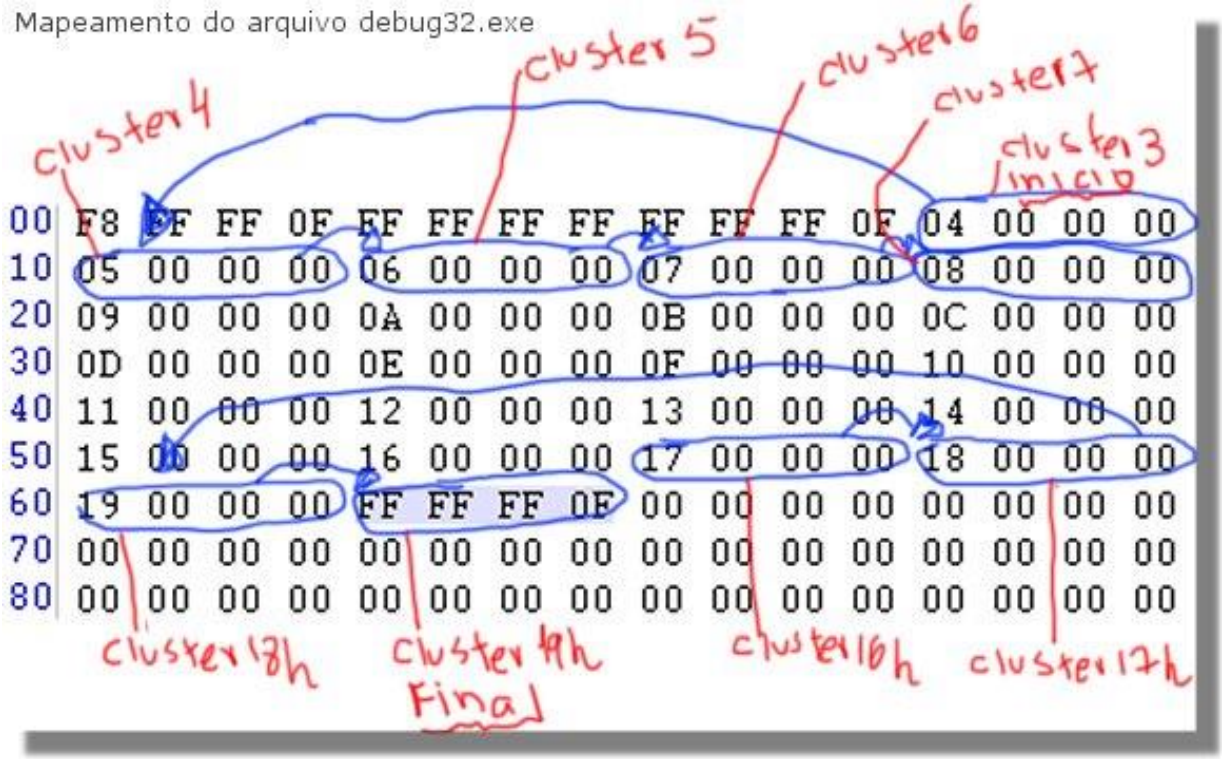
Περιεχόμενα

1.Εισαγωγή

Για την 2η εργαστηριακή άσκηση θα σας παρουσιάσουμε την δική μας προσπάθεια φτάνοντας μέχρι και το τρίτο ερώτημα στο οποίο μας ζητείται να δημιουργήσουμε ένα αρχείο καταγραφής (το οποίο η δική μας ομάδα το ονομάζει testFile.txt) . Στόχος της άσκησης ήταν να υλοποιήσουμε ένα αρχείο καταγραφής τροποποιήσεων για το σύστημα αρχείων FAT στην LKL (Linux Kernel Library). Η χρήση της LKL γίνεται με σκοπό την πρόσβασή μας στον πυρήνα του Linux, γιατί ο κώδικας του Linux εκτελείται σε επίπεδο πυρήνα, πράγμα που τον καθιστά αυτόνομο και έτσι δεν έχει πρόσβαση στο σύστημα αρχείων root. Αυτό έχει ως αποτέλεσμα να μην μπορούν να χρησιμοποιηθούν γνωστές βιβλιοθήκες της C από τον προγραμματιστή όπως malloc() και printf(). Ωστόσο, ο πυρήνας παρέχει παρόμοιες συναρτήσεις όπως kmalloc() και printk(), το οποίο χρησιμοποιούμε για την εκτύπωση δεδομένων στην οθόνη μας στο πρώτο ερώτημα και όχι μόνο. Πολλές λειτουργίες του συστήματος αρχείων FAT αποτελούνται από πολλαπλά στάδια τα οποία πρέπει να ολοκληρωθούν για να θεωρηθεί μία λειτουργία επιτυχημένη. Όμως μία απότομη διακοπή λειτουργίας του συστήματος, καθώς εκτελείται μία λειτουργία του που δεν έχει ολοκληρωθεί μπορεί να οδηγήσει το σύστημα αρχείων σε κατάσταση ασυνέπειας. Για αυτό το λόγο κάνουμε καταγραφή των λειτουργιών σε ένα αρχείο πριν την ενημέρωση των δομών του συστήματος, ώστε σε περίπτωση αποτυχίας ή σφάλματος να γίνει επανάληψη των λειτουργιών που βρίσκονται στο αρχείο καταγραφής. Όλα τα προγράμματα οδήγησης συστήματος αρχείων Linux υποστηρίζουν και τους τρεις τύπους FAT, δηλαδή FAT12, FAT16 και FAT32. Εκεί που διαφέρουν είναι η παροχή υποστήριξης για μεγάλα ονόματα αρχείων, πέρα από τη δομή ονόματος της αρχικής μορφής συστήματος αρχείων FAT δηλαδή, αλλά και στην παροχή σημασιολογίας αρχείων Unix που δεν υπάρχουν ως στάνταρ στη μορφή συστήματος αρχείων FAT, όπως δικαιώματα αρχείων. Τα προγράμματα οδήγησης του συστήματος αρχείων είναι αμοιβαία αποκλειστικά. Μόνο ένας μπορεί να χρησιμοποιηθεί για την προσθήκη οποιουδήποτε δεδομένου όγκου δίσκου ανά πάσα στιγμή. Έτσι, η επιλογή μεταξύ τους καθορίζεται από τα μεγάλα ονόματα αρχείων και τη σημασιολογία Unix που υποστηρίζουν και από ποια χρήση θέλει να κάνει κάποιος από τη χωρητικότητα του δίσκου.

Το σύστημα αρχείων FAT οργανώνει τα δεδομένα σε αρχεία και στη συνέχεια τα αποθηκεύει σε block της συσκευής αποθήκευσης (πχ. σκληρός δίσκος). Για την οργάνωση των αρχείων περιέχει καταλόγους οι οποίοι περιέχουν άλλα αρχεία ή άλλους καταλόγους ενώ το ίδιο το σύστημα δεν χρειάζεται να γνωρίζει το περιεχόμενο ενός αρχείου. Όσον αφορά τον δίσκο, στον οποίο αποθηκεύονται τα δεδομένα, είναι δομημένος σε clusters που περιέχουν συγκεκριμένο αριθμό απο sectors. Συνεπώς τα

δεδομένα αποθηκεύονται σε clusters. Στην αρχή του κάθε cluster βρίσκεται ένας δείκτης ο οποίος μας δείχνει ποιο θα είναι το επόμενο cluster στην σειρά.



