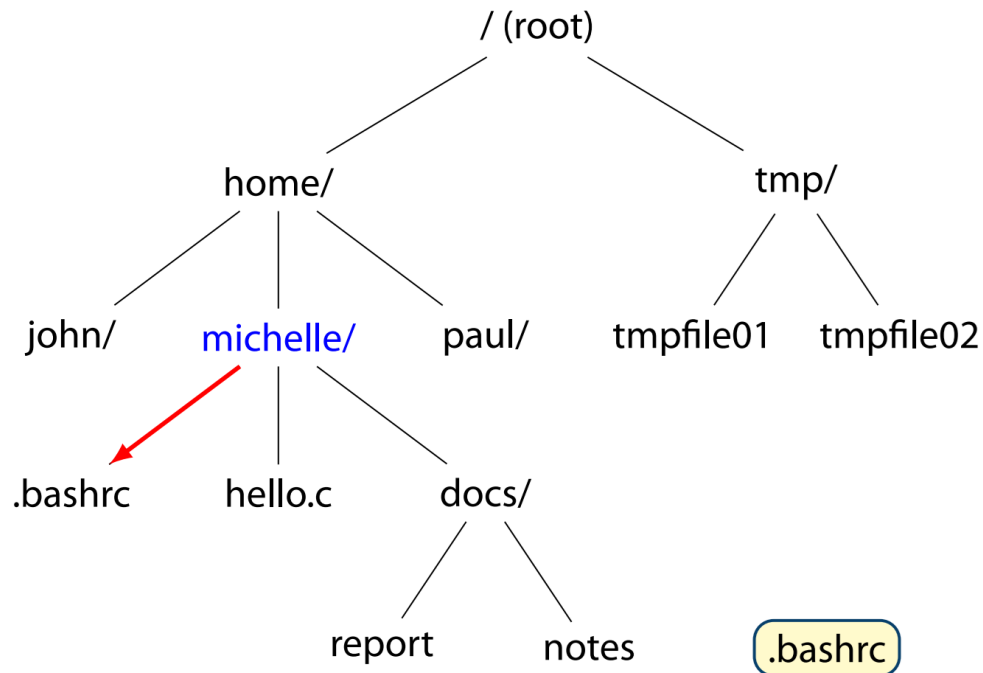


`/home/michelle/hello.c`

# Παραδείγματα μονοπατιών

(σε Σ.Α. τύπου Unix)

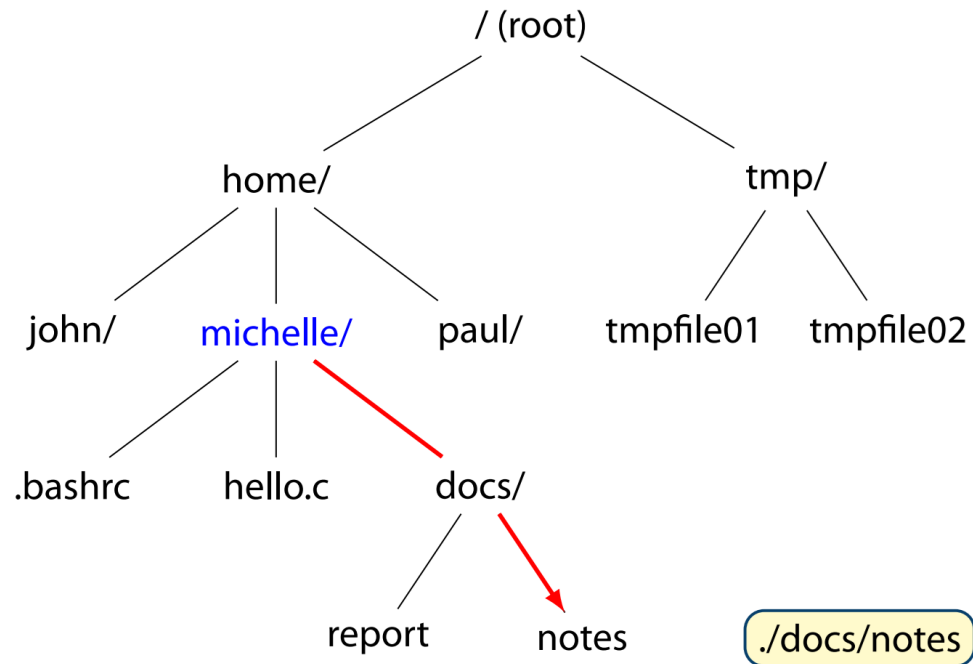
cwd: /home/michelle



# Παραδείγματα μονοπατιών

(σε Σ.Α. τύπου Unix)

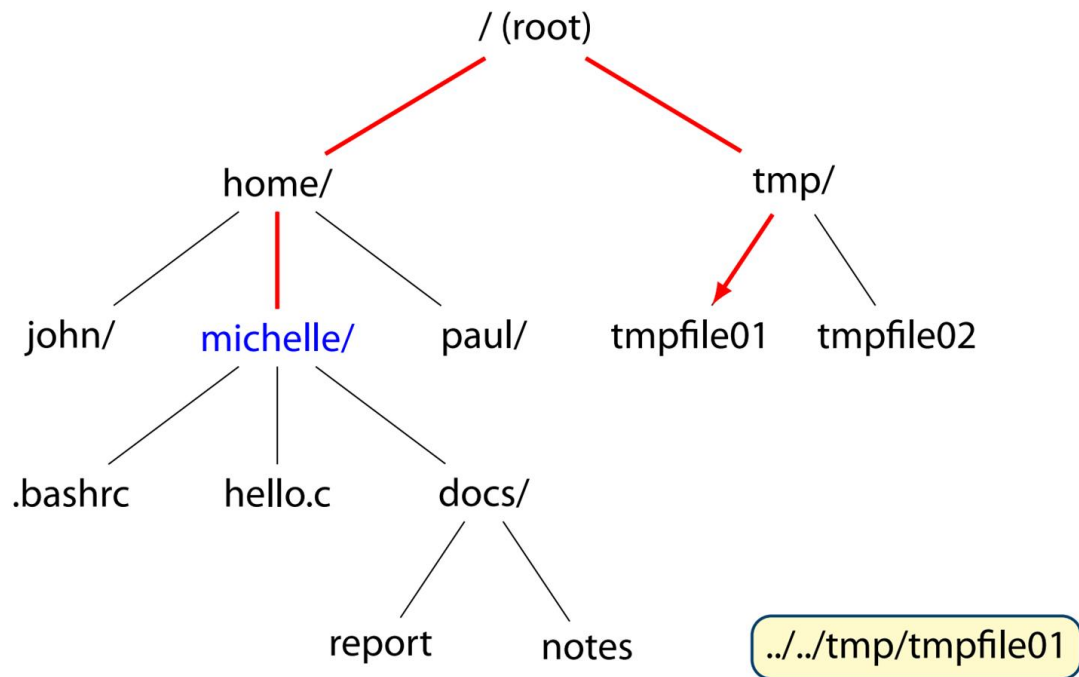
cwd: /home/michelle



# Παραδείγματα μονοπατιών

(σε Σ.Α. τύπου Unix)

cwd: /home/michelle



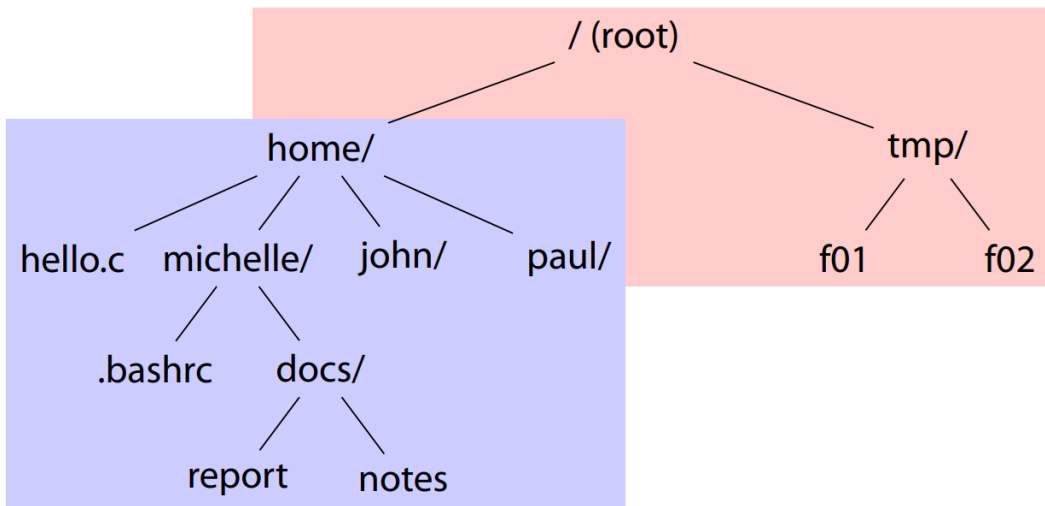


# Προσάρτηση ΣΑ

(mount)

Χρειάζονται:

- Σημείο προσάρτησης (mountpoint)
- Συσκευή αποθήκευσης (σκληρός δίσκος, flash, ...)



# Σύστημα αρχείων από την οπτική του χρήστη - Σύνοψη

- **Αρχείο** - Η διεπαφή για την πληροφορία στο μέσο αποθήκευσης
- **Κατάλογοι** - Περαιτέρω οργάνωση των αρχείων σε καταλόγους/φακέλους
- **Προστασία** - Δικαιώματα πρόσβασης σε κάθε αρχείο/κατάλογο
- **Κλήσεις συστήματος** - Για το χειρισμό αρχείων και καταλόγων

# Τι ΣΑ χρησιμοποιεί ο υπολογιστής σας;

Υπάρχει πληθώρα ΣΑ:

- ext2, ext3, ext4 → Linux
- FAT16, FAT32, exFAT, vFAT → Windows
- NTFS, HPFS → Windows
- UFS
- ZFS
- BTRFS
- XFS, ReiserFS
- NFS, AFS
- ...

