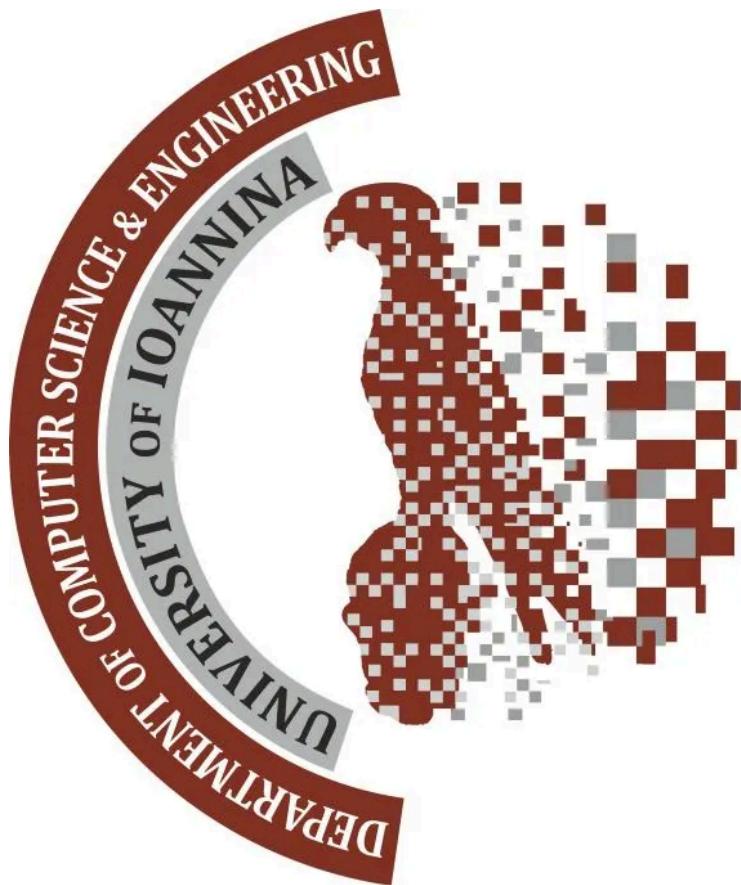


Πανεπιστήμιο Ιωαννίνων
Τμήμα Μηχανικών Η/Υ και Πληροφορικής

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 2Η

ΥΛΟΠΟΙΗΣΗ ΑΡΧΕΙΟΥ ΚΑΤΑΓΡΑΦΗΣ ΣΤΟ ΣΥΣΤΗΜΑ ΑΡΧΕΙΩΝ VFAT ΤΟΥ LINUX



ΟΜΑΔΑ:

Σολδάτου Χριστίνα Ολυμπία, Α.Μ. 4001
Τσιτσιμίκλη Αικατερίνη, Α.Μ. 4821

Περιεχόμενα

❖ Εισαγωγή	3
❖ Εκτέλεση Προγράμματος	3
❖ Παραδοτέο Εξήγηση	5
❖ 1o Βήμα: Κατανόηση της βασικής λειτουργίας του συστήματος αρχείων VFAT	5
❖ Εκσφαλμάτωση Κώδικα με GDB	6
❖ 2o Βήμα: Πρόσθεση συνάρτησης printk() στο πηγαίο κώδικα της LKL	7
❖ Εκτέλεση με δοκιμαστικό αρχείο	11
❖ 3o Βήμα: Καταγραφή των λειτουργιών σε αρχείο	15
❖ Αρχείο Καταγραφής - Journaling με Metadata	15
❖ Περιγραφή της Διαδικασίας Καταγραφής (Journaling) στο Σύστημα Αρχείων VFA	16
❖ Λεπτομέρειες Υλοποίησης της Καταγραφής (Journaling) στο Σύστημα Αρχείων VFA	18
❖ <u>Δομές αποθήκευσης στο δίσκο</u>	19
❖ <u>Βασικές Δομές FAT</u>	28
❖ <u>FILE ALLOCATION TABLE (FAT)</u>	28
❖ <u>SUPERBLOCK</u>	33
❖ <u>INODE</u>	41
❖ <u>DIRECTORY ENTRIES</u>	43
❖ <u>FILES</u>	48
❖ <u>ΜΠΛΟΚ ΜΕΤΑΔΕΔΟΜΕΝΩΝ – INODES</u>	50
❖ Υλοποίηση Ελέγχων με δοκιμαστικά αρχεία και αρχείο journaling	56

Εισαγωγή

Στο πλαίσιο του μαθήματος ΜΥΥ601 "Λειτουργικά Συστήματα", ανατέθηκε η δεύτερη εργαστηριακή άσκηση με θέμα την υλοποίηση ενός αρχείου καταγραφής στο σύστημα αρχείων VFAT του Linux. Ο σκοπός της άσκησης είναι η επέκταση του συστήματος αρχείων VFAT στην βιβλιοθήκη του πυρήνα Linux, με στόχο να παρέχουμε αρχείο καταγραφής (journaling) που αποθηκεύει τροποποιήσεις που συμβαίνουν στο σύστημα αρχείων κατά την εκτέλεση εφαρμογών. Η δημιουργία αυτού του αρχείου έχει σκοπό την επαναφορά του συστήματος σε περίπτωση βλάβης. Μας δίνεται ο πηγαίος κώδικας του πυρήνα του Linux με την μορφή βιβλιοθήκης. Ακολουθεί η αναλυτική εξήγηση των βημάτων που ακολουθήσαμε για να υλοποιήσουμε τα ζητούμενα της άσκησης.

Εκτέλεση Προγράμματος

Για την εκτέλεση του προγράμματος που αναπτύχθηκε στο πλαίσιο της εργασίας, αρχικά ανοίγουμε το τερματικό στην εικονική μηχανή στον κατάλογο **/home/myy601/**.

Αφού ανοίξει το τερματικό, εκτελούμε την εντολή **cd lkl-source/tools/lkl** για να μετακινηθούμε στον κατάλογο lkl-source/tools/lkl.

```
myy601@myy601lab2:~$ cd lkl-source/tools/lkl
```

Έπειτα, εκτελούμε την εντολή **make -j8 clean** για να καθαρίσουμε την τρέχουσα μεταγλώττιση.

```
myy601@myy601lab2:~/lkl-source/tools/lkl$ make -j8 clean
```

Αφού ολοκληρωθεί η διαδικασία, μεταγλωττίζουμε τη βιβλιοθήκη με την εντολή **make -j8**. Η διαδικασία αυτή μπορεί να διαρκέσει μερικά λεπτά.

```
myy601@myy601lab2:~/lkl-source/tools/lkl$ make -j8
```

Σημαντικό είναι να τονίσουμε ότι για τη δημιουργία του /tmp/disk θα πρέπει να εκτελέσουμε μετά από κάθε εκκίνηση του VM την εντολή **make run-tests**.

```
myy601@myy601lab2:~/lkl-source/tools/lkl$ make run-tests  
./tests/run.py
```

SUITE	boot
TEST	ok
TEST	ok mutex
TEST	ok semaphore
TEST	ok join
TEST	ok start_kernel
TEST	ok getpid
TEST	ok syscall_latency
TEST	ok umask
TEST	ok umask2
TEST	ok creat
TEST	-

```

TEST      ok      umask2
TEST      ok      creat
TEST      ok      close
TEST      ok      failopen
TEST      ok      open
TEST      ok      write
TEST      ok      lseek_cur
TEST      ok      lseek_end
TEST      ok      lseek_set
TEST      ok      read
TEST      ok      fstat
TEST      ok      mkdir
TEST      ok      stat
TEST      ok      nanosleep
TEST      ok      pipe2
TEST      ok      epoll
TEST      ok      mount_fs_proc
TEST      ok      chdir_proc
TEST      ok      open_cwd
TEST      ok      getdents64
TEST      ok      close_dir_fd
TEST      ok      chdir_root
TEST      ok      umount_fs_proc
TEST      ok      lo_ifup
TEST      ok      gettid
TEST      ok      syscall_thread
TEST      ok      many_syscall_threads
TEST      ok      stop_kernel
SUITE
TEST      ok      disk_vfat
TEST      ok      prepfs_vfat
TEST      ok      disk_add
TEST      ok      start_kernel
TEST      ok      mount_dev
TEST      ok      chdir_mnt_point
TEST      ok      opendir
TEST      ok      readdir
TEST      ok      closedir
TEST      ok      umount_dev
TEST      ok      stop_kernel
TEST      ok      disk_remove
Summary: 2 suites run, 47 tests, 47 ok, 0 not ok, 0 skipped

```

Έπειτα, αναλόγως με το τι θέλουμε να κάνουμε, εκτελούμε και μία εντολή από τις παρακάτω:

1. Για GNU Debugger: **gdb ./cptofs** (*σελίδα 6*)
2. Για εκτέλεση cptofs: **./cptofs -i /tmp/disk -p -t vfat XFILE /**
όπου XFILE ένα από τα αρχεία που σας έχουμε δώσει για δοκιμή (*σελίδα 56*)

Παραδοτέο Εξήγηση

Στο παραδοτέο μας περιέχεται:

- η περιγραφή της άσκησης και του κώδικα μας στο **Report.pdf** και
- ένας φάκελος **lkl-source.zip**, στον οποίο, όπως μας ζητήθηκε και από την εκφώνηση της άσκησης, περιλαμβάνονται τα αρχεία που έχουμε αλλάξει (**dir.c**, **fat.h**, **inode.c**, **misc.c**, **namei_msdos.c**, **namei_vfat.c**, **fat_test.c**, **cache.c**, **fatent.c**) καθώς και ένας φάκελος **test_files** με τα νέα αρχεία που δημιουργήσαμε για την υλοποίηση και τον έλεγχο του προγράμματος.

Προσδιορίζουμε το μονοπάτι στο οποίο βρίσκονται τα αρχεία που κάναμε αλλαγές στον πηγαίο κώδικα που μας δίνεται:

```
/home/myy601/lklsource/fs/fat/dir.c  
/home/myy601/lklsource/fs/fat/fat.h  
/home/myy601/lklsource/fs/fat/inode.c  
/home/myy601/lklsource/fs/fat/misc.c  
/home/myy601/lklsource/fs/fat/namei_msdos.c  
/home/myy601/lklsource/fs/fat/namei_vfat.c  
/home/myy601/lklsource/fs/fat/fat_test.c  
/home/myy601/lklsource/fs/fat/cache.c  
/home/myy601/lklsource/fs/fat/fatent.c
```

1ο Βήμα: Κατανόηση της βασικής λειτουργίας του συστήματος αρχείων VFAT

Αρχικά, επιχειρήσαμε να κατανοήσουμε τη λειτουργία των διάφορων καταλόγων του πηγαίου κώδικα της LKL. Εξερευνήσαμε τους καταλόγους arch/lkl, tools/lkl, fs/fat και mm όπως αναφέρονται στην εκφώνηση, προσπαθώντας να κατανοήσουμε τον ρόλο κάθε καταλόγου. Σε πρώτο επίπεδο, επικεντρωθήκαμε στην κατανόηση των βασικών λειτουργιών κάθε δομής, όπως περιγράφεται στην εκφώνηση. Μελετήσαμε τα 4 βασικά αντικείμενα που περιλαμβάνει το VFS:

- Superblock**: περιλαμβάνει μεταδεδομένα (ιδιότητες και χαρακτηριστικά) του συστήματος αρχείων.
- Inode**: περιλαμβάνει μεταδεδομένα για ένα συγκεκριμένο αρχείο ή φάκελο.
- Dentry**: μια εγγραφή καταλόγου που αντιπροσωπεύει ένα στοιχείο (φάκελο ή αρχείο) ενός μονοπατιού.
- File**: δομή που περιγράφει ένα ανοιχτό αρχείο που είναι συσχετισμένο με μια διεργασία.

Ακολουθώντας αυτά τα βήματα, προχωρήσαμε στο πρώτο στάδιο της άσκησης. Χρησιμοποιήσαμε το δοκιμαστικό αρχείο που αναφέρεται στην εκφώνηση της άσκησης (**lklfuse.c**) και προσπαθήσαμε να κατανοήσουμε το “μονοπάτι” του κώδικα όταν εκτελείται με τη βοήθεια του αποσφαλματωτή gdb.

Εκσφαλμάτωση Κώδικα με GDB

Το GDB (GNU Debugger) είναι ένα εργαλείο αποσφαλμάτωσης που χρησιμοποιείται για την ανάλυση και επίλυση προβλημάτων σε προγράμματα. Το GDB επιτρέπει την εκτέλεση των προγραμμάτων βήμα προς βήμα, να παρακολουθούν τις τιμές των μεταβλητών, να εκτελούν εντολές σε συνθήκες διακοπής (breakpoints), και να ανιχνεύουν την πορεία της εκτέλεσης του προγράμματος.

Βασικά Χαρακτηριστικά του GDB:

1. **Stepping:** Μας δίνει τη δυνατότητα να εκτελούμε το πρόγραμμα γραμμή-γραμμή ή να εισερχόμαστε σε συναρτήσεις.
2. **Breakpoints:** Επιτρέπει τη θέσπιση σημείων στο πρόγραμμα όπου η εκτέλεση σταματάει, ώστε να εξετάσουμε την κατάσταση του προγράμματος.
3. **Παρακολούθηση Μεταβλητών:** Μπορούμε να δούμε και να τροποποιήσουμε τις τιμές των μεταβλητών κατά την εκτέλεση του προγράμματος.
4. **Back Trace:** Παρέχει πληροφορίες για τη σειρά των κλήσεων των συναρτήσεων που οδήγησαν σε κάποιο σφάλμα.

Χρήση του GDB:

Κάνοντας **cd lklsource/tools/lkl** όπως προείπαμε, ακολουθούμε τα παρακάτω βήματα:

1. Εκτελούμε **make run-tests** ώστε να δημιουργήσουμε τον δίσκο (/tmp/disk) .
2. Στην συνέχεια εκτελούμε **gdb ./cptofs** .

```
myy601@myy601lab2:~/lkl-source/tools/lkl$ gdb ./cptofs
GNU gdb (Debian 13.1-3)
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./cptofs...
(gdb) 
```

3. Αφού ανοίξει το gdb βάζουμε την εντολή **break run_syscall**.

```
(gdb) break run_syscall
Breakpoint 1 at 0x7b89b: file arch/lkl/kernel/syscalls.c, line 40.
```

4. Εκτελούμε την εντολή **run i /tmp/disk p t vfat lklfuse.c** και παρατηρούμε το παρακάτω breakpoint:

```
[New Thread 0x7ffffd5ff36c0 (LWP 26136)]
[New Thread 0x7ffffd67f46c0 (LWP 26137)]
[Thread 0x7ffffd6ff56c0 (LWP 26136) exited]
[ 0.300570] Warning: unable to open an initial console.
[ 0.300931] This architecture does not have kernel memory protection.
[ 0.301079] Run /init as init process
[New Thread 0x7ffffd6ff56c0 (LWP 26138)]
[New Thread 0x7ffffd5ff36c0 (LWP 26139)]
[Thread 0x7ffffd5ff36c0 (LWP 26139) exited]

Thread 1 "cptofs" hit Breakpoint 1, run_syscall (params=0xfffffffffae10, no=34) at arch/lkl/kernel/syscalls.c:40
40          if (no < 0 || no >= __NR_syscalls)
(gdb) ■
```

5. Χρησιμοποιούμε την εντολή **continue** ή **c** για να συνεχίσουμε την εκτέλεση μέχρι το επόμενο breakpoint και παρατηρούμε και τα printk που θα αναλύσουμε στη συνέχεια:

```
Thread 1 "cptofs" hit Breakpoint 1, run_syscall (params=0xfffffffffce90, no=40) at arch/lkl/kernel/syscalls.c:40
40          if (no < 0 || no >= __NR_syscalls)
(gdb) c
Continuing.
[New Thread 0x7ffffd5ff36c0 (LWP 26161)]
[Thread 0x7ffffd5ff36c0 (LWP 26161) exited]
[ 308.771033] * ~ ENTERING fat_alloc_inode (inode.c/struct super_operations fat_sops) ~ *
[ 308.771093] * ~ ENTERING fat_alloc_inode (inode.c/struct super_operations fat_sops) ~ *
[New Thread 0x7ffffd5ff36c0 (LWP 26162)]
[Thread 0x7ffffd5ff36c0 (LWP 26162) exited]
```

Όσον αφορά τα breakpoints, δοκιμάσαμε αρκετά κυρίως για να εξερευνήσουμε τον κώδικα. Μερικά από αυτά είναι τα **break fat_fill_super**, **break lkl_mount_dev**, **break vfat_mount**, **break lkl_sys_mount**, **break lkl_sys_open** κ.α.