Για παράδειγμα όπως φαίνεται και στην παραπάνω εικόνα βλέπουμε κυκλομένους τους clusters μέσα στους οποίους βρίσκονται οι sectors. Παρατηρώντας τους δείκτες καταλαβαίνουμε σε ποιον cluster βρισκόμαστε και ποιος είναι ο επόμενος. Δηλαδή ο cluster με δείκτη 17 είναι ο 16ος cluster ο οποίος μας δείχνει ότι ο επόμενος είναι 17ος με δείκτη στον 18ο. Όταν φτάσουμε στον cluster FF FF FF 0F καταλαβαίνουμε ότι εκείνος είναι ο τελευταίος και οτι εκεί τελειώνει το αρχείο μας, καθώς ο τελευταίος cluster έχει πάντα συγκεκριμένη τιμή (Το παραπάνω παράδειγμα είναι εικόνα από το Google).

2. Πρώτο Ερώτημα

Για να μπορέσουμε να κατανοήσουμε περαιτέρω τη λειτουργία του FAT προσθέσαμε κλήσεις της συνάρτησης printk() στις λειτουργίες για τη διαχείριση του superblock, των inodes, των αρχείων και των καταλόγων, οι οποίες είναι σημαντικές για την λειτουργία του FAT. Οι λειτουργίες του superblock ορίζονται στην δομή struct super_operations fat_sops του αρχείου fs/fat/inode.c. Επομένως αναζητήσαμε όλες τις λειτουργίες που ορίζονται στην συγκεκριμένη δομή χρησιμοποιώντας την εντολή ~/bin/search <string>, όπου string το όνομα της λειτουργίας που αναζητούμε, και προσθέσαμε στην αρχή του κώδικα της κάθε λειτουργίας ένα printk().

Για την λειτουργία fat_alloc_inode σειρα 750-751 στο fs/fat/inode.c:
/*Topothetisi prwtou printk se sinartish tou super struct*/
printk(KERN_INFO "Attention in function fat_alloc 1"); // minima gia idopoihsh xrhsth me skopo thn amesh prosoxh tou sthn sunarthsh

Για την λειτουργία fat_destroy_inode σειρα 769-770 στο fs/fat/inode.c:
/*Deutero pritnk se sinarthsh tou super struct*/
printk(KERN_INFO "Just destroyed inode 2"); /*enhmerwtiko minima pros ton xrhsth */

Για την λειτουργία fat_write_inode σειρα 936-937 στο fs/fat/inode.c:
/*Trito printk message pros ton xrhsth apo sinarthsh tou super stuct*/
printk(KERN_WARNING "Warning message from fat write 3");

Για την λειτουργία fat_evict_inode σειρα 659-660 στο fs/fat/inode.c:
/*Tetarto printk apo sinarthsh super struct*/
printk(KERN_INFO "fat evict 444");

Για την λειτουργία fat_put_super σειρα 733-734 στο fs/fat/inode.c:
/*Pempto printk apo sunarthsh super struct*/
printk(KERN_INFO "fat put 5");

Για την λειτουργία fat_statfs σειρα 854-855 στο fs/fat/inode.c:
/*ekto printk apo sinarthsh super struct*/
printk(KERN_INFO "fat statfs 6");

Για την λειτουργία fat_remount σειρα 835-836 στο fs/fat/inode.c:
/*evdomo printk sinarthshs super struct*/
printk(KERN_INFO "fat remount 7");

Για την λειτουργία fat_show_options σειρα 981-982 στο fs/fat/inode.c:

```
/*ogdoo printk apo sinarthsh super struct */
_____printk(KERN_INFO "fat show options 8");
```

Για τις λειτουργίες της μνήμης (χώρου διευθύνσεων) ορίζονται στην δομή struct address_space_operations fat_aops του αρχείου fs/fat/inode.c. ακολουθήσαμε την ίδια τακτική με τις λειτουργίες του superblock. Επομένως έχουμε:

Για την λειτουργία fat_readpage σειρά 208 στο fs/fat/inode.c.:

```
printk(KERN_INFO "fat_readpage 1");
```

Για την λειτουργία fat_readpages σειρά 215 στο fs/fat/inode.c.:

```
printk(KERN_INFO "fat readpages 2");
```

Για την λειτουργία fat_writepage σειρά 195 στο fs/fat/inode.c.:

```
printk(KERN_INFO " fat writepage 3");
```

Για την λειτουργία fat_writepages σειρά 202 στο fs/fat/inode.c.:

```
printk(KERN_INFO " fat writepages 4");
```

Για την λειτουργία fat_write_begin σειρά 234 στο fs/fat/inode.c.:

```
printk(KERN_INFO " fat write begin 5");
```

Για την λειτουργία fat_write_end σειρά 250 στο fs/fat/inode.c.:

```
printk(KERN_INFO " fat write end 6");
```

Για την λειτουργία fat_direct_IO σειρά 272 στο fs/fat/inode.c.:

```
printk(KERN_INFO "fat direct IO 7");
```

Για την λειτουργία _fat_bmap σειρά 327 στο fs/fat/inode.c.:

```
printk(KERN_INFO " fat bmap 8");
```

Για τις λειτουργίες των εγγραφών FAT που ορίζονται στην δομή struct fatent_operations fat12_ops του αρχείου fs/fat/fatent.c έχουμε:

Για την λειτουργία fat12_ent_blocknr σειρά 28-29 στο fs/fat/fatent.c:

```
/* Prwto printk apo fat12_ops*/
        printk(KERN_INFO "fat ent blocknr 1");
```

Για την λειτουργία fat12_ent_set_ptr σειρά 57-58 στο fs/fat/fatent.c:

```
/* Deutero printk gia fat12_ops*/
        printk(KERN_INFO "fat12 ent set ptr 2");
```

Για την λειτουργία fat12_ent_bread σειρά 95-96 στο fs/fat/fatent.c:

```
/* Trito printk gia fat12_ops */
        printk(KERN_INFO "fat12 end bread 3");
```

Για την λειτουργία fat12_ent_get σειρά 154-155 στο fs/fat/fatent.c:

```
/* Tetarto printk gia fat12_ops */
        printk(KERN_INFO "fat12 ent get 4");
```

Για την λειτουργία fat12_ent_put σειρά 200-201 στο fs/fat/fatent.c:

```
/* Pempto printk gia fat12_ops */
        printk(KERN_INFO "fat12 ent put 5");
```

Για την λειτουργία fat12_ent_next σειρά 250-251 στο fs/fat/fatent.c:

```
/* Ekto printk gia fat12_ops */
        printk(KERN_INFO "fat12 ent next 6");
```

Για τις λειτουργίες των εγγραφών FAT που ορίζονται στην δομή struct fatent_operations fat16_ops του αρχείου fs/fat/fatent.c έχουμε:

Για την λειτουργία fat_ent_blocknr σειρά 42-43 στο fs/fat/fatent.c:

```
/* Prwto printk gia fat16_ops */
        printk(KERN_INFO "fat16 ent blocknr 1");
```

Για την λειτουργία fat16_ent_set_ptr σειρά 74-75 στο fs/fat/fatent.c:

```
/* Deutero printk gia fat16_ops */
```

```
printk(KERN_INFO "fat16 ent set ptr 2");
```

Για την λειτουργία fat_ent_bread σειρά 130-131 στο fs/fat/fatent.c:

```
/* Trito printk gia fat16_ops */
printk(KERN_INFO "fat16 ent bread 3");
```

Για την λειτουργία fat16_ent_get σειρά 174-175 στο fs/fat/fatent.c:

```
/* Tetarto printk gia fat16_ops */
printk(KERN_INFO "fat16 ent get 4");
```

Για την λειτουργία fat16_ent_put σειρά 223-224 στο fs/fat/fatent.c:

```
/* Pempto printk gia fat16_ops */
printk(KERN_INFO "fat16 ent put 5");
```

