ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

3η Εργαστηριακή Άσκηση:

Εξερεύνηση Συστήματος Αρχείων σε Περιβάλλον Linux

Εργασία για το μάθημα του 7ου εξαμήνου:
"Εργαστήριο Λειτουργικών Συστημάτων"

Διδάσκοντες: **Ν. Κοζύρης, Γ. Γκούμας, Β. Κούκης**

Ονοματεπώνυμο: **Κελλάρη Μυρσίνη**

Αριθμός Μητρώου: **03119082**

Ονοματεπώνυμο: **Ξανθόπουλος Παναγιώτης**

Αριθμός Μητρώου: **03119084**

Ομάδα: **1η**

Αθήνα 28 Ιανουαρίου 2023

Άσκηση 1

Η εικόνα fsdisk1.img

1.1

Στην εντολή exec του utopia.sh προσθέσαμε την εξής γραμμή:
`-drive file=./fsdisk1-c8e29319b.img,format=raw,if=virtio`
Η γραμμή αυτή ουσιαστικά κάνει attach ένα νέο block device στο VM που χρησιμοποιεί το virtio paravirtual device driver ως interface και περιέχει τα δεδομένα του αρχείου "fsdisk1-c8e29319b.img". Το format καθορίζει το είδος των δεδομένων που περιέχονται στο αρχείο. Η συσκευή που προστίθεται είναι η `/dev/vdb`.

1.2

Ο δίσκος έχει μέγεθος 51MB (αφού γίνει mount).

Προσέγγιση tools: Εκτελούμε την εντολή `df -T -H /dev/vdb`. Το αποτέλεσμα της εντολής αυτής είναι το παρακάτω:

```
Filesystem      Type  Size  Used Avail Use% Mounted on
/dev/vdb        ext2   51M   17k   49M   1% /mnt
```

Προσέγγιση hexedit: Εκτελούμε την εντολή `hexdump -s $((1024+2)) -n 1024 -C /dev/vdb` για να δούμε τα δεδομένα του superblock του Block Group 0. Τα bytes 2-3 (εξού και το +2) περιέχουν τον συνολικό αριθμό των blocks στον δίσκο. Αυτά είναι 51200 (0xc800). Εφόσον το block size είναι 1024 bytes (τα bytes 24-27 είναι 0 οπότε το block size είναι $1024 \ll 0 = 1024$), το συνολικό μέγεθος του δίσκου είναι $\sim$ 51M.

1.3

Περιέχει το σύστημα αρχείων ext2.

Προσέγγιση tools: Εκτελούμε ξανά την εντολή `df -T -H /dev/vdb`.

Προσέγγιση hexedit: Αν δούμε τα δεδομένα του ίδιου superblock με την εντολή `hexdump -s $((1024+56)) -n 1024 -C /dev/vdb`, τα bytes 56-57 (εξού και το +56) περιέχουν τον λεγόμενο "μαγικό αριθμό", ο οποίος μας δείχνει το σύστημα αρχείων που περιέχει ο δίσκος. Έχουν τιμή 0xEF53, η οποία αντιστοιχεί στο ext2.

1.4

Ημερομηνία δημιουργίας: "Fri Dec 16 15:32:29 2022"

Προσέγγιση tools: Με εκτέλεση της εντολής `dumpe2fs /dev/vdb`, στο πεδίο "Filesystem created:" βλέπουμε την εξής ημερομηνία: "Fri Dec 16 15:32:29 2022".

Προσέγγιση hexedit: Στο ίδιο superblock, τα bytes 264-267 περιέχουν την ημερομηνία αυτή ως δευτερόλεπτα από την ημερομηνία 00:00:00, January 1, 1970, UTC. Η τιμή τους είναι 0x639c736d, η οποία αντιστοιχεί στην ημερομηνία που είδαμε προηγουμένως. Την τιμή των bytes αυτών την βλέπουμε με την εντολή `hexdump -s $((1024+264)) -n 1024 -C /dev/vdb`.

1.5

Ημερομηνία τελευταίας προσάρτησης: "Sat Dec 31 18:22:28 2022"

Προσέγγιση tools: Με εκτέλεση της ίδιας εντολής, στο πεδίο "Last mount time:" βλέπουμε την εξής ημερομηνία: "Sat Dec 31 18:22:28 2022".

Προσέγγιση hexedit: Στο ίδιο superblock, τα bytes 44-47 περιέχουν την ημερομηνία αυτή ως δευτερόλεπτα από την ημερομηνία 00:00:00, January 1, 1970, UTC. Η τιμή τους είναι 0x63b061c4, η οποία αντιστοιχεί στην ημερομηνία που είδαμε προηγουμένως. Την τιμή των bytes αυτών την βλέπουμε με την εντολή `hexdump -s $((1024+44)) -n 1024 -C /dev/vdb`.

1.6

Μονοπάτι τελευταίας προσάρτησης: `/home/jimsiak/cslab-git/fs/mnt`

Προσέγγιση tools: Με εκτέλεση της ίδιας εντολής, στο πεδίο "Last mounted on:" βλέπουμε το εξής μονοπάτι: `/home/jimsiak/cslab-git/fs/mnt`.

Προσέγγιση hexedit: Στο ίδιο superblock, τα bytes 136-201 περιέχουν το directory που αναζητούμε σε ASCII τιμές. Την τιμή των bytes αυτών την βλέπουμε με την εντολή `hexdump -s $((1024+136)) -n 1024 -C /dev/vdb`.

1.7

Ημερομηνία τελευταίας τροποποίησης: "Sat Dec 31 18:22:28 2022"

Προσέγγιση tools: Με εκτέλεση της ίδιας εντολής, στο πεδίο "Last write time:" βλέπουμε την εξής ημερομηνία: "Sat Dec 31 18:22:28 2022".

Προσέγγιση hexedit: Στο ίδιο superblock, τα bytes 48-51 περιέχουν την ημερομηνία αυτή ως δευτερόλεπτα από την ημερομηνία 00:00:00, January 1, 1970, UTC. Η τιμή τους είναι 0x63b061c4, η οποία αντιστοιχεί στην ημερομηνία που είδαμε προηγουμένως. Την τιμή των bytes αυτών την βλέπουμε με την εντολή `hexdump -s $((1024+48)) -n 1024 -C /dev/vdb`.

1.8

Το μπλόκ σε ένα σύστημα αρχείων είναι ένα συνεχόμενο κομμάτι δίσκου που χρησιμεύει στην αποθήκευση δεδομένων. Έιναι σταθερό και προκαθορισμένου μεγέθους. Είναι το μεγαλύτερο δυνατό συνεχόμενο κομμάτι δίσκου που μπορεί να ανατεθεί σε ένα αρχείο και που μπορεί να μεταφερθεί με μία λειτουργία IO.