□ 2ο Βήμα: Πρόσθεση συνάρτησης printk() στο πηγαίο κώδικα της LKL

Αποφασίσαμε να εστιάσουμε σε συγκεκριμένες λειτουργίες του συστήματος αρχείων FAT, όπως το superblock, η μνήμη, οι εγγραφές, τα αρχεία, τα inodes και οι κατάλογοι. Για να κατανοήσουμε τη σειρά εκτέλεσης των πεδίων των δομών, προσθέσαμε τη λειτουργία `printk(KERN_INFO "...")` σε κατάλληλες συναρτήσεις στους καταλόγους `fat/`, προσδίδοντας το όνομα της κάθε συνάρτησης που καλείται κατά την εκτέλεση της εφαρμογής.

Ενδεικτικό παράδειγμα εντολής μέσα στον κώδικα:

```
static void fat_evict_inode(struct inode *inode)
{
    printk(KERN_INFO "* ~ ENTERING fat_evict_inode (inode.c/struct super_operations fat_sops) ~ *");
    truncate_inode_pages_final(&inode->i_data);
```

Η υλοποίηση του συστήματος αρχείων VFAT στο Linux περιλαμβάνει διάφορες λειτουργίες για τη διαχείριση δομών όπως το superblock, τα inodes, τα αρχεία και οι κατάλογοι. Αυτές περιγράφονται παρακάτω:

- **Λειτουργίες Superblock:** Υλοποιούνται στη δομή `fat_sops` με τύπο `struct super_operations` στο αρχείο **fs/fat/inode.c**.
- **Λειτουργίες Μνήμης:** Βρίσκονται στη δομή `fat_aops` με τύπο `struct address_space_operations` στο αρχείο **fs/fat/inode.c**.
- **Λειτουργίες Εγγραφών FAT:** Υλοποιούνται στη δομή `struct fatent_operations` για `fat12/16/32_ops` στο αρχείο **fs/fat/fatent.c**.
- **Λειτουργίες Αρχείων:** Υλοποιούνται στη δομή `fat_file_operations` με τύπο `struct file_operations` στο αρχείο **fs/fat/file.c**.
- **Λειτουργίες Inode:** Υλοποιούνται στη δομή `fat_file_inode_operations` με τύπο `struct inode_operations` στο αρχείο **fs/fat/file.c**.
- **Λειτουργίες Καταλόγων:** Βρίσκονται στη δομή `struct inode_operations` με τύπο `msdos_dir_inode_operations` στο αρχείο **fs/fat/namei_msdos.c** για το σύστημα αρχείων FAT και στο αρχείο **fs/fat/namei_vfat.c** για το σύστημα αρχείων VFAT.

Οι συναρτήσεις που προσθέσαμε `printf(KERN_INFO ".....")` είναι οι παρακάτω:

1. Στο αρχείο **inode.c** έχουμε πολλές προσθήκες της εντολής `printf(KERN_INFO ".....")`. Αυτές οι εντολές βρίσκονται στις εξής συναρτήσεις:

- `fat_writepage`
- `fat_writepages`
- `fat_read_folio`
- `fat_readahead`
- `fat_write_begin`
- `fat_write_end`
- `fat_direct_IO`
- `_fat_bmap`
- `fat_evict_inode`
- `fat_set_state`
- `fat_put_super`
- `fat_alloc_inode`
- `fat_free_inode`
- `fat_remount`
- `fat_statfs`
- `_fat_write_inode`
- `fat_write_inode`
- `fat_show_options`
- `fat_fill_super`

Δομές στις οποίες ανήκουν αυτές οι συναρτήσεις:

Αυτές οι συναρτήσεις ανήκουν σε διάφορες δομές, κυρίως στις δομές *struct address space operations* και *struct super operations*. Συγκεκριμένα:

- ❖ Οι συναρτήσεις `fat_writepage`, `fat_writepages`, `fat_read_folio`, `fat_readahead`, `fat_write_begin`, `fat_write_end`, `fat_direct_IO`, και `_fat_bmap` είναι μέρος της δομής *struct address space operations*.
- ❖ Οι συναρτήσεις `fat_evict_inode`, `fat_put_super`, `fat_alloc_inode`, `fat_free_inode`, `fat_remount`, `fat_statfs`, `fat_write_inode`, και `fat_show_options` είναι μέρος της δομής *struct super operations*.
- ❖ Οι υπόλοιπες συναρτήσεις, όπως `fat_set_state`, `fat_fill_super`, και `_fat_write_inode`, ανήκουν στην γενική υλοποίηση του συστήματος αρχείων FAT.

2. Στο αρχείο *fatent.c* οι εντολές `printk(KERN_INFO "....")` βρίσκονται στις εξής συναρτήσεις:

- `fat12_ent_blocknr`
- `fat_ent_blocknr`
- `fat12_ent_set_ptr`
- `fat16_ent_set_ptr`
- `fat32_ent_set_ptr`
- `fat12_ent_bread`
- `fat_ent_bread`
- `fat12_ent_get`
- `fat16_ent_get`
- `fat32_ent_get`
- `fat12_ent_put`
- `fat16_ent_put`
- `fat32_ent_put`
- `fat12_ent_next`
- `fat16_ent_next`
- `fat32_ent_next`
- `fat_ent_access_init`
- `fat_ent_update_ptr`
- `fat_alloc_clusters`

Δομή στην οποία ανήκουν αυτές οι συναρτήσεις:

Η δομή στην οποία ανήκουν αυτές οι συναρτήσεις είναι η *struct fatent_operations*. Αυτή η δομή περιλαμβάνει δείκτες σε συναρτήσεις που σχετίζονται με την επεξεργασία των FAT entries (FAT12, FAT16, FAT32) στα αρχεία συστήματος FAT.

3. Στο αρχείο **file.c** οι εντολές printk(KERN_INFO "....") βρίσκονται στις εξής συναρτήσεις:

- fat_generic_ioctl
- fat_file_release
- fat_file_fsync
- fat_fallocate
- fat_getattr
- fat_setattr

Δομές στις οποίες ανήκουν αυτές οι συναρτήσεις:

- ❖ Οι συναρτήσεις **fat_generic_ioctl**, **fat_file_release**, **fat_file_fsync**, και **fat_fallocate** ανήκουν στη δομή **struct file_operations** μέσω της μεταβλητής **fat_file_operations**.
- ❖ Οι συναρτήσεις **fat_getattr** και **fat_setattr** ανήκουν στη δομή **struct inode_operations** μέσω της μεταβλητής **fat_file_inode_operations**.

4. Στο αρχείο **namei_msdos.c** οι εντολές printk(KERN_INFO "....") βρίσκονται στις εξής συναρτήσεις:

- msdos_lookup
- msdos_create
- msdos_rmdir
- msdos_mkdir
- msdos_unlink
- msdos_rename

Δομή στην οποία ανήκουν αυτές οι συναρτήσεις:

Οι συναρτήσεις **msdos_lookup**, **msdos_create**, **msdos_rmdir**, **msdos_mkdir**, **msdos_unlink**, και **msdos_rename** ανήκουν στη δομή **struct inode_operations** μέσω της μεταβλητής **msdos_dir_inode_operations**.

5. Στο αρχείο **namei_vfat.c** οι εντολές printk(KERN_INFO "....") βρίσκονται στις εξής συναρτήσεις:

- vfat_lookup
- vfat_create
- vfat_rmdir
- vfat_unlink
- vfat_mkdir
- vfat_rename2

Δομή στην οποία ανήκουν αυτές οι συναρτήσεις:

Οι συναρτήσεις `vfat_lookup`, `vfat_create`, `vfat_rmdir`, `vfat_unlink`, `vfat_mkdir`, και `vfat_rename2` ανήκουν στη δομή `struct inode_operations` μέσω της μεταβλητής `vfat_dir_inode_operations`.

□ Εκτέλεση με δοκιμαστικό αρχείο

Επειτα, όπως μας καθοδηγεί και η εκφώνηση, εκτελούμε τις εντολές στο τερματικό:

- `cd lkl-source/tools/lkl`
- `make -j8 clean`
- `make -j8`
- `make run-tests`
- Εκτελούμε την εντολή `./cptofs -i /tmp/disk -p -t vfat lklfuse.c /`

Παρακάτω φαίνεται η εκτύπωση στο τερματικό μετά την αντιγραφή του `lklfuse.c` στο `disk`.

```
[ 0.082983] Run /init as init process
[ 0.085287] * ~ ENTERING fat_alloc_inode (inode.c/struct super_operations fat_sops) ~ *
[ 0.085331] * ~ ENTERING fat_alloc_inode (inode.c/struct super_operations fat_sops) ~ *
[ 0.085379] * ~ ENTERING fat_alloc_inode (inode.c/struct super_operations fat_sops) ~ *
[ 0.097723] * ~ ENTERING vfat_lookup (namei_vfat.c/struct inode_operations vfat_dir_inode_operations) ~ *
[ 0.097878] * ~ ENTERING vfat_create (namei_vfat.c/struct inode_operations vfat_dir_inode_operations) ~ *
[ 0.098089] * ~ ENTERING fat_alloc_inode (inode.c/struct super_operations fat_sops) ~ *
[ 0.099926] * ~ ENTERING fat_write_begin (inode.c/struct address_space_operations fat_aops) ~ *
[ 0.100004] * ~ ENTERING fat_ent_blocknr (fatent.c/struct fatent_operations fat16_ops AND fat32_ops) ~ *
[ 0.100045] * ~ ENTERING fat_ent_bread (fatent.c/struct fatent_operations fat16_ops AND fat32_ops) ~ *
[ 0.110589] * ~ ENTERING fat16_ent_set_ptr (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.110658] * ~ ENTERING fat16_ent_get (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.110674] * ~ ENTERING fat16_ent_put (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.110954] * ~ ENTERING fat_write_end (inode.c/struct address_space_operations fat_aops) ~ *
[ 0.111028] * ~ ENTERING fat_write_begin (inode.c/struct address_space_operations fat_aops) ~ *
[ 0.111090] * ~ ENTERING fat_write_end (inode.c/struct address_space_operations fat_aops) ~ *
[ 0.111148] * ~ ENTERING fat_write_begin (inode.c/struct address_space_operations fat_aops) ~ *
[ 0.111199] * ~ ENTERING fat_ent_blocknr (fatent.c/struct fatent_operations fat16_ops AND fat32_ops) ~ *
[ 0.111241] * ~ ENTERING fat_ent_bread (fatent.c/struct fatent_operations fat16_ops AND fat32_ops) ~ *
[ 0.111257] * ~ ENTERING fat16_ent_set_ptr (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.111292] * ~ ENTERING fat16_ent_get (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.111306] * ~ ENTERING fat16_ent_put (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.111342] * ~ ENTERING fat_ent_blocknr (fatent.c/struct fatent_operations fat16_ops AND fat32_ops) ~ *
[ 0.111355] * ~ ENTERING fat_ent_bread (fatent.c/struct fatent_operations fat16_ops AND fat32_ops) ~ *
[ 0.111391] * ~ ENTERING fat16_ent_set_ptr (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.111404] * ~ ENTERING fat16_ent_get (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.111439] * ~ ENTERING fat_ent_blocknr (fatent.c/struct fatent_operations fat16_ops AND fat32_ops) ~ *
[ 0.111452] * ~ ENTERING fat_ent_bread (fatent.c/struct fatent_operations fat16_ops AND fat32_ops) ~ *
[ 0.111487] * ~ ENTERING fat16_ent_set_ptr (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.111500] * ~ ENTERING fat16_ent_get (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.111535] * ~ ENTERING fat16_ent_put (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.111549] * ~ ENTERING fat_ent_blocknr (fatent.c/struct fatent_operations fat16_ops AND fat32_ops) ~ *
[ 0.111603] * ~ ENTERING fat_ent_bread (fatent.c/struct fatent_operations fat16_ops AND fat32_ops) ~ *
[ 0.111642] * ~ ENTERING fat16_ent_set_ptr (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.111655] * ~ ENTERING fat16_ent_get (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.111699] * ~ ENTERING fat_write_end (inode.c/struct address_space_operations fat_aops) ~ *
[ 0.111823] * ~ ENTERING fat_write_begin (inode.c/struct address_space_operations fat_aops) ~ *
[ 0.122415] * ~ ENTERING fat_write_end (inode.c/struct address_space_operations fat_aops) ~ *
[ 0.122744] * ~ ENTERING fat_setattr (file.c/struct inode_operations fat_file_inode_operations) ~ *
[ 0.141957] * ~ ENTERING fat_file_release (file.c/struct file_operations fat_file_operations) ~ *
[ 1.131875] * ~ ENTERING fat_remount (inode.c/struct super_operations fat_sops) ~ *
[ 1.131975] * ~ ENTERING fat_write_inode (inode.c/struct super_operations fat_sops) ~ *
[ 1.132078] * ~ ENTERING fat_writepages (inode.c/struct address_space_operations fat_aops) ~ *
[ 1.132162] * ~ ENTERING fat_write_inode (inode.c/struct super_operations fat_sops) ~ *
[ 1.133991] reboot: Restarting system
```

Με βάση τα μηνύματα printk, το μονοπάτι των συναρτήσεων κατά την εντολή αντιγραφής είναι το εξής:

1. fat_alloc_inode	24. fat16_ent_set_ptr
2. fat_alloc_inode	25. fat16_ent_get
3. fat_alloc_inode	26. fat_ent_blocknr
4. vfat_lookup	27. fat_ent_bread
5. vfat_create	28. fat16_ent_set_ptr
6. fat_alloc_inode	29. fat16_ent_get
7. fat_write_begin	30. fat16_ent_put
8. fat_ent_blocknr	31. fat_ent_blocknr
9. fat_ent_bread	32. fat_ent_bread
10. fat16_ent_set_ptr	33. fat16_ent_set_ptr
11. fat16_ent_get	34. fat16_ent_get
12. fat16_ent_put	35. fat_write_end
13. fat_write_end	36. fat_write_begin
14. fat_write_begin	37. fat_write_end
15. fat_write_end	38. fat_setattr
16. fat_write_begin	39. fat_file_release
17. fat_ent_blocknr	40. fat_remount
18. fat_ent_bread	41. fat_write_inode
19. fat16_ent_set_ptr	42. fat_writepages
20. fat16_ent_get	43. fat_write_inode
21. fat16_ent_put	
22. fat_ent_blocknr	
23. fat_ent_bread	

Σε αυτό το σημείο της ανάλυσης, θα περιγράψουμε και θα εξηγήσουμε τη λειτουργία των συναρτήσεων που εμπλέκονται στην εντολή αντιγραφής αρχείου, όπως καταγράφηκαν από τα μηνύματα printk.

Αυτές οι συναρτήσεις μαζί σχηματίζουν τη διαδρομή που ακολουθεί η εντολή αντιγραφής ενός αρχείου στο σύστημα αρχείων FAT. Κάθε συνάρτηση εκτελεί ένα συγκεκριμένο ρόλο και συνεργάζεται με τις υπόλοιπες για να ολοκληρώσει τη διαδικασία.

1.fat_alloc_inode (inode.c/struct super_operations fat_sops)

Περιγραφή: Υπεύθυνη για την εκχώρηση (allocation) νέων inodes για το σύστημα αρχείων FAT.

Δειτουργία: Δημιουργεί ένα νέο inode, αρχικοποιεί τις δομές δεδομένων του και επιστρέφει έναν δείκτη σε αυτό. Χρησιμοποιεί τη συνάρτηση alloc_inode_sb για την κατανομή μνήμης και την αρχικοποίηση της δομής msdos_inode_info. Η συνάρτηση καλείται τρεις φορές για την εκχώρηση νέων inodes.