Για την λειτουργία fat16_ent_next σειρά 284-285 στο fs/fat/fatent.c:

```
/* Ekto printk gia fat16_ops */
printk(KERN_INFO "fat16 ent next 6");
```

Για τις λειτουργίες των εγγραφών FAT που ορίζονται στην δομή struct fatent_operations fat32_ops του αρχείου fs/fat/fatent.c έχουμε:

Για την λειτουργία fat_ent_blocknr σειρά 45-46 στο fs/fat/fatent.c:

```
/*Prwto printk gia fat32_ops */
printk(KERN_INFO "fat32 ent blocknr 1");
```

Για την λειτουργία fat32_ent_set_ptr σειρά 83-84 στο fs/fat/fatent.c:

```
/* Deutero printk gia fat32_ops */
printk(KERN_INFO "fat32 ent set ptr 2");
```

Για την λειτουργία fat_ent_bread σειρά 133-134 στο fs/fat/fatent.c:

```
/* Trito printk gia fat32_ops */
printk(KERN_INFO "fat32 ent bread 3");
```

Για την λειτουργία fat32_ent_get σειρά 187-188 στο fs/fat/fatent.c:

```
/* Tetarto printk gia fat32_ops */
printk(KERN_INFO "fat32 ent get 4");
```

Για την λειτουργία fat32_ent_put σειρά 235-236 στο fs/fat/fatent.c:

```
/* Pempto printk gia fat32_ops */
```

```
printk(KERN_INFO "fat32 ent put 5");
```

Για την λειτουργία fat32_ent_next σειρά 301-302 στο fs/fat/fatent.c:

```
/* Ekto printk gia fat32_ops */
printk(KERN_INFO "fat32 ent next 6");
```

Για τις λειτουργίες αρχείου που ορίζονται στην δομή struct file_operations fat_file_operations του αρχείου fs/fat/file.c.

Για την λειτουργία generic_file_llseek σειρά147-148 στο fs/read_write.c

```
/* Prwto printk gia fat_file_operations*/
printk(KERN_INFO "generic file llseek 1");
```

Για την λειτουργία generic_file_read_iter σειρά 2030-2031 στο mm/filemap.c

```
/* Deutero printk gia fat_file_operatios */
printk(KERN_INFO "generic file read iter 2");
```

Για την λειτουργία generic_file_write_iter σειρά 2991-2992 στο mm/filemap.c

```
/* Trito printk gia fat_file_operatios */
printk(KERN_INFO "generic file write iter 3");
```

Για την λειτουργία generic_file_mmap σειρά 2444-2445 στο mm/filemap.c

```
/* Tetarto printk gia fat_file_operatios */
printk(KERN_INFO "generic file mmap 4");
```

Για την λειτουργία fat_file_release σειρά 158-159 στο fs/fat/file.c

```
/* Pempto printk gia fat_file_operatios */
printk(KERN_INFO "fat file release 5");
```

Για την λειτουργία fat_generic_ioctl σειρά 129-130 στο fs/fat/file.c

```
/* Ekto printk gia fat_file_operatios */
printk(KERN_INFO "fat generic ioctl 5");
```

Για την λειτουργία fat_file_fsync σειρά 174-175 στο fs/fat/file.c

```
/* Ogdoo printk gia fat_file_operatios */
printk(KERN_INFO "fat file fsync 8");
```

Για την λειτουργία generic_file_splice_read σειρά 308-309 στο fs/splice.c

```
/* Enato printk gia fat_file_operatios */
```

```
        printk(KERN_INFO "generic file splice read 9");
```

Για την λειτουργία fat_fallocate σειρά 253-254 στο fs/fat/file.c

```
/* Dekato printk gia fat_file_operatios */
        printk(KERN_INFO "fat fallocate 10");
```

Για τις λειτουργίες inode που ορίζονται στην δομή struct inode_operations fat_file_inode_operations του αρχείου fs/fat/file.c.

Για την λειτουργία fat_setattr σειρά 469-470 στο fs/fat/file.c

```
/* Prwto printk gia fat_file_inode_operations */
        printk(KERN_INFO "fat setattr 1");
```

Για την λειτουργία fat_getattr σειρά 388-389 στο fs/fat/file.c

```
/* Deutero printk gia fat_file_inode_operations */
        printk(KERN_INFO "fat getattr 2");
```

Για τις λειτουργίες των καταλόγων που ορίζονται στην δομή struct inode_operations msdos_dir_inode_operations του αρχείου fs/fat/namei_msdos.c για το FAT:

Για την λειτουργία msdos_create σειρά 275-276 στο fs/fat/namei_msdos.c

```
/* Prwto printk gia msdos_dir_inode_operations */
        printk(KERN_INFO "msdos create 1");
```

Για την λειτουργία msdos_lookup σειρά 210-211 στο fs/fat/namei_msdos.c

```
/* Deutero printk gia msdos_dir_operations */
        printk(KERN_INFO "msdos lookup 2");
```

Για την λειτουργία msdos_unlink σειρά 421-422 στο fs/fat/namei_msdos.c

```
/* Trito printk gia msdos_dir_inode_operations */
        printk(KERN_INFO "msdos unlink 3");
```

Για την λειτουργία msdos_mkdir σειρά 361-362 στο fs/fat/namei_msdos.c

```
/* Tetarto printk gia msdos_dir_inode_operations */
        printk(KERN_INFO "msdos mkdir 4");
```

Για την λειτουργία msdos_rmdir σειρά 321-322 στο fs/fat/namei_msdos.c

```
/* Pempto printk gia msdos_dir_inode_operations */
        printk(KERN_INFO "msdos rmdir 5");
```

Για την λειτουργία msdos_rename σειρά 637-638 στο fs/fat/namei_msdos.c

```
/* Ekto printk gia msdos_dir_inode_operations */
        printk(KERN_INFO "msdos rename 6");
```

Για την λειτουργία fat_setattr σειρά 472-473 στο fs/fat/file.c

```
/*Evdomo printk gia msdos_dir_inode_operations */
        printk(KERN_INFO "fat setattr 7");
```

Για την λειτουργία fat_getattr σειρά 388-389 στο fs/fat/file.c

```
/* Deutero printk gia fat_file_inode_operations */
        printk(KERN_INFO "fat getattr 2");
```

Για τις λειτουργίες των καταλόγων που ορίζονται στην δομή struct inode_operations vfat_dir_inode_operations του αρχείου fs/fat/namei_vfat.c για το VFAT:

Για την λειτουργία vfat_create σειρά 794-795 στο fs/fat/namei_vfat.c

```
/* Prwto printk gia vfat_dir_inode_operations */
        printk(KERN_INFO "vfat_create 1");
```

Για την λειτουργία vfat_lookup σειρά 732-733 στο fs/fat/namei_vfat.c

```
/* Deutero printk gia vfat_dir_inode_operations */
        printk(KERN_INFO "vfat lookup 2");
```

Για την λειτουργία vfat_unlink σειρά 860-861 στο fs/fat/namei_vfat.c

```
/* Trito printk gia vfat_dir_inode_operations */
        printk(KERN_INFO "vfat unlink 3");
```

Για την λειτουργία vfat_mkdir σειρά 891-892 στο fs/fat/namei_vfat.c

```
/* Tetarto printk gia vfat_dir_inode_operations */
        printk(KERN_INFO "vfat mkdir 4");
```

Για την λειτουργία vfat_mkdir σειρά 829-830 στο fs/fat/namei_vfat.c

```
/* Pempto prink gia vfat_dir_inode_operations */
        printk(KERN_INFO "vfat rmdir 5");
```

Για την λειτουργία vfat_rename σειρά 950-951 στο fs/fat/namei_vfat.c

```
/* Ekto printk gia vfat_dir_inode_operations */
        printk(KERN_INFO "vfat rename 6");
```

Για την λειτουργία vfat_setattr σειρά 472-473 στο fs/fat/file.c

```
/*Evdomo printk gia msdos_dir_inode_operations */
        printk(KERN_INFO "fat setattr 7");
```

Για την λειτουργία vfat_getattr σειρά 394-395 στο fs/fat/file.c

```
/*Ogdoo printk gia vfat_dir_inode_operations */
        printk(KERN_INFO "fat getattr 88");
```

3. Δεύτερο Ερώτημα

Την υλοποίηση του δεύτερου ερωτήματος-βήματος την παραλείψαμε καθώς το τρίτο ερώτημα αποτελεί μια πιο ρεαλιστική υλοποίηση της καταγραφής αφού χρησιμοποιούμε κλήσεις του Linux (π.χ., sys_open, κλπ) και όχι τις κανονικές κλήσεις συστήματος Ε/Ε (π.χ., open, write, κλπ).

