

## 3η εργαστηριακή άσκηση

# FileSystems

Βασίλης Κοτρώνης 03121432,

Βασίλης Οικονόμου 03120020

## Πρώτο Μέρος

### 1.Εισαγωγή:

Η άσκηση αυτή αφορά την εξοικείωση με τα συστήματα αρχείων τόσο από την μεριά του χρήστη όσο και από την μεριά του πυρήνα.

Στο 1ο μέρος μας δόθηκαν τρία disk images πάνω στα οποία λύσαμε κάποια challenges χρησιμοποιώντας εργαλεία CLI αλλά και hex editor.

Στο 2ο μέρος θα υλοποιήσουμε κάποιες μεθόδους σε ένα σύστημα αρχείων ext2-lite.

### 2.disk1:

#### 2.1

Στο script `utopia.sh` προσθέσαμε μεταβλητή `DISK_IMAGE="/fsdisk1.img"`, έπειτα error handling για την ύπαρξη της μεταβλητής αυτής και τέλος στο `exec` εντολή `-drive file=$DISK_IMAGE, format=raw, if=virtio`.

Έτσι προσθέτουμε στο VM συσκευή `fsdisk1.img`.

#### 2.2

`cli: lsblk -> size=50M`

`hexedit: hexedit /dev/vdb -> βλέπουμε συνολικό μέγεθος 0x32000000= 50M`

#### 2.3

`cli: file -s /dev/vdb -> ext2 filesystem`

`hexedit: Στο superblock, στα offset 56-57 βλέπουμε 0xef53 -> ext2 filesystem`

## 2.4

cli: dumpe2fs /dev/vdb -> 2024-12-19 19:12:02...

hexedit: ?

## 2.5

cli: ίδια εντολή με πάνω

hexedit: Στα offset 44-47 του superblock

## 2.6

cli: dumpe2fs /dev/vdb -> /cslab-bunker

hexedit: Στα offset 136-199 του superblock

## 2.7

cli: dumpe2fs -> 2024-12-19 19:12:17...

hexedit: offset 48-51 superblock

## 2.8

Τα block σε ένα σύστημα αρχείων είναι η μικρότερη μονάδα αποθήκευσης δεδομένων.

Επίσης το block είναι η μεγαλύτερη ποσότητα δεδομένων που μπορεί να μεταφερθεί σε μία διαδικασία I/O. Στατικό μέγεθος, virtual.

## 2.9

cli: dumpe2fs -> block size = 1024

hexedit: offset 24 του superblock:  $\log \text{block size} = \log_2(\text{blocksize}) = 10 \rightarrow \text{block size} = 1024$

## 2.10

Είναι δομή η οποία αναπαριστά κάποια οντότητα μέσα στο σύστημα αρχείων που καταλαμβάνει χώρο. Περιέχει μια σύνδεση στο block το οποίο το περιέχει.

## 2.11

cli: dumpe2fs -> inode size = 128

hexedit: offset 88-89 superblock ->  $0x0080 = 128$

## 2.12

cli: dumpe2fs -> free blocks = 49552, free inodes = 12810

hexedit: offset 12-15 -> free blocks  $0xC190$ , offset 16-19 -> free inodes  $0x320A = 12810$

## 2.13

Περιλαμβάνει πληροφορίες για τα χαρακτηριστικά ενός συστήματος αρχείων.

## 2.14

Στην αρχή του συστήματος με offset 1024

## 2.15

Τα εφεδρικά superblock χρησιμοποιούνται σε περίπτωση corruption / override στο αρχικό superblock εξασφαλίζοντας έτσι σωστή λειτουργία του filesystem.

## 2.16

cli: dumpe2fs -> backup superblock:  $1 \times 8192 + 1$  ,  $2 \times 8192 + 1$  etc για group 1,2... αντίστοιχα.

hexedit: position = blockspergroup x blocksize x (i-1) + boot ( στο δεξαεδικο μετατροπη)

## 2.17

Τοποθετούμε blocks που σχετίζονται μεταξύ τους σε ένα μόνο Block group για να μειώσουμε τις αναζητήσεις για I/O.