1.9

Το μέγεθος του block είναι 1024 bytes.

Προσέγγιση tools: Με εκτέλεση της ίδιας εντολής, στο πεδίο "Block size:" βλέπουμε την τιμή 1024.

Προσέγγιση hexedit: Με την εντολή `hexdump -s $((1024+24)) -n 1024 -C /dev/vdb` βλέπουμε τα bytes 24-27 του ίδιου superblock. Η τιμή τους είναι 0 οπότε το block size είναι $= 1024 \ll 0 = 1024$ (τα 4 bytes περιέχουν την τιμή s_log_block_size και το block size προκύπτει ως block size $= 1024 \ll$ s_log_block_size).

1.10

Το inode σε ένα σύστημα αρχείων είναι μια δομή δεδομένων που αντιπροσωπεύει κάθε ξεχωριστό αρχείο. Περιέχει πληροφορίες όπως το μέγεθος, τα permissions για τους διάφορους χρήστες και άλλα μεταδεδομένα. Κυρίως, περιέχει τις τοποθεσίες των data blocks του αρχείου, δηλαδή το που βρίσκονται τα δεδομένα του αρχείου, μέσω άμεσων και έμμεσων δεικτών. Κάθε αρχείο έχει μοναδικό inode και κάθε inode αναφέρεται σε μοναδικό αρχείο.

1.11

Το μέγεθος του inode είναι 128 bytes.

Προσέγγιση tools: Με εκτέλεση της ίδιας εντολής, στο πεδίο "Inode size:" βλέπουμε την τιμή 128.

Προσέγγιση hexedit: Με την εντολή `hexdump -s $((1024+88)) -n 1024 -C /dev/vdb` βλέπουμε τα bytes 88-89 του ίδιου superblock. Η τιμή τους είναι 0x0080 οπότε το inode size είναι 128 bytes.

1.12

Έχουμε 49552 διαθέσιμα blocks και 12810 διαθέσιμα inodes.

Προσέγγιση tools: Με εκτέλεση της ίδιας εντολής, στο πεδίο "Free blocks:" βλέπουμε την τιμή 49552 και στο πεδίο "Free inodes:" βλέπουμε την τιμή 12810.

Προσέγγιση hexedit: Με την εντολή `hexdump -s $((1024+12)) -n 1024 -C /dev/vdb` βλέπουμε τα bytes 12-15 του ίδιου superblock. Η τιμή τους είναι 0x0000c190 οπότε τα διαθέσιμα blocks είναι 49552. Επίσης, με την εντολή `hexdump -s $((1024+16)) -n 1024 -C /dev/vdb` βλέπουμε τα bytes 16-19 του ίδιου superblock. Η τιμή τους είναι 0x0000320a οπότε τα διαθέσιμα inodes είναι 12810.

1.13

Το superblock είναι μια δομή δεδομένων, μήκους ενός block, η οποία κρατάει πληροφορίες για το σύστημα αρχείων ως σύνολο. Αποθηκεύει τον συνολικό αριθμό blocks/inodes, τον αριθμό διαθέσιμων blocks/inodes, το μέγεθος του κάθε block/inode, πληροφορίες σχετικά με την τελευταία τροποποίηση, προσάρτηση και δημιουργία του συστήματος αρχείων, και άλλα.

1.14

Το superblock είναι το πρώτο block σε κάθε block group. Το superblock που αντιστοιχεί στο block group 0 (το πρώτο), είναι το κύριο αντίγραφο. Για να το διαβάσουμε, μπορούμε να δούμε 1024 bytes μετά την αρχή

κάποιου δίσκου που περιέχει το ext2. Τα πρώτα 1024 bytes περιέχουν δεδομένα όπως το MBR και το Partition table. Μετά από αυτά, ξεκινούν τα δεδομένα του πρώτου block group.

1.15

Το superblock υπάρχει ως αντίγραφο σε κάθε block group. Σαν δομή δεδομένων είναι πολύ σημαντικό γιατί περιέχει απαραίτητες πληροφορίες που επιτρέπουν την πρόσβαση στα δεδομένα του συστήματος αρχείων. Αν υποστεί corruption, η πρόσβαση σε αυτά καθίσταται δύσκολη έως αδύνατη. Για το λόγο αυτό, υπάρχουν τα αντίγραφα, ώστε αν ένα από αυτά υποστεί corruption, να μπορεί να επανέλθει στη σωστή κατάσταση. Οπότε, για να υπάρχει πραγματική ζημιά, πρέπει να υποστούν corruption όλα τα αντίγραφα, πράγμα δύσκολο. Επίσης, η ύπαρξη αντιγράφων ευνοεί και την ταχύτητα πρόσβασης. Σε μαγνητικούς δίσκους, η πρόσβαση δεδομένων που είναι κοντά τοπικά είναι πιο γρήγορη. Οπότε, η δυνατότητα πρόσβασης του superblock από πολλές τοποθεσίες εκμεταλλεύεται αυτό το χαρακτηριστικό για να αυξήσει την ταχύτητα πρόσβασης.

1.16

Προσέγγιση tools: Με την εντολή `dumpe2fs /dev/vdb`, μπορούμε να δούμε πληροφορίες για κάθε group ξεχωριστά. Σε κάθε block (εκτός του πρώτου), βλέπουμε το πεδίο "Backup superblock at X", όπου X το block στο οποίο περιέχεται το αντίγραφο του superblock. Οπότε, εφόσον έχουμε 7 groups, έχουμε 6 αντίγραφα του superblock που βρίσκονται στα blocks 8193, 16385, 24577, 32769, 40961, 49153.

Προσέγγιση hexedit: Με την εντολή `hexdump -s $((1024+32)) -n 1024 -C /dev/vdb`, βλέπουμε τα bytes 32-35 του κύριου superblock. Έχουν τιμή 0x00002000, οπότε έχουμε 8192 blocks ανά block group. Επίσης, με την εντολή `hexdump -s $((1024+4)) -n 1024 -C /dev/vdb`, βλέπουμε τα bytes 4-7 του κύριου superblock. Έχουν τιμή 0x0000c800, οπότε όλο το σύστημα αρχείων έχει 51200 blocks συνολικά. Αν διαιρέσουμε τις 2 αυτές τιμές (αφαιρώντας 1 από το 51200 γιατί 1 block είναι το boot block), παίρνουμε 6.25. Άρα έχουμε 7 block groups (το τελευταίο έχει λιγότερα blocks). Οπότε, εφόσον το primary superblock βρίσκεται στο block 1, τα υπόλοιπα βρίσκονται σε πολλαπλάσια του $8192 + 1$. Άρα, τα αντίγραφα βρίσκονται στα blocks 8193, 16385, 24577, 32769, 40961, 49153 (έχουμε 6 αντίγραφα).