2.vfat_lookup (namei_vfat.c/struct inode_operations vfat_dir_inode_operations)

Περιγραφή: Αναζητά μια συγκεκριμένη εγγραφή σε έναν κατάλογο και εξάγει το αντίστοιχο inode.

Δειτουργία: Χρησιμοποιεί τη συνάρτηση vfat_find για να ελέγξει αν υπάρχει η εγγραφή στον κατάλογο. Αν υπάρχει, επιστρέφει το inode χρησιμοποιώντας τη συνάρτηση fat_build_inode, διαφορετικά επιστρέφει σφάλμα. Χρησιμοποιεί κλειδαριά (mutex) για αμοιβαίο αποκλεισμό κατά την πρόσβαση στα δεδομένα του συστήματος αρχείων.

3.vfat_create (namei_vfat.c/struct inode_operations vfat_dir_inode_operations)

Περιγραφή: Δημιουργεί μια νέα εγγραφή (inode) σύμφωνα με τα ορίσματα που της δίνονται.

Δειτουργία: Εισάγει μια νέα εγγραφή χρησιμοποιώντας τη συνάρτηση vfat_add_entry και δημιουργεί το νέο inode συσχετίζοντας το με το dentry. Ενημερώνει τα μεταδεδομένα του καταλόγου και δημιουργεί το νέο inode με τη συνάρτηση fat_build_inode.

4.fat_write_begin (inode.c/struct address_space_operations fat_aops)

Περιγραφή: Ξεκινά τη διαδικασία εγγραφής δεδομένων στο αρχείο.

Δειτουργία: Προετοιμάζει τη διαδικασία εγγραφής και διαχειρίζεται τη μνήμη και τους δείκτες για την εγγραφή των δεδομένων στη σωστή θέση στο αρχείο. Χρησιμοποιεί τη συνάρτηση cont_write_begin για να ξεκινήσει την εγγραφή.

5.fat_ent_blocknr (fatent.c/struct fatent_operations fat16_ops AND fat32_ops)

Περιγραφή: Υπολογίζει τον αριθμό του μπλοκ που αντιστοιχεί σε μια εγγραφή του FAT.

Δειτουργία: Μεταφράζει τις λογικές διευθύνσεις σε φυσικές διευθύνσεις στο σύστημα αρχείων FAT. Υπολογίζει την αντιστοιχία των εγγραφών FAT σε μπλοκ και byte offset.

6.fat_ent_bread (fatent.c/struct fatent_operations fat16_ops AND fat32_ops)

Περιγραφή: Διαβάζει το περιεχόμενο του μπλοκ από το δίσκο.

Δειτουργία: Φορτώνει το περιεχόμενο ενός συγκεκριμένου μπλοκ στη μνήμη για να είναι διαθέσιμο για επεξεργασία. Χρησιμοποιεί τη συνάρτηση sb_bread για την ανάγνωση του μπλοκ από το δίσκο.

7.fat16_ent_set_ptr (fatent.c/struct fatent_operations fat16_ops)

Περιγραφή: Ορίζει τον δείκτη εγγραφής για το FAT16.

Δειτουργία: Αρχικοποιεί τον δείκτη για να δείχνει στη σωστή θέση στον πίνακα εγγραφών του FAT16.

8.fat16_ent_get (fatent.c/struct fatent_operations fat16_ops)

Περιγραφή: Ανακτά την εγγραφή από το FAT16.

Δειτουργία: Διαβάζει την τιμή από τον πίνακα εγγραφών του FAT16. Ελέγχει αν η τιμή είναι έγκυρη και επιστρέφει την επόμενη εγγραφή ή το τέλος του πίνακα.

9.fat16_ent_put (fatent.c/struct fatent_operations fat16_ops)

Περιγραφή: Αποθηκεύει μια εγγραφή στο FAT16.

Δειτουργία: Ενημερώνει τον πίνακα εγγραφών του FAT16 με τη νέα τιμή. Χρησιμοποιεί τη συνάρτηση mark_buffer_dirty_inode για να σημειώσει ότι το μπλοκ περιέχει τροποποιημένα δεδομένα.

10.fat_write_end (inode.c/struct address_space_operations fat_aops)

Περιγραφή: Ολοκληρώνει τη διαδικασία εγγραφής δεδομένων.

Λειτουργία: Επιβεβαιώνει την εγγραφή των δεδομένων, ενημερώνει τους δείκτες και τα μεταδεδομένα. Χρησιμοποιεί τη συνάρτηση generic_write_end για να ολοκληρώσει την εγγραφή.

11.fat_setattr (file.c/struct inode_operations fat_file_inode_operations)

Περιγραφή: Ρυθμίζει τα χαρακτηριστικά ενός αρχείου.

Λειτουργία: Ενημερώνει τα μεταδεδομένα του inode με τα νέα χαρακτηριστικά που έχουν οριστεί. Χειρίζεται τις αλλαγές μεγέθους, δικαιωμάτων και χρονικών σφραγίδων (timestamps).

12.fat_file_release (file.c/struct file_operations fat_file_operations)

Περιγραφή: Απελευθερώνει ένα αρχείο.

Λειτουργία: Ελευθερώνει πόρους που σχετίζονται με το αρχείο και διαχειρίζεται την κατάσταση της εγγραφής. Καλεί τη συνάρτηση fat_flush_inodes αν η επιλογή flush είναι ενεργοποιημένη.

13.fat_remount (inode.c/struct super_operations fat_sops)

Περιγραφή: Επανεγκαθιστά το σύστημα αρχείων.

Λειτουργία: Διαχειρίζεται την επαναφορά των ρυθμίσεων του συστήματος αρχείων μετά από αποπροσάρτηση και επαναπροσάρτηση. Ελέγχει τις επιλογές μόνο ανάγνωσης και ενημερώνει την κατάσταση του συστήματος αρχείων.

14.fat_write_inode (inode.c/struct super_operations fat_sops)

Περιγραφή: Εγγράφει τα δεδομένα του inode στον δίσκο.

Λειτουργία: Διαχειρίζεται τη διαδικασία εγγραφής του inode και των μεταδεδομένων του στο δίσκο. Εξασφαλίζει ότι οι αλλαγές στο inode αποθηκεύονται μόνιμα.

15.fat_writepages (inode.c/struct address_space_operations fat_aops)

Περιγραφή: Εγγράφει τις σελίδες του inode στον δίσκο.

Λειτουργία: Χρησιμοποιεί τη συνάρτηση mpage_writepages για να εγγράψει τις σελίδες δεδομένων στο δίσκο. Διαχειρίζεται τις σελίδες μνήμης που περιέχουν τα δεδομένα του αρχείου και εξασφαλίζει την εγγραφή τους.

3ο Βήμα: Καταγραφή των λειτουργιών σε αρχείο

Αρχείο Καταγραφής - Journaling με Metadata

Στην υλοποίηση του συστήματος καταγραφής (journaling) για το σύστημα αρχείων VFAT του Linux, επιλέγουμε τη μέθοδο Metadata Journaling. Η επιλογή αυτή γίνεται με σκοπό να καταγράφουμε μόνο τις δομές μεταδεδομένων, όπως τα superblocks και τα inodes, αφήνοντας τα δεδομένα να γραφούν απευθείας στο δίσκο πριν την εγγραφή των μεταδεδομένων στο journal. Αυτή η προσέγγιση μειώνει τον όγκο των εγγραφών στο journal και βελτιώνει την απόδοση, ενώ εξακολουθεί να **διασφαλίζει την ακεραιότητα των μεταδεδομένων σε περίπτωση σφάλματος**.

Διαδικασία Υλοποίησης:

- Δημιουργία Αρχείου Καταγραφής:**

Χρησιμοποιούμε κανονικές κλήσεις συστήματος για να δημιουργήσουμε και να ανοίξουμε ένα αρχείο καταγραφής στο σύστημα ext4. Το αρχείο αυτό θα χρησιμοποιηθεί για την καταγραφή των μεταδεδομένων.

Αρχεία Καταγραφής - Journaling

To journal αποθηκεύεται ως αρχείο είτε στο ίδιο σύστημα αρχείων, είτε σε κάποιο εξωτερικό σύστημα αρχείων

Full Journaling

- Γράφει στο αρχείο καταγραφής τις δομές μεταδεδομένων (π.χ. superblock, inodes) και τα δεδομένα των αρχείων πριν πραγματοποιήσει τις αντίστοιχες ενημερώσεις στις δομές του συστήματος

Metadata Journaling

- Γράφει στο αρχείο καταγραφής μόνο τις δομές μεταδεδομένων (π.χ. superblock, inodes)
- Τα δεδομένα πρέπει να γραφούν στο δίσκο πριν γίνει η εγγραφή των μεταδεδομένων στο journal

Σε περίπτωση σφάλματος το σύστημα διαβάζει το journal και εκτελεί τις λειτουργίες που έχουν καταγραφεί

ΜΥΥ601: Λειτουργικά Συστήματα, Πανεπιστήμιο Ιωαννίνων

44

- Αποθήκευση των Μεταδεδομένων:**

Κατά την εκτέλεση κρίσιμων λειτουργιών που αλλάζουν τα μεταδεδομένα του συστήματος FAT (π.χ., δημιουργία αρχείων, διαγραφή, τροποποίηση inodes), καταγράφουμε τις αλλαγές στο αρχείο καταγραφής. Χρησιμοποιούμε την κλήση write() για την εγγραφή των μεταδεδομένων στο αρχείο καταγραφής.

- Καταγραφή των Αλλαγών:**

Για κάθε σημαντική λειτουργία που αλλάζει τα μεταδεδομένα, όπως η δημιουργία ή η τροποποίηση ενός inode, καταγράφουμε το όνομα της συνάρτησης, τις παραμέτρους, και τις χρονικές σημάνσεις στο αρχείο καταγραφής. Αυτό περιλαμβάνει τα inodes, τα directory entries, και το superblock.

- Κλείσιμο Αρχείου Καταγραφής:**

Πριν τον τερματισμό της εφαρμογής, κλείνουμε το αρχείο καταγραφής με την close() για να διασφαλίσουμε ότι όλες οι ενημερώσεις έχουν μεταφερθεί στο δίσκο.

- Αποκατάσταση Από Σφάλματα:**

Σε περίπτωση σφάλματος, το σύστημα διαβάζει το journal και εκτελεί τις λειτουργίες που έχουν καταγραφεί για να αποκαταστήσει την κατάσταση του συστήματος αρχείων.

Περιγραφή της Διαδικασίας Καταγραφής (Journaling) στο Σύστημα Αρχείων VFAT

Η διαδικασία εισαγωγής δεδομένων στο αρχείο καταγραφής (journal) κατά την εκτέλεση λειτουργιών στο σύστημα αρχείων VFAT περιλαμβάνει την τροποποίηση των βασικών δομών του συστήματος αρχείων και την ενσωμάτωση κλήσεων συστήματος για την εγγραφή των μεταδεδομένων στο journal. Ακολουθεί αναλυτική περιγραφή των βημάτων που απαιτούνται για την υλοποίηση αυτής της διαδικασίας.

1. Αναζήτηση των Δομών Αποθήκευσης στον Δίσκο

Εξεκινήσαμε αναζητώντας τις δομές αποθήκευσης στον δίσκο που εμπλέκονται στο σύστημα αρχείων VFAT. Οι δομές αυτές βρίσκονται στο αρχείο msdos_fs.h και περιλαμβάνουν τις βασικές δομές μεταδεδομένων, όπως το superblock και τα inodes.

2. Προσδιορισμός της Διαδρομής κατά την Προσάρτηση (Mount)

Για να κατανοήσουμε πού πρέπει να προσθέσουμε τον κώδικα καταγραφής, μελετήσαμε τη διαδικασία που ακολουθείται κατά την προσάρτηση του συστήματος αρχείων. Η συνάρτηση init_msdos_fs() αρχικοποιεί το superblock και είναι υπεύθυνη για τη δημιουργία του στη μνήμη όταν το σύστημα αρχείων προσαρτάται. Η δομή msdos_fs_type περιλαμβάνει την msdos_mount, η οποία καλεί τη συνάρτηση fat_fill_super.

3. Τροποποίηση της δομής msdos_sb_info στο fat.h

Για να καταγράψουμε τις αλλαγές, προσθέσαμε ένα νέο πεδίο στη δομή msdos_sb_info που βρίσκεται στο αρχείο fat.h. Το πεδίο αυτό χρησιμοποιείται για να αποθηκεύσει τον file descriptor του αρχείου καταγραφής μας.

```
struct msdos_sb_info {
    int myJournal;
```

4. Αρχικοποίηση των write, open, close στο fat.h

Στο αρχείο fat.h αρχικοποιήσαμε τις συναρτήσεις write(), open() και close(), οι οποίες είναι υπεύθυνες για την υλοποίηση της καταγραφής, το άνοιγμα και το κλείσιμο του αρχείου καταγραφής journal.

```
ssize_t write(int file_descriptor, const void *buf, size_t length);
int open(const char *pathname, int flags, mode_t mode);
int close(int fd);
```

5. Ανοιγμα του Αρχείου Καταγραφής

Στη συνάρτηση fat_fill_super του αρχείου inode.c, αρχικοποιήθηκε το πεδίο myJournal και ανοίχτηκε το αρχείο καταγραφής:

```
//~~~~~anoigma journal~~~~~
sbi->myJournal = open("/tmp/journal", O_CREAT | O_RDWR | O_TRUNC, S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH);
//~~~~~
```

6. Συνάρτηση Καταγραφής Λεδούμένων στο Journal

Έχουμε δημιουργήσει μια συνάρτηση στο fat.h, τη **write_to_journal**.

Την τοποθετούμε στο fat.h αρχείο, έτσι ώστε να είναι ορατή σε όλα τα αρχεία.c του fs/fat, και έτσι θα μπορούμε να την καλούμε από εκεί σε όσες συναρτήσεις θέλουμε να εγγράψουμε τις αλλαγές στο journal. Παρακάτω δίνεται η αναλυτική εξήγηση της συνάρτησης **write_to_journal**.

```
static void write_to_journal(int file_descriptor, char buf[1024], size_t length) {
    length = strlen(buf); // strlen gia to length tou buffer
    printk(KERN_INFO "-----Journal write started.-----\n");
    write(file_descriptor, buf, length);
    printk(KERN_INFO "-----Journal write completed.-----\n");
    printk(KERN_INFO "-----\n");
}
```

- Δέχεται ως ορίσματα έναν ακέραιο `file_descriptor`, τον `buf`, δηλαδή τον buffer που κρατάμε τις αλλαγές κάθε φορά και το `length` που συμβολίζει το μέγεθος του buffer.
- Χρησιμοποιούμε τη συνάρτηση `strlen` για να μας επιστραφεί το μήκος του `buf` και το δηλώνουμε στη μεταβλητή `length`. Για να μπορέσουμε να καλέσουμε τη `strlen` πρέπει να εισάγουμε στην αρχή του fat.h αρχείου το: `#include <linux/string.h>`
- Με χρήση της `printk` δίνουμε μηνύματα ενημέρωσης για το στάδιο που βρισκόμαστε, στην αρχή ή στο τέλος της εγγραφής.
- Έπειτα εκτελούμε εγγραφη στο αρχείο journal χρησιμοποιώντας τη `write`, την οποία έχουμε ορίσει πιο πάνω .

Μέσα στον κώδικα που γίνονται οι αλλαγές κάνουμε τα εξής:

```
//-----ALLAGH-----
printk(KERN_INFO "-----\n");
printk(KERN_INFO "-----We are in fat_fill_super of inode.c.-----\n");
file_descriptor = sbi->myJournal;
sprintf(journal_buffer, "\n-----We are in fat_fill_super of inode.c.----- %u\n",
        sbi->sec_per_clus);
write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
//-----
```

- Έχουμε προσδιορίσει το μέγεθος του buffer στην αρχή του κάθε αρχείου:
`#define JOURNAL_BUFFER_SIZE 1024`
- Τυπώνουμε στο τερματικό σε ποια συνάρτηση και σε ποιο αρχείο βρισκόμαστε.
- Στην αρχή της συνάρτησης μαζί με τις υπόλοιπες δηλώσεις, ορίζουμε το buffer του journal
`char journal_buffer[JOURNAL_BUFFER_SIZE]; //dhlwsh buffer tou journal`
- Τυπώνουμε στο journal με την `sprintf` το περιεχόμενο του buffer και τη συνάρτηση στην οποία βρισκόμαστε, για την καλύτερη οργάνωση του journal.
- Καλούμε τη **write_to_journal** .