# Το Σύστημα Αρχείων ext2

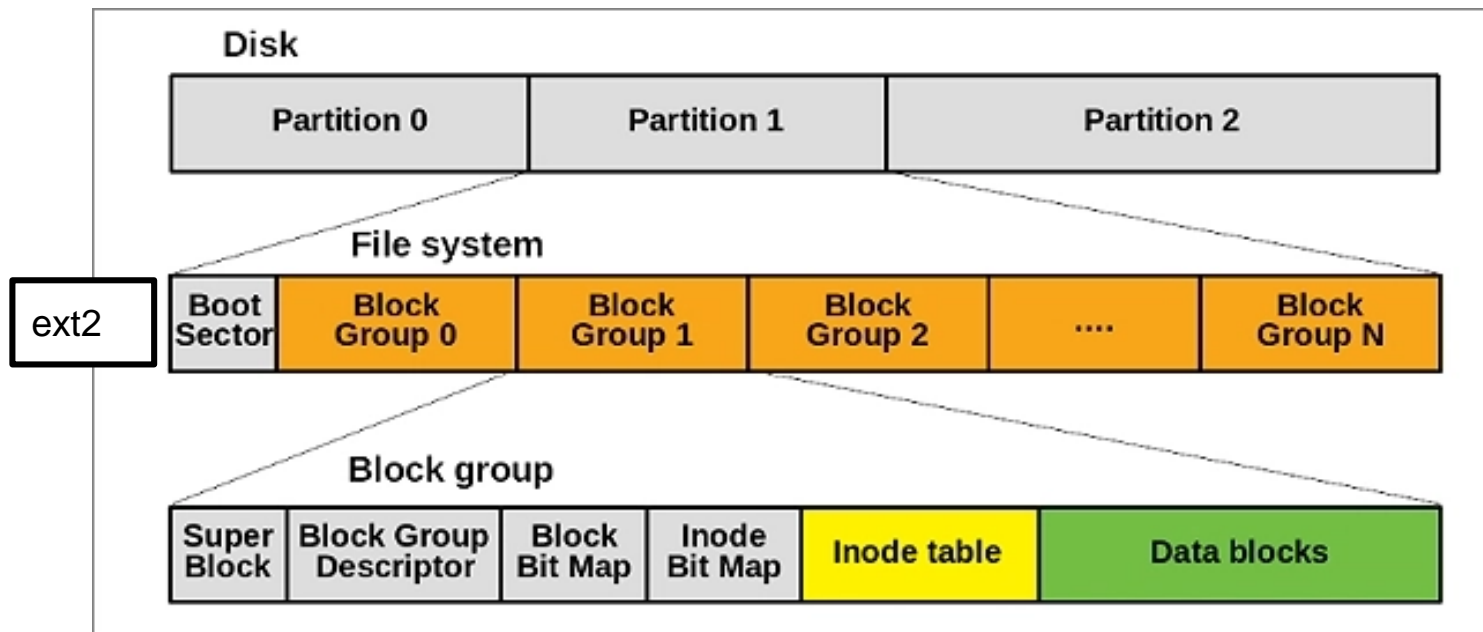
# ext2 - Γενικά

- Οικογένεια ΣΑ “EXTended File System”
  - ext/ext2/ext3/ext4
- Το βασικό ΣΑ στο Linux
- ext: Απρίλιος 1992
  - Αντικατέστησε το σύστημα αρχείων Minix στο Linux
- ext2: Ιανουάριος 1993
  - Έλυσε το βασικό πρόβλημα του ext
    - Προσθήκη timestamps πρόσβασης/τροποποίησης κάθε αρχείου, τροποποίησης inode
  - Εύκολα επεκτάσιμο
- ext3 (Νοέμβριος 2001) / ext4 (Οκτώβριος 2008):
  - Επεκτείνουν το ext2
  - Μία απο τις βασικότερες προσθήκες είναι το journaling

# ext2 - Γενικά

- Δημιουργήθηκε για το Linux.
- Η δομή του ακολουθεί τη δομή του Virtual File System (VFS) του Linux.
  - Η πιο σωστά, το VFS ακολουθεί τη δομή του ext2 ☺
- Οργανώνει τον δίσκο σε ομάδες απο blocks (block groups, BG)
  - Αρχείο/κατάλογοι στο ίδιο BG προσπελαύνονται πιο γρήγορα.
- Οι βασικές του δομές είναι:
  - Block
    - Συνεχόμενο κομμάτι του δίσκου
    - Σταθερού και προκαθορισμένου μεγέθους (1K, 2K, 4K και πιο σπάνια 8K)
  - Block group
    - Μία σειρά απο συνεχόμενα blocks
  - Superblock
    - Αποθηκεύει τα μεταδεδομένα σχετικά με το σύστημα αρχείων
  - Inode
    - Αποθηκεύει μεταδεδομένα σχετικά με κάθε αρχείο/κατάλογο/ειδικό αρχείο του συστήματος αρχείων

# EXT2 - Οργάνωση του δίσκου



# EXT2 - Οργάνωση του δίσκου



- Τα πρώτα 1024 bytes είναι το boot sector
- Μετά το boot sector ξεκινάει η λίστα των block groups
  - Ο συνολικός αριθμός των μπλοκ εξαρτάται από το συνολικό μέγεθος του συστήματος αρχείων και από το μέγεθος του μπλοκ.
  - Τα δεδομένα που αποθηκεύονται στο ίδιο block group μπορούν να προσπελαστούν με μεγαλύτερη ταχύτητα (λόγω του σχεδιασμού του υλικού των HDDs).

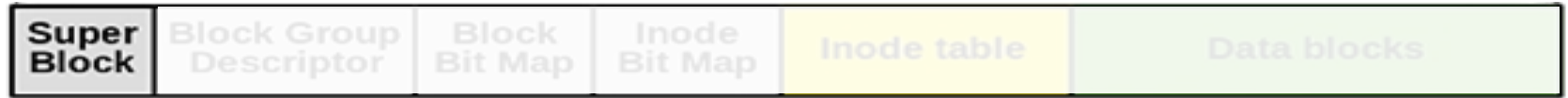


## ext2 - Η οργάνωση των Block Groups



- Superblock - 1 μπλοκ
- Block group descriptors - N μπλοκ (ανάλογα με τον αριθμό των block groups)
- Block bitmap - 1 μπλοκ
- Inode bitmap - 1 μπλοκ
- Inode table - N μπλοκ (ανάλογα με το μέγεθος του inode και το μέγεθος του μπλοκ)
- Data blocks - N μπλοκ (ανάλογα με το μέγεθος του μπλοκ και το μέγεθος των block groups)

## ext2 - Η οργάνωση των Block Groups



# ext2 - Η οργάνωση των Block Groups



Αποθηκεύει τα μεταδεδομένα σχετικά με το σύστημα αρχείων, π.χ.,

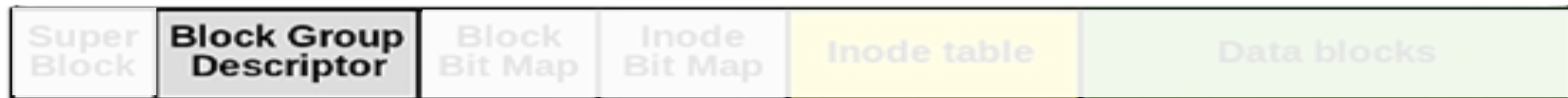
- Συνολικός αριθμός inodes
- Συνολικός αριθμός blocks
- Αριθμός διαθέσιμων inodes
- Αριθμός διαθέσιμων blocks
- ...

# ext2 - Η οργάνωση των Block Groups



Starting Byte	Ending Byte	Size in Bytes	Field Description
0	3	4	Total number of inodes in file system
4	7	4	Total number of blocks in file system
8	11	4	Number of blocks reserved for superuser (see offset 80)
12	15	4	Total number of unallocated blocks
16	19	4	Total number of unallocated inodes
20	23	4	Block number of the block containing the superblock (also the starting block number, NOT always zero.)
24	27	4	$\log_2$ (block size) - 10. (In other words, the number to shift 1,024 to the left by to obtain the block size)
28	31	4	$\log_2$ (fragment size) - 10. (In other words, the number to shift 1,024 to the left by to obtain the fragment size)
32	35	4	Number of blocks in each block group
36	39	4	Number of fragments in each block group
40	43	4	Number of inodes in each block group
44	47	4	Last mount time (in <a href="#">POSIX time</a> )
48	51	4	Last written time (in <a href="#">POSIX time</a> )

## ext2 - Η οργάνωση των Block Groups



# ext2 - Η οργάνωση των Block Groups

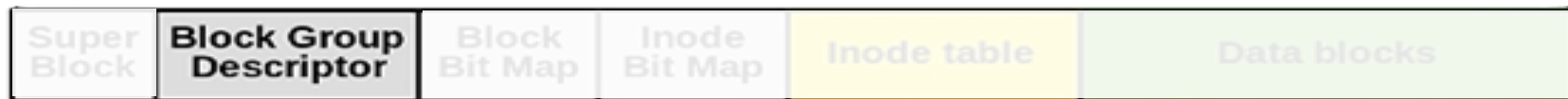


Πίνακας που περιέχει έναν block group descriptor για κάθε block group.

Κάθε block group descriptor αποθηκεύει τα παρακάτω μεταδεδομένα για το block group:

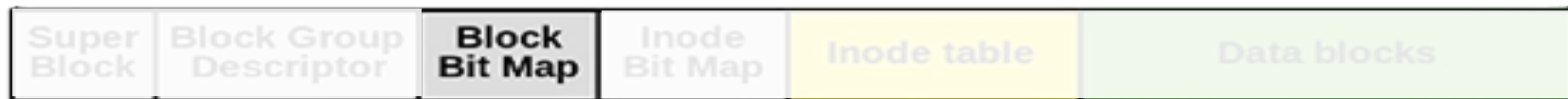
- Θέση του block bitmap του block group
- Θέση του inode bitmap του block group
- Θέση του inode table μέσα στο block group
- Αριθμός διαθέσιμων blocks στο block group
- Αριθμός διαθέσιμων inodes στο block group
- Αριθμός καταλόγων στο block group

# ext2 - Η οργάνωση των Block Groups



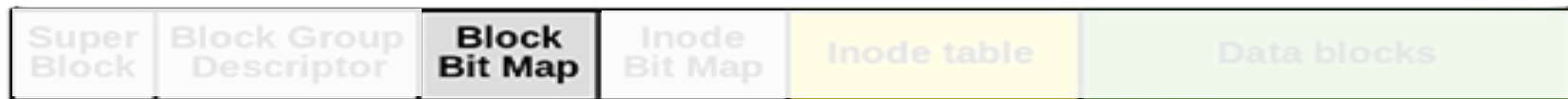
Starting Byte	Ending Byte	Size in Bytes	Field Description
0	3	4	Block address of block usage bitmap
4	7	4	Block address of inode usage bitmap
8	11	4	Starting block address of inode table
12	13	2	Number of unallocated blocks in group
14	15	2	Number of unallocated inodes in group
16	17	2	Number of directories in group
18	31	X	(Unused)