1.17

Το block group είναι μια ομαδοποίση blocks σε ένα συνεχόμενο χώρο. Βοηθάει στην καλύτερη οργάνωση των δεδομένων (των data blocks και των inodes) και περιέχει αντίγραφα σημαντικών δομών όπως το superblock και τα block group descriptors. Περιέχει επίσης δομές που βοηθούν στην οργάνωση των εσωτερικών σε αυτό data blocks και inodes, τα λεγόμενα bitmaps (μας πληροφορούν για το ποια blocks/inodes είναι διαθέσιμα).

1.18

Ο αριθμός των block groups δεν είναι σταθερός και εξαρτάται από το μέγεθος του συστήματος αρχείων. Δηλαδή δεν υπάρχει κάποιο ανώτατο όριο. Όμως προκύπτει άμεσα από τα δεδομένα του superblock όπως δείξαμε στην ερώτηση 2.16. Τα groups τοποθετούνται σειριακά, το ένα μετά το άλλο, στο δίσκο, αμέσως μετα το Boot block. Έχουν ίδιο μέγεθος, εκτός ίσως από το τελευταίο.

1.19

Το συγκεκριμένο σύστημα αρχείων περιέχει 7 block groups. Ο τρόπος εύρεσης του αριθμού αυτού και με τις 2 προσεγγίσεις φαίνεται στο ερώτημα 2.16.

1.20

Ο block group descriptor είναι μια δομή δεδομένων που περιέχει μεταδεδομένα για ένα block group. Συγκεκριμένα, περιέχει την τοποθεσία του block bitmap, του inode bitmap, του inode table. Περιέχει επίσης το πλήθος των διαθέσιμων blocks, inodes και το πλήθος των directories στο group. Κάθε block group έχει τον δικό του descriptor. Όλοι οι descriptors περιέχονται σε μια δομή, το block group descriptor table. Το table περιέχεται σε κάθε group (κύριο table στο πρώτο group και αντίγραφα στα υπόλοιπα), αμέσως μετά το superblock. Έχει μήκος ανάλογο του πλήθους των groups (≥ 1 block βέβαια) και κάθε descriptor έχει μέγεθος 32 bytes.

1.21

Οι descriptors βοηθούν στην καλύτερη οργάνωση και πιο εύκολη πρόσβαση των δεδομένων μέσα σε ένα block group. Η ύπαρξη αντιγράφων βοηθά στην επαναφορά ενός descriptor που πιθανά να υπέστη corruption όπως και στην γρηγορότερη πρόσβαση, κατ' αντιστοιχία με το superblock.

1.22

Προσέγγιση tools: Με την εντολή `dumpe2fs /dev/vdb`, μπορούμε να δούμε πληροφορίες για κάθε group ξεχωριστά. Σε κάθε block (εκτός του πρώτου), βλέπουμε το πεδίο "Superblocks at X-Y", όπου X-Y το εύρος σε block στο οποίο περιέχεται το αντίγραφο του block group descriptor table. Οπότε, εφόσον έχουμε 7 groups, έχουμε 6 αντίγραφα του block group descriptor table που βρίσκονται στα blocks 8194, 16386, 24578, 32770, 40962, 49154. Το κύριο block group descriptor table βρίσκεται στο block 2.

Προσέγγιση hexedit: Εργαζόμαστε όμοια με την 2.16. Γνωρίζουμε ότι το κύριο block group descriptor table βρίσκεται στο block 2. Οπότε, τα υπόλοιπα βρίσκονται σε πολλαπλάσια του 8192 + 2. Άρα, τα αντίγραφα βρίσκονται στα blocks 8194, 16386, 24578, 32770, 40962, 49154 (έχουμε 6 αντίγραφα).

1.23

Το block bitmap είναι μια δομή δεδομένων, μήκους ενός block, ξεχωριστή για κάθε group. Κάθε bit του bitmap αντιστοιχεί σε ένα block από τα data blocks του συγκεκριμένου group. Αν το bit είναι 1, τότε το αντίστοιχο block είναι σε χρήση και αν είναι 0, τότε το αντίστοιχο block είναι διαθέσιμο. Το inode bitmap είναι η αντίστοιχη δομή δεδομένων για τα inodes του εκάστοτε group. Το block bitmap είναι το 1ο block αμέσως μετά το block group descriptor table και το inode bitmap είναι το 2ο block αμέσως μετά το table.

1.24

Το inode table είναι η δομή δεδομένων που κρατά όλα τα inodes για όλα τα αρχεία που υπάρχουν στο συγκεκριμένο group, το ένα μετά το άλλο. Βρίσκεται αμέσως μετά το inode bitmap.

1.25

Μια δομή δεδομένων inode περιέχει μεταδεδομένα για ένα αρχείο. Συγκεκριμένα, περιέχει τα εξής πεδία:
Type and Permissions (bytes 0-1)
User ID (bytes 2-3)
Lower 4 bytes of size (bytes 4-7)
Last Access Time (bytes 8-11)
Creation Time (bytes 12-15)

Last Modification Time (bytes 16-19)
Deletion Time (bytes 20-23)
Group ID (bytes 24-25)
Count of hard links to this inode (bytes 26-27)
Count of disk sectors used by this node (excluding inode struct and dentries to the inode) (bytes 28-31)
Flags (bytes 32-35)
OS specific value (bytes 36-39)
Direct Block Pointer 1-11 (bytes 40-87)
Singly Indirect Block Pointer (bytes 88-91)
Doubly Indirect Block Pointer (bytes 92-95)
Triply Indirect Block Pointer (bytes 96-99)

Το κάθε inode αποθηκεύεται μέσα στο inode table του group στο οποίο ανήκει.

1.26

Προσέγγιση tools: Με την εντολή `dumpe2fs /dev/vdb`, στο πεδίο "Blocks per group:" βλέπουμε την τιμή 8192 και στο πεδίο "Inodes per group:" βλέπουμε την τιμή 1832.

Προσέγγιση hexedit: Με την εντολή `hexdump -s $((1024+32)) -n 1024 -C /dev/vdb`, βλέπουμε τα bytes 32-35 του κύριου superblock. Έχουν τιμή 0x00002000, οπότε έχουμε 8192 blocks ανά block group. Επίσης, με την εντολή `hexdump -s $((1024+40)) -n 1024 -C /dev/vdb`, βλέπουμε τα bytes 40-43 του κύριου superblock. Έχουν τιμή 0x00000728, οπότε έχουμε 1832 inodes ανά block group.

1.27

Προσέγγιση tools: Αρχικά κάνουμε mount το δίσκο στην τοποθεσία `/mnt` μέσω της εντολής `mount -t ext2 /dev/vdb /mnt`. Στη συνέχεια, εκτελούμε την εντολή `stat /mnt/dir2/helloworld`. Η εντολή αυτή μας δίνει πληροφορίες για το αρχείο και μεταξύ άλλων, μας πληροφορεί ότι το αρχείο αυτό αντιστοιχεί στο inode 3666.

