Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εργαστήριο Λειτουργικών Συστημάτων
Αναφορά στην τρίτη εργαστηριακή άσκηση:
Συστήματα Αρχείων σε Περιβάλλον Linux

Ομάδα oslab 37
Παπαδόπουλος Κωνσταντίνος, Α.Μ.: 03120152
Μέη Αρετή, Α.Μ.:03120062

Στην παρούσα αναφορά, θα αναλυθούν οι απαντήσεις που δόθηκαν για το πρώτο μέρος της τρίτης εργαστηριακής άσκησης.

Η εικόνα fsdisk1.img

1. Τροποποιήστε κατάλληλα το αρχείο utopia.sh ώστε να προσθέσετε στην εικονική μηχανή utopia έναν επιπλέον δίσκο για την εικόνα fsdisk1.img. Ποια είναι η προσθήκη που κάνατε; Ποια συσκευή στο utopia είναι αυτή που μόλις προσθέσατε;

Συμπληρώσαμε στο αρχείο utopia.sh τις επόμενες 3 γραμμές, προκειμένου να προσθέσουμε στο virtual machine utopia έναν επιπλέον δίσκο για τις 3 εικόνες fsdisk1.img, fsdisk2.img, fsdisk3.img (και για τα επόμενα ερωτήματα):

```
-drive file=./fsdisk1-7c2b425a6.img,format=raw,if=virtio \
-drive file=./fsdisk2-a0173283d.img,format=raw,if=virtio \
-drive file=./fsdisk3-982902777.img,format=raw,if=virtio
```

Χρησιμοποιούμε την εντολή lsblk, η οποία κάνει display τις πληροφορίες για τα block devices:

```
root@utopia:~# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
fd0      2:0    1    4K  0 disk
sr0     11:0    1 1024M  0 rom
vda    254:0    0   11G  0 disk
└─vda1 254:1    0   11G  0 part /
vdb    254:16   0   50M  0 disk
vdc    254:32   0   20M  0 disk
vdd    254:48   0   20M  0 disk
```

Όπως βλέπουμε, προστέθηκαν οι συσκευές vdb, vdc, vdd.

2. Τι μέγεθος έχει ο δίσκος που προσθέσατε στο utopia;

- Θα χρησιμοποιήσουμε αρχικά την εντολή blkid:

```
root@utopia:~# blkid
/dev/vda1: UUID="ce8cc13a-d361-471d-adac-3061bf165cf4" BLOCK_SIZE="4096" TYPE="e
xt4" PARTUUID="46795e59-01"
/dev/vdb: LABEL="fsdisk1.img" UUID="c63028e5-711b-410d-a263-e7ca2b15a8d3" BLOCK_
SIZE="1024" TYPE="ext2"
/dev/vdc: LABEL="fsdisk2.img" UUID="d1266ad1-dae1-4275-8136-a29a4dfc9d1f" BLOCK_
SIZE="1024" TYPE="ext2"
/dev/vdd: LABEL="fsdisk3.img" UUID="19032143-52ce-4917-8ec5-991c89ee421b" BLOCK_
SIZE="1024" TYPE="ext2"
root@utopia:~#
```

Βλέπουμε πως ο δίσκος έχει μέγεθος 50MB.

- Χρησιμοποιώντας το hexdump:

```
root@utopia:~# hexdump -s 1024 -n 1024 -C /dev/vdb
00000400  18 32 00 00 00 c8 00 00  00 0a 00 00 90 c1 00 00  |.2..............|
00000410  0a 32 00 00 01 00 00 00  00 00 00 00 00 00 00 00  |.2..............|
00000420  00 20 00 00 00 20 00 00  28 07 00 00 f6 01 98 65  |. ... ..(......e|
00000430  f6 01 98 65 03 00 ff ff  53 ef 00 00 01 00 00 00  |...e....S.......|
00000440  e4 7a 78 65 00 00 00 00  00 00 00 00 01 00 00 00  |.zxe............|
00000450  00 00 00 00 0b 00 00 00  80 00 00 00 00 00 00 00  |................|
00000460  00 00 00 00 00 00 00 00  c6 30 28 e5 71 1b 41 0d  |.........0(.q.A.|
00000470  a2 63 e7 ca 2b 15 a8 d3  66 73 64 69 73 6b 31 2e  |.c..+...fsdisk1.|
00000480  69 6d 67 00 00 00 00 00  2f 63 73 6c 61 62 2d 62  |img...../cslab-b|
00000490  75 6e 6b 65 72 00 00 00  00 00 00 00 00 00 00 00  |unker...........|
000004a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000004e0  00 00 00 00 00 00 00 00  00 00 00 00 a7 f1 37 f1  |..............7.|
000004f0  07 8d 4b b9 9b 0d e1 b1  22 bc 50 38 01 00 00 00  |..K.....".P8....|
00000500  0c 00 00 00 00 00 00 00  e4 7a 78 65 00 00 00 00  |.........zxe....|
00000510  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000560  01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000570  00 00 00 00 00 00 00 00  0e 00 00 00 00 00 00 00  |................|
00000580  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000800
```

(-s: σημείο από όπου ξεκινά το hexdump, -n: πόσα bytes θα κάνει display)
Θα έχουμε λοιπόν πως το μέγεθος του δίσκου είναι:
(size) = (block size)*(number of blocks) = (4ο έως 7ο byte, όπου δηλώνεται το block size)*(24ο έως 27ο byte όπου δηλώνεται το total number of blocks) = 51200*1024 = 52428800 bytes = 50MB.

3. Τι σύστημα αρχείων περιέχει;

- Χρησιμοποιώντας την εντολή blkid, βλέπουμε πως χρησιμοποιείται το σύστημα ext2.
- Χρησιμοποιώντας το hexdump, θα κοιτάξουμε τα bytes 56 και 57 του superblock, όπου έχουν την τιμή 0xef53, το οποίο βρήκαμε πως είναι το signature του ext2.

4. Πότε ακριβώς δημιουργήθηκε αυτό το σύστημα αρχείων; Δείξτε τη χρονοσφραγίδα [timestamp].

- Η αρχική μας σκέψη ήταν να χρησιμοποιήσουμε την εντολή stat για το vdb:

```
root@utopia:~# stat vdb
stat: cannot statx 'vdb': No such file or directory
root@utopia:~# stat /dev/vdb
  File: /dev/vdb
  Size: 0               Blocks: 0          IO Block: 4096   block special file
Device: 5h/5d   Inode: 207         Links: 1     Device type: fe,10
Access: (0660/brw-rw----)  Uid: (    0/    root)   Gid: (    6/    disk)
Access: 2024-01-02 21:56:34.492000000 +0200
Modify: 2024-01-02 21:56:34.492000000 +0200
Change: 2024-01-02 21:56:34.492000000 +0200
 Birth: -
root@utopia:~#
```

Βλέπουμε πως δεν επιτρέπει να δούμε το timestamp της δημιουργίας του συστήματος αρχείου. Δοκιμάσαμε έπειτα το dumpe2fs:

```
root@utopia:~# dumpe2fs /dev/vdb
dumpe2fs 1.46.2 (28-Feb-2021)
Filesystem volume name:   fsdisk1.img
Last mounted on:          /cslab-bunker
Filesystem UUID:          c63028e5-711b-410d-a263-e7ca2b15a8d3
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:              49552
Free inodes:              12810
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:       Tue Dec 12 17:23:16 2023
Last mount time:          Fri Jan  5 15:19:50 2024
Last write time:          Fri Jan  5 15:19:50 2024
Mount count:              3
Maximum mount count:      -1
Last checked:             Tue Dec 12 17:23:16 2023
Check interval:           0 (<none>)
Lifetime writes:          14 kB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
Default directory hash:   half_md4
Directory Hash Seed:      a7f137f1-078d-4bb9-9b0d-e1b122bc5038


Group 0: (Blocks 1-8192)
  Primary superblock at 1, Group descriptors at 2-2
  Block bitmap at 3 (+2)
  Inode bitmap at 4 (+3)
  Inode table at 5-233 (+4)
  7944 free blocks, 1821 free inodes, 2 directories
  Free blocks: 248-1024, 1026-8192
  Free inodes: 12-1832
Group 1: (Blocks 8193-16384)
  Backup superblock at 8193, Group descriptors at 8194-8194
  Block bitmap at 8195 (+2)
  Inode bitmap at 8196 (+3)
```

Και βλέπουμε πως η απάντηση στην ερώτηση μας είναι το πεδίο “Filesystem created: Tue Dec 12 17:23:16 2023”.