## ext2 - Η οργάνωση των Block Groups





# ext2 - Η οργάνωση των Block Groups



```
Terminal
root@utopia:~# xxd -s$((202*1024)) -l$((128)) -b /dev/vdb
00032800: 11111111 11111111 11111111 11111111 11111111 11111111 .....
00032806: 11111111 11111111 11111111 11111111 11111111 11111111 .....
0003280c: 11111111 11111111 11111111 11111111 11111111 11111111 .....
00032812: 11111111 11111111 11111111 11111111 11111111 11111111 .....
00032818: 11111111 11111111 11111111 11111111 11111111 11111111 .....
0003281e: 11111111 11111111 11111111 11111111 11111111 11111111 .....
00032824: 11111111 11111111 11111111 11111111 11111111 11111111 .....
0003282a: 11111111 11111111 11111111 11111111 11111111 11111111 .....
00032830: 11111111 11111111 11111111 11111111 11111111 11111111 .....
00032836: 11111111 00111111 00000000 00000000 00000000 00000000 .?...
0003283c: 00000000 00000000 00000000 00000000 00000000 00000000 .....
00032842: 00000000 00000000 00000000 00000000 00000000 00000000 .....
00032848: 00000000 00000000 00000000 00000000 00000000 00000000 .....
0003284e: 00000000 00000000 00000000 00000000 00000000 00000000 .....
00032854: 00000000 00000000 00000000 00000000 00000000 00000000 .....
0003285a: 00000000 00000000 00000000 00000000 00000000 00000000 .....
00032860: 00000000 00000000 00000000 00000000 00000000 00000000 .....
00032866: 00000000 00000000 00000000 00000000 00000000 00000000 .....
0003286c: 00000000 00000000 00000000 00000000 00000000 00000000 .....
00032872: 00000000 00000000 00000000 00000000 00000000 00000000 .....
00032878: 00000000 00000000 00000000 00000000 00000000 00000000 .....
0003287e: 00000000 00000000 .....
root@utopia:~#
```

Κάθε bit αντιπροσωπεύει ένα μπλοκ

- 0 ελεύθερο
- 1 χρησιμοποιείται

Σε ΣΑ με 1K μπλοκ μπορούμε να έχουμε μέχρι  $1024 \times 8$  μπλοκ/BG

**Προσοχή στη μορφή κάθε byte:**

στο 1ο byte π.χ. το MSB είναι το μπλοκ #7 και το LSB είναι το block #0

# ext2 - Η οργάνωση των Block Groups



Αντίστοιχο με το block bitmap αλλά για inodes:

- 0 ελεύθερο
- 1 χρησιμοποιείται

Σε ΣΑ με 1K block μπορούμε να έχουμε μέχρι  $1024 \times 8$  inodes/BG

**Προσοχή στη μορφή κάθε byte:**

στο 1ο byte π.χ. το MSB είναι το block #7 και το LSB είναι το block #0

## ext2 - Η οργάνωση των Block Groups

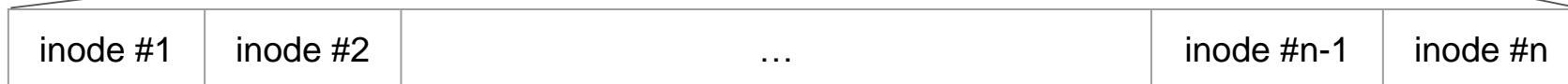


# ext2 - Η οργάνωση των Block Groups



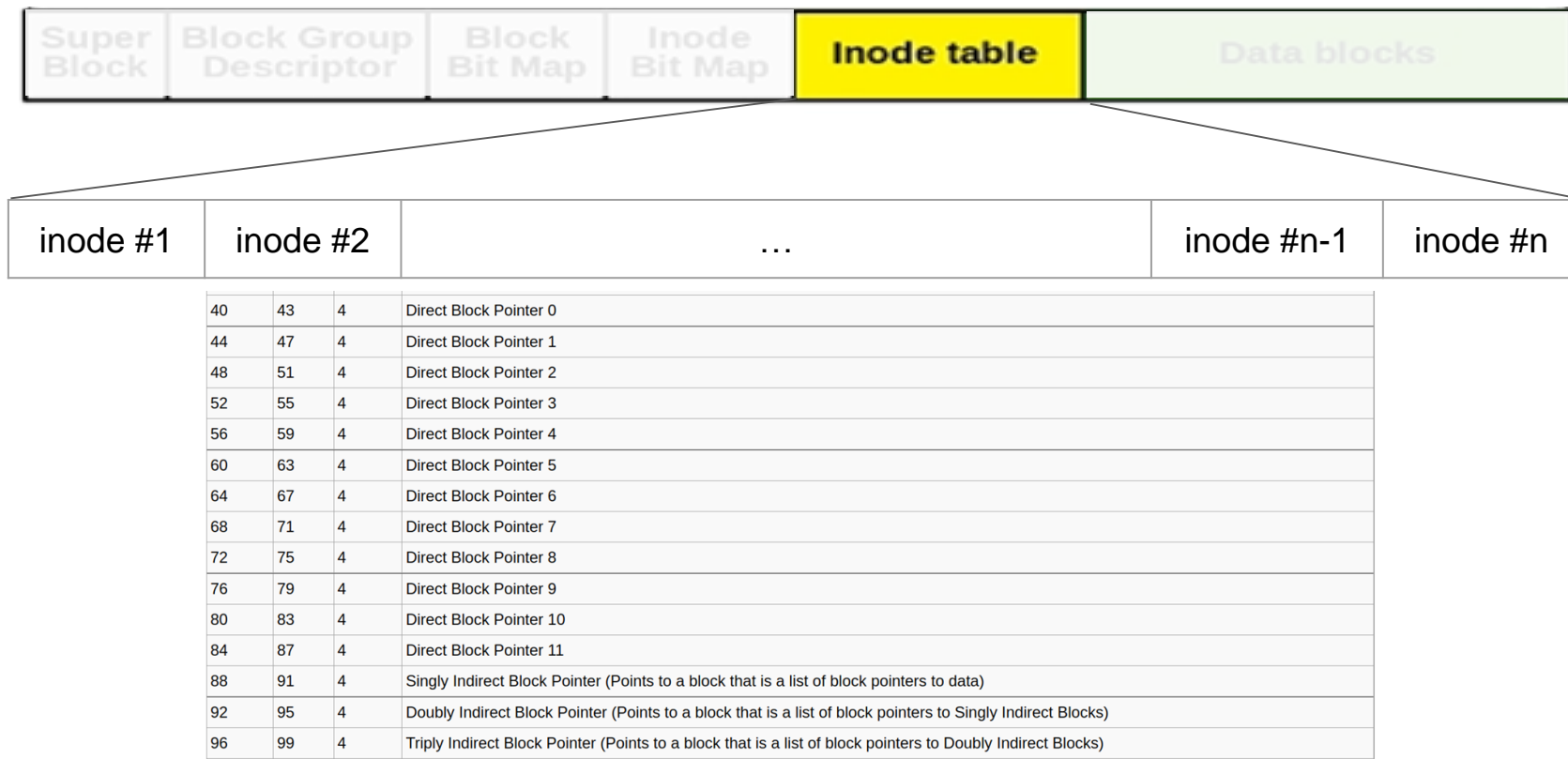
- Πίνακας απο inodes
- Σε παλιές εκδόσεις του ext2 κάθε inode έχει μέγεθος 128 bytes.
- Σε νέες εκδόσεις το μέγεθος του inode καθορίζεται κατα τη δημιουργία του ΣΑ.
- Η αρίθμηση των inodes ξεκινάει απο το 1 (σε αντίθεση με την αρίθμηση των μπλοκ που ξεκινάει απο το 0)
- Τα πρώτα 11 inodes στο BG #1 είναι πάντα δεσμευμένα απο το ΣΑ:
  - inode #2: ο αρχικός κατάλογος του ΣΑ (root)
  - inode #11: ο φάκελος lost+found

# ext2 - Η οργάνωση των Block Groups

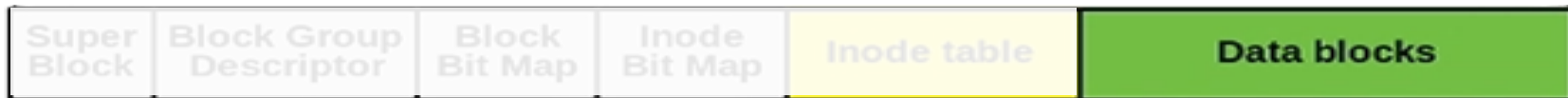


Starting Byte	Ending Byte	Size in Bytes	Field Description
0	1	2	Type and Permissions ( <a href="#">see below</a> )
2	3	2	User ID
4	7	4	Lower 32 bits of size in bytes
8	11	4	Last Access Time (in <a href="#">POSIX time</a> )
12	15	4	Creation Time (in <a href="#">POSIX time</a> )
16	19	4	Last Modification time (in <a href="#">POSIX time</a> )
20	23	4	Deletion time (in <a href="#">POSIX time</a> )
24	25	2	Group ID
26	27	2	Count of hard links (directory entries) to this inode. When this reaches 0, the data blocks are marked as unallocated.
28	31	4	Count of disk sectors (not Ext2 blocks) in use by this inode, not counting the actual inode structure nor directory entries linking to the inode.
32	35	4	Flags ( <a href="#">see below</a> )
36	39	4	<a href="#">Operating System Specific value #1</a>

# ext2 - Η οργάνωση των Block Groups

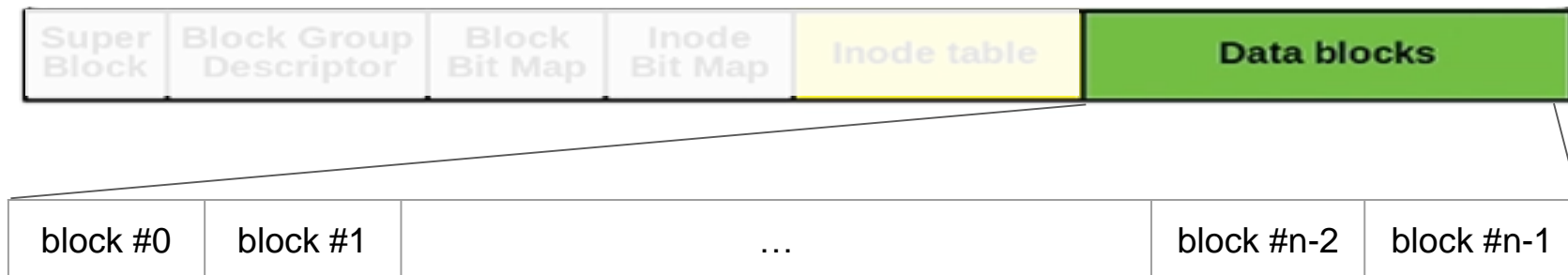


# ext2 - Η οργάνωση των Block Groups



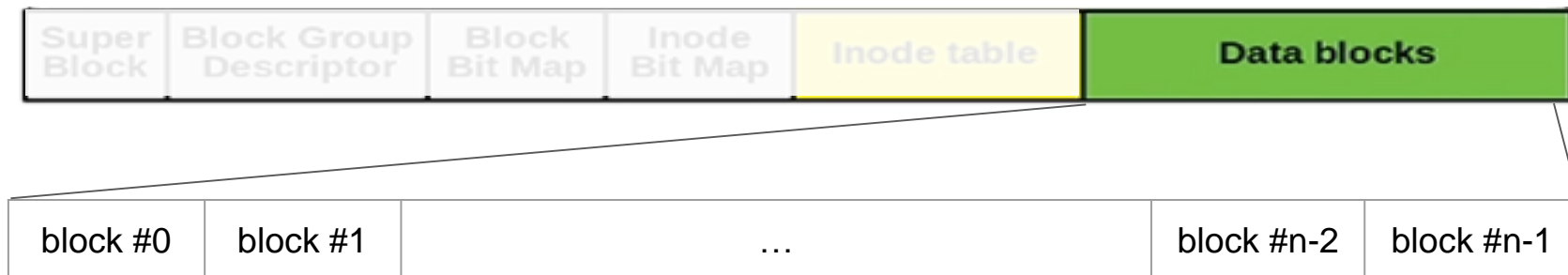
- Πίνακας απο blocks
- Η αρίθμηση των blocks ξεκινάει απο το 0 (σε αντίθεση με την αρίθμηση των inodes που ξεκινάει απο το 1)
- Τα περιεχόμενα κάθε block εξαρτώνται απο το είδος του inode στο οποίο ανήκουν (αρχείο/κατάλογος/ειδικό αρχείο)