Προσέγγιση hexedit: Εργαζόμαστε ως εξής:

Γνωρίζουμε ότι το inode του root directory για τον συγκεκριμένο δίσκο είναι το inode 2 (το δεύτερο inode του BG 0).

Ψάχνουμε το inode table του BG 0. Το βρίσκουμε από το BGD (BG descriptor). Ο descriptor του BG 0 είναι τα 32 πρώτα bytes του BGDT (BGD table). Τα λαμβάνουμε με την εντολή `hexdump -s $((2*1024)) -n 32 -C /dev/vdb` (γνωρίζουμε ότι το BGDT ξεκινάει στο 2ο block). Από αυτά, τα bytes 8-11 είναι ένας δείκτης στην αρχή του inode table του BG 0. Η τιμή του δείκτη αυτού είναι 0x05. Οπότε, το inode table ξεκινάει στο 5ο block.

Εφόσον θέλουμε το 2ο inode του inode table (και inode size = 128 bytes), χρησιμοποιούμε την εντολή `hexdump -s $((5*1024 + 128)) -n 128 -C /dev/vdb` για να λάβουμε τα δεδομένα του inode του root directory.

Θέλουμε τώρα τα δεδομένα του root directory file. Τα bytes 40-43 περιέχουν τον Direct Block Pointer 1 και έχουν τιμή 0xea. Οπότε στο block 234 περιέχεται το directory file του root (είναι και ο μοναδικός μη μηδενικός data pointer). Διαβάζουμε τα δεδομένα του file με την εντολή `hexdump -s $((234*1024)) -n 1024 -C /dev/vdb`.

Γενικά, ένα directory file έχει συνεχόμενα directory entries με τα εξής πεδία: Inode number (4 bytes)
Dentry size (2 bytes)
File name size (1 byte)
File type (1 byte)
File name (X bytes)

Για παράδειγμα, βλέπουμε 1 dentry με τα εξής δεδομένα: 02 00 00 00 0c 00 01 00 2e 00 00 00. Έχουμε ένα αρχείο που αντιστοιχεί στο inode 2, το dentry size είναι 12 bytes, το μήκος του ονόματος είναι 1 byte, το είδος του αρχείου είναι άγνωστο (type = 0x00) και το όνομά του είναι ".".

Αναζητούμε το dentry που αντιστοιχεί στο dir2. Έχει τα εξής δεδομένα: 51 0e 00 00 c8 03 04 00 64 69 72 32 ... (τα υπόλοιπα είναι 0). Από τα πρώτα 4 bytes βλέπουμε ότι το dir2 ανήκει στο inode 0x0e51, δηλαδή στο inode 3665.

Γνωρίζουμε ότι το κάθε group έχει 1832 inodes οπότε το inode 3665 είναι το 1ο inode του BG 2 (η μέτρηση των BG ξεκινά από το 0 ενώ των inodes από το 1). Οπότε, αναζητούμε το inode table του BG 2. Εργαζόμαστε όπως πριν. Δηλαδή, αναζητούμε την τοποθεσία του inode table του BG 2 από το BGD 2. Τον BGD 2 τον βρίσκουμε από το BGDT, σε offset 64 (κάθε BGD έχει μήκος 32 bytes και ο BGD 2 είναι ο 3ος). Τα δεδομένα του BGD 2 τα λαμβάνουμε άρα με την εντολή `hexdump -s $((2*1024 + 64)) -n 32 -C /dev/vdb`. Τα bytes 8-11 έχουν τιμή 0x4005 οπότε το inode table του BG 2 είναι στο block 16389.

Λαμβάνουμε τα δεδομένα του πρώτου inode του inode table που εντοπίσαμε παραπάνω με την εντολή `hexdump -s $((16389*1024)) -n 128 -C /dev/vdb`. Θυμίζουμε ότι το inode αυτό αντιστοιχεί στο directory /dir2. Παίρνουμε την τοποθεσία του πρώτου Data block από τα bytes 40-43 (Direct Data Block Pointer 1). Η τοποθεσία είναι το block 0x80ea = 33002.

Διαβάζουμε τα δεδομένα στο block 33002 ώστε να δούμε τα περιεχόμενα του αρχείου `/dir2` (τα dentries) με την εντολή `hexdump -s $((33002*1024)) -n 128 -C /dev/vdb`. Βρίσκουμε το dentry που αντιστοιχεί στο αρχείο `helloworld`. Έχει τα εξής δεδομένα: 52 0e 00 00 e8 03 0a 00 68 65 6c 6c 6f 77 6f 72 6c 64 ... (τα υπόλοιπα είναι 0). Από τα πρώτα 4 bytes βλέπουμε ότι το helloworld ανήκει στο inode 0x0e52, δηλαδή στο inode 3666.

1.28

Γνωρίζουμε ότι κάθε group έχει 1832 inodes, άρα το inode 3666 είναι το 2ο inode του BG 2.

1.29

Προσέγγιση tools: Εκτελούμε την εντολή `debugfs /dev/vdb`. Έπειτα, στο νέο περιβάλλον εκτελούμε την εντολή `imap <3666>`. Η εντολή αυτή μας δίνει την τοποθεσία του inode αυτού σε block, η οποία ταυτίζεται με την τοποθεσία του inode table που περιέχει το συγκεκριμένο inode. Η τοποθεσία είναι το block 16389.

Προσέγγιση hexedit: Ξέρουμε από το ερώτημα 2.28 ότι το inode 3666 είναι το 2ο inode του BG 2. Στο ερώτημα 2.27 βρήκαμε σε ποιο block βρίσκεται το inode table του BG 2, το οποίο είναι το block 16389.

1.30

Προσέγγιση tools: Στο περιβάλλον της `debugfs`, εκτελούμε την εντολή `stat <3666>`. Λαμβάνουμε τα εξής δεδομένα:

```
Inode: 3666   Type: regular    Mode:  0644   Flags: 0x0
Generation: 969212841    Version: 0x00000001
User:     0   Group:     0   Size: 42
File ACL: 0
Links: 1   Blockcount: 2
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x639c736d -- Fri Dec 16 15:32:29 2022
atime: 0x63b6f9b6 -- Thu Jan  5 18:24:22 2023
mtime: 0x639c736d -- Fri Dec 16 15:32:29 2022
BLOCKS:
(0):33793
TOTAL: 1
```