- Χρησιμοποιώντας το hexdump, θα χρειαστεί να ψάξουμε στα bytes 264-267 του superblock, τα οποία έχουν την τιμή e4 7a 78 65, το οποίο μεταφράζεται σε posix time : Tuesday, December 12, 2023 5:23:16 PM

5. Πότε ακριβώς προσαρτήθηκε τελευταία φορά; Δείξτε τη χρονοσφραγίδα.
   - Και πάλι από το dumpe2fs, βλέπουμε πως η απάντηση στο ερώτημά μας είναι το πεδίο “Last mount time: Fri Jan 5 15:19:50 2024”
   - Μέσω του hexdump, θα εστιάσουμε στα bytes 44-47, τα οποία έχουν την τιμή 0x659801f6, σε POSIX time: “Friday, January 5, 2024 3:19:50 PM GMT+02:00”.

6. Σε ποιο μονοπάτι προσαρτήθηκε τελευταία φορά;
   - Μέσω dumpe2fs: Πεδίο “Last mounted on: /cslab-bunker”
   - Μέσω hexdump: Κοιτάμε στα bytes 136-199, τα οποία μεταφράζονται στο path /mnt.

7. Πότε ακριβώς τροποποιήθηκε τελευταία φορά; Δείξτε τη χρονοσφραγίδα.
   - Μέσω dumpe2fs: Πεδίο “Last write time: Fri Jan 5 15:19:50 2024”
   - Μέσω hexdump: Κοιτάμε στα bytes 48-51, τα οποία μεταφράζονται στο POSIX time: Friday, January 5, 2024 3:19:50 PM GMT+02:00

8. Τι είναι το μπλοκ σε ένα σύστημα αρχείων;
Τα δεδομένα σε ένα σύστημα αρχείων EXT2 οργανώνονται σε blocks, τα οποία είναι σταθερού μεγέθους και αναλαμβάνουν τη μονάδα διαχείρισης αποθηκευτικού χώρου από το σύστημα αρχείων.
Τα blocks χρησιμοποιούνται για να αποθηκεύονται τα δεδομένα των αρχείων και των καταλόγων, καθώς και πληροφορίες διαχείρισης, όπως οι δείκτες προς άλλα blocks ή δείκτες για τον επόμενο block σε μια σειρά από blocks.

9. Τι μέγεθος μπλοκ [block size] χρησιμοποιεί αυτό το σύστημα αρχείων;
   - Μέσω dumpe2fs: Πεδίο “Block size: 1024”.
   - Μέσω hexdump: Κοιτάμε στα bytes 24-27, που μεταφράζονται σε 1024.

10. Τι είναι το inode σε ένα σύστημα αρχείων;
Το inode είναι μια δομή δεδομένων που χρησιμοποιείται για την αποθήκευση πληροφοριών σχετικά με ένα αρχείο ή έναν κατάλογο στο σύστημα αρχείων. Κάθε αρχείο ή κατάλογος στο σύστημα αρχείων έχει ένα μοναδικό inode που το αναπαριστά. Το inode περιέχει πληροφορίες όπως: Τύπος αρχείου (κανονικό αρχείο, κατάλογος, συμβολικός σύνδεσμος κλπ.), άδειες πρόσβασης (permissions) που καθορίζουν ποιος χρήστης ή ομάδα χρηστών μπορεί να διαβάσει, γράψει ή εκτελέσει το αρχείο, μέγεθος του αρχείου σε bytes, χρόνοι δημιουργίας, τροποποίησης και πρόσβασης του αρχείου, δείκτες προς τα blocks που περιέχουν τα πραγματικά δεδομένα του αρχείου.

11. Τι μέγεθος έχει το inode σε αυτό το σύστημα αρχείων;
- Μέσω dumpe2fs: Πεδίο “Inode size: 128”.
- Μέσω hexdump: Κοιτάμε στα bytes 88-89, που μεταφράζονται σε 128.

12. Πόσα διαθέσιμα μπλοκ και πόσα διαθέσιμα inodes υπάρχουν σε αυτό το σύστημα αρχείων;
- Μέσω dumpe2fs: Πεδίο “Free inodes:12810, Free Blocks: 49952”.
- Μέσω hexdump: Κοιτάμε στα bytes 12-15 για το πεδίο των free blocks, που μεταφράζεται σε 49952, στα επόμενα 4 bytes(16-19) για τα free inodes, που μεταφράζονται σε 12810.

13. Τι είναι το superblock στο σύστημα αρχείων ext2;
Το superblock αποτελεί την πρώτη δομή δεδομένων στον δίσκο και περιέχει πληροφορίες που αφορούν το συνολικό σχήμα και τη διαχείριση του EXT2 συστήματος αρχείων. Ο βασικός ρόλος του superblock είναι να παρέχει κρίσιμες πληροφορίες για το σύστημα αρχείων, συμπεριλαμβανομένων:

1.Μέγεθος του συστήματος αρχείων: Το superblock περιέχει πληροφορίες σχετικά με το συνολικό μέγεθος του συστήματος αρχείων, συμπεριλαμβανομένου του συνολικού αριθμού των blocks και τον αριθμό των inodes που διατίθενται.

2.Μέγεθος block: Ορίζει το μέγεθος των blocks στο σύστημα αρχείων, το οποίο είναι σταθερό και καθορίζεται κατά τη δημιουργία του συστήματος.

3.Αποθηκευτική διάταξη: Περιέχει πληροφορίες σχετικά με τον τρόπο οργάνωσης των blocks στο δίσκο, όπως η διάταξη των διαφορετικών ζωνών αποθήκευσης.

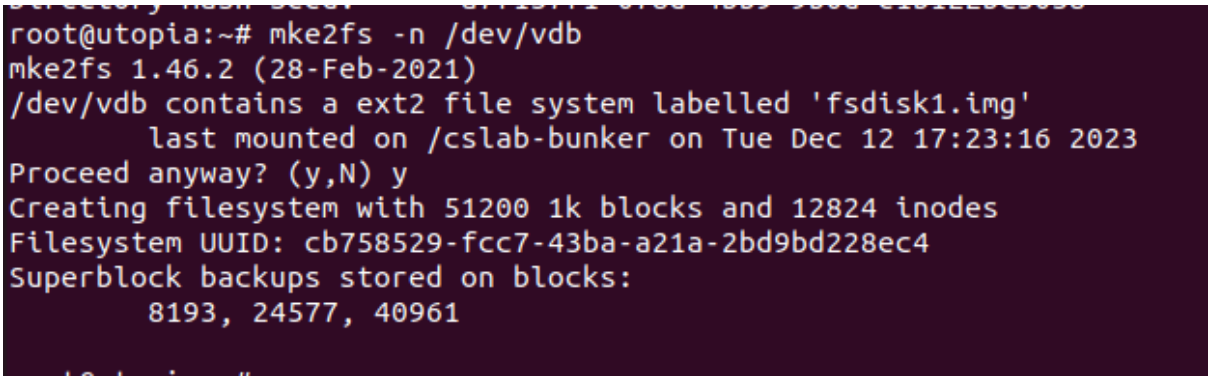
4.Λεπτομέρειες σχετικά με το mount: Περιλαμβάνει πληροφορίες σχετικά με το πότε και πώς πρέπει να γίνει το mount του συστήματος αρχείων.