7. Δυναμική Κατανομή Μνήμης

Η χρήση της kmalloc εξυπηρετεί την ανάγκη για δυναμική κατανομή μνήμης στο kernel space του Linux, ιδιαίτερα για τη διαχείριση του buffer που χρησιμοποιείται για την καταγραφή των δεδομένων στο journal.

Παράδειγμα Χρήσης της kmalloc:

Ας δούμε ένα απόσπασμα κώδικα όπου χρησιμοποιείται η **kmalloc** για τη δημιουργία buffer:

```
static void fat_set_state(struct super_block *sb,
    unsigned int set, unsigned int force)
{
    struct buffer_head *bh;
    struct fat_boot_sector *b;
    struct msdos_sb_info *sbi = MSDOS_SB(sb);
    char *journal_buffer; //dhlwsh buffer tou journal
    int file_descriptor;

    journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
    if (!journal_buffer) {
        printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
        kfree(journal_buffer);
        return;
    }
    /* do not change anything if mounted read only */
}
```

8. Κλείσιμο του Αρχείου Καταγραφής

Στη συνάρτηση delayed_free του αρχείου inode.c, κλείνουμε το αρχείο καταγραφής:

```
close(myJournal);
```

Λεπτομέρειες Υλοποίησης της Καταγραφής (Journaling) στο Σύστημα Αρχείων VFAT

Λεπτομέρειες Υλοποίησης

Η υλοποίηση του FAT βρίσκεται στο φάκελο **fs/fat**

Δομές αποθήκευσης στο δίσκο

- **include/uapi/linux/msdos.h**
- **struct __fat_dirent, struct fat_boot_sector, struct fat_boot_finfo, struct msdos_dir_entry, struct msdos_dir_slot**

File Allocation Table

- **FAT entry: struct fat_entry (fs/fat/fat.h)**
- **Λειτουργίες: fs/fat/fat.h, fs/fat/fatten.h**

Superblock

- **Δομή: struct msdos_sb_info (fs/fat/fat.h)**
- **Λειτουργίες: struct super_operations fat_sops (fs/fat/inode.c)**
- **Αρχικοποίηση: int __init init_msdos_fs() (fs/fat/namei_msdos.c)**

Λεπτομέρειες Υλοποίησης

Inode

- **Δομή: struct msdos_inode_info (fs/fat/fat.h)**
- **Λειτουργίες: struct inode_operations fat_file_inode_operations (fs/fat/file.c)**
- **Δημιουργία: new_inode() (fs/fat/inode.c)**

Directory entries

- **Δομή: struct msdos_slot_info (fs/fat/fat.h)**
- **Λειτουργίες: struct inode_operations msdos_dir_inode_operations (fs/fat/namei_msdos.c)**

Files

- **Δομή: struct file (include/linux/fs.h)**
- **Λειτουργίες: struct file_operations fat_file_operations (fs/fat/file.c)**

Σύμφωνα με τις παραπάνω διαφάνειες του μαθήματος η υλοποίηση του FAT βρίσκεται στο φάκελο **fs/fat**. Στον φάκελο αυτό περιέχονται τα παρακάτω:

- ❖ Δομές αποθήκευσης στο δίσκο
- ❖ File Allocation Table (FAT)
- ❖ Superblock
- ❖ Inode
- ❖ Directory entries
- ❖ Files

Παρακάτω ακολουθεί η αναλυτική διερεύνηση του φακέλου **fs/fat** και των περιεχομένων του.

□ Δομές αποθήκευσης στο δίσκο

Στο αρχείο **msdos_fs.h** υπάρχουν οι δομές:

- ❖ **_fat dirent**
- ❖ **fat_boot_sector**
- ❖ **fat_boot_finfo**
- ❖ **msdos_dir_entry**
- ❖ **msdos_dir_slot**

Για κάθε μία από αυτές τις δομές θα δούμε που αλλάζουν τιμή οι μεταβλητές τους. Θα χρησιμοποιήσουμε την εντολή **grep -r** για να κάνουμε αναζήτηση για μια συμβολοσειρά σε έναν κατάλογο αναδρομικά. Στις δομές που παρατηρούμε αλλαγές στις τιμές των μεταβλητών προσθέτουμε κώδικα για να εγγράψουμε αυτές τις αλλαγές που γίνονται σε ένα αρχείο καταγραφής journal, έτσι ώστε να υπάρχουν εκεί αποθηκευμένες σε περίπτωση βλάβης.

> Ξεκινάμε από την **_fat dirent**. Παρακάτω έχουμε το screenshot από την εκτέλεση της εντολής **grep -r _fat dirent**.

```
myy601@myy601lab2:~/lkl-source$ grep -r _fat dirent
include/uapi/linux/msdos_fs.h:struct __fat_dirent {
include/uapi/linux/msdos_fs.h:#define VFAT_IOCTL_READDIR_BOTH      _IOR('r', 1, struct __fat_dirent[2])
include/uapi/linux/msdos_fs.h:#define VFAT_IOCTL_READDIR_SHORT _IOR('r', 2, struct __fat_dirent[2])
grep: vmlinux: binary file matches
grep: vmlinux.o: binary file matches
grep: tools/lkl/tests/disk: binary file matches
grep: tools/lkl/tests/boot: binary file matches
grep: tools/lkl/tests/net-test: binary file matches
grep: tools/lkl/tests/disk-vfio-pci: binary file matches
grep: tools/lkl/tests/test-dlmopen: binary file matches
grep: tools/lkl/liblkl.a: binary file matches
grep: tools/lkl/cptofs: binary file matches
grep: tools/lkl/lib/liblkl.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-zpoline.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-hijack.so: binary file matches
grep: tools/lkl/lib/lkl.o: binary file matches
grep: tools/lkl/fs2tar: binary file matches
fs/fat/dir.c:FAT_IOCTL_FILLDIR_FUNC(fat_ioctl_filldir, __fat_dirent)
fs/fat/dir.c:    struct __fat_dirent __user *d1 = (struct __fat_dirent __user *)arg;
grep: fs/fat/dir.o: binary file matches
grep: lkl.o: binary file matches
myy601@myy601lab2:~/lkl-source$
```

Μας κατευθύνει στις **msdos_fs.h** και **dir.c**.

- ❖ Περιηγηθήκαμε στην ***dir.c*** και με Ctrl+F, ψάχαμε μεταβλητές **fat dirent** και είδαμε ότι δεν υπάρχουν αλλαγές στις τιμές των μεταβλητών της.

- Συνεχίζουμε με την **fat_boot_sector**. Παρακάτω έχουμε το screenshot από την εκτέλεση της εντολής **grep -r fat_boot_sector**.

```
myy601@myy601lab2:~/lkl-source$ grep -r fat_boot_sector
include/uapi/linux/msdos_fs.h:struct fat_boot_sector {
grep: vmlinux: binary file matches
grep: vmlinux.o: binary file matches
grep: block/partitions/msdos.o: binary file matches
block/partitions/msdos.c:           struct fat_boot_sector *fb;
block/partitions/msdos.c:           fb = (struct fat_boot_sector *) data;
grep: tools/lkl/tests/disk: binary file matches
grep: tools/lkl/tests/boot: binary file matches
grep: tools/lkl/tests/net-test: binary file matches
grep: tools/lkl/tests/disk-vfio-pci: binary file matches
grep: tools/lkl/tests/test-dlmopen: binary file matches
grep: tools/lkl/liblkl.a: binary file matches
grep: tools/lkl/cptofs: binary file matches
grep: tools/lkl/lib/liblkl.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-zpoline.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-hijack.so: binary file matches
grep: tools/lkl/lib/lkl.o: binary file matches
grep: tools/lkl/fs2tar: binary file matches
grep: fs/fat/inode.o: binary file matches
fs/fat/inode.c: * fat_boot_sector.
fs/fat/inode.c: struct fat_boot_sector *b;
fs/fat/inode.c: b = (struct fat_boot_sector *) bh->b_data;
fs/fat/inode.c:static bool fat_bpb_is_zero(struct fat_boot_sector *b)
fs/fat/inode.c:static int fat_read_bpb(struct super_block *sb, struct fat_boot_sector *b,
fs/fat/inode.c: struct fat_boot_sector *b, int silent,
fs/fat/inode.c: error = fat_read_bpb(sb, (struct fat_boot_sector *)bh->b_data, silent,
fs/fat/inode.c:                                     (struct fat_boot_sector *)bh->b_data, silent, &bpb);
grep: lkl.o: binary file matches
```

Μας κατευθύνει στις ***msdos_fs.h***, ***msdos.c*** και ***inode.c***.

- ❖ Περιηγηθήκαμε στην ***msdos.c*** και με Ctrl+F ψάχαμε μεταβλητές **fat boot sector** και είδαμε ότι δεν υπάρχουν αλλαγές στις τιμές των μεταβλητών της.
- ❖ Έπειτα ακολουθήσαμε την ίδια διαδικασία για την ***inode.c*** και είδαμε ότι υπάρχουν αλλαγές στη **static void fat_set_state** στις τιμές των μεταβλητών **fat32.state** και **fat16.state**. Αφού υπάρχουν αλλαγές στις τιμές των μεταβλητών, τότε θα πρέπει να προσθέσουμε κώδικα για να εγγράψουμε τις αλλαγές αυτές στο journal.

```
801 static void fat_set_state(struct super_block *sb,
802                           unsigned int set, unsigned int force)
803 {
804     struct buffer_head *bh;
805     struct fat_boot_sector *b;
806     struct msdos_sb_info *sbi = MSDOS_SB(sb);
807     char *journal_buffer; //dhlwsh buffer tou journal
808     int file_descriptor;
809
810     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
811     if (!journal_buffer) {
812         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
813         kfree(journal_buffer);
814         return;
815     }
816     /* do not change any thing if mounted read only */
817     if (sb_rdonly(sb) && !force) {
818         kfree(journal_buffer);
819         return;
820     }
821     /* do not change state if fs was dirty */
822     if (sbi->dirty) {
823         /* warn only on set (mount).
824          if (set)
825              fat_msg(sb, KERN_WARNING, "Volume was not properly "
826                      "unmounted. Some data may be corrupt. "
827                      "Please run fsck."); */
828         kfree(journal_buffer);
829         return;
830     }
831     bh = sb_bread(sb, 0);
832     if (bh == NULL) {
833         fat_msg(sb, KERN_ERR, "unable to read boot sector "
834                 "to mark fs as dirty");
835         kfree(journal_buffer);
836         return;
837     }
```

```

838     b = (struct fat_boot_sector *) bh->b_data;
839     if (is_fat32(sbi)) {
840         if (set)
841             b->fat32.state |= FAT_STATE_DIRTY;
842         else
843             b->fat32.state &= ~FAT_STATE_DIRTY;
844         //-----ALLAGH-----
845         printk(KERN_INFO "-----\n");
846         printk(KERN_INFO "-----We are in fat_set_state of inode.c.\n");
847         file_descriptor = sbi->myJournal;
848         sprintf(journal_buffer, "\n-----We are in fat_set_state of inode.c.\n fat32.state Changed: %u\n", b->fat32.state);
849         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
850         //-----
851     } else /* fat 16 and 12 */
852     if (set)
853         b->fat16.state |= FAT_STATE_DIRTY;
854     else
855         b->fat16.state &= ~FAT_STATE_DIRTY;
856     //-----ALLAGH-----
857     printk(KERN_INFO "-----\n");
858     printk(KERN_INFO "-----We are in fat_set_state of inode.c.\n");
859     file_descriptor = sbi->myJournal;
860     sprintf(journal_buffer, "\n-----We are in fat_set_state of inode.c.\n fat16.state Changed: %u\n", b->fat16.state);
861     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
862     //-----
863 }
864 mark_buffer_dirty(bh);
865 sync_dirty_buffer(bh);
866 brelse(bh);
867 kfree(journal_buffer);
868 }

```

- Συνεχίζουμε με την **fat_boot_fsinfo**. Παρακάτω έχουμε το screenshot από την εκτέλεση της εντολής **grep -r fat_boot_fsinfo**.

```

myy601@myy601lab2:~/lkl-source$ grep -r fat_boot_fsinfo
include/uapi/linux/msdos_fs.h:struct fat_boot_fsinfo {
grep: vmlinux: binary file matches
grep: vmlinux.o: binary file matches
grep: tools/lkl/tests/disk: binary file matches
grep: tools/lkl/tests/boot: binary file matches
grep: tools/lkl/tests/net-test: binary file matches
grep: tools/lkl/tests/disk-vfio-pci: binary file matches
grep: tools/lkl/tests/test-dlmopen: binary file matches
grep: tools/lkl/liblkl.a: binary file matches
grep: tools/lkl/cptofs: binary file matches
grep: tools/lkl/lib/liblkl.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-zpoline.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-hijack.so: binary file matches
grep: tools/lkl/lib/lkl.o: binary file matches
grep: tools/lkl/fs2tar: binary file matches
grep: fs/fat/misc.o: binary file matches
grep: fs/fat/inode.o: binary file matches
fs/fat/misc.c: struct fat_boot_fsinfo *fsinfo;
fs/fat/misc.c: fsinfo = (struct fat_boot_fsinfo *)bh->b_data;
fs/fat/inode.c:         struct fat_boot_fsinfo *fsinfo;
fs/fat/inode.c:         fsinfo = (struct fat_boot_fsinfo *)fsinfo_bh->b_data;
grep: lkl.o: binary file matches

```

Μας κατευθύνει στις ***msdos_fs.h***, ***misc.c*** και ***inode.c***.

- ❖ Περιηγηθήκαμε στην ***misc.c*** και με Ctrl+F ψάξαμε μεταβλητές **fat_boot_fsinfo** και είδαμε ότι υπάρχουν αλλαγές στη συνάρτηση ***int fat_clusters_flush*** στις μεταβλητές ***free_clusters*** και ***next_cluster***. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές αυτές στο journal.

```

69 int fat_clusters_flush(struct super_block *sb)
70 {
71     struct msdos_sb_info *sbi = MSDOS_SB(sb);
72     struct buffer_head *bh;
73     struct fat_boot_fsinfo *fsinfo;
74     char *journal_buffer; //dhlwsh buffer tou journal
75     int file_descriptor;
76
77     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
78     if (!journal_buffer) {
79         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
80         return -ENOMEM;
81     }

```

```

92         fsinfo = (struct fat_boot_fsinfo *)bh->b_data;
93         /* Sanity check */
94         if (!IS_FSINFO(fsinfo)) {
95             fat_msg(sb, KERN_ERR, "Invalid FSINFO signature: "
96                     "0x%08x, 0x%08x (sector = %lu)",
97                     le32_to_cpu(fsinfo->signature1),
98                     le32_to_cpu(fsinfo->signature2),
99                     sbi->fsinfo_sector);
100     } else {
101         if (sbi->free_clusters != -1)
102             fsinfo->free_clusters = cpu_to_le32(sbi->free_clusters);
103         ////////////////////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////////////////////
104         printk(KERN_INFO "-----\n");
105         printk(KERN_INFO "-----We are in fat_clusters_flush of misc.c.-----\n");
106         file_descriptor = sbi->myJournal;
107         sprintf(journal_buffer, "\n-----We are in fat_clusters_flush of misc.c.-----\n fsinfo->free_clusters Changed: %u\n",
108                 fsinfo->free_clusters);
109         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
110         ////////////////////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////////////////////
111         if (sbi->prev_free != -1)
112             fsinfo->next_cluster = cpu_to_le32(sbi->prev_free);
113         ////////////////////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////////////////////
114         printk(KERN_INFO "-----\n");
115         printk(KERN_INFO "-----We are in fat_clusters_flush of misc.c.-----\n");
116         file_descriptor = sbi->myJournal;
117         sprintf(journal_buffer, " fsinfo->next_cluster Changed: %u\n", fsinfo->next_cluster);
118         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
119         ////////////////////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////////////////////
120         mark_buffer_dirty(bh);
121     }
122     brelse(bh);
123     kfree(journal_buffer);
124     return 0;
125 }

```

- ❖ Έπειτα ακολουθήσαμε την ίδια διαδικασία για την **inode.c** και είδαμε ότι δεν υπάρχουν αλλαγές σε μεταβλητές.