4. Τρίτο Ερώτημα

Στο τρίτο ερώτημα μας ζητήθηκε να δημιουργήσουμε ένα αρχείο καταγραφής στο ίδιο το σύστημα FAT που παρακολουθούμε χρησιμοποιώντας τις κλήσεις του Linux (π.χ., sys_open, κλπ) . Σε αυτό το αρχείο έπρεπε να καταγράψουμε μετατροπές των βασικών δομών του FAT οι οποίες ορίζονται στο header file του καταλόγου fs/fat ( ). Οι βασικές δομές αυτές αντιστοιχούν σε δομές του FAT όπως είναι το superblock (Δομή: struct msdos_sb_info (fs/fat/fat.h)), το file allocation table (FAT entry: struct fat_entry (fs/fat/fat.h)), τα directory entries (Δομή: struct msdos_slot__info (fs/fat/fat.h)) και τα δεδομένα των αρχείων (Δομή: struct msdos_inode_info (fs/fat/fat.h) που αφορά τα μεταδεδομένα των αρχείων) όπως αναφέρεται και στο φροντιστήριο.

Η πρώτη μας κίνηση ήταν να ορίσουμε στη δομή του superblock του FAT έναν ακέραιο περιγραφέα αρχείου (int file_descriptor; /* My file descriptor */ σειρα

63 fs/fat/fa.h) που χρησιμοποιείται για την αναφορά ενός αρχείου που θα ανοίξει μέσω της συνάρτησης sys_open του Linux. Επομένως έπρεπε να βρούμε σε ποιες συναρτήσεις του fs/fat αρχικοποιούνται πεδία του struct msdos_sb_info έτστ ώστε να αρχικοποιήσουμε τον περιγραφέα αρχείου (πχ sbi->file_descriptor = sys_open()). Τέτοιες συναρτήσεις υπάρχουν πολλές όπως η fat_ent_access_init() στο fs/fat/fatent.c. Ωστόσο εμείς αποφασίσαμε να τον αρχικοποιήσουμε στη συνάρτηση όπου γίνονται οι περισσότερες αρχικοποιήσεις του msdos_sb_info, δηλαδή στην συνάρτηση fat_fill_super() στον κατάλογο fs/fat/inode.c ("safe" επιλογή). Εκεί λοιπόν αρχικοποιούμε τον περιγραφέα αρχείου ώς εξής : sbi->file_descriptor = sys_open("testFile.txt",O_CREAT|O_RDWR|O_APPEND,S_IRUSR|S_IWUSR|S_IRGRP|S_IWGRP|S_IROTH|S_IWOTH); στη σειρά 1749, όπου σαν πρώτο όρισμα μπαίνει το όνομα του αρχείου που δημιουργούμε, ως δεύτερο τα απαραίτητα flags , δηλαδή O_CREAT για να δημιουργηθεί το αρχείο σε περίπτωση που δεν υπάρχει(επειδή δεν υπήρχε έπρεπε να το δημιουργήσουμε εμείς), O_RDWR για δυνατότητα διαβάσματος και γραψίματος σε αυτό και O_APPEND καθώς η λειτουργία αυτή προκαλεί όλες τις ενέργειες εγγραφής στο τέλος του αρχείου. Τέλος ως τρίτο όρισμα έχουν μπει όλα τα modes δηλαδή τα δικαιώματα των χρηστών, δηλαδή την δυνατότητα για διάβασμα και γράψιμο στον user, group και other (το mode other πολυ πιθανόν να είναι περιττό αλλά εμείς τα βάλαμε όλα για να είμαστε "safe"). Επίσης για δικό μας έλεγχο προσθέσαμε στις σειρές 1750, 2022 και 800 ένα printk() (printk(KERN_INFO "***** File Descriptor = %d",sbi->file_descriptor);) για να δούμε τι τιμές θα πάρει ο file_descriptor. Στην περίπτωση που ο file_descriptor πάρει τιμή μη αρνητική (δηλαδή file_descriptor >=0) τότε αυτό σημαίνει ότι το αρχείο μας έχει δημιουργηθεί και έχει ανοίξει, ενώ στην περίπτωση που πάρει αρνητική τιμή αυτό σημαίνει ότι υπάρχει το αντίθετο αποτέλεσμα (στην δική μας περίπτωση ο file_descriptor είχε πάντα τιμή 0).

Στη συνέχεια μετά τον εντοπισμό των τροποποιήσεων των δομών έπρεπε να γράψουμε τις τροποποιήσεις που ανιχνεύσαμε στο file_descriptor με την χρήση sys_write() του Linux. Για αυτό τον λόγο για να μην καλούμε συνεχώς την sys_write αποφασίσαμε να να βάλουμε μέσα σε ένα char πίνακα (char *descriptor_array) όλες αυτές τις τροποποιήσεις που συμβαίνουν μέσα σε μία συνάρτηση, να κάνουμε sys_write τον πίνακα αυτό και τέλος να κάνουμε flush τα δεδομένα και τα μεταδεδομένα στον πυρήνα (με την χρήση των sys_fsync και sys_fdatasync αντίστοιχα που δίνονται στο φροντιστήριο). Έτσι σε κάθε αρχείο που παρατηρήθηκε τροποποίηση σε μία από τις δομές που αναφέρθηκε παραπάνω ορίσαμε με #define N 1024 το μέγεθος αυτού του πίνακα. Επομένως ψάχνοντας σε κάθε συνάρτηση των αρχείων στο fs/fat μόλις βρίσκαμε μία τροποποίηση, ορίζαμε έναν πίνακα, στην αρχή της κάθε μία συνάρτησης από αυτές,ο οποίος δεσμεύει χώρο στην μνήμη δυναμικά (όχι με την χρήση malloc καθώς ο πυρήνας δεν έχει πρόσβαση στο σύστημα αρχείων

root και συνεπώς σε βιβλιοθήκες της C, αλλά με την χρήση της συνάρτησης kmalloc αποδεσμευοντας στο τέλος την μνήμη με χρήση kfree()).