Κατά την αρχικοποίηση του συστήματος, δημιουργείται μόνο ένα superblock, αλλά υπάρχουν αντίγραφα του σε διαφορετικές θέσεις στον δίσκο για λόγους ανάκτησης δεδομένων σε περίπτωση προβλημάτων.

14. Πού βρίσκεται μέσα στον δίσκο σε ένα σύστημα αρχείων ext2;
Στο σύστημα αρχείων ext2, το superblock είναι σταθερού μεγέθους 1024 bytes και βρίσκεται στο 1ο block κάθε block group.

15. Για ποιο λόγο έχει νόημα να υπάρχουν εφεδρικά αντίγραφα του superblock στο σύστημα αρχείων ext2;
Για λόγους ανάκτησης δεδομένων σε περίπτωση προβλημάτων.

16. Σε ποια μπλοκ βρίσκονται αποθηκευμένα εφεδρικά αντίγραφα του superblock σε αυτό το σύστημα αρχείων;
Μέσω της εντολής mke2fs, βλέπουμε:



Άρα στα blocks 8193, 24577, 40961.

17. Τι είναι ένα block group στο σύστημα αρχείων ext2;
Το block group είναι μια οργανωτική μονάδα στο σύστημα αρχείων EXT2. Ο δίσκος στο EXT2 σύστημα αρχείων είναι χωρισμένος σε πολλαπλά block groups, κάθε ένα από τα οποία περιλαμβάνει ένα σύνολο από blocks. Κάθε block group περιέχει μια πληθώρα διαφορετικών δομών δεδομένων που αφορούν τα αρχείαπου αποθηκεύονται στον δίσκο. Αυτές οι δομές περιλαμβάνουν:

1.Το block bitmap: Ένας πίνακας bit που δείχνει ποια blocks στο block group είναι κενά και ποια χρησιμοποιούνται.

2.Το inode bitmap: Ένας πίνακας bit που δείχνει ποια inodes στο block group είναι κενά και ποια είναι αντιστοιχισμένα σε αρχεία ή καταλόγους.

3.Τα inodes: Τα inodes περιέχουν πληροφορίες σχετικά με τα αρχεία και τους καταλόγους που αποθηκεύονται στο block group, όπως δικαιώματα πρόσβασης, χρόνος δημιουργίας, μέγεθος και δείκτες προς τα blocks που περιέχουν τα δεδομένα των αρχείων.

4.Τα data blocks: Τα blocks που περιέχουν τα δεδομένα των αρχείων και των καταλόγων.

18. Πόσα block groups έχει ένα σύστημα αρχείων ext2 και πώς κατανέμονται;
Ο αριθμός των συνολικών Block groups σε ένα ext2 σύστημα αρχείων εξαρτάται από το ext2 partition καθώς και το size, και είναι όλα του ίδιου μεγέθους εκτός από το τελευταίο που είναι το μικρότερο.

19. Πόσα block groups περιέχει αυτό το σύστημα αρχείων;

- Χρησιμοποιούμε και πάλι το dumpe2fs, ενώ δίνουμε το option “grep ’Group’ ” για να βρούμε τις πληροφορίες που αφορούν στα block groups, και μετράμε 7:

```
root@utopia:~# dumpe2fs /dev/vdb | grep "Group"
dumpe2fs 1.46.2 (28-Feb-2021)
Group 0: (Blocks 1-8192)
  Primary superblock at 1, Group descriptors at 2-2
Group 1: (Blocks 8193-16384)
  Backup superblock at 8193, Group descriptors at 8194-8194
Group 2: (Blocks 16385-24576)
  Backup superblock at 16385, Group descriptors at 16386-16386
Group 3: (Blocks 24577-32768)
  Backup superblock at 24577, Group descriptors at 24578-24578
Group 4: (Blocks 32769-40960)
  Backup superblock at 32769, Group descriptors at 32770-32770
Group 5: (Blocks 40961-49152)
  Backup superblock at 40961, Group descriptors at 40962-40962
Group 6: (Blocks 49153-51199)
  Backup superblock at 49153, Group descriptors at 49154-49154
```

- Μέσω hexdump:
  $\frac{(bytes \text{ στις θέσεις } 0-3, \text{ όπου κωδικοποιούνται τα } inode\ counts)}{(bytes \text{ στις θέσεις } 40-43, \text{ όπου κωδικοποιούνται τα } inodes\ per\ group)}$ = 7.

20. Τι είναι ο block group descriptor στο σύστημα αρχείων ext2;
Οι block group descriptors είναι δομές δεδομένων που περιέχουν πληροφορίες σχετικά με κάθε block group στο σύστημα αρχείων EXT2. Κάθε block group στο EXT2 έχει έναν αντίστοιχο block group descriptor που περιέχει σημαντικές πληροφορίες για το εκάστοτε block group.

Οι βασικές πληροφορίες που περιέχονται σε ένα block group descriptor περιλαμβάνουν:

1.Το δείκτη στο πρώτο block του συγκεκριμένου block group, το οποίο περιλαμβάνει το superblock, το block bitmap, το inode bitmap και τους πρώτους inodes.

2.Τον αριθμό των διαθέσιμων inodes στο συγκεκριμένο block group.

3.Τον συνολικό αριθμό των blocks στο συγκεκριμένο block group.

4.Δείκτες block bitmap και inode bitmap: Δείχνουν στη θέση των bitmap του block και των inodes στο συγκεκριμένο block group.

Κατά τη λειτουργία του συστήματος αρχείων, οι block group descriptors χρησιμοποιούνται για να προσδιορίσουν τη δομή και την τοποθεσία των δεδομένων σε κάθε block group.

21. Για ποιο λόγο έχει νόημα να υπάρχουν εφεδρικά αντίγραφα των block group descriptors στο σύστημα αρχείων ext2;
Η ύπαρξη εφεδρικών αντιγράφων των block group descriptors στο σύστημα αρχείων EXT2 είναι σημαντική για την ανθεκτικότητα και την αξιοπιστία του συστήματος αρχείων. Αυτά τα εφεδρικά αντίγραφα παρέχουν αντίγραφα των πληροφοριών των block group descriptors σε περίπτωση που τα αρχικά block group descriptors καταστραφούν ή χαθούν λόγω βλάβης του δίσκου ή άλλων ανεπιθύμητων συμβάντων.

22. Σε ποια μπλοκ βρίσκονται αποθηκευμένα εφεδρικά αντίγραφα των block group descriptors σε αυτό το σύστημα αρχείων;
Θα το ψάξουμε μέσω του dumpe2fs και πάλι:

```
root@utopia:~# dumpe2fs /dev/vdb | grep "Group"
dumpe2fs 1.46.2 (28-Feb-2021)
Group 0: (Blocks 1-8192)
  Primary superblock at 1, Group descriptors at 2-2
Group 1: (Blocks 8193-16384)
  Backup superblock at 8193, Group descriptors at 8194-8194
Group 2: (Blocks 16385-24576)
  Backup superblock at 16385, Group descriptors at 16386-16386
Group 3: (Blocks 24577-32768)
  Backup superblock at 24577, Group descriptors at 24578-24578
Group 4: (Blocks 32769-40960)
  Backup superblock at 32769, Group descriptors at 32770-32770
Group 5: (Blocks 40961-49152)
  Backup superblock at 40961, Group descriptors at 40962-40962
Group 6: (Blocks 49153-51199)
  Backup superblock at 49153, Group descriptors at 49154-49154
```

23. Τι είναι το block bitmap και τι το inode bitmap; Πού βρίσκονται μέσα στον δίσκο;
Τόσο το block bitmap όσο και το inode bitmap είναι δομές δεδομένων που χρησιμοποιούνται στο σύστημα αρχείων EXT2 για τη διαχείριση των blocks και των inodes αντίστοιχα. Αυτά τα bitmaps παρέχουν μια εικόνα της διαθεσιμότητας των blocks και των inodes σε κάθε block group του συστήματος αρχείων.