- Συνεχίζουμε με την **msdos_dir_entry**. Παρακάτω έχουμε το screenshot από την εκτέλεση της εντολής **grep -r msdos_dir_entry**.

```

myy601@myy601lab2:~/lkl-sources$ grep -r msdos_dir_entry
include/uapi/linux/msdos_fs.h:#define MSDOS_DPS (SECTOR_SIZE / sizeof(struct msdos_dir_entry))
include/uapi/linux/msdos_fs.h:#define MSDOS_DIR_BITS 5           /* log2(sizeof(struct msdos_dir_entry)) */
include/uapi/linux/msdos_fs.h:struct msdos_dir_entry {
grep: vmlinux: binary file matches
grep: vmlinu.o: binary file matches
grep: tools/lkl/tests/disk: binary file matches
grep: tools/lkl/tests/boot: binary file matches
grep: tools/lkl/tests/net-test: binary file matches
grep: tools/lkl/tests/disk-vfio-pci: binary file matches
grep: tools/lkl/tests/test-dlmopen: binary file matches
grep: tools/lkl/liblkl.a: binary file matches
grep: tools/lkl/cptofs: binary file matches
grep: tools/lkl/lib/liblkl.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-zpoline.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-hijack.so: binary file matches
grep: tools/lkl/lib/lkl.o: binary file matches
grep: tools/lkl/fs2tar: binary file matches
grep: fs/fat/nfs.o: binary file matches
fs/fat/dir.c:                                     struct msdos_dir_entry *de)
fs/fat/dir.c: | (de - (struct msdos_dir_entry *)bh->b_data);
fs/fat/dir.c: |           struct buffer_head **bh, struct msdos_dir_entry **de)
fs/fat/dir.c: *pos += sizeof(struct msdos_dir_entry);
fs/fat/dir.c: *de = (struct msdos_dir_entry *)((*bh)->b_data + offset);
fs/fat/dir.c: |           struct msdos_dir_entry **de)
fs/fat/dir.c: (*de - (struct msdos_dir_entry *)(*bh)->b_data) <
fs/fat/dir.c: |           *pos += sizeof(struct msdos_dir_entry);
fs/fat/dir.c: |           struct buffer_head **bh, struct msdos_dir_entry **de,
fs/fat/dir.c: |           const struct msdos_dir_entry *de,
fs/fat/dir.c: struct msdos_dir_entry *de;
fs/fat/dir.c: struct msdos_dir_entry *de;
fs/fat/dir.c: if (cpos & (sizeof(struct msdos_dir_entry) - 1)) {
fs/fat/dir.c: ctx->pos = cpos - (nr_slots + 1) * sizeof(struct msdos_dir_entry);
fs/fat/dir.c: |           struct msdos_dir_entry **de)
fs/fat/dir.c: |           struct msdos_dir_entry **de)
fs/fat/dir.c: struct msdos_dir_entry *de;
fs/fat/dir.c: struct msdos_dir_entry *de;
fs/fat/dir.c: struct msdos_dir_entry *de, *endp;
fs/fat/dir.c: |           endp = (struct msdos_dir_entry *) (bh->b_data + sb->s_blocksize);
fs/fat/dir.c: struct msdos_dir_entry *de;
fs/fat/dir.c: while (nr_slots && de >= (struct msdos_dir_entry *)bh->b_data) {
fs/fat/dir.c: |           struct msdos_dir_entry *de;
fs/fat/dir.c: |           de = (struct msdos_dir_entry *)bhs[0]->b_data;
fs/fat/dir.c: |           int *nr_cluster, struct msdos_dir_entry **de,
fs/fat/dir.c: |           size = nr_slots * sizeof(struct msdos_dir_entry);
fs/fat/dir.c: |           offset = copy - sizeof(struct msdos_dir_entry);

```

```

fs/fat/dir.c:     *de = (struct msdos_dir_entry *)((*bh)->b_data + offset);
fs/fat/dir.c:     struct msdos_dir_entry *de;
grep: fs/fat/inode.o: binary file matches
fs/fat/fat.h:     struct msdos_dir_entry *de;
fs/fat/fat.h:         const struct msdos_dir_entry *de)
fs/fat/fat.h: static inline void fat_set_start(struct msdos_dir_entry *de, int cluster)
fs/fat/fat.h:         struct msdos_dir_entry **de);
fs/fat/fat.h:             struct msdos_dir_entry *de, loff_t i_pos);
fs/fat/fat.h: extern int fat_fill_inode(struct inode *inode, struct msdos_dir_entry *de);
fs/fat/namei_msdos.c: struct msdos_dir_entry de;
fs/fat/namei_msdos.c: struct msdos_dir_entry *dotdot_de;
grep: fs/fat/dir.o: binary file matches
grep: fs/fat/namei_vfat.o: binary file matches
fs/fat/nfs.c:     struct msdos_dir_entry *de ;
fs/fat/nfs.c:     de = (struct msdos_dir_entry *)bh->b_data;
fs/fat/nfs.c:     struct msdos_dir_entry *de;
fs/fat/nfs.c:     de = (struct msdos_dir_entry *) parent_bh->b_data;
fs/fat/nfs.c:     struct msdos_dir_entry *de;
fs/fat/inode.c:int fat_fill_inode(struct inode *inode, struct msdos_dir_entry *de)
fs/fat/inode.c:         struct msdos_dir_entry *de, loff_t i_pos)
fs/fat/inode.c: struct msdos_dir_entry *raw_entry;
fs/fat/inode.c: raw_entry = &((struct msdos_dir_entry *) (bh->b_data))[offset];
fs/fat/inode.c:     inode->i_size = sbi->dir_entries * sizeof(struct msdos_dir_entry);
fs/fat/inode.c: sbi->dir_per_block = sb->s_blocksize / sizeof(struct msdos_dir_entry);
fs/fat/inode.c:     * sizeof(struct msdos_dir_entry) / sb->s_blocksize;
fs/fat/namei_vfat.c: struct msdos_dir_entry *de;
fs/fat/namei_vfat.c:     de = (struct msdos_dir_entry *)slots;
fs/fat/namei_vfat.c:     de = (struct msdos_dir_entry *)ps;
fs/fat/namei_vfat.c:         struct msdos_dir_entry **de)
fs/fat/namei_vfat.c:             struct msdos_dir_entry *dotdot_de)
fs/fat/namei_vfat.c:     struct msdos_dir_entry *dotdot_de = NULL;
fs/fat/namei_vfat.c:     struct msdos_dir_entry *old_dotdot_de = NULL, *new_dotdot_de = NULL;
grep: lkl.o: binary file matches

```

Μας κατευθύνει στις ***msdos_fs.h***, ***dir.c***, ***fat.h***, ***namei_msdos.c***, ***nfc.c***, ***inode.c*** και ***namei_vfat.c***.

- ❖ Αρχικά ψάχνω με Ctrl+F στην ***dir.c*** μεταβλητές **msdos_dir_entry** και βλέπω ότι υπάρχουν αλλαγές στην **static int __fat_remove_entries** στη μεταβλητή ***name***. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

1086 static int __fat_remove_entries(struct inode *dir, loff_t pos, int nr_slots)
1087 {
1088     struct super_block *sb = dir->i_sb;
1089     struct buffer_head *bh;
1090     struct msdos_dir_entry *de, *endp;
1091     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthettw gia na mporw na kalesw to sbi->myJournal
1092     char *journal_buffer; //dhlwsh buffer tou journal
1093     int err = 0, orig_slots;
1094     int file_descriptor;
1095
1096     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
1097     if (!journal_buffer) {
1098         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
1099         return -ENOMEM;
1100     }
1101     while (nr_slots) {
1102         bh = NULL;
1103         if (fat_get_entry(dir, &pos, &bh, &de) < 0) {
1104             err = -EIO;
1105             break;
1106         }
1107
1108         orig_slots = nr_slots;
1109         endp = (struct msdos_dir_entry *) (bh->b_data + sb->s_blocksize);
1110         while (nr_slots && de < endp) {
1111             de->name[0] = DELETED_FLAG;
1112             de++;
1113             nr_slots--;
1114         }
1115         //~~~~~ALLAGH~~~~~
1116         printk(KERN_INFO "-----\n");
1117         printk(KERN_INFO "-----We are in __fat_remove_entries of dir.c.-----\n");
1118         file_descriptor = sbi->myJournal;
1119         sprintf(journal_buffer, "-----We are in __fat_remove_entries of dir.c.-----\n de->name[0] Changed: %u\n", de->name[0]);
1120         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1121
1122         mark_buffer_dirty_inode(bh, dir);
1123         if (IS_DIRSYNC(dir))
1124             err = sync_dirty_buffer(bh);
1125         brelse(bh);
1126         if (err)
1127             break;
1128
1129         /* pos is *next* de's position, so this does `~ sizeof(de)` */
1130         pos += ((orig_slots - nr_slots) * sizeof(*de)) - sizeof(*de);
1131     }
1132     kfree(journal_buffer);
1133     return err;
1134 }

```

- ❖ Έπειτα ψάχνω με Ctrl+F στην **namei_msdos.c** μεταβλητές **msdos_dir_entry** και βλέπω ότι υπάρχουν αλλαγές στην **static int msdos_add_entry** στις μεταβλητές **name, attr, lcase, cdate, adate, ctime, ctime_cs, time, date, size**. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

228 static int msdos_add_entry(struct inode *dir, const unsigned char *name,
229                         int is_dir, int is_hid, int cluster,
230                         struct timespec64 *ts, struct fat_slot_info *sinfo)
231 {
232     struct msdos_sb_info *sbi = MSDOS_SB(dir->i_sb);
233     struct msdos_dir_entry de;
234     __le16 time, date;
235     int err;
236     char *journal_buffer; //dhlwsh buffer tou journal
237     int file_descriptor;
238
239     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
240     if (!journal_buffer) {
241         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
242         return;
243     }
244     memcpy(de.name, name, MSDOS_NAME);
245     //-----ALLAGH-----
246     printk(KERN_INFO "-----\n");
247     printk(KERN_INFO "-----We are in msdos_add_entry of namei_msdos.c-----\n");
248     file_descriptor = sbi->myJournal;
249     sprintf(journal_buffer, "\n-----We are in msdos_add_entry of namei_msdos.c.\n de.name Changed: %u\n", de.name);
250     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
251     //-----
252
253     de.attr = is_dir ? ATTR_DIR : ATTR_ARCHIVE;
254     if (is_hid)
255         de.attr |= ATTR_HIDDEN;
256     //-----ALLAGH-----
257     printk(KERN_INFO "-----\n");
258     printk(KERN_INFO "-----We are in msdos_add_entry of namei_msdos.c-----\n");
259     file_descriptor = sbi->myJournal;
260     sprintf(journal_buffer, " de.attr Changed: %u\n", de.attr);
261     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
262     //-----
263
264     de.lcase = 0;
265     //-----ALLAGH-----
266     printk(KERN_INFO "-----\n");
267     printk(KERN_INFO "-----We are in msdos_add_entry of namei_msdos.c-----\n");
268     file_descriptor = sbi->myJournal;
269     sprintf(journal_buffer, " de.lcase Changed: %u\n", de.lcase);
270     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
271     //-----
272
273     fat_time_unix2fat(sbi, ts, &time, &date, NULL);
274     de.cdate = de.adate = 0;
275     //-----ALLAGH-----
276     printk(KERN_INFO "-----\n");
277     printk(KERN_INFO "-----We are in msdos_add_entry of namei_msdos.c-----\n");
278     file_descriptor = sbi->myJournal;
279     sprintf(journal_buffer, " de.cdate Changed: %u\n de.adate Changed: %u\n", de.cdate, de.adate);
280     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
281     //-----
282
283     de.ctime = 0;
284     de.ctime_ns = 0;
285     //-----ALLAGH-----
286     printk(KERN_INFO "-----\n");
287     printk(KERN_INFO "-----We are in msdos_add_entry of namei_msdos.c-----\n");
288     file_descriptor = sbi->myJournal;
289     sprintf(journal_buffer, " de.ctime Changed: %u\n de.ctime_ns Changed: %u\n", de.ctime, de.ctime_ns);
290     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
291     //-----
292
293     de.time = time;
294     de.date = date;
295     //-----ALLAGH-----
296     printk(KERN_INFO "-----\n");
297     printk(KERN_INFO "-----We are in msdos_add_entry of namei_msdos.c-----\n");
298     file_descriptor = sbi->myJournal;
299     sprintf(journal_buffer, " de.time Changed: %u\n de.date Changed: %u\n", de.time, de.date);
300     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
301     //-----
302
303     fat_set_start(&de, cluster);
304     de.size = 0;
305     //-----ALLAGH-----
306     printk(KERN_INFO "-----\n");
307     printk(KERN_INFO "-----We are in msdos_add_entry of namei_msdos.c-----\n");
308     file_descriptor = sbi->myJournal;
309     sprintf(journal_buffer, " de.size Changed: %u\n", de.size);
310     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
311     //-----

```

- ❖ Στη συνέχεια ψάχνω με Ctrl+F στην **nfc.c** μεταβλητές **msdos_dir_entry** και βλέπω ότι δεν υπάρχουν αλλαγές.

- ❖ Ακολουθώ την ίδια διαδικασία και στην **inode.c** και βλέπω ότι υπάρχουν αλλαγές στη **static int __fat_write_inode** στις μεταβλητές **size** και **attr**. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

1050 static int __fat_write_inode(struct inode *inode, int wait)
1051 {
1052     struct super_block *sb = inode->i_sb;
1053     struct msdos_sb_info *sbi = MSDOS_SB(sb);
1054     struct buffer_head *bh;
1055     struct msdos_dir_entry *raw_entry;
1056     loff_t i_pos;
1057     sector_t blocknr;
1058     int err, offset;
1059     char *journal_buffer; //dhlwsh buffer tou journal
1060     int file_descriptor;
1061
1062     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
1063     if (!journal_buffer) {
1064         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
1065         return -ENOMEM;
1066     }
1067
1092     raw_entry = &((struct msdos_dir_entry *) (bh->b_data))[offset];
1093     if (S_ISDIR(inode->i_mode))
1094         raw_entry->size = 0;
1095     else
1096         raw_entry->size = cpu_to_le32(inode->i_size);
1097     //-----ALLAGH-----
1098     printk(KERN_INFO "-----\n");
1099     printk(KERN_INFO "-----We are in __fat_write_inode of inode.c.-----\n");
1100     file_descriptor = sbi->myJournal;
1101     sprintf(journal_buffer, "\n-----We are in __fat_write_inode of inode.c.-----\n raw_entry->size Changed: %u\n", raw_entry->size);
1102     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1103     //-----ALLAGH-----
1104
1105     raw_entry->attr = fat_make_attrs(inode);
1106     //-----ALLAGH-----
1107     printk(KERN_INFO "-----\n");
1108     printk(KERN_INFO "-----We are in __fat_write_inode of inode.c.-----\n");
1109     file_descriptor = sbi->myJournal;
1110     sprintf(journal_buffer, " raw_entry->attr Changed: %u\n", raw_entry->attr);
1111     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1112     //-----ALLAGH-----
1113
1114     fat_set_start(raw_entry, MSDOS_I(inode)->i_logstart);
1115     fat_time_unix2fat(sbi, &inode->i_mtime, &raw_entry->time,
1116                         &raw_entry->date, NULL);
1117     if (sbi->options.isvfat) {
1118         _le16 atime;
1119         fat_time_unix2fat(sbi, &inode->i_atime, &atime,
1120                           &raw_entry->adate, NULL);
1121         fat_time_unix2fat(sbi, &MSDOS_I(inode)->i_crttime, &raw_entry->ctime,
1122                           &raw_entry->cdate, &raw_entry->ctime_cs);
1123     }
1124     spin_unlock(&sbi->inode_hash_lock);
1125     mark_buffer_dirty(bh);
1126     err = 0;
1127     if (wait)
1128         err = sync_dirty_buffer(bh);
1129     brelse(bh);
1130     kfree(journal_buffer);
1131     return err;
1132 }

```

- ❖ Τέλος ψάχνω με Ctrl+F στην **namei_vfat.c** μεταβλητές **msdos_dir_entry** και βλέπω ότι υπάρχουν αλλαγές στην **static int vfat_build_slots** στις μεταβλητές **name, attr, lcase, time, ctime, date, adate, ctime_cs, size**. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

580 static int vfat_build_slots(struct inode *dir, const unsigned char *name,
581                             int len, int is_dir, int cluster,
582                             struct timespec64 *ts,
583                             struct msdos_dir_slot *slots, int *nr_slots)
584 {
585     struct msdos_sb_info *sbi = MSDOS_SB(dir->i_sb);
586     struct fat_mount_options *opts = &sbi->options;
587     struct msdos_dir_slot *ps;
588     struct msdos_dir_entry *de;
589     unsigned char cksum, lcase;
590     unsigned char msdos_name[MSDOS_NAME];
591     char *journal_buffer; //dhlwsh buffer tou journal
592     int file_descriptor;
593     wchar_t *uname;
594     _le16 time, date;
595     u8 time_cs;
596     int err, ulen, usize, i;
597     loff_t offset;
598     *nr_slots = 0;
599
600     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
601     if (!journal_buffer)
602         return -ENOMEM;