Επόμενη μας σκέψη ήταν να γράφουμε στον πίνακα descriptor_array τις τροποποιήσεις με την χρήση της συνάρτησης sprintf (δίνεται στο φροντιστήριο σελίδα 9 η δυνατότητα χρήσης της sprint για strings). Για αυτό τον λόγο μόλις ορίζαμε στην αρχή των συναρτήσεων (συναρτήσεων που πραγματοποιούνταν τροποποιήσεις δομών) έναν ακέραιο αριθμό (int bravo : ορίστηκε ως ακέραιος μετά από ψάξιμο στο διαδίκτυο) στον οποίο κάθε φορά αποθηκεύουμε τον πίνακα descriptor_array, έτσι ώστε να μην χαθούν τα προηγούμενα strings που βάλαμε σε αυτόν μέσω της sprintf(). Με άλλα λόγια κάθε φορά που κάναμε sprintf ένα string στο descriptor_array προσθέταμε σε αυτόν τον bravo για να εμφανιστούν και τα προηγούμενα strings (θα δείτε παράδειγμα στην συνέχεια στον κώδικα). Αφού λοιπόν προσθέταμε στο descriptor_array όλες τις τροποποιήσεις της συγκεκριμένης συνάρτησης τοποθετήσαμε printk() έτσι ώστε να δούμε στο τερματικό τι περιέχει μέσα ο πίνακας τον οποίο θα κάνουμε sys_write() (printk(KERN_INFO "Descriptor array data ---------> %s \n",descriptor_array)). Αποφασίσαμε να προβούμε σε αυτή την κίνηση διότι εκτελώντας την εντολή ./cptofs -i /tmp/vfatfile -p -t vfat lklfuse.c / το αρχείο testFile.txt που δημιουργούμε με την κλήση της sys_open() δημιουργείται μέσα στον κατάλογο /tmp/vfatfile (το vfatfile δημιουργείται κάθε φορά που εκτελούμε την συγκεκριμένη εντολή). Αυτός ο κατάλογος αποτελεί μία “συσκευή” που εμπεριέχει ένα ξεχωριστό σύστημα αρχείων από αυτό της εικονικής μηχανής που χρησιμοποιούμε, με αποτέλεσμα να μην μπορούμε να έχουμε πρόσβαση σε αυτό και συνεπώς πρόσβαση στο αρχείο μας.

Τέλος κάναμε καταγραφή του πίνακα descriptor_array στο file_descriptor με την χρήση της συνάρτησης sys_write() και μετά κάναμε flush τα δεδομένα και τα μετα δεδομένα στον πυρήνα με την χρησης sys_fsync() και sys_fdatasync(). Τελευταία μας κίνηση ήταν να αποδεσμεύσουμε χώρο από την μνήμη για τον descriptor_array κάνοντας kfree(descriptor_array).

ΣΗΜΕΙΩΣΗ 1 : Σε κάθε αρχείο που κάναμε κλήση των λειτουργιών syscalls του Linux κάναμε #inlcude την απαραίτητη βιβλιοθήκη (#include <linux/syscalls.h>).

ΣΗΜΕΙΩΣΗ 2 : Σε κάθε αρχείο που καταγράψαμε τροποποιήσεις των δομών κάνουμε #define το μέγεθος του descriptor_array (#define N 1024).

ΣΗΜΕΙΩΣΗ 3 : Σε ορισμένες συναρτήσεις που πραγματοποιούνταν τροποποιήσεις δεν αρχικοποιούνταν με κάποιο τρόπο ο τύπος sbi του struct msdos_sb_info ή δεν οριζόταν καθόλου. Επομένως προβήκαμε στις ανάλογες ενέργειες. Δηλαδή στις συναρτήσεις που δεν ορίζονταν το sbi το ορίσαμε και το

αρχικοποιήσαμε όπως και στη συνάρτηση fat_fill_super() στο fs/fat/inode.c (ήταν ορισμένο ως struct msdos_sb_info *sbi; και αρχικοποιημένο ως sbi = kzalloc(sizeof(struct msdos_sb_info), GFP_KERNEL); αποδεσμεύοντας στο τέλος την μνήμη του sbi ως kfree(sbi);). Αυτή η κίνηση ήταν αναγκαία καθώς ο τύπος του file_descriptor είναι τύπου sbi. Στις συναρτήσεις όπου αρχικοποιούνταν το sbi (πχ συνάρτηση fat_ent_access_init() στο fs/fat/fatent.c ως struct msdos_sb_info *sbi = MSDOS_SB(sb);) δεν κάναμε τίποτα.

Οι τροποποιήσεις τον δομων για το **superlock** παρατηρήθηκαν στο αρχείο /fs/fat/fatent.c στην συνάρτηση fat_ent_access_init() και προσθέσαμε τα ακόλουθα κομμάτια στον κώδικα:

Σειρες 353-365:

```
char *descriptor_array;                    /* Pinakas pou tha kratw mesa tis allages twn domwn gia na tis kanw sth sunexeia sys_write */
        int bravo;
        int write_value;                   /* value retuend for sys_write */

        descriptor_array = kmalloc(N, GFP_KERNEL);          /* Desmeush xwrou gia ton descriptor_array sthn mnhmh*/
```

Σειρες 366-367:

```
bravo = sprintf(descriptor_array,"sbi->fatent_shift = %d \n ",sbi->fatent_shift);
                bravo += sprintf(descriptor_array + bravo,"sbi->fatent_ops = %lu \n",sbi->fatent_ops);
```

Σειρες 371-372:

```
bravo = sprintf(descriptor_array,"sbi->fatent_shift = %d \n ",sbi->fatent_shift);
                bravo += sprintf(descriptor_array + bravo,"sbi->fatent_ops = %lu \n",sbi->fatent_ops);
```

Σειρες 378-379:

```
bravo = sprintf(descriptor_array,"sbi->fatent_shift = %d \n ",sbi->fatent_shift);
                bravo += sprintf(descriptor_array + bravo,"sbi->fatent_ops = %lu \n",sbi->fatent_ops);
```

Σειρες 383-387:

```
printk(KERN_INFO "Descriptor array data  ---------->  %s \n",descriptor_array);
                /* Voththitiko printk gia na kserw ti tha graftei sto description file */
```

```
sys_write(sbi->file_descriptor,descriptor_array,sizeof(descriptor_array));
        /* Grafw sto arxeio ton descriptor array */
sys_fsync(sbi->file_descriptor);                                        /*
Kane flush ta dedomena ston purhna */
sys_fdatasync(sbi->file_descriptor);
printk(KERN_INFO "-------sys_write = %d",write_value);


kfree(descriptor_array);
```

Στη συνάρτηση fat_alloc_clusters() στο /fs/fat/fatent.c:

Σειρες 555-557:

```
char *descriptor_array;                 /* Pinakas pou tha kratw mesa tis allages twn
domwn gia na tis kanw sth sunexeia sys_write */
int bravo;
int write_value;
```

Σειρα 562:

```
descriptor_array = kmalloc(N, GFP_KERNEL);          /* Desmeush xwrou gia ton
descriptor_array sthn mnhmh*/
```

Σειρα 580:

```
bravo = sprintf(descriptor_array,"fatent.entry = %d \n",fatent.entry);
```

Σειρα 599:

```
bravo += sprintf(descriptor_array+bravo,"sbi->prev_free = %u \n",sbi->prev_free);
```

Σειρα 604:

```
bravo += sprintf(descriptor_array + bravo,"sbi->free_clusters = %u \n",sbi-
>free_clusters);
```

Σειρες 625-634:

```
bravo = sprintf(descriptor_array,"sbi->free_clusters = %u \n ",sbi->free_clusters);
        bravo += sprintf(descriptor_array + bravo,"sbi->free_clus_valid = %u \n",sbi-
>free_clus_valid);
        printk(KERN_INFO "Descriptor array data  ---------->  %s \n",descriptor_array);
                        /* Vohthitiko printk gia na kserw ti tha graftei sto description file
*/
```

```
        sys_write(sbi->file_descriptor,descriptor_array,sizeof(descriptor_array));
                /* Grafw sto arxeio ton descriptor array */
        sys_fsync(sbi->file_descriptor);
        /* Kane flush ta dedomena ston purhna */
        sys_fdatasync(sbi->file_descriptor);
                /* Kanw flush ta metadedomena ston purhna */
        err = -ENOSPC;
        printk(KERN_INFO "-------sys_write = %d",write_value);

        kfree(descriptor_array);      /* Apodesmeuw ton xwro stin mnhmh gia
descriptor array*/
```