1.Block Bitmap: Το block bitmap είναι ένας πίνακας bit που αναπαριστά κάθε block σε ένα block group. Κάθε bit στο block bitmap αντιστοιχεί σε ένα block στο block group, όπου το 0 δείχνει ότι το αντίστοιχο block είναι διαθέσιμο και το 1 δείχνει ότι το block έχει ήδη καταχωρηθεί. Ο block bitmap επιτρέπει στο σύστημα αρχείων να γνωρίζει ποια blocks είναι ελεύθερα για αποθήκευση δεδομένων και ποια έχουν ήδη χρησιμοποιηθεί.

2.Inode Bitmap: Το inode bitmap είναι ένας πίνακας bit που αναπαριστά κάθε inode σε ένα block group. Κάθε bit στο inode bitmap αντιστοιχεί σε ένα inode στο block group, όπου το 0 δείχνει ότι το αντίστοιχο inode είναι διαθέσιμο και το 1 δείχνει ότι το inode έχει ήδη καταχωρηθεί. Ο inode bitmap επιτρέπει στο σύστημα αρχείων να γνωρίζει ποια inodes είναι διαθέσιμα για χρήση και ποια έχουν ήδη αντιστοιχιστεί σε αρχεία ή καταλόγους.

Τα block bitmap και inode bitmap βρίσκονται σε κάθε block group του συστήματος αρχείων EXT2. Συνήθως το block bitmap βρίσκεται αμέσως μετά το superblock, ενώ το inode bitmap ακολουθεί μετά το block bitmap στον ίδιο block group.

24. Τι είναι τα inode tables; Πού βρίσκονται μέσα στον δίσκο;
Τα inode tables είναι δομές δεδομένων στο σύστημα αρχείων EXT2 που περιέχουν τις πληροφορίες των inodes. Κάθε inode αντιπροσωπεύει ένα αρχείο ή έναν κατάλογο στο σύστημα αρχείων και περιέχει πληροφορίες όπως τα δικαιώματα πρόσβασης, τον τύπο του

αρχείου, τους δείκτες προς τα blocks που περιέχουν τα δεδομένα του αρχείου και άλλες σχετικές πληροφορίες.

Τα inode tables περιέχουν τις εγγραφές για όλα τα inodes που ανήκουν σε ένα συγκεκριμένο block group του συστήματος αρχείων. Η δομή του inode table εξαρτάται από την υλοποίηση του EXT2 συστήματος αρχείων, αλλά συνήθως περιλαμβάνει έναν συνεχόμενο πίνακα από inodes, κάθε ένα από τα οποία έχει σταθερό μέγεθος.

Τα inode tables βρίσκονται σε κάθε block group του δίσκου. Ακριβώς μετά τα bitmaps στη δομή του block group, τα inode tables περιέχουν τις εγγραφές των inodes για τα αρχεία και τους καταλόγους που ανήκουν σε αυτό το συγκεκριμένο block group.

25. Τι πεδία περιέχει το κάθε inode; Πού αποθηκεύεται μέσα στον δίσκο;
Το κάθε inode στο σύστημα αρχείων EXT2 περιλαμβάνει διάφορα πεδία που περιγράφουν τα χαρακτηριστικά του αντίστοιχου αρχείου ή καταλόγου. Αυτά τα πεδία περιλαμβάνουν τα ακόλουθα:

1.Τύπος αρχείου (File Type): Ο τύπος του αρχείου, όπως κανονικό αρχείο, κατάλογος, σύνδεσμος ή ειδικό αρχείο.

2.Δικαιώματα (Permissions): Τα δικαιώματα πρόσβασης του αρχείου σε όλους τους χρήστες και ομάδες χρηστών.

3.Αριθμός συνδέσεων (Links Count): Ο αριθμός των συνδέσεων προς το αρχείο από διαφορετικούς καταλόγους.

4.Κάτοχος (Owner): Ο χρήστης που έχει δικαιώματα ιδιοκτησίας του αρχείου.

5.Ομάδα (Group): Η ομάδα που έχει δικαιώματα πρόσβασης στο αρχείο.

6.Μέγεθος (Size): Το μέγεθος του αρχείου σε bytes.

7.Ώρα τελευταίας πρόσβασης (Last Access Time): Η τελευταία φορά που ένας χρήστης έχει ανακτήσει το αρχείο.

8.Ώρα τελευταίας τροποποίησης (Last Modification Time): Η τελευταία φορά που το περιεχόμενο του αρχείου έχει τροποποιηθεί.

9.Ώρα τελευταίας αλλαγής (Last Change Time): Η τελευταία φορά που οι πληροφορίες του αρχείου (όπως τα δικαιώματα πρόσβασης) έχουν τροποποιηθεί.

10.Δείκτες blocks (Block Pointers): Δείκτες προς τα blocks που περιέχουν τα δεδομένα του αρχείου.

Το κάθε inode αποθηκεύεται μέσα στον δίσκο σε μια δομή που ονομάζεται inode table.

26. Πόσα μπλοκ και πόσα inodes περιέχει το κάθε block group σε αυτό το σύστημα αρχείων;
    - Με dumpe2fs: Τα πεδία “Blocks per group: 8192, Inodes per group: 1832”
    - Μέσω hexdump: Κοιτάζουμε τα bytes 32-35, που αντιστοιχούν στα blocks per group και μεταφράζονται σε 8192, καθώς και τα bytes 40-43, που αντιστοιχούν στα inodes per group και μεταφράζονται σε 1832.

Οι ερωτήσεις 27-33 θα απαντηθούν και θα αναλυθούν στις επόμενες γραμμές:

- Αρχικά χρησιμοποιώντας την εντολή stat, βρίσκουμε:

```
Inode: 9162   Type: regular    Mode:  0644   Flags: 0x0
Generation: 2739270588    Version: 0x00000001
User:     0   Group:     0   Size: 42
File ACL: 0
Links: 1   Blockcount: 2
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x65787ae4 -- Tue Dec 12 17:23:16 2023
atime: 0x65981516 -- Fri Jan  5 16:41:26 2024
mtime: 0x65787ae4 -- Tue Dec 12 17:23:16 2023
BLOCKS:
(0):1025
TOTAL: 1
```

Άρα στο inode 9162.
Γνωρίζουμε πως κάθε block group περιέχει 1832 inodes, άρα το αρχείο θα βρίσκεται στο 5ο block group και καταλαμβάνει μόνο 1 block, επομένως θα έχουμε ότι το μέγεθος του θα είναι ίσο με το block size = 1024.
Το block που περιέχει το τα δεδομένα του αρχείου /dir2/helloworld είναι το 1025.
Γνωρίζουμε ότι το Inode table βρίσκεται στο block 1, και ότι είναι το 3ο στη σειρά inode(3*128 < 1024), άρα το block που περιέχει το inode για το file είναι το:
1(πρώτο block) + 4*8192(blocks per group*4 groups) + 10 (superblock, gdt, block bitmap, inode bitmap) = 32779.