## ext2 - Η οργάνωση των Block Groups





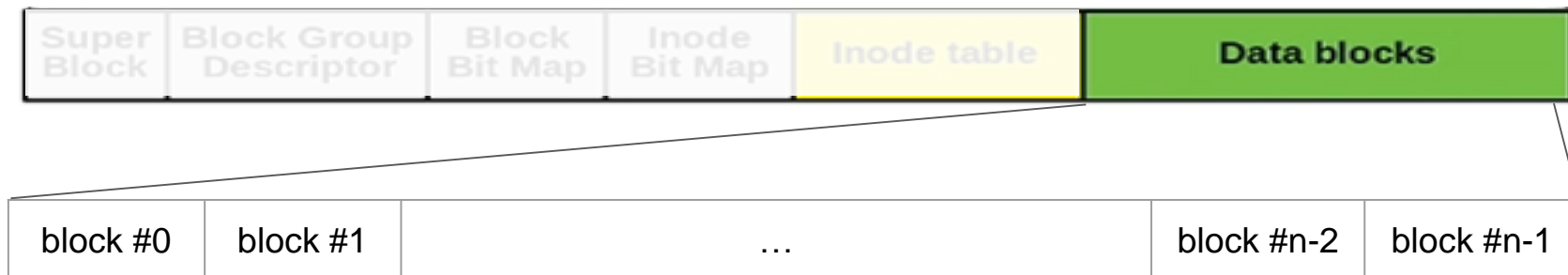
# ext2 - Η οργάνωση των Block Groups



Blocks απλού αρχείου (regular file): περιέχει τα περιεχόμενα του αρχείου

```
root@utopia:~# hexdump -s$((447*1024)) -n$((64)) -C /dev/vdb
0006fc00  48 65 6c 6c 6f 2c 20 57  6f 72 6c 64 0a 00 00 00  |Hello, World....|
0006fc10  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |.....|
*
0006fc40
root@utopia:~#
```

# ext2 - Η οργάνωση των Block Groups



Blocks καταλόγου: περιέχει μία λίστα απο directory entries

Starting Byte	Ending Byte	Size in Bytes	Field Description
0	3	4	Inode
4	5	2	Total size of this entry (Including all subfields)
6	6	1	Name Length least-significant 8 bits
7	7	1	Type indicator (only if the feature bit for "directory entries have file type byte" is set, else this is the most-significant 8 bits of the Name Length)
8	8+N-1	N	Name characters

# ext2 - Η οργάνωση των Block Groups

Blocks καταλόγου: περιέχει μία λίστα απο directory entries

Starting Byte	Ending Byte	Size in Bytes	Field Description
0	3	4	Inode
4	5	2	Total size of this entry (Including all subfields)
6	6	1	Name Length least-significant 8 bits
7	7	1	Type indicator (only if the feature bit for "directory entries have file type byte" is set, else this is the most-significant 8 bits of the Name Length)
8	8+N-1	N	Name characters

```
root@utopia:~# hexdump -s$((16616*1024)) -n$((1024)) -C /dev/vdb
0103a000  51 0e 00 00 0c 00 01 02  2e 00 00 00 02 00 00 00  |Q.....|
0103a010  0c 00 02 02 2e 2e 00 00  52 0e 00 00 10 00 05 01  |.....R.....|
0103a020  66 69 6c 65 31 00 00 00  54 0e 00 00 d8 03 07 02  |file1...T.....|
0103a030  64 69 72 6e 61 6d 65 00  00 00 00 00 00 00 00 00  |dirname.....|
0103a040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |.....|
*
0103a400
root@utopia:~#
```

# ext2 - Η οργάνωση των Block Groups

Blocks καταλόγου: περιέχει μία λίστα απο directory entries

Starting Byte	Ending Byte	Size in Bytes	Field Description
0	3	4	Inode
4	5	2	Total size of this entry (Including all subfields)
6	6	1	Name Length least-significant 8 bits
7	7	1	Type indicator (only if the feature bit for "directory entries have file type byte" is set, else this is the most-significant 8 bits of the Name Length)
8	8+N-1	N	Name characters

inode: 2

size: 12

len: 1

type: 2 (dir)

name: "."

```
root@utopia:~# hexdump -s$((16616*1024)) -n$((1024)) -C /dev/vdb
0103a000  51 0e 00 00 0c 00 01 02 2e 00 00 00 02 00 00 00 |Q.....|
0103a010  0c 00 02 02 2e 2e 00 00 52 0e 00 00 10 00 05 01 |.....R.....|
0103a020  66 69 6c 65 31 00 00 00 54 0e 00 00 d8 03 07 02 |file1...T.....|
0103a030  64 69 72 6e 61 6d 65 00 00 00 00 00 00 00 00 00 |dirname.....|
0103a040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
0103a400
root@utopia:~#
```

# ext2 - Η οργάνωση των Block Groups

Blocks καταλόγου: περιέχει μία λίστα απο directory entries

Starting Byte	Ending Byte	Size in Bytes	Field Description
0	3	4	Inode
4	5	2	Total size of this entry (Including all subfields)
6	6	1	Name Length least-significant 8 bits
7	7	1	Type indicator (only if the feature bit for "directory entries have file type byte" is set, else this is the most-significant 8 bits of the Name Length)
8	8+N-1	N	Name characters

inode: 2

size: 12

len: 2

type: 2 (dir)

name: ".."

```
root@utopia:~# hexdump -s$((16616*1024)) -n$((1024)) -C /dev/vdb
0103a000  51 0e 00 00 0c 00 01 02  2e 00 00 00 02 00 00 00 |Q.....|
0103a010  0c 00 02 02 2e 2e 00 00  52 0e 00 00 10 00 05 01 |.....R.....|
0103a020  66 69 6c 65 31 00 00 00  54 0e 00 00 d8 03 07 02 |file1...T.....|
0103a030  64 69 72 6e 61 6d 65 00  00 00 00 00 00 00 00 00 |dirname.....|
0103a040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |.....|
*
0103a400
root@utopia:~#
```

# ext2 - Η οργάνωση των Block Groups

Blocks καταλόγου: περιέχει μία λίστα απο directory entries

Starting Byte	Ending Byte	Size in Bytes	Field Description
0	3	4	Inode
4	5	2	Total size of this entry (Including all subfields)
6	6	1	Name Length least-significant 8 bits
7	7	1	Type indicator (only if the feature bit for "directory entries have file type byte" is set, else this is the most-significant 8 bits of the Name Length)
8	8+N-1	N	Name characters

inode: 3666  
(0x0e52)

size: 16

len: 5

type: 1 (reg. file)

name: "file1"

```
root@utopia:~# hexdump -s$((16616*1024)) -n$((1024)) -C /dev/vdb
0103a000  51 0e 00 00 0c 00 01 02  2e 00 00 00 02 00 00 00  |Q.....|
0103a010  0c 00 02 02 2e 2e 00 00  52 0e 00 00 10 00 05 01  |.....R.....|
0103a020  66 69 6c 65 31 00 00 00  54 0e 00 00 d8 03 07 02  |file1...T.....|
0103a030  64 69 72 6e 61 6d 65 00  00 00 00 00 00 00 00 00  |dirname.....|
0103a040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |.....|
*
0103a400
root@utopia:~#
```

# ext2 - Η οργάνωση των Block Groups

Blocks καταλόγου: περιέχει μία λίστα απο directory entries

Starting Byte	Ending Byte	Size in Bytes	Field Description
0	3	4	Inode
4	5	2	Total size of this entry (Including all subfields)
6	6	1	Name Length least-significant 8 bits
7	7	1	Type indicator (only if the feature bit for "directory entries have file type byte" is set, else this is the most-significant 8 bits of the Name Length)
8	8+N-1	N	Name characters

**inode:** 3668  
(0x0e52)

**size:** 984  
(0x0ed8)

**len:** 7

**type:** 2 (dir)

**name:** "dirname"

```
root@utopia:~# hexdump -s$((16616*1024)) -n$((1024)) -C /dev/vdb
0103a000  51 0e 00 00 0c 00 01 02  2e 00 00 00 02 00 00 00  |Q.....|
0103a010  0c 00 02 02 2e 2e 00 00  52 0e 00 00 10 00 05 01  |.....R.....|
0103a020  66 69 6c 65 31 00 00 00  54 0e 00 00 d8 03 07 02  |file1...T.....|
0103a030  64 69 72 6e 61 6d 65 00  00 00 00 00 00 00 00 00  |dirname.....|
0103a040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |.....|
*
0103a400
root@utopia:~#
```

# Χρήσιμα Εργαλεία/Εντολές

- `cat /proc/filesystems`
  - Δείχνει ποια συστήματα αρχείων είναι διαθέσιμα στον πυρήνα του Linux
- `cat /proc/mounts`
  - Δείχνει ποια συστήματα αρχείων είναι προσαρτημένα
- `fdisk, lsblk`
  - Πληροφορίες σχετικά με τις συσκευές μπλοκ που υπάρχουν στο σύστημα
- `mount`
  - Προσάρτηση ενός συστήματος αρχείων
- `dumpe2fs, e2fsck, mke2fs, tune2fs (e2fsprogs utilities)`
  - Εξερεύνηση, έλεγχος, δημιουργία και τροποποίηση ενός συστήματος αρχείων ext2/3/4
- `debugfs (e2fsprogs)`
  - Εξέταση και τροποποίηση ενός συστήματος αρχείων ext2/3/4
- `stat`
  - Προβολή πληροφοριών για το σύστημα αρχείων ή για κάποιο αρχείο
- `hexdump, hexedit, xxd`
  - Ανάγνωση και τροποποίηση αρχείων σε δεκαεξαδική μορφή
- `dd`
  - Ανάγνωση και τροποποίηση αρχείων σε επίπεδο byte

...manpages are your best friend!