Στη συνάρτηση fat_free_clusters() στο /fs/fat/fatent.c:

Σειρές 664-669:

```
char *descriptor_array;                 /* Pinakas pou tha kratw mesa tis allages twn
domwn gia na tis kanw sth sunexeia sys_write */
        int bravo;
        int write_value;
        nr_bhs = 0;

        descriptor_array = kmalloc(N, GFP_KERNEL);          /* Desmeush xwrou gia
ton descriptor_array sthn mnhmh*/
```

Σειρα 709:

```
bravo = sprintf(descriptor_array,"bi->free_clusters = %u \n",sbi->free_clusters);
```

Σειρές 732-738:

```
printk(KERN_INFO "Descriptor array data  ---------->  %s \n",descriptor_array);
                /* Vohthitiko printk gia na kserw ti tha graftei sto description file */
        sys_write(sbi->file_descriptor,descriptor_array,sizeof(descriptor_array));
                /* Grafw sto arxeio ton descriptor array */
        sys_fsync(sbi->file_descriptor);
        /* Kane flush ta dedomena ston purhna */
        sys_fdatasync(sbi->file_descriptor);
                /* Kanw flush ta metadedomena ston purhna */
printk(KERN_INFO "-------sys_write = %d",write_value);
```

```
        kfree(descriptor_array);      /* Apodesmeuw ton xwro stin mnhmh gia
descriptor array*/
```

Στη συνάρτηση fat_count_free_clusters() στο /fs/fat/fatent.c:

Σειρές 774-779:

```
char *descriptor_array;                    /* Pinakas pou tha kratw mesa tis allages twn
domwn gia na tis kanw sth sunexeia sys_write */
        int bravo;
        int write_value;

descriptor_array = kmalloc(N, GFP_KERNEL);          /* Desmeush xwrou gia ton
descriptor_array sthn mnhmh*/
```

Σειρές 810-820:

```
bravo = sprintf(descriptor_array,"sbi->free_clusters = %u \n ",sbi->free_clusters);
        bravo += sprintf(descriptor_array + bravo,"sbi->free_clus_valid = %u \n",sbi-
>free_clus_valid);
        printk(KERN_INFO "Descriptor array data  ---------->  %s \n",descriptor_array);
                        /* Vohthitiko printk gia na kserw ti tha graftei sto description file
*/
        sys_write(sbi->file_descriptor,descriptor_array,sizeof(descriptor_array));
                /* Grafw sto arxeio ton descriptor array */
        sys_fsync(sbi->file_descriptor);
        /* Kane flush ta dedomena ston purhna */
        sys_fdatasync(sbi->file_descriptor);
                /* Kanw flush ta metadedomena ston purhna */
        printk(KERN_INFO "-------sys_write = %d",write_value);

kfree(descriptor_array);      /* Apodesmeuw ton xwro stin mnhmh gia descriptor
array*/
```

Στη συνάρτηση fat_fill_super() στο /fs/fat/inode.c:

Σειρές 1661-1665:

```
char *descriptor_array;                    /* Pinakas pou tha kratw mesa tis allages twn
domwn gia na tis kanw sth sunexeia sys_write */
        int bravo;
```

```
int write_value;
descriptor_array = kmalloc(N, GFP_KERNEL);          /* Desmeush xwrou gia
ton descriptor_array sthn mnhm
```

Σειρά 1717:

```
bravo = sprintf(descriptor_array,"sbi->sec_per_clus = %u \n",sbi->sec_per_clus);  /*
Grafoume sto descriptor_array to pedio pou allazei kai tin timi tou */
```

Σειρές 1752-1753:

```
sbi->file_descriptor =
sys_open("testFile.txt",O_CREAT|O_RDWR|O_APPEND,S_IRUSR|S_IWUSR|S_IR
GRP|S_IWGRP|S_IROTH|S_IWOTH);
        printk(KERN_INFO "***** File Descriptor = %d",sbi->file_descriptor);
```

Σειρές 1767-1787:

```
/* Parapanw exoume allages ths domhs msdos_sb_info */
        bravo += sprintf(descriptor_array + bravo,"sbi->cluster_size = %d \n",sbi-
>cluster_size); // u


        bravo += sprintf(descriptor_array + bravo,"sbi->cluster_bits = %d \n",sbi-
>cluster_bits); //u

        bravo += sprintf(descriptor_array + bravo,"sbi->fats = %d \n",sbi->fats); // u

        bravo += sprintf(descriptor_array + bravo,"sbi->fat_bits = %u \n",sbi-
>fat_bits); // u

        bravo += sprintf(descriptor_array + bravo,"sbi->fat_start =%u \n",sbi-
>fat_start);

        bravo += sprintf(descriptor_array + bravo,"sbi->fat_length = %lu \n",sbi-
>fat_length);

        bravo += sprintf(descriptor_array + bravo,"sbi->root_cluster = %lu \n",sbi-
>root_cluster);

        bravo += sprintf(descriptor_array + bravo,"sbi->free_clusters = %u \n",sbi-
>free_clusters);
```

bravo += sprintf(descriptor_array + bravo,"sbi->free_clus_valid = %u \n",sbi->free_clus_valid);

bravo += sprintf(descriptor_array + bravo,"sbi->prev_free = %u \n",sbi->prev_free);

Σειρές 1802-1808:
bravo += sprintf(descriptor_array + bravo,"sbi->fat_bits = %u \n",sbi->fat_bits);

bravo += sprintf(descriptor_array + bravo,"sbi->fat_length = %lu \n",sbi->fat_length);

bravo += sprintf(descriptor_array + bravo,"sbi->root_cluster = %lu \n",sbi->root_cluster);

bravo += sprintf(descriptor_array + bravo,"sbi->fsinfo_sector = %lu \n",sbi->fsinfo_sector);

Σειρά1813:
bravo += sprintf(descriptor_array + bravo,"sbi->fsinfo_sector = %lu \n",sbi->fsinfo_sector);

Σειρά1833:
bravo += sprintf(descriptor_array + bravo,"sbi->free_clus_valid = %u \n",sbi->free_clus_valid);

Σειρές 1838-1840
bravo += sprintf(descriptor_array + bravo,"sbi->free_clusters = %u \n",sbi->free_clusters);

bravo += sprintf(descriptor_array + bravo,"sbi->prev_free = %u \n",sbi->prev_free);

Σειρά 1851:
bravo += sprintf(descriptor_array + bravo,"sbi->vol_id = %u \n",sbi->vol_id);

Σειρά 1855:
bravo += sprintf(descriptor_array + bravo,"sbi->vol_id = %u \n",sbi->vol_id);

Σειρές 1864-1870:

```
bravo += sprintf(descriptor_array + bravo,"sbi->dir_per_block = %d \n",sbi->dir_per_block);

        bravo += sprintf(descriptor_array + bravo,"sbi->dir_per_block_bits = %d \n",sbi->dir_per_block_bits);

        bravo += sprintf(descriptor_array + bravo,"sbi->dir_start = %lu \n",sbi->dir_start);

        bravo += sprintf(descriptor_array + bravo,"sbi->dir_entries = %u \n",sbi->dir_entries);
```

Σειρά 1884:

```
bravo += sprintf(descriptor_array + bravo,"sbi->data_start = %lu \n",sbi->data_start);
```

Σειρά 1894:

```
bravo += sprintf(descriptor_array + bravo,"sbi->fat_bits = %u \n",sbi->fat_bits);
```