- Μέσω hexdump:
  Αρχικά, ξεκινάμε την αναζήτηση μέσω hexdump, χρησιμοποιώντας ως offset 2*1024, προκειμένου να πάμε στο block group 1. Βρίσκουμε που ξεκινά το inode table, και από εκεί θα πάρουμε το δεύτερο inode, το οποίο αντιστοιχεί πάντα στο root. Έτσι θα πάμε στο root inode, και από εκεί θα κοιτάξουμε τα bytes 40-43 τα οποία αντιστοιχούν στον pointer προς το data block για τα περιεχόμενα του “/”, που στην προκειμένη περίπτωση είναι το dir2:

```
root@utopia:~# hexdump -s 2048 -n 128 -C /dev/vdb
00000800  03 00 00 00 04 00 00 00  05 00 00 00 08 1f 1d 07  |................|
00000810  02 00 04 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000820  03 20 00 00 04 20 00 00  05 20 00 00 16 1f 27 07  |. ... ... ....'.|
00000830  01 00 04 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000840  03 40 00 00 04 40 00 00  05 40 00 00 17 1f 28 07  |.@...@...@....(.|
00000850  00 00 04 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000860  03 60 00 00 04 60 00 00  05 60 00 00 17 1f 28 07  |.`...`...`....(.|
00000870  00 00 04 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000880
root@utopia:~# hexdump -s 5248 -n 128 -C /dev/vdb
00001480  ed 41 00 00 00 04 00 00  e5 14 98 65 e4 7a 78 65  |.A.........e.zxe|
00001490  e4 7a 78 65 00 00 00 00  00 00 05 00 02 00 00 00  |.zxe............|
000014a0  00 00 00 00 02 00 00 00  ea 00 00 00 00 00 00 00  |................|
000014b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

```
root@utopia:~# hexdump -s 239616 -n 1024 -C /dev/vdb
0003a800  02 00 00 00 0c 00 01 00  2e 00 00 00 02 00 00 00  |................|
0003a810  0c 00 02 00 2e 2e 00 00  0b 00 00 00 14 00 0a 00  |................|
0003a820  6c 6f 73 74 2b 66 6f 75  6e 64 00 00 29 07 00 00  |lost+found..)...|
0003a830  0c 00 04 00 64 69 72 31  c9 23 00 00 c8 03 04 00  |....dir1.#......|
0003a840  64 69 72 32 00 00 00 00  00 00 00 00 00 00 00 00  |dir2............|
0003a850  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

Σε αυτό το σημείο βρισκόμαστε μέσα στο “/”, και ψάχνουμε το dir2.
Το ddir2 βρίσκεται στο inode 0x23c9 = 9161, και είμαστε στο 5ο block group, όπως βρήκαμε και παραπάνω με την χρήση του stat, άρα το offset για το hexdump θα είναι: 2*1024 + 5*block size = 2208:

```
root@utopia:~# hexdump -s 2208 -n 132 -C /dev/vdb
000008a0  03 a0 00 00 04 a0 00 00  05 a0 00 00 17 1f 26 07  |..............&.|
000008b0  01 00 04 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000008c0  03 c0 00 00 04 c0 00 00  05 c0 00 00 16 07 28 07  |..............(.|
000008d0  00 00 04 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000008e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000920  00 00 00 00                                       |....|
```

Από εδώ θα πάρουμε τα 8-11 bytes, για την αρχή του inode table του block group 5, και στην συνέχεια θα χρησιμοποιήσουμε πάλι το hexdump με κατάλληλο offest για το Inode που βρήκαμε σε αυτά τα Bytes:

```
root@utopia:~# hexdump -s$((40965*1024)) -n 128 -C /dev/vdb
02801400  ed 41 00 00 00 04 00 00  f7 14 98 65 e4 7a 78 65  |.A.........e.zxe|
02801410  e4 7a 78 65 00 00 00 00  00 00 02 00 02 00 00 00  |.zxe............|
02801420  00 00 00 00 02 00 00 00  f7 00 00 00 00 00 00 00  |................|
02801430  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
02801460  00 00 00 00 c5 b2 72 92  00 00 00 00 00 00 00 00  |......r.........|
02801470  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

Από εδώ, διαβάζουμε τα bytes 40-43, τα οποία αντιστοιχούν στον δείκτη προς το datablock που περιέχει τα δεδομένα(στην προκειμένη περίπτωση, το helloworld) και στην συνέχεια στησιμοποιούμε αυτό ως κατάλληλο offset του hexdump, προκειμένου να βρούμε το helloworld αρχείο:

```
root@utopia:~# hexdump -s$((247*1024)) -n 128 -C /dev/vdb
0003dc00  c9 23 00 00 0c 00 01 00  2e 00 00 00 02 00 00 00  |.#..............|
0003dc10  0c 00 02 00 2e 2e 00 00  ca 23 00 00 e8 03 0a 00  |.........#......|
0003dc20  68 65 6c 6c 6f 77 6f 72  6c 64 00 00 00 00 00 00  |helloworld......|
0003dc30  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

Από το τελευταίο hexdump, παρατηρούμε πως το inode του helloworld είναι το 0x23ca = 9162, όπως και περιμέναμε από το stat.

Η εικόνα fsdisk2.img

1. Συνδέστε την εικόνα του δίσκου στην εικονική μηχανή σας, όπως κάνατε και για την εικόνα fsdisk1.img και προσαρτήστε τη στον κατάλογο /mnt.
Το κάναμε στο πρώτο ερώτημα του πρώτου μέρους.

2. Χρησιμοποιήστε την εντολή touch για να δημιουργήσετε ένα νέο κενό αρχείο /file1 μέσα στο συγκεκριμένο σύστημα αρχείων. Βεβαιωθείτε ότι η εντολή σας αναφέρεται πράγματι στο συγκεκριμένο σύστημα αρχείων [σε ποιον κατάλογο το έχετε προσαρτήσει;], κι όχι στον ριζικό κατάλογο του συστήματος.

```
root@utopia:/# mount /dev/vdc /mnt
mount: /mnt: /dev/vdc already mounted on /mnt.
root@utopia:/# touch /mnt/file1
touch: cannot touch '/mnt/file1': No space left on device
root@utopia:/#
```

3. Πέτυχε η εντολή; Αν όχι, τι πρόβλημα υπήρξε;
Η εντολή δεν πετυχε και το πρόβλημα μας είναι ότι δεν υπάρχει αρκετός χώρος.
Έχουμε μάλλον ξεμείνει από Blocks ή από inodes!

```
root@utopia:/# dumpe2fs /dev/vdc
dumpe2fs 1.46.2 (28-Feb-2021)
Filesystem volume name:   fsdisk2.img
Last mounted on:          /cslab-bunker
Filesystem UUID:          d1266ad1-dae1-4275-8136-a29a4dfc9d1f
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:              5136
Block count:              20480
Reserved block count:     0
Free blocks:              19555
Free inodes:              0
First block:              1
Block size:               1024
Fragment size:            1024
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         1712
Inode blocks per group:   214
Filesystem created:       Tue Dec 12 17:23:17 2023
Last mount time:          Fri Jan  5 17:50:53 2024
Last write time:          Fri Jan  5 17:50:53 2024
Mount count:              2
Maximum mount count:      -1
Last checked:             Tue Dec 12 17:23:17 2023
Check interval:           0 (<none>)
Lifetime writes:          915 kB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
Default directory hash:   half_md4
Directory Hash Seed:      e01a1438-a9b7-4218-b892-e0e1ae23d891
```

Όπως υποθέσαμε έχουμε 0 free inodes, άρα δεν μπορούμε να δημιουργήσουμε νέα αρχεία. Το ΄το έχουμε ξεμείνει από διαθέσιμα inodes, Μπορούμε να το διαπιστώσουμε και με κατάλληλη χρήση του hexdump στο πηγαίνοντας sto superblock στο πεδίο “Free inodes”, στα bytes 16-19.