## 2.18

Εξαρτάται από το μέγεθος του partition και του block.

## 2.19

cli: dumpe2fs -> 7 groups

hexedit: δεν υπάρχει θέση 0x3800400

## 2.20

Block descriptor περιγράφει Block group-> block bitmap, inode bitmap, inode table, free block count, free inode count, used dir count

## 2.21

Εφεδρικό για κύριο descriptor σε περίπτωση που αποτύχει.

## 2.22

Όμοια με τα εφεδρικά superblock απλά με +2 αντί για +1. ( 8194, 16386 ... κλπ)

## 2.23

Μας δείχνει αν ένα block του group χρησιμοποιείται ή είναι ελεύθερο.

όμοια με superblock αλλά σε θέση +3

Το inode bitmap δείχνει αν ένα Inode είναι σε χρήση ή όχι.

Θεση +4

## 2.24

Περιέχει όλα τα inodes, θέση +5

## 2.25

To inode περιέχει πληροφορίες για το αρχείο: permissions, type, user id, size, create time, modify time, mount time, group id, hardlink count, flags, block pointers etc.

Είναι αποθηκευμένα μέσα στα inode tables.

## 2.26

cli: dumpe2fs -> 8192 blocks per group, 1832 inodes per group

hexedit: offset 0-3 superblock->inode count= 0x3218=12824 / 7 groups = 1832 inodes/group

offset 4-7-> block count=0xC800=51200/7=8192 blocks per group.

## 2.27

cli: mkdir /mnt/vdb -> mount /dev/vdb /mnt/vdb -> cd -> stat helloworld -> 9162

hexedit: 5x1024 -> inode table.

inode 2= root directory : 0x1480: offset 40-43 -> block 234 -> 0x3A800

->dir2 inode = 9161 -> 1o inode στο block group 5-> 0x2801400-> helloworld -> inode=0x23CA=9162

## 2.28

block group = inode/inodes per group

## 2.29

cli: dumpe2fs -> inode table at 40965

hexedit: 2.27 -> inode table of block group 5

## 2.30

cli: debugfs /dev/vdb -> stat /dir2/helloworld -> inode info

hexedit: 0x02801480 - 0x028014F0 ( 128 bytes )

## 2.31

from 2.27 -> block 1025 of block group 0

## 2.32

cli: stat helloworld -> 42 bytes

hexedit: 2.27

## 2.33

cli: cat helloworl: "Welcome to the Mighty World of Filesystems"

hexedit:0x02100400-0x02100420

## 3. disk2:

### 3.1

`mount /dev/vdb /mnt/vdb`

### 3.2

`touch /mnt/vdb/file1`

### 3.3

Απέτυχε λόγω έλλειψης χώρου. `dumpe2fs-> free inodes=0`

### 3.4

`open()` με κωδικό `ENOSPC`

### 3.5

cli: `du -a /mnt/vdb | wc -l -> 5127` total files and dir used ( 5136 because 9 are used by the filesystem)