## ext2 - Παράδειγμα 1: διάβασμα superblock

```

root@utopia:~#
root@utopia:~# dumpe2fs /dev/vdb
dumpe2fs 1.46.2 (28-Feb-2021)
Filesystem volume name:   fsdisk1.img
Last mounted on:         /mnt
Filesystem UUID:          7e7896bc-d444-4397-93f1-0f5b0bf3cea
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      (none)
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         not clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              12824
Block count:              51200
Reserved block count:     2560
Free blocks:              49545
Free inodes:              12809
First block:              1
Block size:               1024
Fragment size:            1024
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         1832
Inode blocks per group:   229
Filesystem created:       Thu Nov 24 17:59:41 2022
Last mount time:          Mon Dec 12 13:36:10 2022
Last write time:          Mon Dec 12 13:36:10 2022
Mount count:              37
Maximum mount count:      -1
Last checked:             Thu Nov 24 17:59:41 2022
Check interval:           0 (<none>)

```

[illegible]

## ext2 - Παράδειγμα 1: διάβασμα superblock

```

root@utopia:~#
root@utopia:~# dumpe2fs /dev/vdb
dumpe2fs 1.46.2 (28-Feb-2021)
Filesystem volume name:   fsdisk1.img
Last mounted on:         /mnt
Filesystem UUID:          7e7896bc-d444-4397-93f1-0f5b0bf3cea
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      (none)
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         not clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              12824
Block count:              51200
Reserved block count:     2560
Free blocks:              49545
Free inodes:              12809
First block:              1
Block size:               1024
Fragment size:            1024
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         1832
Inode blocks per group:   229
Filesystem created:       Thu Nov 24 17:59:41 2022
Last mount time:          Mon Dec 12 13:36:10 2022
Last write time:          Mon Dec 12 13:36:10 2022
Mount count:              37
Maximum mount count:      -1
Last checked:             Thu Nov 24 17:59:41 2022
Check interval:           0 (<none>)

```

[illegible]

## ext2 - Παράδειγμα 1: διάβασμα superblock

```

root@utopia:~#
root@utopia:~# dumpe2fs /dev/vdb
dumpe2fs 1.46.2 (28-Feb-2021)
Filesystem volume name:   fsdisk1.img
Last mounted on:         /mnt
Filesystem UUID:          7e7896bc-d444-4397-93f1-0f5b0bf3b3cea
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      (none)
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         not clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              12809
Block count:              51200
Reserved block count:     2560
Free blocks:              49545
Free inodes:              12809
First block:              1
Block size:               1024
Fragment size:            1024
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         1832
Inode blocks per group:   229
Filesystem created:       Thu Nov 24 17:59:41 2022
Last mount time:          Mon Dec 12 13:36:10 2022
Last write time:          Mon Dec 12 13:36:10 2022
Mount count:              37
Maximum mount count:      -1
Last checked:             Thu Nov 24 17:59:41 2022
Check interval:           0 (<none>)

```

[illegible]

## ext2 - Παράδειγμα 1: διάβασμα superblock

```

root@utopia:~#
root@utopia:~# dumpe2fs /dev/vdb
dumpe2fs 1.46.2 (28-Feb-2021)
Filesystem volume name:   fsdisk1.img
Last mounted on:         /mnt
Filesystem UUID:         7e7896bc-d444-4397-93f1-0f5b0bf3cea
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      (none)
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         not clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              12824
Block count:              51200
Reserved block count:     2560
Free blocks:              49545
Free inodes:              12809
First block:              1
Block size:               1024
Fragment size:            1024
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         1832
Inode blocks per group:   229
Filesystem created:       Thu Nov 24 17:59:41 2022
Last mount time:          Mon Dec 12 13:36:10 2022
Last write time:          Mon Dec 12 13:36:10 2022
Mount count:              37
Maximum mount count:      -1
Last checked:             Thu Nov 24 17:59:41 2022
Check interval:           0 (<none>)

```

[illegible]

## ext2 - Παράδειγμα 1: διάβασμα superblock

```

root@utopia:~#
root@utopia:~# dumpe2fs /dev/vdb
dumpe2fs 1.46.2 (28-Feb-2021)
Filesystem volume name:   fsdisk1.img
Last mounted on:         /mnt
Filesystem UUID:          7e7896bc-d444-4397-93f1-0f5b0bf3cea
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      (none)
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         not clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              12824
Block count:              51200
Reserved block count:     2560
Free blocks:              40545
Free inodes:              12809
First block:              1
Block size:               1024
Fragment size:            1024
Blocks per group:         8192
Fragments per group:     8192
Inodes per group:         1832
Inode blocks per group:   229
Filesystem created:       Thu Nov 24 17:59:41 2022
Last mount time:          Mon Dec 12 13:36:10 2022
Last write time:          Mon Dec 12 13:36:10 2022
Mount count:              37
Maximum mount count:      -1
Last checked:             Thu Nov 24 17:59:41 2022
Check interval:           0 (<none>)

```

[illegible]

## ext2 - Παράδειγμα 2: εύρεση δεδομένων αρχείου

Θέλουμε τα περιεχόμενα του `/dir1/file1`

# ext2 - Παράδειγμα 2: εύρεση δεδομένων αρχείου

Θέλουμε τα περιεχόμενα του `/dir1/file1`

**Βήμα 1:** Σε ποιο inode αντιστοιχεί το filename;











# ext2 - Παράδειγμα 2: εύρεση δεδομένων αρχείου

Θέλουμε τα περιεχόμενα του /dir1/file1

**Βήμα 1:** Σε ποιο inode αντιστοιχεί το filename;

Διαβάζουμε το directory entry block #433

```
root@utopia:~# export blocksz=1024
root@utopia:~# export GDT_OFFSET=$((2*blocksz))
root@utopia:~# export GD_SIZE=32
root@utopia:~# hexdump -s$((GDT_OFFSET)) -n$((GD_SIZE)) -C /dev/vdb
00000800 ca 00 00 00 cb 00 00 00 cc 00 00 00 42 1e 1d 07 |.....B...|
00000810 02 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
00000820
root@utopia:~# export INODE_SIZE=128
root@utopia:~# hexdump -s$((204*blocksz+INODE_SIZE)) -n$((INODE_SIZE)) -C /dev/vdb
00033080 ed 41 00 00 00 04 00 00 70 70 98 63 77 70 98 63 |.A.....pp.cwp.c|
00033090 77 70 98 63 00 00 00 00 00 00 04 00 02 00 00 00 |wp.C.....|
000330a0 00 00 00 00 01 00 00 00 b1 01 00 00 00 00 00 00 |.....|
000330b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00033100
root@utopia:~# hexdump -s$((433*blocksz)) -n$((64)) -C /dev/vdb
0006c400 02 00 00 00 0c 00 01 02 2e 00 00 00 02 00 00 00 |.....|
0006c410 0c 00 02 02 2e 2e 00 00 0b 00 00 00 14 00 0a 02 |.....|
0006c420 6c 6f 73 74 2b 66 6f 75 6e 64 00 00 51 0e 00 00 |lost+found..Q...|
0006c430 d4 03 04 02 64 69 72 31 00 00 00 00 00 00 00 00 |....dir1.....|
0006c440
root@utopia:~#
```

# ext2 - Παράδειγμα 2: εύρεση δεδομένων αρχείου

Θέλουμε τα περιεχόμενα του /dir1/file1

**Βήμα 1:** Σε ποιο inode αντιστοιχεί το filename;

Διαβάζουμε το directory entry block #433

Ψάχνουμε μέσα σε αυτό το όνομα dir1

```
root@utopia:~# export blocksz=1024
root@utopia:~# export GDT_OFFSET=$((2*blocksz))
root@utopia:~# export GD_SIZE=32
root@utopia:~# hexdump -s$((GDT_OFFSET)) -n$((GD_SIZE)) -C /dev/vdb
00000800 ca 00 00 00 cb 00 00 00 cc 00 00 00 42 1e 1d 07 |.....B...|
00000810 02 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
00000820
root@utopia:~# export INODE_SIZE=128
root@utopia:~# hexdump -s$((204*blocksz+INODE_SIZE)) -n$((INODE_SIZE)) -C /dev/vdb
00033080 ed 41 00 00 00 04 00 00 70 70 98 63 77 70 98 63 |.A.....pp.cwp.c|
00033090 77 70 98 63 00 00 00 00 00 00 04 00 02 00 00 00 |wp.C.....|
000330a0 00 00 00 00 01 00 00 00 b1 01 00 00 00 00 00 00 |.....|
000330b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00033100
root@utopia:~# hexdump -s$((433*blocksz)) -n$((64)) -C /dev/vdb
0006c400 02 00 00 00 0c 00 01 02 2e 00 00 00 02 00 00 00 |.....|
0006c410 0c 00 02 02 2e 2e 00 00 0b 00 00 00 14 00 0a 02 |.....|
0006c420 6c 6f 73 74 2b 66 6f 75 6e 64 00 00 51 0e 00 00 |lost+found..Q...|
0006c430 d4 03 04 02 64 69 72 31 00 00 00 00 00 00 00 00 |....dir1.....|
0006c440
root@utopia:~#
```

# ext2 - Παράδειγμα 2: εύρεση δεδομένων αρχείου

Θέλουμε τα περιεχόμενα του /dir1/file1

**Βήμα 1:** Σε ποιο inode αντιστοιχεί το filename;

Διαβάζουμε το directory entry block #433

Ψάχνουμε μέσα σε αυτό το όνομα dir1

directory entry structure

inode: 2
size: 12
len: 1
type: 2 (dir)
name: "."

```
root@utopia:~# export blocksz=1024
root@utopia:~# export GDT_OFFSET=$((2*blocksize))
root@utopia:~# export GD_SIZE=32
root@utopia:~# hexdump -s$((GDT_OFFSET)) -n$((GD_SIZE)) -C /dev/vdb
00000800 ca 00 00 00 cb 00 00 00 cc 00 00 00 42 1e 1d 07 |.....B...|
00000810 02 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
00000820
root@utopia:~# export INODE_SIZE=128
root@utopia:~# hexdump -s$((204*blocksize+INODE_SIZE)) -n$((INODE_SIZE)) -C /dev/vdb
00033080 ed 41 00 00 00 04 00 00 70 70 98 63 77 70 98 63 |.A.....pp.cwp.c|
00033090 77 70 98 63 00 00 00 00 00 00 04 00 02 00 00 00 |wp.C.....|
000330a0 00 00 00 00 01 00 00 00 b1 01 00 00 00 00 00 00 |.....|
000330b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00033100
root@utopia:~# hexdump -s$((433*blocksize)) -n$((64)) -C /dev/vdb
0006c400 02 00 00 00 0c 00 01 32 2e 00 00 00 02 00 00 00 |.....|
0006c410 0c 00 02 02 2e 2e 00 00 00 00 00 00 14 00 0a 02 |.....|
0006c420 6c 6f 73 74 2b 66 6f 75 6e 64 00 00 51 0e 00 00 |lost+found..Q...|
0006c430 d4 03 04 02 64 69 72 31 00 00 00 00 00 00 00 00 |....dir1.....|
0006c440
root@utopia:~#
```