Σειρά 1900:

```
bravo += sprintf(descriptor_array + bravo,"sbi->dirty = %u \n",sbi->dirty);
```

Σειρά 1904:

```
bravo += sprintf(descriptor_array + bravo,"sbi->dirty = %u \n",sbi->dirty);
```

Σειρά 1918:

```
bravo += sprintf(descriptor_array + bravo,"sbi->max_cluster = %lu \n",sbi->max_cluster);
```

Σειρά 1923:

```
bravo += sprintf(descriptor_array + bravo,"sbi->free_clusters = %u \n",sbi->free_clusters);
```

Σειρά 1928:

```
bravo += sprintf(descriptor_array + bravo,"sbi->prev_free = %u \n",sbi->prev_free);
```

Σειρά 1932:

```
bravo += sprintf(descriptor_array + bravo,"sbi->prev_free = %u \n",sbi->prev_free);
```

Σειρά 1951:

```
bravo += sprintf(descriptor_array + bravo,"sbi->nls_disk = %lu \n",sbi->nls_disk);
```

Σειρά 1961:

```
bravo += sprintf(descriptor_array + bravo,"sbi->nls_io = %lu \n",sbi->nls_io);
```

Σειρά 1976:

```
bravo += sprintf(descriptor_array + bravo,"sbi->fat_inode = %lu \n",sbi->fat_inode);
```

Σειρά 1985:

```
bravo += sprintf(descriptor_array + bravo,"sbi->fsinfo_inode = %lu \n",sbi->fsinfo_inode);
```

Σειρές 2010-2025

```
printk(KERN_INFO "Descriptor array data --------->  %s \n",descriptor_array);
        /* Vohthitiko printk gia na kserw ti tha graftei sto description file */
    sys_write(sbi->file_descriptor,descriptor_array,sizeof(descriptor_array));
        /* Grafw sto arxeio ton descriptor array */
    sys_fsync(sbi->file_descriptor);
    /* Kane flush ta dedomena ston purhna */
    sys_fdatasync(sbi->file_descriptor);
        /* Kanw flush ta metadedomena ston purhna */
    printk(KERN_INFO "*******file descriptor = %d",sbi->file_descriptor);
    printk(KERN_INFO "-------sys_write = %d",write_value);
```

Σειρά 2043:

```
kfree(descriptor_array);      /* apodesmeush mnimis gia descriptor array */
```

Οι τροποποιήσεις τον δομων για το **inode** παρατηρήθηκαν στο αρχείο /fs/fat/cache.c στην συνάρτηση void __fat_cache_inval_inode() και προσθέσαμε τα ακόλουθα κομμάτια στον κώδικα:

Σειρές 191-198:

```
struct msdos_sb_info *sbi; /* Vazw to msdos_sb_info gia ta sys calls */
    char *descriptor_array;                 /* Pinakas pou tha kratw mesa tis allages twn domwn gia na tis kanw sth sunexeia sys_write */
    int bravo;
    int write_value;
```

descriptor_array = kmalloc(N, GFP_KERNEL); /* Desmeush xwrou gia ton descriptor_array sthn mnhmh*/

sbi = kzalloc(sizeof(struct msdos_sb_info), GFP_KERNEL); /* Arxikopoiw to sbi */

Σειρές 205:
bravo = sprintf(descriptor_array,"i->nr_caches = %d \n",i->nr_caches);

Σειρά 210:
bravo = sprintf(descriptor_array,"i->cache_valid_id = %u \n",i->cache_valid_id);

Σειρά 213:
bravo += sprintf(descriptor_array + bravo,"i->cache_valid_id = %u \n",i->cache_valid_id);

Σειρές 215-222:
```
printk(KERN_INFO "Descriptor array data  ---------->  %s \n",descriptor_array);
        /* Vohthitiko printk gia na kserw ti tha graftei sto description file */
    sys_write(sbi->file_descriptor,descriptor_array,sizeof(descriptor_array));
        /* Grafw sto arxeio ton descriptor array */
    sys_fsync(sbi->file_descriptor);
    /* Kane flush ta dedomena ston purhna */
    sys_fdatasync(sbi->file_descriptor);
        /* Kanw flush ta metadedomena ston purhna */
    printk(KERN_INFO "-------sys_write = %d",write_value);

    kfree(sbi);                 /* Apodesmeuw ton xwro stin mnhmh gia sbi*/
    kfree(descriptor_array);    /* Apodesmeuw ton xwro stin mnhmh gia descriptor array*/
```

Στη συνάρτηση init_once() στο /fs/fat/inode.c:

Σειρές 779-786:
```
struct msdos_sb_info *sbi; /* Arxikopoiw to msdos_sb_info gia ta sys calls */
    char *descriptor_array;             /* Pinakas pou tha kratw mesa tis allages twn domwn gia na tis kanw sth sunexeia sys_write */
    int bravo;
```

```
int write_value;

descriptor_array = kmalloc(N, GFP_KERNEL);          /* Desmeush xwrou gia
ton descriptor_array sthn mnhmh*/

sbi = kzalloc(sizeof(struct msdos_sb_info), GFP_KERNEL);  /* Arxikopoiw to
sbi */
```

Σειρές 791-796:

```
bravo = sprintf(descriptor_array,"ei->nr_caches = %d \n",ei->nr_caches);
        bravo += sprintf(descriptor_array + bravo,"ei->cache_valid_id = %u \n",ei-
>cache_valid_id);
        printk(KERN_INFO "Descriptor array data  ---------->  %s \n",descriptor_array);
                        /* Vohthitiko printk gia na kserw ti tha graftei sto description file
*/
        sys_write(sbi->file_descriptor,descriptor_array,sizeof(descriptor_array));
                /* Grafw sto arxeio ton descriptor array */
        sys_fsync(sbi->file_descriptor);
        /* Kane flush ta dedomena ston purhna */
        sys_fdatasync(sbi->file_descriptor);
                /* Kanw flush ta metadedomena ston purhna */
```

Σειρές 800-806:

```
printk(KERN_INFO "*******file descriptor = %d",sbi->file_descriptor);
printk(KERN_INFO "-------sys_write = %d",write_value);


kfree(sbi);                     /* Apodesmeuw ton xwro stin mnhmh gia sbi*/
kfree(descriptor_array);        /* Apodesmeuw ton xwro stin mnhmh gia descriptor
array*/
```

Οι τροποποιήσεις τον δομων για το **directory entries** παρατηρήθηκαν στο αρχείο /fs/fat/namei_msdos.c στην συνάρτηση void int do_msdos_rename() και προσθέσαμε τα ακόλουθα κομμάτια στον κώδικα:

Σειρά 455:

```
struct msdos_sb_info *sbi;
```

Σειρά 458-464:

```
char *descriptor_array;                    /* Pinakas pou tha kratw mesa tis allages twn domwn gia na tis kanw sth sunexeia sys_write */
        int bravo;
        int write_value;

        descriptor_array = kmalloc(N, GFP_KERNEL);          /* Desmeush xwrou gia ton descriptor_array sthn mnhmh*/

        sbi = kzalloc(sizeof(struct msdos_sb_info), GFP_KERNEL);  /* Arxikopoiw to sbi */
```

Σειρά 4678:

```
bravo = sprintf(descriptor_array,"old_sinfo.bh = %lu \n",old_sinfo.bh);
```