Προσέγγιση hexedit: Ξέρουμε ότι το inode 3666 είναι το 2ο inode του BG 2 και ότι το inode table του BG 2 βρίσκεται στο block 16389. Για να πάρουμε τα περιεχόμενα του inode 3666 εκτελούμε την εντολή `hexdump -s $((16389*1024 + 128)) -n 128 -C /dev/vdb`. Παίρνουμε τα εξής δεδομένα:

```
01001480  a4 81 00 00 2a 00 00 00  b6 f9 b6 63 6d 73 9c 63  |....*......cms.c|
01001490  6d 73 9c 63 00 00 00 00  00 00 01 00 02 00 00 00  |ms.c............|
010014a0  00 00 00 00 01 00 00 00  01 84 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 a9 03 c5 39  00 00 00 00 00 00 00 00  |.......9........|
010014f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

Τα δεδομένα αυτά ταυτίζονται απόλυτα με τα δεδομένα που είδαμε στην άλλη προσέγγιση. Για παράδειγμα, τα bytes 16-19 που αντιστοιχούν στην ημερομηνία τελευταίας τροποποίησης έχουν τιμή 0x639c736d, η οποία ταυτίζεται με την τιμή που βλέπουμε στην άλλη προσέγγιση.

1.31

Προσέγγιση tools: Στο αποτέλεσμα της προηγούμενης εντολής, βλέπουμε τον αριθμό του πρώτου (και μοναδικού) data block για το αρχείο μας. Αυτός είναι ο 33793.

Προσέγγιση hexedit: Από τα δεδομένα του inode που είδαμε στο 2.30, παίρνουμε την τοποθεσία του πρώτου Data block από τα bytes 40-43 (Direct Data Block Pointer 1). Η τοποθεσία είναι το block 0x8401 = 33793.

1.32

2.32: Προσέγγιση tools: Εκτελούμε την εντολή `stat /mnt/dir2/helloworld` και στο πεδίο "Size", βλέπουμε ότι το αρχείο έχει μέγεθος 42 bytes.

Προσέγγιση hexedit: Από τα δεδομένα του inode που είδαμε στο 2.30, παίρνουμε το μέγεθος του αρχείου από τα bytes 4-7. Το μέγεθος είναι ίσο με 0x2a = 42 bytes.

Προσέγγιση tools: Εκτελούμε την εντολή `cat /mnt/dir2/helloworld`. Τα δεδομένα του αρχείου είναι `Welcome to the Mighty World of Filesystems`.

Προσέγγιση hexedit: Γνωρίζουμε ότι το data block του αρχείου είναι το 33793 και ότι το αρχείο έχει μέγεθος 42 bytes. Οπότε διαβάζουμε τα πρώτα 42 bytes του block αυτού με την εντολή `hexdump -s $((33793*1024)) -n 42 -C /dev/vdb`. Τα δεδομένα που παίρνουμε (σε ASCII) είναι τα ίδια με πριν. Θα μπορούσαμε να διαβάσουμε και ολόκληρο το block καθώς τα υπόλοιπα bytes θα είναι σίγουρα 0. Αυτό συμβαίνει γιατί δεν μπορεί ένα block να το μοιράζονται πολλά αρχεία.

Άσκηση 2

Η εικόνα fsdisk2.img

2.1

Προσθέτουμε νέα γραμμή στο `exec` του `utopia.sh`:
`-drive file=./fsdisk2-4acfb06a7.img,format=raw,if=virtio`

Την προσαρτούμε στον κατάλογο `/mnt` με την παρακάτω εντολή: `mount -t ext2 /dev/vdc /mnt`

2.2

Εφόσον έχουμε προσαρτήσει το δίσκο στον κατάλογο `/mnt`, η εντολή που εκτελούμε είναι η `touch /mnt/file1`.

2.3

Η εντολή δεν πέτυχε. Επέστρεψε το μήνυμα "No space left on device".

2.4

Η touch προσπάθησε να εκτελέσει την παρακάτω κλήση συστήματος για να ανοίξει (και ταυτόχρονα να δημιουργήσει με την σημαία O_CREAT) το αρχείο `/mnt/file1`:

```
openat(AT_FDCWD, "/mnt/file1", O_WRONLY|O_CREAT|O_NOCTTY|O_NONBLOCK, 0666)
```

Η κλήση αυτή απέτυχε και επέστρεψε τον κωδικό λάθους "ENOSPC", ο οποίος υποδεικνύει ότι δεν υπάρχει χώρος στην συσκευή.

2.5

Προσέγγιση tools: Για να βρούμε το πλήθος των αρχείων που περιέχονται στο σύστημα αρχείων εκτελούμε την εντολή `find /mnt/ -type f | wc -l`. Η εντολή αυτή επιστρέφει το 4868 ως τον αριθμό των αρχείων. Για να βρούμε το πλήθος των καταλόγων, εκτελούμε την εντολή `find /mnt/ -type d | wc -l`. Η εντολή αυτή επιστρέφει το 259 ως τον αριθμό των καταλόγων. Η εντολή αυτή μετρά και τον ριζικό κατάλογο, και τον κατάλογο `lost+found`.

Τα παραπάνω αποτελέσματα μπορούμε να τα βρούμε και μέσω της εντολής `dumpe2fs`. Εκτελούμε την εντολή `dumpe2fs /dev/vdc`. Βλέπουμε για κάθε block group τον αριθμό των χρησιμοποιούμενων directories. Έχουμε 3 groups που έχουν 84, 83 και 92 directories αντίστοιχα, σύνολο 259.

Για τα αρχεία, εργαζόμαστε ως εξής: Έχουμε τον συνολικό αριθμό των inodes που είναι ίσος με 5136. Από αυτόν αφαιρούμε τον αριθμό των ελεύθερων inodes που είναι 0, τον αριθμό των καταλόγων που είναι 259 και το 9 (θα εξηγηθεί στη συνέχεια), οπότε καταλήγουμε στον αριθμό 4868 που βρήκαμε και πριν.

Το 9 προκύπτει ως εξής. Γνωρίζουμε ότι τα πρώτα 11 inodes χρησιμοποιούνται για ειδικούς σκοπούς από το ΣΑ. Μάλιστα το 2ο είναι για το root directory ενώ το 11ο είναι για το lost+found. Η find όμως βλέπει και το root directory και το lost+found, άρα αδυνατεί να μετρήσει μόνο 9 από τα ειδικά inodes. Οπότε, για να καταλήξουμε στον ίδιο αριθμό πρέπει να αφαιρέσουμε το 9 από τον αριθμό των εναπομείναντων inodes.

Προσέγγιση hexedit: Εργαζόμαστε όπως στην δεύτερη περίπτωση προηγουμένως. Γνωρίζουμε ότι σε κάθε group descriptor, στα bytes 16-17 υπάρχει ο αριθμός των συνολικών χρησιμοποιούμενων directories στο συγκεκριμένο group. Γνωρίζουμε ότι οι descriptors περιέχονται στο δεύτερο block του group 0 και ο καθένας έχει μέγεθος 32 bytes. Βρίσκουμε τον συνολικό αριθμό των groups όπως στο ερώτημα 2.16, και ο αριθμός αυτός προκύπτει ίσος με 3. Οπότε, βρίσκουμε τον αριθμό των directories για κάθε block μέσω των εντολών `hexdump -s $((2*1024 + 16)) -n 1024 -C /dev/vdc`, `hexdump -s $((2*1024 + 32 + 16)) -n 1024 -C /dev/vdc`, `hexdump -s $((2*1024 + 2*32 + 16)) -n 1024 -C /dev/vdc`, διαβάζοντας τα byte 16-17 του κάθε descriptor. Ο συνολικός αριθμός των directories προκύπτει ίσος με 259.

Στη συνέχεια, βρίσκουμε τον συνολικό αριθμό των inodes μέσω της εντολής `hexdump -s $((1024)) -n 4 -C /dev/vdc` (τα πρώτα 4 bytes του superblock). Αυτά είναι ίσα με 5136. Βρίσκουμε τον αριθμό των διαθέσιμων inodes στο σύστημα μέσω της εντολής `hexdump -s $((1024 + 16)) -n 4 -C /dev/vdc` (βλ. 2.12). Αυτά είναι ίσα με 0. Οπότε έχουμε 5136 χρησιμοποιούμενα inodes. Από αυτά αφαιρούμε τον αριθμό των directories (259) οπότε καταλήγουμε με τον αριθμό 4877. Από αυτά τα 4868 είναι σίγουρα κανονικά αρχεία, και τα υπόλοιπα 9 είναι ειδικά inodes που χρησιμοποιεί το ΣΑ (9 και όχι 11 γιατί το `root` και το `lost+found` περιλαμβάνονται στα 259 directories).

2.6

Προσέγγιση tools: Εκτελούμε την εντολή `dumpe2fs /dev/vdc`. Βλέπουμε στο πεδίο "Block count:" το συνολικό αριθμό των block. Αυτά είναι 20480. Στο πρώτο group, τα superblock, descriptors, bitmaps και inode table παίρνουν τα πρώτα 218 blocks, οπότε έχουμε 218 blocks για metadata. Επίσης, τα blocks 219-405 έχουν δεδομένα μέσα τους οπότε έχουμε 187 blocks για data. Τέλος, έχουμε και 7787 ελεύθερα blocks (406-8192). Ομοίως, το 2ο group έχει 218 blocks για metadata, 83 blocks για data και 7891 ελεύθερα blocks. Το 3ο group έχει 218 blocks για metadata, 0 blocks για data και 3877 ελεύθερα blocks. Οπότε, συνολικά, έχουμε 654 blocks για metadata και 19826 blocks για data (εκ των οποίων το 1 είναι το boot block, τα 270 χρησιμοποιούμενα από δεδομένα και τα 19555 είναι διαθέσιμα).

Προσέγγιση hexedit: Εκτελούμε την εντολή `hexdump -s $((1024 + 4)) -n 1024 -C /dev/vdc` για να διαβάσουμε τον συνολικό αριθμό των blocks (bytes 4-7 του superblock). Αυτά είναι 20480. Έχουμε 3 groups (το βρήκαμε στην προηγούμενη ερώτηση). Γνωρίζουμε ότι κάθε group έχει 1 block για το superblock και 2 blocks για τα bitmaps συνολικά. Θέλουμε να βρούμε το μέγεθος του descriptor table και του inode table για το κάθε group ώστε να προσδιορίσουμε το συνολικό μέγεθος των metadata.

Το μέγεθος του inode table μπορούμε να το βρούμε από το superblock ως εξής. Στα bytes 88-89 βρίσκουμε το μέγεθος του inode. Τα διαβάζουμε με την εντολή `hexdump -s $((1024 + 88)) -n 2 -C /dev/vdc` και το μέγεθος προκύπτει ίσο με 128 bytes. Στη συνέχεια, διαβάζουμε τα bytes 40-43 του superblock με την εντολή `hexdump -s $((1024 + 40)) -n 4 -C /dev/vdc`. Εκεί, βρίσκουμε τον αριθμό των inodes σε κάθε group. Αυτός είναι ίσος με 1712. Πολλαπλασιάζουμε το 128 με το 1712, οπότε προκύπτει ότι το μέγεθος του inode table σε κάθε group είναι 219136 bytes ή 214 blocks.

Μένει να βρούμε το μέγεθος του descriptor table. Αυτό το βρίσκουμε από το ίδιο το descriptor table ως εξής. Γνωρίζουμε ότι η αρχή του descriptor table είναι το 2ο block σε κάθε group. Επίσης, τα bytes 8-13 του κάθε descriptor είναι η τοποθεσία σε block της αρχής του inode table. Οπότε, αν αφαιρέσουμε από την τοποθεσία αυτή την τοποθεσία του superblock και το 2 (καθώς ανάμεσα βρίσκονται τα bitmaps) βρίσκουμε το μήκος του descriptor table. Περιγράφουμε τη διαδικασία μόνο για το πρώτο group.

Εκτελούμε την εντολή `hexdump -s $((2*1024 + 8 + 32)) -n 4 -C /dev/vdc` για να διαβάσουμε τα bytes 8-13 του descriptor 0. Έχουν τιμή 5. Το descriptor table του group 0 βρίσκεται στο block 2. 5-2-2 = 1 άρα το μέγεθος του descriptor table είναι ίσο με 1. Ομοίως, το μήκος όλων των descriptor tables είναι 1.

Οπότε, έχουμε για κάθε group 1 block για το superblock, 2 blocks για τα bitmaps, 214 blocks για το inode table και 1 block για το descriptor table, σύνολο 218 blocks για κάθε group. Άρα, ο δίσκος έχει σύνολο 654 blocks για metadata.

Τώρα μένει να βρούμε την κατανομή διαθέσιμων και χρησιμοποιούμενων data blocks για κάθε group. Κάθε group έχει 8192 blocks (bytes 32-35 του superblock, εντολή `hexdump -s $((1024 + 32)) -n 4 -C /dev/vdc`). Από αυτά τα 218 είναι metadata και τα 7974 είναι data blocks. Τα bytes 12-13 του κάθε descriptor περιέχουν τον αριθμό των free data blocks, οπότε μπορούμε να βρούμε την κατανομή που αναζητούμε.

Για το πρώτο group, τα free data blocks είναι 7787 (εντολή `hexdump -s $((2*1024 + 12)) -n 2 -C /dev/vdc`), για το δεύτερο είναι 7891 (εντολή `hexdump -s $((2*1024 + 32 + 12)) -n 2 -C /dev/vdc`) και για το 3ο είναι 3877 (εντολή `hexdump -s $((2*1024 + 64 + 12)) -n 2 -C /dev/vdc`).

Οπότε, το 1ο group έχει 218 blocks για metadata, 187 blocks για data και 7787 ελεύθερα blocks. Το 2ο group έχει 218 blocks για metadata, 83 blocks για data και 7891 ελεύθερα blocks. Το 3ο group έχει 218 blocks για metadata, 0 blocks για data και 3877 ελεύθερα blocks. Οπότε, συνολικά, έχουμε 654 blocks για metadata και 19826 blocks για data (εκ των οποίων το 1 είναι το boot block, τα 270 χρησιμποιούμενα από δεδομένα και τα 19555 είναι διαθέσιμα).

2.7

Ο δίσκος έχει μέγεθος 51MB (αφού γίνει mount).

Προσέγγιση tools: Εκτελούμε την εντολή `df -T -H /dev/vdc`. Το αποτέλεσμα της εντολής αυτής είναι το παρακάτω:

```
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/vdc       ext2   21M  277k   21M   2% /mnt
```

Προσέγγιση hexedit: Ο συνολικός αριθμός των block είναι 20480 όπως είδαμε πριν. Το μέγεθος του κάθε block είναι 1024 bytes (τα bytes 24-27 του superblock είναι 0 οπότε το block size είναι $1024 \ll 0 = 1024$). Τα bytes αυτά τα διαβάζουμε με την εντολή `hexdump -s $((1024 + 24)) -n 4 -C /dev/vdc`. Πολλαπλασιάζουμε και προκύπτει το μέγεθος του δίσκου ίσο με 20971520 bytes ($\sim$21M).

2.8

Όπως είδαμε στο ερώτημα 3.6, έχουμε σύνολο 19555 διαθέσιμα blocks.

2.9

Ο δίσκος μας δεν έχει διαθέσιμα inodes. Έχουμε ακριβώς όσα αρχεία χωράνε στα inode tables του δίσκου μας. Τα δεδομένα όμως αυτών των αρχείων δεν καταλαμβάνουν όλο το διαθέσιμο χώρο του δίσκου.

Άσκηση 3

Η εικόνα fsdisk3.img

3.1

Το `fsck` (συγκεκριμένα το `e2fsck`) είναι ένα εργαλείο στο Linux που μπορεί να ανιχνεύσει και πιθανώς να διορθώσει αλλοιώσεις ενός συστήματος αρχείων.

3.2

Αλλοιώσεις στο σύστημα αρχείων θα μπορούσαν να προκληθούν από πληθώρα παραγόντων. Αρχικά αν προκληθούν σφάλματα στην εκκίνηση ή τερματισμό του συστήματος οι διαδικασίες εγγραφής μπορεί να αποτύχουν και τα αρχεία να μην αποθηκευθούν σωστά. Επιπλέον, σε περίπτωση ελαττωματικού υλισμικού (π.χ. ελαττωματική RAM) οι διαδικασίες εγγραφής και ανάγνωσης πιθανόν να μην γίνονται σωστά με απότελεσμα την απώλεια δεδομένων. Ακόμη, εάν γίνεται χρήση του συστήματος αρχείων με απενεργοποιημένο το journaling. Επίσης η εκτέλεση κάποιου malware μπορεί να τροποποιήσει ή/και να διαγράψει αρχεία. Τέλος υπάρχει η περίπτωση κάποιου bug στο λογίσμικο ή και λάθους από τον χρήστη. Δέκα πιθανές αλλοιώσεις αναφέρονται παρακάτω : 1. Λανθασμένος αριθμός ελεύθερων block στο superblock.
2. Λανθασμένος αριθμός ελεύθερων block στο superblock.
3. Αλλοιώσεις στο block bitmap, ένα block που δεν είναι σε χρήση φαίνεται ότι χρησιμοποιείται και το αντίστροφο.
4. Λάθη στο inode bitmap, ένα inode που δεν είναι σε χρήση φαίνεται ότι χρησιμοποιείται και το αντίστροφο.
5. Αλλοιώσεις σε directory entry, για παράδειγμα λανθασμένη δομή (απουσία directory "." ή/και "..").
6. Ελλιπής συνδεσιμότητα των directory entries, αν απουσιάζει η εγγραφή "/dir-2" δεν μπορούμε να βρούμε την εγγραφή "/dir-2/eg".
7. Ένα directory entry μπορεί να δείχνει σε inode που δεν υπάρχει.
8. Λάθος μέγεθος αρχείου στο inode με αποτέλεσμα να γίνει truncate.
9. Ο αριθμός αναφορών σε ένα inode μπορεί να είναι διφορετικός από το reference count.
10. Τα δικαιώματα χρήστη δεν είναι αυτά που αναγράφονται.

3.3

Αρχικά χρησιμοποιούμε την εντολή `e2fsck -n /dev/vdd` για να εντοπίσουμε όλες τις αλλοιώσεις στο σύστημα αρχείων χωρίς να τις διορθώσουμε. Έπειτα χρησιμοποιούμε την εντολή `e2fsck -y /dev/vdd -z undo`. Ο δίσκος μας είναι ο `/dev/vdd` και δημιουργούμε ένα undo file σε περίπτωση που θέλουμε να γυρίσουμε τον δίσκο στην πρότερή του κατάσταση. Η πρώτη αλλοίωση που εντοπίστηκε είναι ότι η πρώτη εγγραφή του inode 1717 (directory inode που αντιστοιχεί στο `/dir-2`) δεν είναι η σωστή (αντί για "." έχουμε "BOO". Η δεύτερη αλλοίωση αφορά τις αναφορές στα inode, συγκεκριμένα στο inode 3425 το reference count είναι 1 ενώ θα έπρεπε να είναι 2. Η τρίτη αλλοίωση εντοπίζεται στο block bitmap, όπου υπάρχει διαφορά στο 34ο bit (+34) και

αντί για την τιμή 1 έχει 0 (ενώ το block 34 χρησιμοποιείται). Η τέταρτη αλλοίωση αφορά το πλήθος ελεύθερων block στο group #0, όπου εμφανίζεται η τιμή 7957 ενώ τα ελεύθερα block είναι 7958 (η αλλοίωση αυτή διορθώνεται με την αλλαγή του bitmap και για αυτό δεν εμφανίζεται στο δεύτερο τρέξιμο). Η πέμπτη αλλοίωση αφορά το πλήθος ελεύθερων block σε όλο το σύστημα αρχείων, όπου το free block count έχει την τιμή 925906233, ενώ τα ελεύθερα block είναι 19800.

3.4

Επαναφέρουμε τον δίσκο στην αρχική του εικόνα. Χρησιμοποιούμε το undo file που δημιουργήσαμε προηγουμένως μέσω της εντολής `e2undo undo /dev/vdd`.

3.5

Πρώτη αλλοίωση: Το πρώτο dentry στο directory που έχει inode number 1717 (το `/dir-2`) έχει όνομα "BOO" ενώ θα έπρεπε να έχει όνομα ".".

Αρχικά εντοπίζουμε πόσα inodes έχει το κάθε group. Αυτό το κάνουμε μέσω της εντολής `hexdump -s $((1024+40)) -n 1024 -C /dev/vdd`. Προκύπτει ότι κάθε group έχει 1712 inodes. Άρα, το inode 1717 είναι το 5ο inode του 2ου group.