`find /mnt/vdb -type f | wc -l -> 4868` files

`find /mnt/vdb -type d | wc -l -> 259` directories

hexedit: bg descriptor offset 16-17 = used dir count

για κάθε block group -> 0: 84 dir , 1: 83 dir , 2: 92 dir, total 259 dir

offset 0-3 superblock inodes=0x1410=5136 , free inodes =0

$5136 - 9 - 259 = 4868$  files

### 3.6

cli: `du -sh /mnt/vdb -> 270K`

hexedit: blocks - free blocks = 925 blocks used

### 3.7

cli: `df -h /mnt/vdb -> 20M`

hexedit: κάτω βλέπουμε 0x01400000 = 20M

### 3.8

cli: `dumpe2fs-> available blocks = 19555`

`df -h /dev/vdb -> 98% free space`

hexedit: offset 12-15 superblock -> free blocks =19555

### 3.9

cli: `dumpe2fs-> free inodes =0`

hexedit: offset 16-19 free inodes =0

## 4.disk3:

### 4.1

fsck για επιδιορθώσεις στο σύστημα αρχείων

### 4.2

(i) πιθανές αιτίες: improper shutdown , φυσική φθορά, malware , λάθη user, bugs  
(ii) πιθανές αλλοιώσεις: λάθος δικαιώματα, απώλεια superblock, αλλοιώσεις superblock, απώλεια blocks, λάθος Inodes, λάθος Links , πολλαπλή χρήση blocks, orphan files, destroyed file headers, repeated entries in directories

### 4.3

e2fsck: 1. First entry 'BOO' (inode=1717) in dir inode 1717(/dir-2) should be '  
2. inode 3425 ref count is 1, should be 2  
3. block bitmap differences +34  
4. free blocks count wrong for group #0 (7960, counted 7961)  
5. free blocks count wrong (926431538, counted=19800)

### 4.4

1. hexedit -> 0x00837000 αλλάζουμε το μέγεθος σε 01 και το περιεχόμενο σε 2E απο 42 2F 2F.  
2. 3425 belongs to group 2-> 16389 (0x01001400) offset 26: change 01 to 02  
3. στο 1ο block bitmap στη 0xC04 κάνουμε allocate το block 34 άρα κάνουμε D->f  
4. απο το 3 προστέθηκε επιπλέον block άρα διορθώθηκε  
5. στο superblock αλλάζουμε free blocks offset σε 58 4D 00 00 ( little endian για 19800)

### 4.5

μετά τις διορθώσεις δεν εμφανίζεται σφάλμα στο dry run.

## Δεύτερο Μέρος

Στο δεύτερο μέρος της άσκησης ασχοληθήκαμε με τα συστήματα αρχείων από τη μεριά του πυρήνα του Linux. Κληθήκαμε να υλοποιήσουμε κάποιες συναρτήσεις που αφορούν στο σύστημα αρχείων ext2-lite που μας δόθηκε (μια απλοποιημένη έκδοχή του ext2).

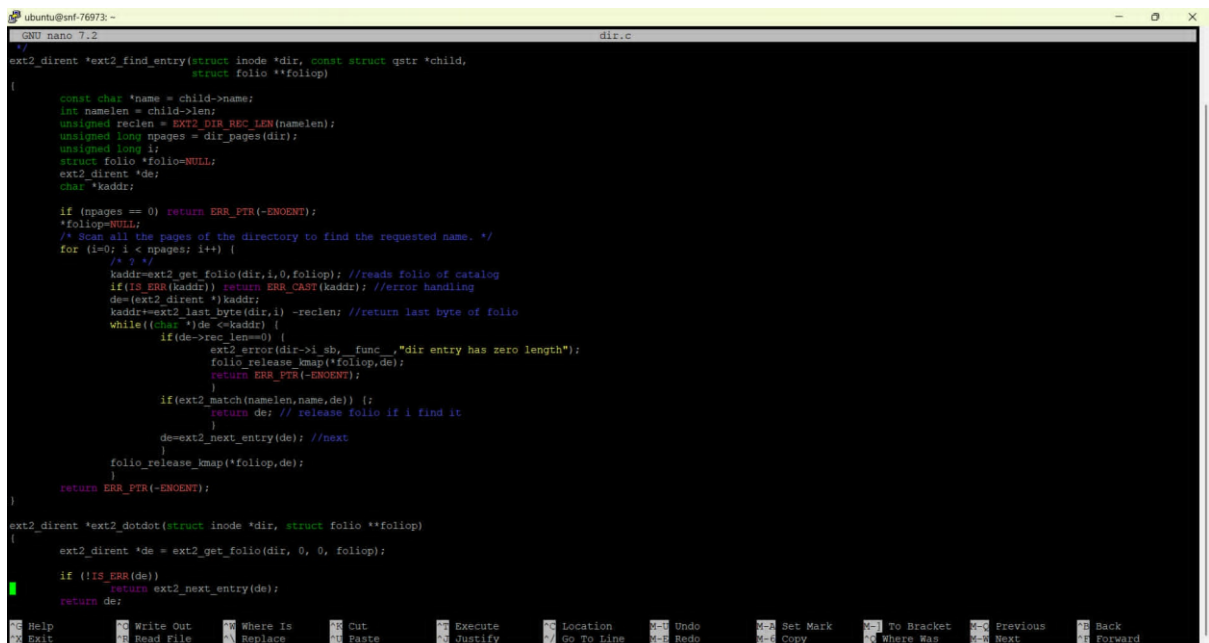
### 1. συναρτήσεις `init_ext2_fs` και `exit_ext2_fs` (\*\* στο τέλος)