Σειρές 567-575:

```
        bravo += sprintf(descriptor_array + bravo,"old_sinfo.bh = %lu \n",old_sinfo.bh);
        printk(KERN_INFO "Descriptor array data  ---------->  %s \n",descriptor_array);
                        /* Vohthitiko printk gia na kserw ti tha graftei sto description file */
        sys_write(sbi->file_descriptor,descriptor_array,sizeof(descriptor_array));
                /* Grafw sto arxeio ton descriptor array */
        sys_fsync(sbi->file_descriptor);
        /* Kane flush ta dedomena ston purhna */
        sys_fdatasync(sbi->file_descriptor);
                /* Kanw flush ta metadedomena ston purhna */
        printk(KERN_INFO "-------sys_write = %d",write_value);

        kfree(sbi);                     /* Apodesmeuw ton xwro stin mnhmh gia sbi*/
        kfree(descriptor_array);        /* Apodesmeuw ton xwro stin mnhmh gia descriptor array*/
```

Στη συνάρτηση vfat_rename() στον κατάλογο /fs/fat/namei_vfat.c:

Σειρές 946-956:

```
struct msdos_sb_info *sbi;  /* Vazw to msdos_sb_info gia ta sys calls */
        char *descriptor_array;                     /* Pinakas pou tha kratw mesa tis allages twn domwn gia na tis kanw sth sunexeia sys_write */
```

```
        int bravo;
        int write_value;

        /* Ekto printk gia vfat_dir_inode_operations */
        printk(KERN_INFO "vfat rename 6");

        descriptor_array = kmalloc(N, GFP_KERNEL);          /* Desmeush xwrou gia
ton descriptor_array sthn mnhmh*/

        sbi = kzalloc(sizeof(struct msdos_sb_info), GFP_KERNEL);  /* Arxikopoiw to
sbi */
```

Σειρά 961:

```
bravo = sprintf(descriptor_array,"old_sinfo.bh = %lu \n",old_sinfo.bh);
```

Σειρές 1020-1028:

```
bravo += sprintf(descriptor_array + bravo,"old_sinfo.bh = %lu \n",old_sinfo.bh);
        printk(KERN_INFO "Descriptor array data  ---------->  %s \n",descriptor_array);
                        /* Vohthitiko printk gia na kserw ti tha graftei sto description file
*/
        sys_write(sbi->file_descriptor,descriptor_array,sizeof(descriptor_array));
                /* Grafw sto arxeio ton descriptor array */
        sys_fsync(sbi->file_descriptor);
        /* Kane flush ta dedomena ston purhna */
        sys_fdatasync(sbi->file_descriptor);
                /* Kanw flush ta metadedomena ston purhna */
        printk(KERN_INFO "-------sys_write = %d",write_value);

        kfree(sbi);                     /* Apodesmeuw ton xwro stin mnhmh gia sbi*/
        kfree(descriptor_array);        /* Apodesmeuw ton xwro stin mnhmh gia
descriptor array*/
```

Για το **file allocation table** δεν παρατηρήθηκε κάποια τροποποίηση οπότε δεν καταγράφηκε τίποτα στον file_descriptor επομένως δεν χρειάστηκε να προσθέσουμε κάτι στον κώδικα για αυτό.

5. Τέταρτο Ερώτημα

Για τους λόγους που αναφέραμε παραπάνω στο τρίτο ερώτημα, το αρχείο καταγραφής δημιουργείται τρέχοντας την εντολή ./cptofs -i /tmp/vfatfile -p -t vfat lklfuse.c / μέσα στον κατάλογο tmp/vfatfile με αποτέλεσμα να μην μπορούμε να έχουμε πρόσβαση σε αυτό. Επομένως, έπρεπε να ελέγξουμε διαφορετικά ότι το αρχείο δημιουργείται. Για αυτό το λόγο αποφασίσαμε να χρησιμοποιήσουμε ένα printk() (printk(KERN_INFO "***** File Descriptor = %d",sbi->file_descriptor);) το οποίο θα μας επιστρέφει την τιμή του file_descriptor (όταν επιστρέφει μη αρνητική τιμή τότε το αρχείο μας δημιουργείται). Επιπλέον αφού σιγουρευτούμε ότι το αρχείο μας δημιουργείται-ανοίγει έπρεπε να ελέγξουμε τη λειτουργία της εντολής sys_write και ότι γράφει τα δεδομένα που παρακολουθούμε στο αρχείο μας. Έχοντας καταγράψει όλες τις αναμενόμενες τροποποιήσεις στον πίνακα μας descriptor_array, χρειάστηκε να επιβεβαιώσουμε ότι οι εγγραφές στον πίνακα αυτόν είναι αυτές που παρακολουθούμε. Για αυτό το λόγο χρησιμοποιήσαμε ένα printk() (printk(KERN_INFO "Descriptor array data ---------> %s \n",descriptor_array);). Στη συνέχεια, χρειάστηκε να τεστάρουμε την επιτυχία της εντολής sys_write(). Συνεπώς, σε κάθε συνάρτηση που παρατηρήσαμε τροποποιήσεις ορίσαμε έναν ακέραιο (int write_value) μιας και γνωρίζαμε ότι το sys_write επιστρέφει τον αριθμό των bytes που έχουν γραφτεί ή -1 σε περίπτωση αποτυχίας ή σφάλματος. Ο έλεγχος γίνεται με printk αμέσως μετά όπως παρακάτω. Έτσι, επιβεβαιώνουμε ότι τα στοιχεία του πίνακα descriptor_array μεταφέρονται στο αρχείο καταγραφής μας.

```
write_value=sys_write(sbi->file_descriptor,descriptor_array,sizeof(descriptor_array));
printk(KERN_INFO "-------sys_write = %d",write_value);
```

```
[    0.018372] Descriptor array data ---------->  sbi->sec_per_clus = 4
[    0.018372] sbi->cluster_size = 2048
[    0.018372] sbi->cluster_bits = 11
[    0.018372] sbi->fats = 2
[    0.018372] sbi->fat_bits = 0
[    0.018372] sbi->fat_start =4
[    0.018372] sbi->fat_length = 200
[    0.018372] sbi->root_cluster = 0
[    0.018372] sbi->free_clusters = 4294967295
[    0.018372] sbi->free_clus_valid = 0
[    0.018372] sbi->prev_free = 2
[    0.018372] sbi->vol_id = 2471789940
[    0.018372] sbi->dir_per_block = 16
[    0.018372] sbi->dir_per_block_bits = 4
[    0.018372] sbi->dir_start = 404
[    0.018372] sbi->dir_entries = 512
[    0.018372] sbi->data_start = 436
[    0.018372] sbi->fat_bits = 16
[    0.018372] sbi->dirty = 0
[    0.018372] sbi->max_cluster = 51093
[    0.018372] sbi->prev_free = 2
[    0.018372] sbi->nls_disk = 93845208077856
[    0.018372] sbi->nls_io = 93845208079392
[    0.018372] sbi->fat_inode = 140380456673392
[    0.018385] generic file write iter 3
[    0.018387] *******file descriptor = 0
[    0.018388] -------sys write = 8
```

Στην παραπάνω εικόνα φαίνονται αυτά που προαναφέραμε, δηλαδή τα δεδομένα μέσα στον πίνακα (descriptor_array). Τα συγκεκριμένα δεδομένα είναι οι αναμενόμενες τροποποιήσεις που γίνονται μέσα στο συνάρτηση fat_fill_super() στο αρχείο fs/fat/inode.c . Επίσης, βλέπουμε στο τέλος την τιμή του file_descriptor να είναι μηδέν και άρα το αρχείο μας δημιουργείται, ενώ η τιμή του sys_write είναι 8, κάτι που σημαίνει ότι η εντολή μας sys_write() δουλεύει σωστά και μεταφέρει τα δεδομένα μας μεγέθους 8 bytes του πίνακα στο αρχείο μας.

Σας ευχαριστούμε πολύ!