Στη συνέχεια εντοπίζουμε την τοποθεσία του inode table για το 2ο group. Αυτό το κάνουμε διαβάζοντας τον 2ο descriptor από το descriptor table του 1ου group. Χρησιμοποιούμε την εντολή `hexdump -s $((2*1024+32)) -n 32 -C /dev/vdd`. Τα bytes 8-12 του descriptor περιέχουν την τοποθεσία 0x2005 (block 8197) που είναι και η αρχή του inode table του 2ου group.

Παίρνουμε τα δεδομένα του inode αυτού μέσω της εντολής `hexdump -s $((8197*1024 + 4*128)) -n 128 -C /dev/vdd`. Τα bytes 40-43 είναι ο direct block pointer 1 και έχουν τιμή 0x20dc. Άρα, το πρώτο data block του inode αυτού (άρα και του `/dir-2`) είναι το 8412.

Διαβάζουμε τα δεδομένα του block αυτού μέσω της εντολής `hexdump -s $((8412*1024)) -n 1024 -C /dev/vdd`. Το πρώτο dentry είναι το εξής: b5 06 00 00 0c 00 03 00 42 4f 4f 00. Γνωρίζουμε ότι θα έπρεπε να είναι b5 06 00 00 0c 00 01 00 2e 00 00 00, ώστε να αντιστοιχούν στο directory ".", δηλαδή το `/dir-2`. Γνωρίζουμε ότι το `/dir-2` έχει inode number 1717 (0χ06b5, εξού και τα πρώτα 4 bytes), έχει dentry size 12 (το ελάχιστο), έχει μήκος ονόματος 1, έχει άγνωστο τύπο και το όνομά του είναι ".". Οπότε, αλλάζουμε τα bytes αυτά στις σωστές τιμές ακολουθώντας την παρακάτω διαδικασία:

Εκτελούμε την εντολή `xxd /dev/vdd > dump` για να δημιουργήσουμε ένα hexdump του δίσκου. Ανοίγουμε στο vi το αρχείο `dump`, πάμε στην τοποθεσία που περιέχει τα bytes αυτά και γράφουμε τις σωστές τιμές. Κάνουμε overwrite το δίσκο με τα νέα δεδομένα από το αρχείο `dump` μέσω της εντολής `xxd -r dump > /dev/vdd` και τρέχουμε την εντολή `e2fsck -n /dev/vdd` (dry run) για να δούμε ότι η αλλοίωση που διορθώσαμε δεν εμφανίζεται πια.

Δεύτερη αλλοίωση: Το reference count του inode 3425 είναι 1 ενώ πρέπει να είναι 2.

Εφόσον κάθε group έχει 1712 inodes, το 3425 είναι το 1ο inode του 3ου group. Βρίσκουμε την τοποθεσία του inode table του 3ου group από τον 3ο descriptor του descriptor table μέσω της εντολής `hexdump -s $((2*1024 + 2*32)) -n 32 -C /dev/vdd`. Τα bytes 8-11 περιέχουν την τοποθεσία που αναζητούμε και αυτή είναι 0x4005 (block 16389).

Διαβάζουμε το inode 3425 μέσω της εντολής `hexdump -s $((16389*1024)) -n 128 -C /dev/vdd`. Τα bytes 26-27 περιέχουν το reference count και παρατηρούμε ότι είναι 1 ενώ θα έπρεπε να είναι 2. Ακολουθούμε την ίδια διαδικασία με πριν για να αλλάξουμε την τιμή αυτή σε 2 (εντολή xxd, κτλ). Τρέχουμε ξανά την εντολή `e2fsck -n /dev/vdd` (dry run) για να δούμε ότι η αλλοίωση που διορθώσαμε δεν εμφανίζεται πια.

Τρίτη αλλοίωση: Το 34ο bit στα block bitmaps είναι 0 ενώ θα έπρεπε να ήταν 1. Δηλαδή το block 34 είναι σε χρήση αλλά το bitmap στο οποίο αντιστοιχεί το block 34 έχει 0 στο bit 34. Κάθε group έχει 8192 blocks (εντολή `hexdump -s $((1024+32)) -n 1024 -C /dev/vdd`) οπότε το bitmap στο οποίο αντιστοιχεί το block

34 είναι το πρώτο bitmap (BG 0).

Βρίσκουμε την τοποθεσία του block bitmap μέσω του πρώτου block descriptor. Διαβάζουμε τα περιεχόμενά του από το πρώτο descriptor table με την εντολή `hexdump -s $((2*1024)) -n 32 -C /dev/vdd`. Τα bytes 0-4 περιέχουν την τοποθεσία του block bitmap και αυτή είναι η 0x03 (block 3).

Διαβάζουμε τα περιεχόμενα του bitmap μέσω της εντολής `hexdump -s $((3*1024)) -n 1024 -C /dev/vdd`. Πράγματι το 34ο bit είναι 0 (το 5ο byte είναι 0xfd). Αλλάζουμε το byte αυτό σε 0xff και τρέχουμε την εντολή `e2fsck -n /dev/vdd` (dry run) για να δούμε ότι η αλλοίωση που διορθώσαμε δεν εμφανίζεται πια.

Τέταρτη αλλοίωση: Η τέταρτη αλλοίωση ανέφερε ότι το free block count για το group 0 είναι λάθος καθώς η τιμή στο superblock είναι 7957 ενώ πρέπει να είναι 7958. Η αλλοίωση αυτή δεν εμφανίζεται πια καθώς σχετιζόταν με το λάθος bit στο bitmap.

Πέμπτη αλλοίωση: Το συνολικό free blocks count είναι λάθος καθώς έχει την τιμή 925906233 ενώ θα έπρεπε να είναι 19800. Η τιμή αυτή περιέχεται στο superblock. Διαβάζουμε τα δεδομένα του superblock μέσω της εντολής `hexdump -s $((1024)) -n 1024 -C /dev/vdd`. Τα bytes 12-16 περιέχουν το συνολικό free blocks count. Τα bytes αυτά έχουν τιμή 0x37303539 ενώ πρέπει να έχουν τιμή 0x4d58. Αλλάζουμε τα bytes αυτά και τρέχουμε την εντολή `e2fsck -n /dev/vdd` (dry run) για να δούμε ότι η αλλοίωση που διορθώσαμε δεν εμφανίζεται πια.

Τρέχουμε μια τελευταία φορά την εντολή `e2fsck -n /dev/vdd` και ο δίσκος μας είναι καθαρός.