# ext2 - Παράδειγμα 2: εύρεση δεδομένων αρχείου

Θέλουμε τα περιεχόμενα του /dir1/file1

**Βήμα 1:** Σε ποιο inode αντιστοιχεί το filename;

Διαβάζουμε το directory entry block #433

Ψάχνουμε μέσα σε αυτό το όνομα dir1

directory entry structure

inode: 2
size: 12
len: 2
type: 2 (dir)
name: ".."

```
root@utopia:~# export blocksz=1024
root@utopia:~# export GDT_OFFSET=$((2*blocksize))
root@utopia:~# export GD_SIZE=32
root@utopia:~# hexdump -s$((GDT_OFFSET)) -n$((GD_SIZE)) -C /dev/vdb
00000800 ca 00 00 00 cb 00 00 00 cc 00 00 00 42 1e 1d 07 |.....B...|
00000810 02 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
00000820
root@utopia:~# export INODE_SIZE=128
root@utopia:~# hexdump -s$((204*blocksize+INODE_SIZE)) -n$((INODE_SIZE)) -C /dev/vdb
00033080 ed 41 00 00 00 04 00 00 70 70 98 63 77 70 98 63 |.A.....pp.cwp.c|
00033090 77 70 98 63 00 00 00 00 00 00 04 00 02 00 00 00 |wp.C.....|
000330a0 00 00 00 00 01 00 00 00 b1 01 00 00 00 00 00 00 |.....|
000330b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00033100
root@utopia:~# hexdump -s$((433*blocksize)) -n$((51)) -C /dev/vdb
0006c400 02 00 00 00 0c 00 01 02 2e 00 00 00 02 00 00 00 |.....|
0006c410 0c 00 02 02 2e 2e 30 00 0b 00 00 00 14 00 0a 02 |.....|
0006c420 0c 01 73 74 20 00 5f 75 6e 64 00 00 51 0e 00 00 |lost+found..Q...|
0006c430 d4 03 04 02 64 69 72 31 00 00 00 00 00 00 00 00 |....dir1.....|
0006c440
root@utopia:~#
```



# ext2 - Παράδειγμα 2: εύρεση δεδομένων αρχείου

Θέλουμε τα περιεχόμενα του /dir1/file1

**Βήμα 1:** Σε ποιο inode αντιστοιχεί το filename;

Διαβάζουμε το directory entry block #433

Ψάχνουμε μέσα σε αυτό το όνομα dir1

directory entry structure

inode: 11 (0x0a)
size: 20
len: 10
type: 2 (dir)
name: "lost+found"

```
root@utopia:~# export blocksz=1024
root@utopia:~# export GDT_OFFSET=$((2*blocksize))
root@utopia:~# export GD_SIZE=32
root@utopia:~# hexdump -s$((GDT_OFFSET)) -n$((GD_SIZE)) -C /dev/vdb
00000800 ca 00 00 00 cb 00 00 00 cc 00 00 00 42 1e 1d 07 |.....B...|
00000810 02 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
00000820
root@utopia:~# export INODE_SIZE=128
root@utopia:~# hexdump -s$((204*blocksize+INODE_SIZE)) -n$((INODE_SIZE)) -C /dev/vdb
00033080 ed 41 00 00 00 04 00 00 70 70 98 63 77 70 98 63 |.A.....pp.cwp.c|
00033090 77 70 98 63 00 00 00 00 00 00 04 00 02 00 00 00 |wp.C.....|
000330a0 00 00 00 00 01 00 00 00 b1 01 00 00 00 00 00 00 |.....|
000330b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00033100
root@utopia:~# hexdump -s$((433*blocksize)) -n$((64)) -C /dev/vdb
0006c400 02 00 00 00 0c 00 01 02 2e 00 00 00 02 00 00 00 |.....|
0006c410 0c 00 02 02 2e 2e 00 00 0b 00 00 00 14 00 0a 02 |.....|
0006c420 6c 6f 73 74 2b 66 6f 75 6e 64 00 00 51 0e 00 00 |lost+found..Q...|
0006c430 d4 03 04 02 64 69 72 31 00 00 00 00 00 00 00 00 |....dir1.....|
0006c440
root@utopia:~#
```



# ext2 - Παράδειγμα 2: εύρεση δεδομένων αρχείου

Θέλουμε τα περιεχόμενα του /dir1/file1

**Βήμα 1:** Σε ποιο inode αντιστοιχεί το filename;

Διαβάζουμε το directory entry block #433

Ψάχνουμε μέσα σε αυτό το όνομα dir1

Βρήκαμε το inode του /dir1 (inode #3665)

directory entry structure

inode: 3665 (0x0e51)
size: 980
len: 4
type: 2 (dir)
name: "dir1"

```
root@utopia:~# export blocksiz=1024
root@utopia:~# export GDT_OFFSET=$((2*blocksiz))
root@utopia:~# export GD_SIZE=32
root@utopia:~# hexdump -s$((GDT_OFFSET)) -n$((GD_SIZE)) -C /dev/vdb
00000800 ca 00 00 00 cb 00 00 00 cc 00 00 00 42 1e 1d 07 |.....B...|
00000810 02 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
00000820
root@utopia:~# export INODE_SIZE=128
root@utopia:~# hexdump -s$((204*blocksiz+INODE_SIZE)) -n$((INODE_SIZE)) -C /dev/vdb
00033080 ed 41 00 00 00 04 00 00 70 70 98 63 77 70 98 63 |.A.....pp.cwp.c|
00033090 77 70 98 63 00 00 00 00 00 00 04 00 02 00 00 00 |wp.C.....|
000330a0 00 00 00 00 01 00 00 00 b1 01 00 00 00 00 00 00 |.....|
000330b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00033100
root@utopia:~# hexdump -s$((433*blocksiz)) -n$((64)) -C /dev/vdb
0006c400 02 00 00 00 0c 00 01 02 2e 00 00 00 02 00 00 00 |.....|
0006c410 0c 00 02 02 2e 2e 00 00 0b 00 00 00 14 00 0a 02 |.....|
0006c420 0c 00 02 02 2e 2e 00 00 0b 00 00 00 51 0e 00 00 |lost+found..Q...|
0006c430 d4 03 04 02 64 69 72 31 00 00 00 00 00 00 00 00 |....dir1.....|
0006c440
root@utopia:~#
```

# ext2 - Παράδειγμα 2: εύρεση δεδομένων αρχείου

Θέλουμε τα περιεχόμενα του `/dir1/file1`

**Βήμα 1:** Σε ποιο inode αντιστοιχεί το filename;

Το inode #3665 είναι το 1ο inode του BG #2 (1832 inodes per BG) (τα inodes ξεκινάνε από το #1)

Διαβάζουμε από το BGD του BG #2 το block του inode table







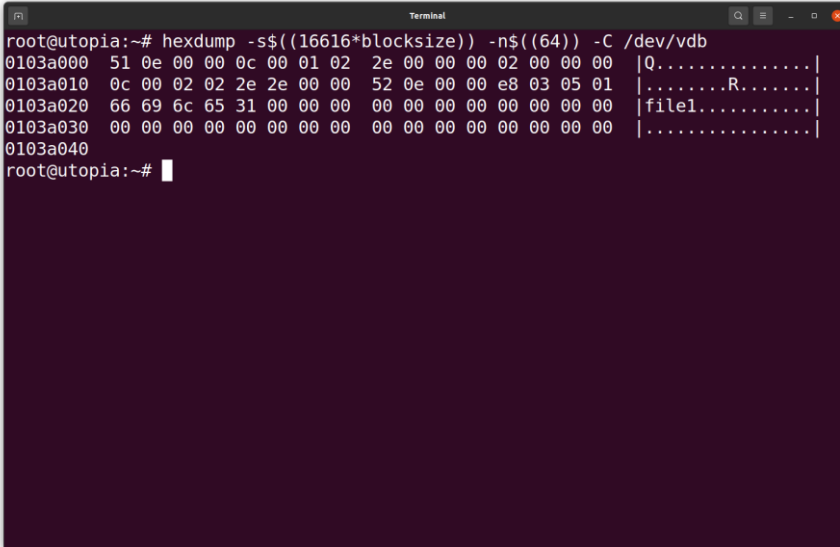


## ext2 - Παράδειγμα 2: εύρεση δεδομένων αρχείου

Θέλουμε τα περιεχόμενα του /dir1/file1

**Βήμα 1:** Σε ποιο inode αντιστοιχεί το filename;

Διαβάζουμε το directory entry block #16616

A terminal window titled "Terminal" showing the output of a hexdump command. The command is: `root@utopia:~# hexdump -s$((16616*blocksize)) -n$((64)) -C /dev/vdb`. The output shows four lines of hexadecimal data with their corresponding ASCII representations in a column on the right. The third line shows the filename "file1".

```
root@utopia:~# hexdump -s$((16616*blocksize)) -n$((64)) -C /dev/vdb
0103a000  51 0e 00 00 0c 00 01 02  2e 00 00 00 02 00 00 00  |Q.....|
0103a010  0c 00 02 02 2e 2e 00 00  52 0e 00 00 e8 03 05 01  |.....R....|
0103a020  66 69 6c 65 31 00 00 00  00 00 00 00 00 00 00 00  |file1.....|
0103a030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |.....|
0103a040
root@utopia:~#
```

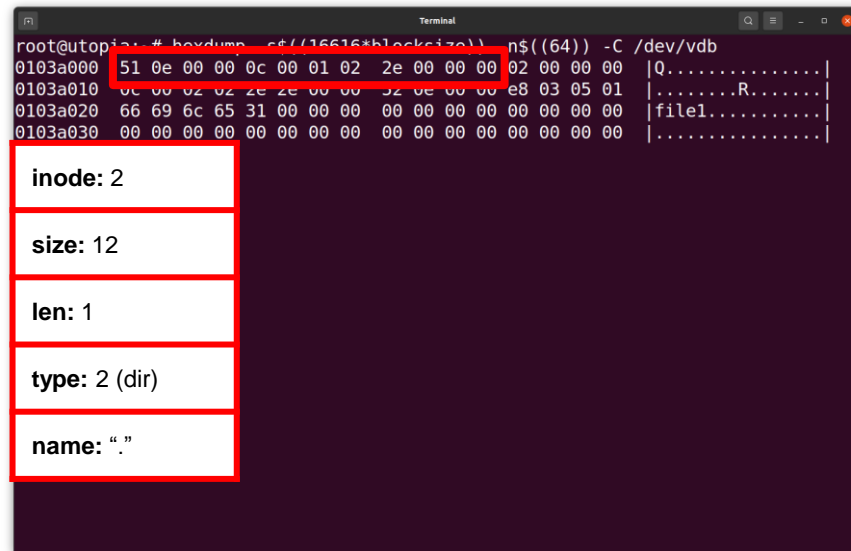
# ext2 - Παράδειγμα 2: εύρεση δεδομένων αρχείου

Θέλουμε τα περιεχόμενα του `/dir1/file1`

**Βήμα 1:** Σε ποιο inode αντιστοιχεί το filename;

Διαβάζουμε το directory entry block #16616

Ψάχνουμε μέσα σε αυτό το όνομα `file1`



```
root@utopia:~# hexdump -s/(16616*blocksize) -n$( (64)) -C /dev/vdb
0103a000  51 0e 00 00 0c 00 01 02  2e 00 00 00  02 00 00 00  |Q.....|
0103a010  0c 00 02 02 2e 2e 00 00  32 0e 00 00  e8 03 05 01  |.....R....|
0103a020  66 69 6c 65 31 00 00 00  00 00 00 00  00 00 00 00  |file1.....|
0103a030  00 00 00 00 00 00 00 00  00 00 00 00  00 00 00 00  |.....|
```

inode: 2
size: 12
len: 1
type: 2 (dir)
name: "."