4. Ποια κλήση συστήματος προσπάθησε να τρέξει η touch, και με ποιον κωδικό λάθους απέτυχε; Υποστηρίξτε την απάντησή σας με χρήση της εντολής strace.
Χρησιμοποιούμε την strace:

```
root@utopia:/# strace touch /mnt/file1
execve("/usr/bin/touch", ["touch", "/mnt/file1"], 0x7ffe57fb1f48 /* 31 vars */) = 0
brk(NULL)                               = 0x558045aad000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=32678, ...}) = 0
mmap(NULL, 32678, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f29f1a00000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0@>\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1905632, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f29f19fe000
mmap(NULL, 1918592, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f29f1829000
mmap(0x7f29f184b000, 1417216, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) = 0x7f29f184b000
mmap(0x7f29f19a5000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17c000) = 0x7f29f19a5000
mmap(0x7f29f19f4000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1ca000) = 0x7f29f19f4000
mmap(0x7f29f19fa000, 13952, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f29f19fa000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f29f1827000
arch_prctl(ARCH_SET_FS, 0x7f29f19ff5c0) = 0
mprotect(0x7f29f19f4000, 16384, PROT_READ) = 0
mprotect(0x558043ff0000, 4096, PROT_READ) = 0
mprotect(0x7f29f1a32000, 4096, PROT_READ) = 0
munmap(0x7f29f1a00000, 32678)           = 0
brk(NULL)                               = 0x558045aad000
brk(0x558045ace000)                     = 0x558045ace000
openat(AT_FDCWD, "/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=3369792, ...}) = 0
mmap(NULL, 3369792, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f29f14f0000
close(3)                                = 0
openat(AT_FDCWD, "/mnt/file1", O_WRONLY|O_CREAT|O_NOCTTY|O_NONBLOCK, 0666) = -1 ENOSPC (No space left on device)
utimensat(AT_FDCWD, "/mnt/file1", NULL, 0) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/share/locale/locale.alias", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=2996, ...}) = 0
read(3, "# Locale name alias data base.\n#"..., 4096) = 2996
read(3, "", 4096)                       = 0
close(3)                                = 0
openat(AT_FDCWD, "/usr/share/locale/en_US/LC_MESSAGES/coreutils.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/share/locale/en/LC_MESSAGES/coreutils.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
write(2, "touch: ", 7touch: )                   = 7
write(2, "cannot touch '/mnt/file1'", 25cannot touch '/mnt/file1') = 25
openat(AT_FDCWD, "/usr/share/locale/en_US/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/share/locale/en/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
write(2, ": No space left on device", 25: No space left on device) = 25
write(2, "\n", 1
)                       = 1
close(1)                                = 0
close(2)                                = 0
exit_group(1)                           = ?
+++ exited with 1 +++
root@utopia:/#
```

Βλέπουμε και πάλι ότι υπάρχει θέμα με τον χώρο.

5. Πόσα αρχεία και πόσους καταλόγους περιέχει το συγκεκριμένο σύστημα αρχείων;

- Μέσω dumpe2fs:

```
Group 0: (Blocks 1-8192)
  Primary superblock at 1, Group descriptors at 2-2
  Block bitmap at 3 (+2)
  Inode bitmap at 4 (+3)
  Inode table at 5-218 (+4)
  7787 free blocks, 0 free inodes, 84 directories
  Free blocks: 406-8192
  Free inodes:
Group 1: (Blocks 8193-16384)
  Backup superblock at 8193, Group descriptors at 8194-8194
  Block bitmap at 8195 (+2)
  Inode bitmap at 8196 (+3)
  Inode table at 8197-8410 (+4)
  7891 free blocks, 0 free inodes, 83 directories
  Free blocks: 8494-16384
  Free inodes:
Group 2: (Blocks 16385-20479)
  Backup superblock at 16385, Group descriptors at 16386-16386
  Block bitmap at 16387 (+2)
  Inode bitmap at 16388 (+3)
  Inode table at 16389-16602 (+4)
  3877 free blocks, 0 free inodes, 92 directories
  Free blocks: 16603-20479
  Free inodes:
```

Βλέπουμε τα directories σε κάθε block group, ενώ για τα regural files θα κάνουμε το εξής:

Ξέρουμε πως κάθε regural file & directory προσδιορίζεται από ένα μοναδικό Inode, επομένως θα ισχύει πως:

regural files = (ο συνολικός αριθμός των χρησιμοποιούμενων inodes)-(τα αρχικά πιασμένα inodes, μέσα στα οποία συμπεριλαμβάνεται και αυτό του root) - (84+83+92) = (3*1712) -9 -84- 83- 92 = 4868.

- Μέσω hexdump:
  Οι υπολογισμοί μας είναι οι ίδιοι, απλά βρίσκουμε τον αριθμό των free inodes, κάνοντας navigate στο superblock:

```
root@utopia:/# hexdump -s 1024 -n 1024 -C /dev/vdc
00000400  10 14 00 00 00 50 00 00  00 00 00 00 63 4c 00 00  |.....P......cL..|
00000410  00 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00  |................|
00000420  00 20 00 00 00 20 00 00  b0 06 00 00 5d 25 98 65  |. ... ......]%.e|
00000430  5d 25 98 65 02 00 ff ff  53 ef 00 00 01 00 00 00  |]%.e....S.......|
00000440  e5 7a 78 65 00 00 00 00  00 00 00 00 01 00 00 00  |.zxe............|
00000450  00 00 00 00 0b 00 00 00  80 00 00 00 00 00 00 00  |................|
00000460  00 00 00 00 00 00 00 00  d1 26 6a d1 da e1 42 75  |.........&j...Bu|
00000470  81 36 a2 9a 4d fc 9d 1f  66 73 64 69 73 6b 32 2e  |.6..M...fsdisk2.|
00000480  69 6d 67 00 00 00 00 00  2f 63 73 6c 61 62 2d 62  |img...../cslab-b|
00000490  75 6e 6b 65 72 00 00 00  00 00 00 00 00 00 00 00  |unker...........|
000004a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000004e0  00 00 00 00 00 00 00 00  00 00 00 00 e0 1a 14 38  |...............8|
000004f0  a9 b7 42 18 b8 92 e0 e1  ae 23 d8 91 01 00 00 00  |..B......#......|
00000500  0c 00 00 00 00 00 00 00  e5 7a 78 65 00 00 00 00  |.........zxe....|
00000510  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000560  01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000570  00 00 00 00 00 00 00 00  93 03 00 00 00 00 00 00  |................|
00000580  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000800
```

Bytes 40-43: Μας δίνουν τον αριθμό inodes per group = 0x06b0=1712, και στην συνέχεια πάμε στο block group descriptor και θα πάρουμε τα Bytes 16-17 για κάθε block group, που στην ουσία είναι τα bytes για το used directories count:

```
root@utopia:/# hexdump -s 2048 -n 1024 -C /dev/vdc
00000800  03 00 00 00 04 00 00 00  05 00 00 00 6b 1e 00 00  |............k...|
00000810  54 00 04 00 00 00 00 00  00 00 00 00 00 00 00 00  |T...............|
00000820  03 20 00 00 04 20 00 00  05 20 00 00 d3 1e 00 00  |. ... ... ......|
00000830  53 00 04 00 00 00 00 00  00 00 00 00 00 00 00 00  |S...............|
00000840  03 40 00 00 04 40 00 00  05 40 00 00 25 0f 00 00  |.@...@...@..%...|
00000850  5c 00 04 00 00 00 00 00  00 00 00 00 00 00 00 00  |\...............|
00000860  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000c00
```

6. Πόσο χώρο καταλαμβάνουν τα δεδομένα και τα μεταδεδομένα του συγκεκριμένου συστήματος αρχείων;

- Για metadata έχουμε: 3(superblocks) +3 (group descriptor) +6 (bitmaps) + 3*214(214 blocks per inode table)=661
  661*blocksize = 654KB.

```
Group 0: (Blocks 1-8192)
  Primary superblock at 1, Group descriptors at 2-2
  Block bitmap at 3 (+2)
  Inode bitmap at 4 (+3)
  Inode table at 5-218 (+4)
  7787 free blocks, 0 free inodes, 84 directories
  Free blocks: 406-8192
  Free inodes:
Group 1: (Blocks 8193-16384)
  Backup superblock at 8193, Group descriptors at 8194-8194
  Block bitmap at 8195 (+2)
  Inode bitmap at 8196 (+3)
  Inode table at 8197-8410 (+4)
  7891 free blocks, 0 free inodes, 83 directories
  Free blocks: 8494-16384
  Free inodes:
Group 2: (Blocks 16385-20479)
  Backup superblock at 16385, Group descriptors at 16386-16386
  Block bitmap at 16387 (+2)
  Inode bitmap at 16388 (+3)
  Inode table at 16389-16602 (+4)
  3877 free blocks, 0 free inodes, 92 directories
  Free blocks: 16603-20479
  Free inodes:
```

Για data: total number of blocks - free blocks - datablocks - first block for boot = 270 KB.