```

```

671    shortname:  

672        /* build the entry of 8.3 alias name */  

673        (*nr_slots)++;  

674        memcpy(de->name, msdos_name, MSDOS_NAME);  

675        ////////////////////////////////////////////////////ALLAGH  

676        printk(KERN_INFO "-----\n");  

677        printk(KERN_INFO "-----We are in vfat_build_slots of namei_vfat.c.\n");  

678        file_descriptor = sbi->myJournal;  

679        sprintf(journal_buffer, "\n-----We are in vfat_build_slots of namei_vfat.c.\n-----\n de->name Changed: %s\n", de->name);  

680        write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);  

681        ////////////////////////////////////////////////////  

682  

683        de->attr = is_dir ? ATTR_DIR : ATTR_ARCH;  

684        de->lcase = lcase;  

685        ////////////////////////////////////////////////////ALLAGH  

686        printk(KERN_INFO "-----\n");  

687        printk(KERN_INFO "-----We are in vfat_build_slots of namei_vfat.c.\n");  

688        file_descriptor = sbi->myJournal;  

689        sprintf(journal_buffer, " de->attr Changed: %u\n de->lcase Changed: %u\n", de->attr, de->lcase);  

690        write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);  

691        ////////////////////////////////////////////////////  

692  

693        fat_time_unix2fat(sbi, ts, &time, &date, &time_cs);  

694        de->time = de->ctime = time;  

695        ////////////////////////////////////////////////////ALLAGH  

696        printk(KERN_INFO "-----\n");  

697        printk(KERN_INFO "-----We are in vfat_build_slots of namei_vfat.c.\n");  

698        file_descriptor = sbi->myJournal;  

699        sprintf(journal_buffer, " de->time Changed: %u\n de->ctime Changed: %u\n", de->time, de->ctime);  

700        write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);  

701        ////////////////////////////////////////////////////  

702        de->date = de->cdate = de->adate = date;  

703        ////////////////////////////////////////////////////ALLAGH  

704        printk(KERN_INFO "-----\n");  

705        printk(KERN_INFO "-----We are in vfat_build_slots of namei_vfat.c.\n");  

706        file_descriptor = sbi->myJournal;  

707        sprintf(journal_buffer, " de->date Changed: %u\n de->cdate Changed: %u\n de->adate Changed: %u\n",  

708        de->date, de->cdate, de->adate);  

709        write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);  

710        ////////////////////////////////////////////////////  

711  

712        de->ctime_cs = time_cs;  

713        ////////////////////////////////////////////////////ALLAGH  

714        printk(KERN_INFO "-----\n");  

715        printk(KERN_INFO "-----We are in vfat_build_slots of namei_vfat.c.\n");  

716        file_descriptor = sbi->myJournal;  

717        sprintf(journal_buffer, " de->ctime_cs Changed: %u\n", de->ctime_cs);  

718        write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);  

719        ////////////////////////////////////////////////////  

720  

721        fat_set_start(de, cluster);  

722        de->size = 0;  

723        ////////////////////////////////////////////////////ALLAGH  

724        printk(KERN_INFO "-----\n");  

725        printk(KERN_INFO "-----We are in vfat_build_slots of namei_vfat.c.\n");  

726        file_descriptor = sbi->myJournal;  

727        sprintf(journal_buffer, " de->size Changed: %u\n", de->size);  

728        write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);  

729        ////////////////////////////////////////////////////  

730  

731        out_free:  

732            _putname(uname);  

733            kfree(journal_buffer);  

734            return err;  

735    }

```

- Καταλήγουμε με την **msdos_dir_slot**. Παρακάτω έχουμε το screenshot από την εκτέλεση της εντολής **grep -r msdos_dir_slot**.

```

myy601@myy601lab2:~/lkl-source$ grep -r msdos_dir_slot
include/uapi/linux/msdos_fs.h:struct msdos_dir_slot {
grep: vmlinuz: binary file matches
grep: vmlinuz.o: binary file matches
grep: tools/lkl/tests/disk: binary file matches
grep: tools/lkl/tests/boot: binary file matches
grep: tools/lkl/tests/net-test: binary file matches
grep: tools/lkl/tests/disk-vfio-pci: binary file matches
grep: tools/lkl/tests/test-dlmopen: binary file matches
grep: tools/lkl/liblkl.a: binary file matches
grep: tools/lkl/cptofs: binary file matches
grep: tools/lkl/lib/liblkl.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-zpoline.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-hijack.so: binary file matches
grep: tools/lkl/lib/lkl.o: binary file matches
grep: tools/lkl/fs2tar: binary file matches
fs/fat/dir.c:     struct msdos_dir_slot *ds;
fs/fat/dir.c:     ds = (struct msdos_dir_slot *)*de;
fs/fat/dir.c:         ds = (struct msdos_dir_slot *)*de;

```

```

grep: fs/fat/dir.o: binary file matches
grep: fs/fat/namei_vfat.o: binary file matches
fs/fat/namei_vfat.c:                                     struct msdos_dir_slot *slots, int *nr_slots)
fs/fat/namei_vfat.c:     struct msdos_dir_slot *ps;
fs/fat/namei_vfat.c:     struct msdos_dir_slot *slots;
grep: lkl.o: binary file matches

```

Μας κατευθύνει στις ***msdos_fs.h***, ***dir.c*** και ***namei_vfat.c***.

- ❖ Αρχικά ψάχνω με Ctrl+F στην ***dir.c*** μεταβλητές ***msdos_dir_slot*** και βλέπω ότι δεν υπάρχουν αλλαγές.
- ❖ Στη συνέχεια ακολουθώ την ίδια διαδικασία για τη ***namei_vfat.c*** και βλέπω ότι υπάρχουν αλλαγές στην ***static int vfat_build_slots*** στις μεταβλητές ***id***, ***attr***, ***reserved***, ***alias_checksum***, ***start***. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

580 static int vfat_build_slots(struct inode *dir, const unsigned char *name,
581                           int len, int is_dir, int cluster,
582                           struct timespec64 *ts,
583                           struct msdos_dir_slot *slots, int *nr_slots)
584 {
585     struct msdos_sb_info *sbi = MSDOS_SB(dir->i_sb);
586     struct fat_mount_options *opts = &sbi->options;
587     struct msdos_dir_slot *ps;
588     struct msdos_dir_entry *de;
589     unsigned char cksm, lcase;
590     unsigned char msdos_name[MSDOS_NAME];
591     char *journal_buffer; //dhlwsh buffer tou journal
592     int file_descriptor;
593     wchar_t *uname;
594     _le16 time, date;
595     u8 time_cs;
596     int err, ulen, usize, i;
597     loff_t offset;
598     *nr_slots = 0;
599
600     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
601     if (!journal_buffer)
602         return -ENOMEM;
603
604     *nr_slots = usize / 13;
605     for (ps = slots, i = *nr_slots; i > 0; i--, ps++) {
606         ps->id = i;
607         ps->attr = ATTR_EXT;
608         //-----ALLAGH-----
609         printk(KERN_INFO "-----\n");
610         printk(KERN_INFO "-----We are in vfat_build_slots of namei_vfat.c.\n");
611         file_descriptor = sbi->myJournal;
612         sprintf(journal_buffer, "\n-----We are in vfat_build_slots of namei_vfat.c.\n\n ps->id Changed: %u\n ps->attr Changed: %u\n",
613                 ps->id, ps->attr);
614         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
615         //-----ALLAGH-----
616         ps->reserved = 0;
617         ps->alias_checksum = cksm;
618         //-----ALLAGH-----
619         printk(KERN_INFO "-----\n");
620         printk(KERN_INFO "-----We are in vfat_build_slots of namei_vfat.c.\n");
621         file_descriptor = sbi->myJournal;
622         sprintf(journal_buffer, " ps->reserved Changed: %u\n ps->alias_checksum Changed: %u\n",
623                 ps->reserved, ps->alias_checksum);
624         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
625         //-----ALLAGH-----
626
627         ps->start = 0;
628         //-----ALLAGH-----
629         printk(KERN_INFO "-----\n");
630         printk(KERN_INFO "-----We are in vfat_build_slots of namei_vfat.c.\n");
631         file_descriptor = sbi->myJournal;
632         sprintf(journal_buffer, " ps->start Changed: %u\n", ps->start);
633         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
634         //-----ALLAGH-----
635         offset = (i - 1) * 13;
636         fatwchar_tol6(ps->name0_4, uname + offset, 5);
637         fatwchar_tol6(ps->name5_10, uname + offset + 5, 6);
638         fatwchar_tol6(ps->name11_12, uname + offset + 11, 2);
639     }
640     slots[0].id |= 0x40;
641     de = (struct msdos_dir_entry *)ps;

```

□ Βασικές Δομές FAT

Αφού ολοκληρώσαμε με το αρχείο **msdos_fs.h**, προσπαθήσαμε να κατανοήσουμε τις επιπλέον παρεμβάσεις που απαιτούνται για την καταγραφή των αλλαγών στο journal. Διαπιστώσαμε, λοιπόν, ότι κάποια συγκεκριμένα αρχεία συνδέονται με κάθε δομή της FAT (File Allocation Table, Superblock, Inode, Dentries, Files) και ότι οι δομές για τις πρώτες τέσσερις εντοπίζονται στο αρχείο **fs/fat/fat.h**. Θα εξετάσουμε κάθε δομή FAT ξεχωριστά.

□ FILE ALLOCATION TABLE (FAT)

Παρατηρούμε ότι η δομή File allocation table ορίζεται στο **fs/fat/fat.h** στη δομή **struct fat_entry**.

```
359 struct fat_entry {
360     int entry;
361     union {
362         u8 *entl2_p[2];
363         __le16 *ent16_p;
364         __le32 *ent32_p;
365     } u;
366     int nr_bhs;
367     struct buffer_head *bhs[2];
368     struct inode *fat_inode;
369 };
```

Επομένως, όπως και παραπάνω, θα χρησιμοποιήσουμε την εντολή **grep -r** για να κάνουμε αναζήτηση. Παρακάτω έχουμε το screenshot από την εκτέλεση της εντολής **grep -r fat_entry**.

```
myy601@myy601lab2:~/lkl-source$ grep -r fat_entry
grep: vmlinux: binary file matches
grep: vmlinux.o: binary file matches
grep: tools/lkl/tests/disk: binary file matches
grep: tools/lkl/tests/boot: binary file matches
grep: tools/lkl/tests/net-test: binary file matches
grep: tools/lkl/tests/disk-vfio-pci: binary file matches
grep: tools/lkl/tests/test-dlmopen: binary file matches
grep: tools/lkl/liblkl.a: binary file matches
grep: tools/lkl/cptofs: binary file matches
grep: tools/lkl/lib/liblkl.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-zpoline.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-hijack.so: binary file matches
grep: tools/lkl/lib/lkl.o: binary file matches
grep: tools/lkl/fs2tar: binary file matches
grep: fs/fat/fatent.o: binary file matches
grep: fs/fat/misc.o: binary file matches
fs/fat/fatent.c:        void (*ent_set_ptr)(struct fat_entry *, int);
fs/fat/fatent.c:        int (*ent_bread)(struct super_block *, struct fat_entry *,
fs/fat/fatent.c:        int (*ent_get)(struct fat_entry *);
fs/fat/fatent.c:        void (*ent_put)(struct fat_entry *, int);
fs/fat/fatent.c:        int (*ent_next)(struct fat_entry *);
fs/fat/fatent.c:static void fat12_ent_set_ptr(struct fat_entry *fatent, int offset)
fs/fat/fatent.c:static void fat16_ent_set_ptr(struct fat_entry *fatent, int offset)
fs/fat/fatent.c:static void fat32_ent_set_ptr(struct fat_entry *fatent, int offset)
fs/fat/fatent.c:static int fat12_ent_bread(struct super_block *sb, struct fat_entry *fatent,
fs/fat/fatent.c:static int fat_ent_bread(struct super_block *sb, struct fat_entry *fatent,
fs/fat/fatent.c:static int fat12_ent_get(struct fat_entry *fatent)
fs/fat/fatent.c:static int fat16_ent_get(struct fat_entry *fatent)
fs/fat/fatent.c:static int fat32_ent_get(struct fat_entry *fatent)
fs/fat/fatent.c:static void fat12_ent_put(struct fat_entry *fatent, int new)
fs/fat/fatent.c:static void fat16_ent_put(struct fat_entry *fatent, int new)
fs/fat/fatent.c:static void fat32_ent_put(struct fat_entry *fatent, int new)
fs/fat/fatent.c:static int fat12_ent_next(struct fat_entry *fatent)
fs/fat/fatent.c:static int fat16_ent_next(struct fat_entry *fatent)
fs/fat/fatent.c:static int fat32_ent_next(struct fat_entry *fatent)
                                         struct fat_entry *fatent,
```