### 2. Συνάρτηση `ext2_find_entry`

Στο αρχείο `dir.c` η οποία, σκανάρει τη λίστα με τα directory entries ενός δοσμένου καταλόγου.

Αναζητά ένα αρχείο/κατάλογο ανάμεσα στα entries ενός συγκεκριμένου directory. Από τα ορίσματα της μαθαίνει: το όνομα του αρχείου/καταλόγου προς αναζήτηση, το μήκος του ονόματος και κατά συνέπεια το μήκος ενός directory entry με αυτό το όνομα (χρησιμοποιώντας το `EXT2_DIR_REC_LEN`) και τον αριθμό σελίδων που περιλαμβάνουν τα entries του καταλόγου (data blocks του `dir`).

Ύστερα, καλεί με την σειρά κάθε σελίδα και με την `ext2_get_folio()` ελέγχει διαδοχικά τα directory entries, μέχρι να βρει το ζητούμενο. Η διάσχιση βασίζεται στην `ext2_next_entry()` και η σύγκριση πραγματοποιείται μέσω της `ext2_match()`. Αν διανύσει ολόκληρη τη σελίδα (ή πριν η συνάρτηση τερματίσει), τότε την αποδεσμεύει με τη `folio_release_kmap()`.



```
GNU nano 7.2 dir.c
ext2_direct *ext2_find_entry(struct inode *dir, const struct qstr *child,
                           struct folio **folio)
{
    const char *name = child->name;
    int namelen = child->len;
    unsigned reclen = EXT2_DIR_REC_LEN(namelen);
    unsigned long npages = dir_pages(dir);
    unsigned long i;
    struct folio *folio=NULL;
    ext2_direct *de;
    char *kaddr;

    if (npages == 0) return ERR_PTR(-ENOENT);
    *folio=NULL;
    /* Scan all the pages of the directory to find the requested name. */
    for (i=0; i < npages; i++) {
        /* i */
        kaddr=ext2_get_folio(dir,i,0,folio); //reads folio of catalog
        if(!IS_ERR(kaddr)) return ERR_CAST(kaddr); //error handling
        de=(ext2_direct *)kaddr;
        kaddr=ext2_last_byte(dir,i) - reclen; //return last byte of folio
        while((char *)de < kaddr) {
            if(de->rec_len==0) {
                ext2_error(dir->i_sb, func, "dir entry has zero length");
                folio_release_kmap(*folio,de);
                return ERR_PTR(-ENOENT);
            }
            if(ext2_match(namelen,name,de)) {
                return de; // release folio if i find it
            }
            de=ext2_next_entry(de); //next
            folio_release_kmap(*folio,de);
        }
        return ERR_PTR(-ENOENT);
    }
}

ext2_direct *ext2_dotdot(struct inode *dir, struct folio **folio)
{
    ext2_direct *de = ext2_get_folio(dir, 0, 0, folio);
    if (!IS_ERR(de))
        return ext2_next_entry(de);
    return de;
}
```