# ext2 - Παράδειγμα 2: εύρεση δεδομένων αρχείου

Θέλουμε τα περιεχόμενα του /dir1/file1

**Βήμα 1:** Σε ποιο inode αντιστοιχεί το filename;

Διαβάζουμε το directory entry block #16616

Ψάχνουμε μέσα σε αυτό το όνομα file1

```
root@utopia:~# hexdump -s$((16616*blocksize)) -n$((64)) -C /dev/vdb
0103a000  51 00 00 00 0c 00 01 02  2e 00 00 00 02 00 00 00  |Q.....|
0103a010  0c 00 02 02 2e 2e 00 00  52 0e 00 00 e0 03 03 01  |.....R...|
0103a020  00 03 0c 03 31 00 00 00  00 00 00 00 00 00 00 00  |file1.....|
0103a030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |.....|
0103a040
root@utopia:~#
```

inode: 2
size: 12
len: 2
type: 2 (dir)
name: ".."

# ext2 - Παράδειγμα 2: εύρεση δεδομένων αρχείου

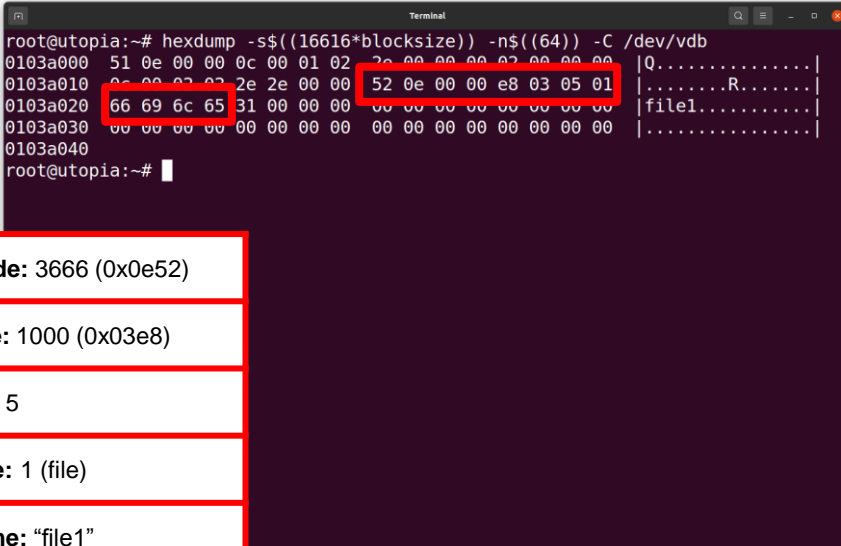
Θέλουμε τα περιεχόμενα του `/dir1/file1`

**Βήμα 1:** Σε ποιο inode αντιστοιχεί το filename;

Διαβάζουμε το directory entry block #16616

Ψάχνουμε μέσα σε αυτό το όνομα `file1`

Βρήκαμε το inode του `/dir1/file1` (inode #3666)



The terminal screenshot shows the command `hexdump -s$((16616*blocksize)) -n$((64)) -C /dev/vdb` being executed. The output displays hexadecimal data with corresponding ASCII characters on the right. Two red boxes highlight specific parts of the output: one box highlights the bytes `66 69 6c 65` (which correspond to the ASCII string "file") and another box highlights the bytes `52 0e 00 00 e8 03 05 01` (which correspond to the ASCII string "Q.....R.....file1.....").

inode: 3666 (0x0e52)
size: 1000 (0x03e8)
len: 5
type: 1 (file)
name: "file1"

# ext2 - Παράδειγμα 2: εύρεση δεδομένων αρχείου

Θέλουμε τα περιεχόμενα του `/dir1/file1`

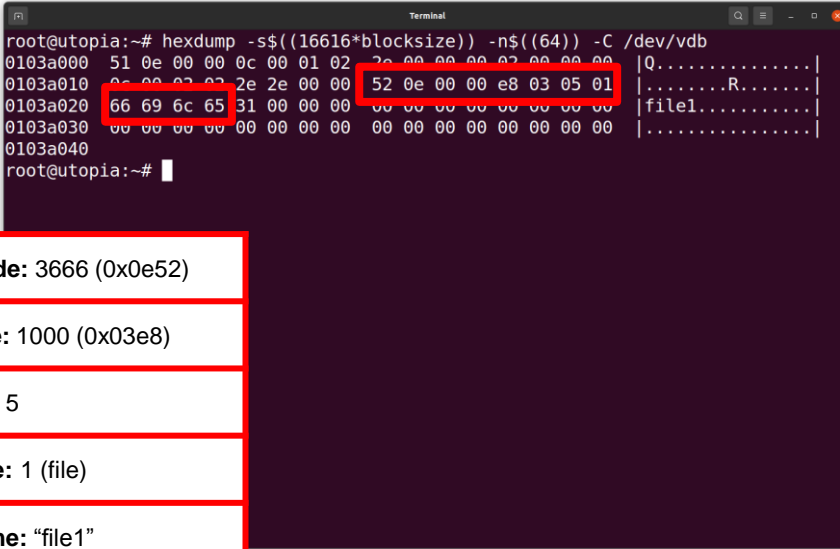
**Βήμα 1:** Σε ποιο inode αντιστοιχεί το filename;

Διαβάζουμε το directory entry block #16616

Ψάχνουμε μέσα σε αυτό το όνομα `file1`

Βρήκαμε το inode του `/dir1/file1` (inode #3666)

Τέλος 1ου Βήματος!



The terminal screenshot shows the command `hexdump -s$((16616*blocksize)) -n$((64)) -C /dev/vdb` being executed. The output displays hexadecimal data with corresponding ASCII characters on the right. Two red boxes highlight specific parts of the output: one box highlights the bytes `66 69 6c 65` (which correspond to the ASCII string "file") and another box highlights the bytes `52 0e 00 00 e8 03 05 01`. Below the terminal output, a red-bordered box contains a table of file metadata extracted from the directory entry.

inode:	3666 (0x0e52)
size:	1000 (0x03e8)
len:	5
type:	1 (file)
name:	"file1"

# ext2 - Παράδειγμα 2: εύρεση δεδομένων αρχείου

Θέλουμε τα περιεχόμενα του `/dir1/file1`

**Βήμα 2:** Σε ποιο block υπάρχουν τα περιεχόμενα του αρχείου;

Το inode #3666 είναι το 2ο inode του BG #2 (1832 inodes per BG) (τα inodes ξεκινάνε από το #1)

Ξέρουμε από πριν ότι το inode table του BG #2 είναι στο block #16387.

Διαβάζουμε το 2ο inode του inode table και βρίσκουμε το 1ο block του αρχείου.











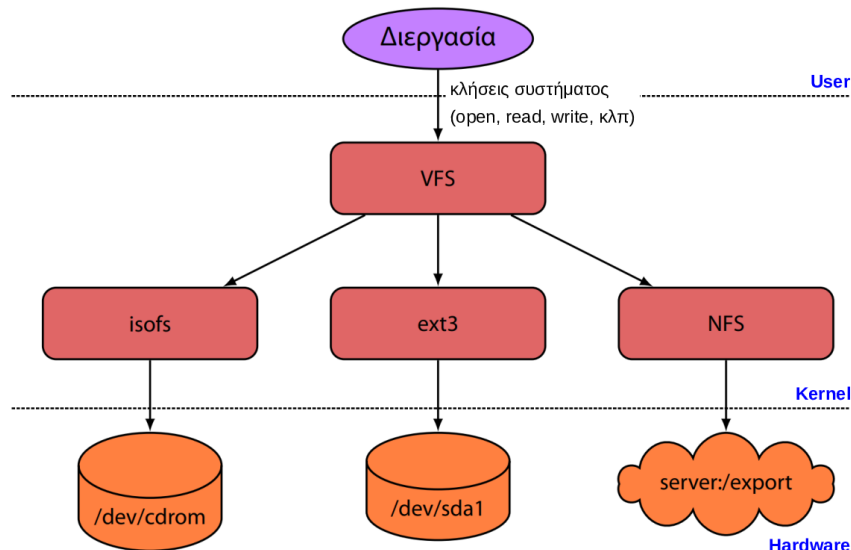
# Linux VFS

## 2ο μέρος της άσκησης

- Στο 1ο μέρος της άσκησης θα δούμε:
  - Το ext2 ΣΑ με τη χρήση εργαλείων του userspace.
  - Πώς είναι οργανωμένα τα δεδομένα σε έναν δίσκο που περιέχει ext2 ΣΑ.
  - Πώς μπορούμε με εργαλεία του userspace να αναζητήσουμε αρχεία.
  - Πώς μπορούμε με εργαλεία του userspace να εντοπίσουμε τα δεδομένα κάποιου αρχείου.
  - ...
- Στο 2ο μέρος θα δούμε:
  - Τα ΣΑ απο τη μεριά του πυρήνα του Linux.
  - Πώς υποστηρίζει το Linux πολλαπλά συστήματα αρχείων (VFS).
  - Πώς μπορείτε να προσθέσετε το δικό σας σύστημα αρχείων.
  - ext2-lite: μία πολύ απλούστερη έκδοση του ext2.
    - Ο κώδικας είναι κώδικας ext2 αλλά έχουμε αφαιρέσει τεράστιο μέρος ώστε να αποφύγουμε την πολυπλοκότητα.
    - Σας δίνουμε το μεγαλύτερο κομμάτι κώδικα και πρέπει απλά να συμπληρώσετε μικρά κομμάτια.
    - Στόχος είναι να εξοικειωθείτε με τον κώδικα του VFS και όχι να υλοποιήσετε κάποιο production-level ΣΑ.
    - Θέλουμε να επικεντρωθείτε στα κομμάτια του VFS και όχι σε αλγοριθμικές λεπτομέρειες που μπορεί να περιλαμβάνει η πλήρης υλοποίηση ενός ΣΑ.