```

fs/fat/fatent.c:int fat_ent_read(struct inode *inode, struct fat_entry *fatent, int entry)
fs/fat/fatent.c:int fat_ent_write(struct inode *inode, struct fat_entry *fatent,
fs/fat/fatent.c:                                struct fat_entry *fatent)
grep: fs/fat/cache.o: binary file matches
fs/fat/fat.h:struct fat_entry {
fs/fat/fat.h:static inline void fatent_init(struct fat_entry *fatent)
fs/fat/fat.h:static inline void fatent_set_entry(struct fat_entry *fatent, int entry)
fs/fat/fat.h:extern int fat_ent_read(struct inode *inode, struct fat_entry *fatent,
fs/fat/fat.h:extern int fat_ent_write(struct inode *inode, struct fat_entry *fatent,
fs/fat/misc.c:                                struct fat_entry *fatent);
grep: fs/fat/file.o: binary file matches
fs/fat/file.c:                                struct fat_entry *fatent;
fs/fat/cache.c: struct fat_entry *fatent;
fs/exfat/fatent.c:                                __le32 *fat_entry;
fs/exfat/fatent.c:                                fat_entry = (__le32 *)(&(bh->_b_data[off]));
fs/exfat/fatent.c:                                *fat_entry = cpu_to_le32(content);
fs/exfat/exfat_fs.h:struct exfat_entry_set_cache {
fs/exfat/exfat_fs.h:void exfat_update_dir_chksm_with_entry_set(struct exfat_entry_set_cache *es)
fs/exfat/exfat_fs.h:struct exfat_dentry *exfat_get_dentry_cached(struct exfat_entry_set_cache *es,
fs/exfat/exfat_fs.h:struct exfat_entry_set_cache *exfat_get_dentry_set(struct super_block *sb,
fs/exfat/exfat_fs.h:int exfat_free_dentry_set(struct exfat_entry_set_cache *es, int sync);
fs/exfat/dir.c: struct exfat_entry_set_cache *es;
fs/exfat/dir.c:void exfat_update_dir_chksm_with_entry_set(struct exfat_entry_set_cache *es)
fs/exfat/dir.c:int exfat_free_dentry_set(struct exfat_entry_set_cache *es, int sync)
fs/exfat/dir.c: struct exfat_entry_set_cache *es, int num)
fs/exfat/dir.c:struct exfat_entry_set_cache *exfat_get_dentry_set(struct super_block *sb,
fs/exfat/dir.c: struct exfat_entry_set_cache *es;
fs/exfat/inode.c:                                struct exfat_entry_set_cache *es = NULL;
fs/exfat/namei.c:                                struct exfat_entry_set_cache *es;
grep: lkl.o: binary file matches

```

Μας κατευθύνει στις ***fatent.c***, ***fat.h***, ***misc.c***, ***file.c***, ***cache.c*** και σε κάποιες συναρτήσεις ακόμα που δε βρίσκονται στον fs/fat, οπότε δε θα τις ερευνήσουμε.

❖ Αρχικά ψάχνουμε με Ctrl+F στην ***fatent.c*** μεταβλητές ***fat_entry*** και παρατηρούμε ότι υπάρχουν αλλαγές σε πολλές συναρτήσεις:

→ Στη ***fat12_ent_bread*** αλλάζουν οι μεταβλητές ***fat_inode*** και ***nr_bhs***. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

78 static int fat12_ent_bread(struct super_block *sb, struct fat_entry *fatent,
79                           int offset, sector_t blocknr)
80 {
81     struct buffer_head **bhs = fatent->bhs;
82     char *journal_buffer; //dhlwsh buffer tou journal
83     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthettw gia na mporw na kalesw to sbi->myJournal
84     int file_descriptor;
85
86     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
87     if (!journal_buffer) {
88         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
89         return -ENOMEM;
90     }
91     printk(KERN_INFO "* ~ ENTERING fat12_ent_bread (%s.c/struct fatent_operations fat12_ops) ~ *\n");
92     WARN_ON(blocknr < MSDOS_SB(sb)->fat_start);
93     fatent->fat_inode = MSDOS_SB(sb)->fat_inode;
94
95     //~~~~~ALLAGH~~~~~
96     printk(KERN_INFO "-----\n");
97     printk(KERN_INFO "-----We are in fat12_ent_bread of %s.c.-----\n", __FILE__);
98     file_descriptor = sbi->myJournal;
99     sprintf(journal_buffer, "\n-----We are in fat12_ent_bread of %s.c.-----\n %s->fat_inode Changed: %px\n",
100            fatent->fat_inode);
101    write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
102
103    bhs[0] = sb_bread(sb, blocknr);
104    if (!bhs[0])
105        goto err;
106
107    if ((offset + 1) < sb->s_blocksize)
108        fatent->nr_bhs = 1;
109    else {
110        /* This entry is block boundary, it needs the next block */
111        blocknr++;
112        bhs[1] = sb_bread(sb, blocknr);
113        if (!bhs[1])
114            goto err_brelse;
115        fatent->nr_bhs = 2;
116    }

```

```

117 //~~~~~ALLAGH~~~~~
118 printk(KERN_INFO "-----\n");
119 printk(KERN_INFO "-----We are in fat12_ent_bread of fatent.c.\n");
120 file_descriptor = sbi->myJournal;
121 sprintf(journal_buffer, "%u\n", fatent->nr_bhs);
122 write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
123 //~~~~~ALLAGH~~~~~
124 fat12_ent_set_ptr(fatent, offset);
125 kfree(journal_buffer);
126 return 0;
127
128 err_brelse:
129 brelse(bhs[0]);
130 err:
131     fat_msg_ratelimit(sb, KERN_ERR, "FAT read failed (blocknr %llu)",
132     (llu)blocknr);
133     kfree(journal_buffer);
134 }
135

```

→ Στη `fat_ent_bread` αλλάζουν οι μεταβλητές `fat_inode`, `bhs[0]` και `nr_bhs`.
Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

137 static int fat_ent_bread(struct super_block *sb, struct fat_entry *fatent,
138                         int offset, sector_t blocknr)
139 {
140     const struct fatent_operations *ops = MSDOS_SB(sb)->fatent_ops;
141     char *journal_buffer; //dhlwsh buffer tou journal
142     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthew gia na mporw na kalesw to sbi->myJournal
143     int file_descriptor;
144     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
145     if (!journal_buffer)
146         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
147     return -ENOMEM;
148 }
149 printk(KERN_INFO "* ~ ENTERING fat_ent_bread (%s.c/struct fatent_operations fat16_ops AND fat32_ops) ~ *");
150 WARN_ON(blocknr < MSDOS_SB(sb)->fat_start);
151 fatent->fat_inode = MSDOS_SB(sb)->fat_inode;
152 fatent->bhs[0] = sb_bread(sb, blocknr);
153 //~~~~~ALLAGH~~~~~
154 printk(KERN_INFO "-----\n");
155 printk(KERN_INFO "-----We are in fat_ent_bread of fatent.c.\n");
156 file_descriptor = sbi->myJournal;
157 sprintf(journal_buffer, "\n-----We are in fat_ent_bread of fatent.c.\n\tfatent->fat_inode Changed: %px\n\tfatent->bhs[0] Changed: %px\n",
158         fatent->fat_inode, fatent->bhs[0]);
159 write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
160 //~~~~~ALLAGH~~~~~
161 if (fatent->bhs[0]) {
162     fat_msg_ratelimit(sb, KERN_ERR, "FAT read failed (blocknr %llu)",
163     (llu)blocknr);
164     kfree(journal_buffer);
165     return -EIO;
166 }
167 fatent->nr_bhs = 1;
168 //~~~~~ALLAGH~~~~~
169 printk(KERN_INFO "-----\n");
170 printk(KERN_INFO "-----We are in fat_ent_bread of fatent.c.\n");
171 file_descriptor = sbi->myJournal;
172 sprintf(journal_buffer, "fatent->nr_bhs Changed: %u\n", fatent->nr_bhs);
173 write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
174 //~~~~~ALLAGH~~~~~
175 ops->ent_set_ptr(fatent, offset);
176 kfree(journal_buffer);
177 return 0;
178 }

```

→ Στη `fat12_ent_next` αλλάζουν οι μεταβλητές `entry` και `nr_bhs`. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

260 static int fat12_ent_next(struct fat_entry *fatent)
261 {
262     char *journal_buffer; //dhlwsh buffer tou journal
263     struct super_block *sb = fatent->bhs[0]->b_page->mapping->host->i_sb; //to prosthew gia na mporw na kalesw to MSDOS_SB(sb)
264     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthew gia na mporw na kalesw to sbi->myJournal
265     int file_descriptor;
266     u8 **ent12_p = fatent->u.ent12_p;
267     struct buffer_head **bhs = fatent->bhs;
268     u8 *nextp = ent12_p[1] + 1 + (fatent->entry & 1);
269
270     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
271     if (!journal_buffer)
272         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
273     return -ENOMEM;
274 }
275 printk(KERN_INFO "* ~ ENTERING fat12_ent_next (%s.c/struct fatent_operations fat12_ops) ~ *");
276 fatent->entry++;
277 //~~~~~ALLAGH~~~~~
278 printk(KERN_INFO "-----\n");
279 printk(KERN_INFO "-----We are in fat12_ent_next of fatent.c.\n");
280 file_descriptor = sbi->myJournal;
281 sprintf(journal_buffer, "\n-----We are in fat12_ent next of fatent.c.\n\tfatent->entry Changed: %u\n", fatent->entry);
282 write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
283 //~~~~~ALLAGH~~~~~

```

```

295 } else {
296     WARN_ON(ent12_p[0] != (u8 *) (bhs[0]->b_data +
297                                     (bhs[0]->b_size - 1)));
298     WARN_ON(ent12_p[1] != (u8 *) bhs[1]->b_data);
299     ent12_p[0] = nextp - 1;
300     ent12_p[1] = nextp;
301     brelse(bhs[0]);
302     bhs[0] = bhs[1];
303     fatent->nr_bhs = 1;
304     //-----ALLAGH-----
305     printk(KERN_INFO "-----\n");
306     printk(KERN_INFO "----We are in fat12_ent_next of fatent.c.\n");
307     file_descriptor = sbi->myJournal;
308     sprintf(journal_buffer, " fatent->nr_bhs Changed: %u\n", fatent->nr_bhs);
309     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
310     //-----
311     kfree(journal_buffer);
312     return 1;
313 }
314 ent12_p[0] = NULL;
315 ent12_p[1] = NULL;
316 kfree(journal_buffer);
317 return 0;
318 }

```

→ Στη `fat16_ent_next` αλλάζουν οι μεταβλητές `entry` και `u.ent16_p`. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

320 static int fat16_ent_next(struct fat_entry *fatent)
321 {
322     const struct buffer_head *bh = fatent->bhs[0];
323     char *journal_buffer; //dhlwsh buffer tou journal
324     struct super_block *sb = bh->b_page->mapping->host->i_sb; //to prosthetw gia na mporw na kalesw to MSDOS_SB(sb)
325     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthetw gia na mporw na kalesw to sbi->myJournal
326     int file_descriptor;
327
328     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
329     if (!journal_buffer) {
330         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
331         return -ENOMEM;
332     }
333     fatent->entry++;
334     //-----ALLAGH-----
335     printk(KERN_INFO "-----\n");
336     printk(KERN_INFO "----We are in fat16_ent_next of fatent.c.\n");
337     file_descriptor = sbi->myJournal;
338     sprintf(journal_buffer, "\n----We are in fat16_ent_next of fatent.c.\n fatent->entry Changed: %u\n",
339             fatent->entry);
340     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
341     //-----
342     printk(KERN_INFO "* ~ ENTERING fat16_ent_next (fatent.c/struct fatent_operations fat16_ops) ~ *");
343     if (fatent->u.ent16_p < (_le16 *) (bh->b_data + (bh->b_size - 2))) {
344         fatent->u.ent16_p++;
345         return 1;
346     }
347     fatent->u.ent16_p = NULL;
348     //-----ALLAGH-----
349     printk(KERN_INFO "-----\n");
350     printk(KERN_INFO "----We are in fat16_ent_next of fatent.c.\n");
351     file_descriptor = sbi->myJournal;
352     sprintf(journal_buffer, " fatent->u.ent16_p Changed: %px\n", fatent->u.ent16_p);
353     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
354     //-----
355     kfree(journal_buffer);
356     return 0;
357 }

```

→ Στη `fat32_ent_next` αλλάζουν οι μεταβλητές `entry` και `u.ent32_p`. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

359 static int fat32_ent_next(struct fat_entry *fatent)
360 {
361     const struct buffer_head *bh = fatent->bhs[0];
362     char *journal_buffer; //dhlwsh buffer tou journal
363     struct super_block *sb = bh->b_page->mapping->host->i_sb; //to prosthetw gia na mporw na kalesw to MSDOS_SB(sb)
364     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthetw gia na mporw na kalesw to sbi->myJournal
365     int file_descriptor;
366
367     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
368     if (!journal_buffer) {
369         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
370         return -ENOMEM;
371     }
372     printk(KERN_INFO "* ~ ENTERING fat32_ent_next (fatent.c/struct fatent_operations fat32_ops) ~ *");
373     fatent->entry++;
374     //-----ALLAGH-----
375     printk(KERN_INFO "-----\n");
376     printk(KERN_INFO "----We are in fat32_ent_next of fatent.c.\n");
377     file_descriptor = sbi->myJournal;
378     sprintf(journal_buffer, "\n----We are in fat32_ent_next of fatent.c.\n fatent->entry Changed: %u\n", fatent->entry);
379     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
380     //-----

```

```

382     if (fatent->u.ent32_p < (_le32 *) (bh->b_data + (bh->b_size - 4))) {
383         fatent->u.ent32_p++;
384         return 1;
385     }
386     fatent->u.ent32_p = NULL;
387     ////////////////////////////////////////////////////ALLAGH
388     printk(KERN_INFO "-----\n");
389     printk(KERN_INFO "-----We are in fat32_ent_next of fatent.c.\n");
390     file_descriptor = sbi->myJournal;
391     sprintf(journal_buffer, " fatent->u.ent32_p Changed: %px\n", fatent->u.ent32_p);
392     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
393     //////////////////////////////////////////////////
394     kfree(journal_buffer);
395     return 0;
396 }

```

→ Στη `fat_ent_update_ptr` αλλάζει η μεταβλητή `nr_bhs`. Προσθέτουμε κώδικα για να εγγράφουμε τις αλλαγές στο journal.

```

497 static inline int fat_ent_update_ptr(struct super_block *sb,
498                                     struct fat_entry *fatent,
499                                     int offset, sector_t blocknr)
500 {
501     struct msdos_sb_info *sbi = MSDOS_SB(sb);
502     const struct fatent_operations *ops = sbi->fatent_ops;
503     struct buffer_head **bhs = fatent->bhs;
504     char *journal_buffer; //dhlwsh buffer tou journal
505     int file_descriptor;
506
507     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
508     if (!journal_buffer) {
509         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
510         return -ENOMEM;
511     }
512     /* Is this fatent's blocks including this entry? */
513     if (!fatent->nr_bhs || bhs[0]->b_blocknr != blocknr){
514         kfree(journal_buffer);
515         return 0;
516     }
517     if (is_fat12(sbi)) {
518         if ((offset + 1) < sb->s_blocksize) {
519             /* This entry is on bhs[0]. */
520             if (fatent->nr_bhs == 2) {
521                 brelse(bhs[1]);
522                 fatent->nr_bhs = 1;
523                 ////////////////////////////////////////////////////ALLAGH
524                 printk(KERN_INFO "-----\n");
525                 printk(KERN_INFO "-----We are in fat_ent_update_ptr of fatent.c.\n");
526                 file_descriptor = sbi->myJournal;
527                 sprintf(journal_buffer, "\n-----We are in fat_ent_update_ptr of fatent.c.\n fatent->nr_bhs Changed: %u\n",
528                         fatent->nr_bhs);
529                 write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
530                 //////////////////////////////////////////////////
531             } else {
532                 /* This entry needs the next block. */
533                 if (fatent->nr_bhs != 2){
534                     kfree(journal_buffer);
535                     return 0;
536                 }
537                 if (bhs[1]->b_blocknr != (blocknr + 1)){
538                     kfree(journal_buffer);
539                     return 0;
540                 }
541             }
542         }
543     }
544     ops->ent_set_ptr(fatent, offset);
545     kfree(journal_buffer);
546     return 1;
547 }

```

→ Στη `fat_alloc_clusters` αλλάζει η μεταβλητή `entry`. Προσθέτουμε κώδικα για να εγγράφουμε τις αλλαγές στο journal.

```

666     int fat_alloc_clusters(struct inode *inode, int *cluster, int nr_cluster)
667 {
668     struct super_block *sb = inode->i_sb;
669     struct msdos_sb_info *sbi = MSDOS_SB(sb);
670     const struct fatent_operations *ops = sbi->fatent_ops;
671     struct fat_entry fatent, prev_ent;
672     struct buffer_head *bhs[MAX_BUF_PER_PAGE];
673     int i, count, err, nr_bhs, idx_clus;
674     char *journal_buffer; //dhlwsh buffer tou journal
675     int file_descriptor;
676
677     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
678     if (!journal_buffer) {
679         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
680         return -ENOMEM;
681     }
682     BUG_ON(nr_cluster > (MAX_BUF_PER_PAGE / 2));    /* fixed limit */
683
684     lock_fat(sbi);
685     if (sbi->free_clusters != -1 && sbi->free_clus_valid &&
686         sbi->free_clusters < nr_cluster) {
687         unlock_fat(sbi);
688         kfree(journal_buffer);
689         return -ENOSPC;
690     }

```

```

697     while (count < sbi->max_cluster) {
698         if (fatent.entry >= sbi->max_cluster){
699             fatent.entry = FAT_START_ENT;
700             //-----ALLAGH-----
701             printk(KERN_INFO "-----\n");
702             printk(KERN_INFO "-----We are in fat_alloc_clusters of fatent.c.\n");
703             file_descriptor = sbi->myJournal;
704             sprintf(journal_buffer, "\n-----We are in fat_alloc_clusters of fatent.c.\n\n fatent.entry Changed: %u\n",
705                     fatent.entry);
706             write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
707             //-----
708         }
709         fatent_set_entry(&fatent, fatent.entry);
710         err = fat_ent_read_block(sb, &fatent);
711         if (err)
712             goto out;
713

```

- ❖ Τέλος, ψάχνουμε με Ctrl+F στις *misc.c*, *file.c* και *cache.c* μεταβλητές *fat_entry* και παρατηρούμε ότι δεν υπάρχουν αλλαγές.