- Αντίστοιχα με hexdump, βρίσκουμε τα αντίστοιχα πεδία κάνοντας navigate και διαβάζοντας byte προς byte.

7. Πόσο είναι το μέγεθος του συγκεκριμένου συστήματος αρχείων;

- Εντολή df -h:

```
root@utopia:/# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            482M     0  482M   0% /dev
tmpfs            98M  448K   98M   1% /run
/dev/vda1        11G  2,6G  7,6G  26% /
tmpfs           489M     0  489M   0% /dev/shm
tmpfs           5,0M     0  5,0M   0% /run/lock
shared           48G   43G  3,0G  94% /home/user/shared
tmpfs            98M     0   98M   0% /run/user/0
/dev/vdc         20M  270K   20M   2% /mnt
```

- Με hexdump: number of blocks * blocksize = 20971520 = 20MB.

8. Πόσα μπλοκ είναι διαθέσιμα/ελεύθερα στο συγκεκριμένο σύστημα αρχείων; Ισοδύναμα, έχει ελεύθερο χώρο το συγκεκριμένο σύστημα αρχείων;
Έχουμε ελεύθερα blocks, αλλά όχι ελεύθερα Inodes, άρα δεν υπάρχει περιθώριο δημιουργίας νέων αρχείων στο συγκεκριμένο filesystem.

9. Αφού υπάρχουν διαθέσιμα μπλοκ, τι σας αποτρέπει από το να δημιουργήσετε νέο αρχείο;
Η απουσία ελεύθερων inodes.

Η εικόνα fsdisk3.img

1. Ποιο εργαλείο στο Linux αναλαμβάνει τον έλεγχο ενός συστήματος αρχείων ext2 για αλλοιώσεις;
Το εργαλείο που χρησιμοποιείται για τον έλεγχο και την επισκευή ενός συστήματος αρχείων ext2/3/4 στο Linux είναι το `e2fsck` (επίσης γνωστό ως ext2 filesystem consistency check and repair tool).
Η εντολή `e2fsck` επιτρέπει να ελέγχουμε τη συντομία και τη συνοχή ενός συστήματος αρχείων ext2, ενώ μπορεί επίσης να επιδιορθώσει αλλοιώσεις που ενδέχεται να ανιχνευθούν κατά τον έλεγχο.

2. Ποιοι παράγοντες θα μπορούσαν δυνητικά να οδηγήσουν σε αλλοιώσεις στο σύστημα αρχείων; Αναφέρετε ενδεικτικά δέκα πιθανές αλλοιώσεις.

3. Τρέξτε το εργαλείο αυτό και επιδιορθώστε το σύστημα αρχείων. Αναφέρετε όλες τις αλλοιώσεις που εντοπίσατε, εξαντλητικά.

```
root@utopia:/# e2fsck /dev/vdd
e2fsck 1.46.2 (28-Feb-2021)
fsdisk3.img contains a file system with errors, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
First entry 'BOO' (inode=1717) in directory inode 1717 (/dir-2) should be '.'
Fix<y>? yes
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Inode 3425 ref count is 1, should be 2.  Fix<y>? yes
Pass 5: Checking group summary information
Block bitmap differences:  +34
Fix<y>? yes
Free blocks count wrong (926431538, counted=19800).
Fix<y>? yes

fsdisk3.img: ***** FILE SYSTEM WAS MODIFIED *****
fsdisk3.img: 23/5136 files (0.0% non-contiguous), 680/20480 blocks
```

Pass 2: Πρώτο πρόβλημα είναι το ότι σε κάθε directory τα 2 πρώτα αρχεία είναι πάντα τα “.”, “,,”(current wording directory, father of cwd)άρα χρειάζεται rename το πρώτο directory, Που εδώ λέγεται boo.

Pass 4: Μς λέει ότι κανονικά στο Inode 3425 δείχνουν 2 hardlinks, ωστόσο το reference count του συγκεκριμένου αρχείου=1, άρα δεν έχει ενημερωθεί σωστά και χρειάζεται αλλαγή.

Pass 5: Μας λέει ότι έχουμε block bitmap differences, που συμβαίνει όταν υπάρχει διαφορά μεταξύ της πληροφορίας που είναι αποθηκευμένη στο Block bitmap Και στην πραγματική

χρήση των blocks(π.χ. στο block bitmap φαίνεται ότι χρησιμοποιείται κάποιο Block, ενώ στην πραγματικότητα όχι). Εδώ μας λέει ότι το block 34 είναι σε χρήση, ενώ το αντίστοιχο bit στο block bitmap είναι 0.

Έπειτα εμφανίζεται θέμα στο free blocks count Του superblock, που δεν αντιστοιχεί στην πραγματικότητα οπότε το φτιάχνουμε και αυτό πατώντας “y”.

4. Επαναφέρετε το δίσκο στην πρότερη του κατάσταση, από την αρχική εικόνα. Εντοπίστε τις αλλοιώσεις με χρήση της μεθόδου hexedit.
Κάνουμε Restore:

```
root@utopia:/home/user/shared# dd if=fsdisk3-982902777.img of=/dev/vdd bs=4M
5+0 records in
5+0 records out
20971520 bytes (21 MB, 20 MiB) copied, 0,0879756 s, 238 MB/s
```

5. Επιδιορθώστε κάθε αλλοίωση ξεχωριστά με χρήση της μεθόδου hexedit. Για κάθε μία από τις αλλοιώσεις που επιδιορθώνετε, τρέξτε το εργαλείο fsck με τρόπο που δεν προκαλεί καμία αλλαγή [“dry run”] και επιβεβαιώστε ότι πλέον δεν την εντοπίζει.

Θα προσπαθήσουμε μέσω του hexdump να αλλάξουμε τα λάθη που βρήκαμε πριν.
Αρχικά κάνουμε ένα hexdump Για να φτάσουμε στο superblock, να βρούμε το όνομα “boo”, και να το αντικαταστήσουμε με το “.”. Γνωρίζοντας πως κάθε group έχει έως 1712 inodes, το Inode Που χρήζει διόρθωσης βρίσκεται στο δεύτερο bg και στο index 5(1717):

```
root@utopia:~# hexdump -s$((2048+32)) -n 128 -C /dev/vdd
00000820  03 20 00 00 04 20 00 00  05 20 00 00 1b 1f a8 06  |. ... ... ......|
00000830  02 00 04 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000840  03 40 00 00 04 40 00 00  05 40 00 00 25 0f ac 06  |.@...@...@..%...|
00000850  01 00 04 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000860  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000008a0
root@utopia:~# hexdump -s$((8197*1024+4*128)) -n 256 -C /dev/vdd
00801600  ed 41 00 00 00 04 00 00  e9 7a 78 65 e9 7a 78 65  |.A.......zxe.zxe|
00801610  e9 7a 78 65 00 00 00 00  00 00 02 00 02 00 00 00  |.zxe............|
00801620  00 00 00 00 04 00 00 00  dc 20 00 00 00 00 00 00  |......... ......|
00801630  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00801660  00 00 00 00 da 39 00 7a  00 00 00 00 00 00 00 00  |.....9.z........|
00801670  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00801680  a4 81 00 00 0b 00 00 00  e9 7a 78 65 e9 7a 78 65  |.........zxe.zxe|
00801690  e9 7a 78 65 00 00 00 00  00 00 01 00 02 00 00 00  |.zxe............|
008016a0  00 00 00 00 01 00 00 00  01 24 00 00 00 00 00 00  |.........$......|
008016b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
008016e0  00 00 00 00 51 ad cf 21  00 00 00 00 00 00 00 00  |....Q..!........|
008016f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00801700
root@utopia:~# hexdump -s$((8412*1024)) -n 256 -C /dev/vdd
00837000  b5 06 00 00 0c 00 03 00  42 4f 4f 00 02 00 00 00  |........BOO.....|
00837010  0c 00 02 00 2e 2e 00 00  b6 06 00 00 10 00 06 00  |................|
00837020  66 69 6c 65 2d 31 00 00  b7 06 00 00 10 00 06 00  |file-1..........|
00837030  66 69 6c 65 2d 32 00 00  b8 06 00 00 c8 03 06 00  |file-2..........|
00837040  66 69 6c 65 2d 33 00 00  00 00 00 00 00 00 00 00  |file-3..........|
00837050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00837100
```