### `ext2_get_inode()`

Η `ext2_get_inode()` γράφει στην μνήμη ένα inode αφού πρώτα το εντοπίσει με βάση τον αριθμό του στον δίσκο. Ελέγχει ότι ο αριθμός inode που ζητείται είναι έγκυρος (2 ή 11-Total Inodes), και έπειτα υπολογίζει σε ποιο block group ανήκει. Μετά, με την `ext2_get_group_desc()` ανακτά τον block group descriptor του. Ύστερα υπολογίζει το offset του inode (σε bytes) στο inode table του συγκεκριμένου group. Με βάση αυτά, υπολογίζει τον αριθμό του block που το περιλαμβάνει, το φέρνει στην μνήμη και επιστρέφει τη διεύθυνση του inode στο buffer head(bh).

```
ubuntu@nrf-76973: ~
GNU nano 7.2                               inode.c
static struct ext2_inode *ext2_get_inode(struct super_block *sb, ino_t ino,
                                         struct buffer_head **p)
{
    struct buffer_head *bh;
    unsigned long block_group;
    unsigned long block;
    unsigned long offset;
    struct ext2_group_desc *gdp;
    unsigned long inodes_pg = EXT2_INODES_PER_GROUP(sb);
    int inode_sz = EXT2_INODE_SIZE(sb);
    unsigned long blocksize = sb->s_blocksize;

    *p = NULL;
    /* Check the validity of the given inode number. */
    if ((ino != EXT2_ROOT_INO && ino < EXT2_FIRST_INO(sb)) ||
        ino > le32_to_cpu(EXT2_SB(sb)->s_es->s_inodes_count))
        goto eio;

    /* Figure out in which block is the inode we are looking for and get
     * its group block descriptor. */
    /* * */
    block_group = (ino-1)/inodes_pg;
    gdp = ext2_get_group_desc(sb, block_group, NULL);
    if (!gdp) goto eio;
    /* Figure out the offset within the block group inode table */
    /* * */
    offset = ((ino-1)%inodes_pg)*inode_sz;
    block = le32_to_cpu(gdp->bg_inode_table) + (offset >> EXT2_BLOCK_SIZE_BITS(sb));
    if (! (bh = sb_bread(sb, block))) goto eio;
    /* Return the pointer to the appropriate ext2_inode */
    /* * */
    *p = bh;
    offset &= (blocksize-1);
    return (struct ext2_inode *) (bh->b_data + offset);
eio:
    ext2_error(sb, __func__, "bad inode number: %lu", (unsigned long)ino);
    return ERR_PTR(-EINVAL);
eio:
    ext2_error(sb, __func__, "unable to read inode block - inode=%lu, block=%lu",
               (unsigned long)ino, block);
    return ERR_PTR(-EIO);
}
```

## ext2\_iget()

Η ext2\_iget(), δεδομένου ενός inode number, (καλώντας δηλαδή την get\_inode, δημιουργεί ένα inode structure και το αρχικοποιεί με βάση τα πραγματικά πεδία του στο δίσκο. Τα πεδία που αφορούν τη δική μας υλοποίηση είναι τα

inode->i\_op

inode->i\_fop

inode->i\_mapping->a\_ops

- Σε περίπτωση που ο τύπος του inode είναι regular file, τα 2 πρώτα πεδία αντιστοιχίζονται στα structs ext2\_file\_inode\_operations και ext2\_file\_operations (τα οποία ορίζονται στο αρχείο file.c),
- Αν ο τύπος του inode είναι directory, αντιστοιχίζονται στα structs ext2\_dir\_inode\_operations και ext2\_dir\_operations (τα οποία ορίζονται στο namei.c)

Το 3ο πεδίο σε κάθε περίπτωση αντιστοιχίζεται στο struct address\_space\_operations ext2\_aops, που ορίζεται στο inode.c.