# Linux VFS

- Η διεπαφή του πυρήνα ανάμεσα στο χρήστη και τα συστήματα αρχείων.
- Όλες οι κλήσεις συστήματος που αφορούν αρχεία περνάνε από το VFS.
- Υποστηρίζει μεγάλο αριθμό συστημάτων αρχείων και μπορούν «εύκολα» να προστεθούν νέα.



# Linux VFS: βασικές δομές

- **struct super\_block**
  - Κρατάει πληροφορία σχετικά με κάποιο προσαρτημένο σύστημα αρχείων.
- **struct inode**
  - Κρατάει πληροφορία σχετική με κάποιο αρχείο (κανονικό αρχείο, κατάλογο, ειδικό αρχείο, ...).
- **struct file**
  - Κρατάει πληροφορία σχετική με την αλληλεπίδραση κάποιας διεργασίας και ενός ανοιχτού αρχείου.
- **struct dentry**
  - Κρατάει πληροφορία σχετική με τη σύνδεση κάποιου ονόματος εντός κάποιου καταλόγου με ένα αρχείο (inode).

## Linux VFS: struct super\_block

<https://elixir.bootlin.com/linux/v5.10/source/include/linux/fs.h#L1416>

```
struct super_block {
    ...
    unsigned long                s_blocksize;
    ...
    const struct super_operations *s_op;
    ...
    struct dentry                *s_root;
    ...
    void                          *s_fs_info;
    ...
}
```

# Linux VFS: struct inode

<https://elixir.bootlin.com/linux/v5.10/source/include/linux/fs.h#L1416>

```
struct inode {  
    umode_t          i_mode;  
    kuid_t           i_uid;  
    kgid_t           i_gid;  
    ...  
    const struct inode_operations *i_op;  
    struct super_block *i_sb;  
    struct address_space *i_mapping;  
  
    unsigned long    i_ino;  
    unsigned int     i_nlink;  
  
    loff_t i_size;  
    struct timespec64 i_atime;  
    struct timespec64 i_mtime;  
    struct timespec64 i_ctime;  
    ...  
    const struct file_operations *i_fop;  
    ...  
}
```

# Linux VFS: struct file

<https://elixir.bootlin.com/linux/v5.10/source/include/linux/fs.h#L916>

```
struct file {
    ...
    struct path          f_path;
    struct inode         *f_inode;
    const struct file_operations *fop;
    ...
    loff_t               f_pos;
    ...
    void                *private_data;
    ...
}
```

# Linux VFS: struct dentry

<https://elixir.bootlin.com/linux/v5.10/source/include/linux/dcache.h#L89>

```
struct dentry {  
    ...  
    struct dentry    *d_parent;  
    struct qstr       d_name;  
    struct inode      *d_inode;  
    ...  
}
```



# Linux VFS: Προσθήκη/Διαγραφή τύπου ΣΑ

Για την προσθήκη/διαγραφή ενός νέου τύπου συστήματος αρχείων χρειάζεται να ορίσουμε μία μεταβλητή τύπου:

- `struct file_system_type`

Και να χρησιμοποιήσουμε τις παρακάτω συναρτήσεις του πυρήνα:

- `int register_filesystem(struct file_system_type *)`
- `int unregister_filesystem(struct file_system_type *)`

Ο πυρήνας διατηρεί μια λίστα με τους διαθέσιμους τύπους ΣΑ στη μεταβλητή `file_systems` (<https://elixir.bootlin.com/linux/v5.10/source/fs/filesystems.c#L72>)

# Linux VFS: Προσθήκη/Διαγραφή τύπου ΣΑ

<https://elixir.bootlin.com/linux/v5.10/source/include/linux/fs.h#L2228>

```
struct file_system_type {  
    const char *name;  
    int fs_flags;  
  
    ...  
    struct dentry *(*mount)(struct file_system_type *, int, const char *, void *);  
    void          (*kill_sb)(struct super_block *);  
  
    ...  
}
```

# Linux VFS: Προσθήκη/Διαγραφή τύπου ΣΑ

Απο το userspace μπορούμε να δούμε τη λίστα με τους διαθέσιμους τύπους ΣΑ μέσω του /proc/filesystems:

```
user@utopia:~/$ cat /proc/filesystems
```

```
nodev    sysfs
```

```
nodev    tmpfs
```

```
ext2
```

```
ext3
```

```
ext4
```

```
ext2-lite
```

# Linux VFS: Προσάρτηση ΣΑ

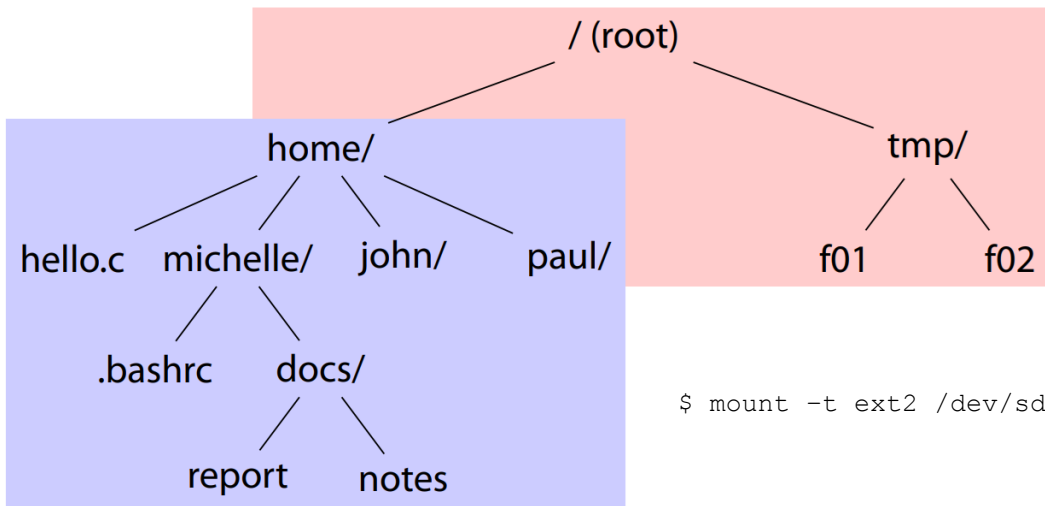
- Η προσάρτηση ενός ΣΑ γίνεται με την κλήση συστήματος `mount`
- Ο τύπος του ΣΑ δίνεται είτε σαν όρισμα στην `mount` είτε εντοπίζεται αυτόματα απο τον πυρήνα για συγκεκριμένα ΣΑ.
- Με βάση τον τύπο του ΣΑ ο πυρήνας βρίσκει το αντίστοιχο `file_system_type` και καλεί τη συνάρτηση στην οποία δείχνει το πεδίο `.mount`
- Η συνάρτηση αυτή είναι διαφορετική για κάθε τύπο ΣΑ και δημιουργεί και αρχικοποιεί κατάλληλα το αντίστοιχο `super_block` για το ΣΑ που προσαρτάται.
- Μετά την προσάρτηση ο πυρήνας χρησιμοποιεί αυτό το `super_block` για να εκτελέσει τις λειτουργίες που χρειάζεται.

# Προσάρτηση ΣΑ (επανάληψη)

(mount syscall)

Χρειάζονται:

- Σημείο προσάρτησης (mountpoint)
- Συσκευή αποθήκευσης (σκληρός δίσκος, flash, ...)



```
$ mount -t ext2 /dev/sdc1 /home/
```

# Linux VFS: Προσάρτηση ΣΑ

- Μετά την προσάρτηση ενός ΣΑ ο πυρήνας μπορεί να εντοπίσει το root inode και απο εκεί να χρησιμοποιήσει τις inode\_operations για να εκτελέσει όποια λειτουργία χρειάζεται.
- Απο το root inode του ΣΑ και μέσω του πεδίου i\_ops του το VFS μπορεί να κάνει για παράδειγμα:
  - Δημιουργία νέου inode εντός κάποιου καταλόγου
  - Αναζήτηση για κάποιο όνομα αρχείου εντός κάποιου καταλόγου
  - Δημιουργία νέου καταλόγου
  - Διαγραφή αρχείων/καταλόγων
  - ...

# Πηγές - Η δομή του ext2

<https://wiki.osdev.org/Ext2>

<https://www.nongnu.org/ext2-doc/>

<http://www.science.smith.edu/~nhowe/262/oldlabs/ext2.html>

# Πηγές - Wikipedia

ΣΑ γενικά: [https://en.wikipedia.org/wiki/File\\_system](https://en.wikipedia.org/wiki/File_system)

ΣΑ γενικά: <https://tldp.org/LDP/sag/html/filesystems.html>

Λίστα ΣΑ: [https://en.wikipedia.org/wiki/List\\_of\\_file\\_systems](https://en.wikipedia.org/wiki/List_of_file_systems)

Σύγκριση ΣΑ: [https://en.wikipedia.org/wiki/Comparison\\_of\\_file\\_systems](https://en.wikipedia.org/wiki/Comparison_of_file_systems)

Το ΣΑ ext2: <https://en.wikipedia.org/wiki/Ext2>

Το πακέτο εφαρμογών e2fsprogs: <https://en.wikipedia.org/wiki/E2fsprogs>



# Πηγές - Linux VFS

<https://terenceli.github.io/%E6%8A%80%E6%9C%AF/2019/02/23/linux-system-call-mount>

<http://pages.cpsc.ucalgary.ca/~crwth/programming/VFS/VFS.php>

<https://elixir.bootlin.com/linux/v5.10/source/Documentation/filesystems/vfs.rst>

# Βιβλιογραφία

- **Understanding the Linux Kernel, 3<sup>rd</sup> Edition, Daniel P. Bovet & Marco Cesati**
  - Καλύπτει αρκετά παλιά έκδοση του πυρήνα (v2.6) αλλά τα βασικά στοιχεία έχουν μείνει αναλλοίωτα.
  - Τα βασικά κεφάλαια που ασχολούνται με τα ΣΑ είναι τα:
    - Κεφάλαιο 12: The Virtual Filesystem
    - Κεφάλαιο 15: The Page Cache
    - Κεφάλαιο 16: Accessing Files
    - Κεφάλαιο 18: The Ext2 and Ext3 Filesystems