Χρησιμοποιούμε την εντολή “hexedit” και αλλάζουμε το όνομα, και το length:

```
00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0
00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0
00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0
00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0
00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0
00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0
00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0
00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0
00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0

            New position ? 0x837000

00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0
00 00   10 00 06 00   66 69 6C 65   2D 31 00 00   B3 0
00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0
00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0
00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0
00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0
00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0
00 00   00 00 00 00   00 00 00 00   00 00 00 00   00 0
```

```
00836FF8   00 00 00 00   00 00 00 00   B5 06 00 00   0C 00 01 00   2E 00 00 00   02 00 00 00   0C 00 02 00   2E 2E 00 00   B6 06 00 00   10 00 06 00   66 69 6C 65
```

Κάνουμε Ctrl-X για save and exit.
Για να τσεκάρουμε αν όλα είναι εντάξει ξανατρέχεουμε το e2fsck ή βρίσκουμε το νέο όνομα μέσω του hexdump, που το κάνουμε εδώ, και βλέπουμε ότι διορθώθηκε:

```
root@utopia:~# hexdump -s$((8412*1024)) -n 256 -C /dev/vdd
00837000  b5 06 00 00 0c 00 01 00  2e 00 00 00 02 00 00 00  |................|
00837010  0c 00 02 00 2e 2e 00 00  b6 06 00 00 10 00 06 00  |................|
00837020  66 69 6c 65 2d 31 00 00  b7 06 00 00 10 00 06 00  |file-1..........|
00837030  66 69 6c 65 2d 32 00 00  b8 06 00 00 c8 03 06 00  |file-2..........|
00837040  66 69 6c 65 2d 33 00 00  00 00 00 00 00 00 00 00  |file-3..........|
00837050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00837100
```

Δεύτερο error: αλλαγή του reference count, άρα θα βρούμε το inode 3425 στο bg 2 με index 1:

```
root@utopia:~# hexdump -s$((2112)) -n 256 -C /dev/vdd
00000840  03 40 00 00 04 40 00 00  05 40 00 00 25 0f ac 06  |.@...@...@..%...|
00000850  01 00 04 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000860  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000940
root@utopia:~# hexdump -s$((1024*16389)) -n 256 -C /dev/vdd
01001400  ed 41 00 00 00 04 00 00  e9 7a 78 65 e9 7a 78 65  |.A.......zxe.zxe|
01001410  e9 7a 78 65 00 00 00 00  00 00 01 00 02 00 00 00  |.zxe............|
01001420  00 00 00 00 04 00 00 00  e8 00 00 00 00 00 00 00  |................|
01001430  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
01001460  00 00 00 00 e1 ce a8 18  00 00 00 00 00 00 00 00  |................|
01001470  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
01001480  a4 81 00 00 0b 00 00 00  e9 7a 78 65 e9 7a 78 65  |.........zxe.zxe|
01001490  e9 7a 78 65 00 00 00 00  00 00 01 00 02 00 00 00  |.zxe............|
010014a0  00 00 00 00 01 00 00 00  04 24 00 00 00 00 00 00  |.........$......|
010014b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
010014e0  00 00 00 00 1f 7f e6 fe  00 00 00 00 00 00 00 00  |................|
010014f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
01001500
```

Με hexedit:

```
010013B4   FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF
010013E0   FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF
0100140C   E9 7A 78 65  E9 7A 78 65  00 00 00 00  00 00 02 00
01001438   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
```

Και διορθώθηκε!
3ο σφάλμα: bitmap difference, πρέπει και πάλι να διορθώσουμε το bit από 0 σε 1 στο Bitmap με hexedit, πρέπει από FD(11111101) να γίνει FF(11111111):

```
00000C08   FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF  FF
```

Έτοιμο.
Για το τελευταίο σφάλμα: Είναι λανθασμένος ο αριθμός των Free blocks count, που είναι το πεδίο στα bytes 12-15 του superblock, άρα αφού τα αλλάξουμε και αυτά:

```
000003F4   00 00 00 00  00 00 00 00  00 00 00 00  10 14 00 00  00 50 00 00  00 04 00 00  58 4D 0
00000420   00 20 00 00  00 20 00 00  B0 06 00 00  7D 7E 99 65  51 80 99 65  02 00 FF FF  53 EF 01
```

Κάνουμε και ένα τελευταίο e2fsck:

```
root@utopia:~# e2fsck /dev/vdd
e2fsck 1.46.2 (28-Feb-2021)
fsdisk3.img: clean, 23/5136 files, 680/20480 blocks
```

Και βλέπουμε πως όλα είναι εντάξει.

**Για την απαντήσεις των ερωτήσεων χρησιμοποιήθηκαν οι παρκάτω φωτογραφίες, καθώς και η ιστοσελίδα:

- https://www.nongnu.org/ext2-doc/ext2.html
- Inode:

Inode Data Structure

Starting Byte	Ending Byte	Size in Bytes	Field Description
0	1	2	Type and Permissions (see below)
2	3	2	User ID
4	7	4	Lower 32 bits of size in bytes
8	11	4	Last Access Time (in POSIX time)
12	15	4	Creation Time (in POSIX time)
16	19	4	Last Modification time (in POSIX time)
20	23	4	Deletion time (in POSIX time)
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 (see below)
36	39	4	Operating System Specific value #1
40	43	4	Direct Block Pointer 0
44	47	4	Direct Block Pointer 1
48	51	4	Direct Block Pointer 2
52	55	4	Direct Block Pointer 3
56	59	4	Direct Block Pointer 4
60	63	4	Direct Block Pointer 5
64	67	4	Direct Block Pointer 6
68	71	4	Direct Block Pointer 7
72	75	4	Direct Block Pointer 8
76	79	4	Direct Block Pointer 9
80	83	4	Direct Block Pointer 10
84	87	4	Direct Block Pointer 11
88	91	4	Singly Indirect Block Pointer (Points to a block that is a list of block pointers to data)
92	95	4	Doubly Indirect Block Pointer (Points to a block that is a list of block pointers to Singly Indirect Blocks)
96	99	4	Triply Indirect Block Pointer (Points to a block that is a list of block pointers to Doubly Indirect Blocks)
100	103	4	Generation number (Primarily used for NFS)
104	107	4	In Ext2 version 0, this field is reserved. In version >= 1, Extended attribute block (File ACL).
108	111	4	In Ext2 version 0, this field is reserved. In version >= 1, Upper 32 bits of file size (if feature bit set) if it's a file, Directory ACL if it's a directory
112	115	4	Block address of fragment
116	127	12	Operating System Specific Value #2

- Superblock:

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 POSIX time)
48	51	4	Last written time (in POSIX time)
52	53	2	Number of times the volume has been mounted since its last consistency check (fsck)
54	55	2	Number of mounts allowed before a consistency check (fsck) must be done
56	57	2	Ext2 signature (0xef53), used to help confirm the presence of Ext2 on a volume
58	59	2	File system state (see below)
60	61	2	What to do when an error is detected (see below)
62	63	2	Minor portion of version (combine with Major portion below to construct full version field)
64	67	4	POSIX time of last consistency check (fsck)
68	71	4	Interval (in POSIX time) between forced consistency checks (fsck)
72	75	4	Operating system ID from which the filesystem on this volume was created (see below)
76	79	4	Major portion of version (combine with Minor portion above to construct full version field)
80	81	2	User ID that can use reserved blocks
82	83	2	Group ID that can use reserved blocks