□ SUPERBLOCK

Στη συνέχεια παρατηρούμε ότι η δομή superblock ορίζεται στο *fs/fat/fat.h* στη δομή *msdos_sb_info*.

```

79 struct msdos_sb_info {
80     int myJournal;
81     unsigned short sec_per_clus; /* sectors/cluster */
82     unsigned short cluster_bits; /* log2(cluster_size) */
83     unsigned int cluster_size; /* cluster size */
84     unsigned char fats, fat_bits; /* number of FATs, FAT bits (12,16 or 32) */
85     unsigned short fat_start;
86     unsigned long fat_length; /* FAT start & length (sec.) */
87     unsigned long dir_start;
88     unsigned short dir_entries; /* root dir start & entries */
89     unsigned long data_start; /* first data sector */
90     unsigned long max_cluster; /* maximum cluster number */
91     unsigned long root_cluster; /* first cluster of the root directory */
92     unsigned long fsinfo_sector; /* sector number of FAT32 fsinfo */
93     struct mutex fat_lock;
94     struct mutex nfs_build_inode_lock;
95     struct mutex s_lock;
96     unsigned int prev_free; /* previously allocated cluster number */
97     unsigned int free_clusters; /* -1 if undefined */
98     unsigned int free_clus_valid; /* is free_clusters valid? */
99     struct fat_mount_options options;
100    struct nls_table *nls_disk; /* Codepage used on disk */
101    struct nls_table *nls_io; /* Charset used for input and display */
102    const void *dir_ops; /* Opaque; default directory operations */
103    int dir_per_block; /* dir entries per block */
104    int dir_per_block_bits; /* log2(dir_per_block) */
105    unsigned int vol_id; /*volume ID*/
106
107    int fatent_shift;
108    const struct fatent_operations *fatent_ops;
109    struct inode *fat_inode;
110    struct inode *fsinfo_inode;
111
112    struct ratelimit_state ratelimit;
113
114    spinlock_t inode_hash_lock;
115    struct hlist_head inode_hashtable[FAT_HASH_SIZE];
116
117    spinlock_t dir_hash_lock;
118    struct hlist_head dir_hashtable[FAT_HASH_SIZE];
119
120    unsigned int dirty; /* fs state before mount */
121    struct rcu_head rcu;
122 };

```

Όπως νωρίτερα, θα χρησιμοποιήσουμε την εντολή **grep -r** για να κάνουμε αναζήτηση.
Παρακάτω έχουμε το screenshot από την εκτέλεση της εντολής **grep -r msdos_sb_info**.

```
myy601@myy601lab2:~/lkl-source$ grep -r msdos_sb_info
grep: vmlinux: binary file matches
grep: vmlinux.o: binary file matches
grep: tools/lkl/tests/disk: binary file matches
grep: tools/lkl/tests/boot: binary file matches
grep: tools/lkl/tests/net-test: binary file matches
grep: tools/lkl/tests/disk-vfio-pci: binary file matches
grep: tools/lkl/tests/test-dlmopen: binary file matches
grep: tools/lkl/liblkl.a: binary file matches
grep: tools/lkl/cptofs: binary file matches
grep: tools/lkl/lib/liblkl.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-zpoline.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-hijack.so: binary file matches
grep: tools/lkl/lib/lklo.o: binary file matches
grep: tools/lkl/fs2tar: binary file matches
grep: fs/fat/fatent.o: binary file matches
grep: fs/fat/misc.o: binary file matches
fs/fat/fatent.c:     struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/fatent.c:     struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/fatent.c:static inline void lock_fat(struct msdos_sb_info *sbi)
fs/fat/fatent.c:static inline void unlock_fat(struct msdos_sb_info *sbi)
fs/fat/fatent.c:     struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/fatent.c:     struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/fatent.c:     struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
fs/fat/fatent.c:     struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/fatent.c:static inline int fat_ent_next(struct msdos_sb_info *sbi,
fs/fat/fatent.c:     struct msdos_sb_info *sbi = MSDOS_SB(sb);
grep: fs/fat/nfs.o: binary file matches
fs/fat/dir.c:   struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/dir.c:   struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/dir.c:static inline int fat_name_match(struct msdos_sb_info *sbi,
fs/fat/dir.c: const struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/dir.c:   struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/dir.c:   struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/dir.c:   struct msdos_sb_info *sbi = MSDOS_SB(sb);
grep: fs/fat/inode.o: binary file matches
grep: fs/fat/cache.o: binary file matches
fs/fat/fat.h:struct msdos_sb_info {
fs/fat/fat.h:static inline struct msdos_sb_info *MSDOS_SB(struct super_block *sb)
fs/fat/fat.h:static inline bool is_fat12(const struct msdos_sb_info *sbi)
fs/fat/fat.h:static inline bool is_fat16(const struct msdos_sb_info *sbi)
fs/fat/fat.h:static inline bool is_fat32(const struct msdos_sb_info *sbi)
fs/fat/fat.h:   struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/fat.h:   struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
fs/fat/fat.h:static inline umode_t fat_make_mode(struct msdos_sb_info *sbi,
fs/fat/fat.h:static inline sector_t fat_clus_to_blknr(struct msdos_sb_info *sbi, int clus)
fs/fat/fat.h:static inline void fat_get_blknr_offset(struct msdos_sb_info *sbi,
fs/fat/fat.h:static inline loff_t fat_i_pos_read(struct msdos_sb_info *sbi,
fs/fat/fat.h:static inline int fat_get_start(const struct msdos_sb_info *sbi,
fs/fat/fat.h:static inline bool fat_valid_entry(struct msdos_sb_info *sbi, int entry)
fs/fat/fat.h:extern void fat_time_fat2unix(struct msdos_sb_info *sbi, struct timespec64 *ts,
fs/fat/fat.h:extern void fat_time_unix2fat(struct msdos_sb_info *sbi, struct timespec64 *ts,
fs/fat/fat.h:extern struct timespec64 fat_truncate_atime(const struct msdos_sb_info *sbi,
fs/fat/fat.h:extern struct timespec64 fat_truncate_mtime(const struct msdos_sb_info *sbi,
fs/fat/misc.c:   struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/misc.c:   struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/misc.c:static inline int fat_tz_offset(const struct msdos_sb_info *sbi)
fs/fat/misc.c:void fat_time_fat2unix(struct msdos_sb_info *sbi, struct timespec64 *ts,
fs/fat/misc.c:void fat_time_unix2fat(struct msdos_sb_info *sbi, struct timespec64 *ts,
fs/fat/misc.c:struct timespec64 fat_truncate_atime(const struct msdos_sb_info *sbi,
fs/fat/misc.c:struct timespec64 fat_truncate_mtime(const struct msdos_sb_info *sbi,
fs/fat/misc.c:   struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
fs/fat/namei_msdos.c:   struct msdos_sb_info *sbi = MSDOS_SB(dir->i_sb);
fs/fat/namei_msdos.c:   struct msdos_sb_info *sbi = MSDOS_SB(dir->i_sb);
grep: fs/fat/dir.o: binary file matches
grep: fs/fat/file.o: binary file matches
fs/fat/file.c:   struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
fs/fat/file.c:   struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
fs/fat/file.c:   struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/file.c:   struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
fs/fat/file.c:   struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
fs/fat/file.c:static int fat_SANITIZE_MODE(const struct msdos_sb_info *sbi,
fs/fat/file.c:                           struct msdos_sb_info *sbi, struct inode *inode)
fs/fat/file.c:   struct msdos_sb_info *sbi = MSDOS_SB(dentry->d_sb);
grep: fs/fat/namei_vfat.o: binary file matches
```

```

fs/fat/nfs.c: struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/nfs.c: struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
fs/fat/nfs.c: struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/fat_test.c: static struct msdos_sb_info fake_sb;
fs/fat/fat_test.c: static struct msdos_sb_info fake_sb;
fs/fat/cache.c: struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/cache.c: struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/cache.c: struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
fs/fat/inode.c: struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/inode.c: struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/inode.c: struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/inode.c: struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
fs/fat/inode.c: struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
fs/fat/inode.c: struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/inode.c: struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
fs/fat/inode.c: static inline void fat_lock_build_inode(struct msdos_sb_info *sbi)
fs/fat/inode.c: static inline void fat_unlock_build_inode(struct msdos_sb_info *sbi)
fs/fat/inode.c: struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/inode.c: struct msdos_sb_info *sbi = container_of(p, struct msdos_sb_info, rCU);
fs/fat/inode.c: struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/inode.c: struct msdos_sb_info *sbi = MSDOS_SB(root->dir);
fs/fat/inode.c: struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
fs/fat/inode.c: struct msdos_sb_info *sbi = MSDOS_SB(sb);
fs/fat/inode.c: struct msdos_sb_info *sbi;
fs/fat/inode.c: sbi = kzalloc(sizeof(struct msdos_sb_info), GFP_KERNEL);
fs/fat/namei_vfat.c: struct msdos_sb_info *sbi = MSDOS_SB(dir->i_sb);
grep: lkl.o: binary file matches

```

Μας κατευθύνει στις ***fatent.c***, ***dir.c***, ***fat.h***, ***misc.c***, ***namei_msdos.c***, ***file.c***, ***nfs.c***, ***fat_test.c***, ***cache.c***, ***inode.c*** και ***namei_vfat.c***.

- ❖ Αρχικά ψάχνουμε με Ctrl+F στην ***fatent.c*** μεταβλητές ***msdos_sb_info*** και παρατηρούμε ότι υπάρχουν αλλαγές στη ***fat_ent_access_init*** στις μεταβλητές ***fatent_shift*** και ***fatent_ops***. Προσθέτουμε κώδικα για να εγγράφουμε τις αλλαγές στο journal.

```

435 void fat_ent_access_init(struct super_block *sb)
436 {
437     struct msdos_sb_info *sbi = MSDOS_SB(sb);
438     char *journal_buffer; //dhlwsh buffer tou journal
439     int file_descriptor;
440
441     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
442     if (!journal_buffer) {
443         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
444         return;
445     }
446     mutex_init(&sbi->fat_lock);
447
448     if (is_fat32(sbi)) {
449         sbi->fatent_shift = 2;
450         sbi->fatent_ops = &fat32_ops;
451         //-----ALLAGH-----
452         printk(KERN_INFO "-----\n");
453         printk(KERN_INFO "-----We are in fat_ent_access_init of fatent.c.-----\n");
454         file_descriptor = sbi->myJournal;
455         sprintf(journal_buffer, "\n-----We are in fat_ent_access_init of fatent.c.-----\n %s->fatent_shift Changed: %u\n %s->fatent_ops Changed: %px\n",
456                 sbi->fatent_shift, sbi->fatent_ops);
457         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
458         //-----
459     } else if (is_fat16(sbi)) {
460         sbi->fatent_shift = 1;
461         sbi->fatent_ops = &fat16_ops;
462         //-----ALLAGH-----
463         printk(KERN_INFO "-----\n");
464         printk(KERN_INFO "-----We are in fat_ent_access_init of fatent.c.-----\n");
465         file_descriptor = sbi->myJournal;
466         sprintf(journal_buffer, "\n-----We are in fat_ent_access_init of fatent.c.-----\n %s->fatent_shift Changed: %u\n %s->fatent_ops Changed: %px\n",
467                 sbi->fatent_shift, sbi->fatent_ops);
468         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
469         //-----
470     } else if (is_fat12(sbi)) {
471         sbi->fatent_shift = -1;
472         sbi->fatent_ops = &fat12_ops;
473         //-----ALLAGH-----
474         printk(KERN_INFO "-----\n");
475         printk(KERN_INFO "-----We are in fat_ent_access_init of fatent.c.-----\n");
476         file_descriptor = sbi->myJournal;
477         sprintf(journal_buffer, "\n-----We are in fat_ent_access_init of fatent.c.-----\n %s->fatent_shift Changed: %u\n %s->fatent_ops Changed: %px\n",
478                 sbi->fatent_shift, sbi->fatent_ops);
479         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
480         //-----
481     } else {
482         fat_fs_error(sb, "invalid FAT variant, %u bits", sbi->fat_bits);
483     }
484     kfree(journal_buffer);
485 }

```

- ❖ Έπειτα ψάχνουμε με Ctrl+F μεταβλητές **msdos_sb_info** στις ***dir.c, misc.c, namei_msdos.c, file.c*** και ***nfs.c*** και βλέπουμε ότι δεν υπάρχουν αλλαγές.
- ❖ Στη συνέχεια ψάχνουμε με Ctrl+F μεταβλητές **msdos_sb_info** στη ***fat_test.c*** και βλέπουμε ότι υπάρχουν αλλαγές στις ακόλουθες συναρτήσεις:
 - ➔ Στη ***fat_time_fat2unix_test*** στις μεταβλητές **options.tz_set** και **options.time_offset**.
Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

132 static void fat_time_fat2unix_test(struct kunit *test)
133 {
134     static struct msdos_sb_info fake_sb;
135     struct timespec64 ts;
136     struct fat_timestamp testcase =
137         ({struct fat_timestamp testcase *)test->param_value;
138     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthew gia na mporw na kalesw to sbi->myJournal
139     char *journal_buffer; //dhlwsh buffer tou journal
140     int file_descriptor;
141     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
142     if (!journal_buffer) {
143         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
144         return;
145     }
146     fake_sb.options.tz_set = 1;
147     fake_sb.options.time_offset = testcase->time_offset;
148     ////////////////////////////////////////////////////////////////////ALLAGH
149     printk(KERN_INFO "\n");
150     printk(KERN_INFO "-----We are in fat_time_fat2unix_test of fat_test.c.-----\n");
151     sprintf(journal_buffer, "\n-----We are in fat_time_fat2unix_test of fat_test.c.-----\n fake_sb.options.tz_set Changed: %u\n fake_sb.options.time_offset Changed: %u\n",
152             fake_sb.options.tz_set, fake_sb.options.time_offset);
153     file_descriptor = sbi->myJournal;
154     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
155     //////////////////////////////////////////////////////////////////
156
157     fat_time_fat2unix(&fake_sb, &ts,
158                     testcase->time,
159                     testcase->date,
160                     testcase->cs);
161     KUNIT_EXPECT_EQ_MSG(test,
162                         testcase->ts.tv_sec,
163                         ts.tv_sec,
164                         "Timestamp mismatch (seconds)\n");
165     KUNIT_EXPECT_EQ_MSG(test,
166                         testcase->ts.tv_nsec,
167                         ts.tv_nsec,
168                         "Timestamp mismatch (nanoseconds)\n");
169     kfree(journal_buffer);
170 }
```

- ➔ Στη ***fat_time_unix2fat_test*** στις μεταβλητές **options.tz_set** και **options.time_offset**.
Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

172 static void fat_time_unix2fat_test(struct kunit *test)
173 {
174     static struct msdos_sb_info fake_sb;
175     __le16 date, time;
176     u8 cs;
177     struct fat_timestamp testcase =
178         ({struct fat_timestamp testcase *)test->param_value;
179     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthew gia na mporw na kalesw to sbi->myJournal
180     char *journal_buffer; //dhlwsh buffer tou journal
181     int file_descriptor;
182     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
183     if (!journal_buffer) {
184         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
185         return;
186     }
187     fake_sb.options.tz_set = 1;
188     fake_sb.options.time_offset = testcase->time_offset;
189     ////////////////////////////////////////////////////////////////////ALLAGH
190     printk(KERN_INFO "\n");
191     printk(KERN_INFO "-----We are in fat_time_unix2fat_test of fat_test.c.-----\n");
192     file_descriptor = sbi->myJournal;
193     sprintf(journal_buffer, "\n-----We are in fat_time_unix2fat_test of fat_test.c.-----\n fake_sb.options.tz_set Changed: %u\n fake_sb.options.time_offset Changed: %u\n",
194             fake_sb.options.tz_set, fake_sb.options.time_offset);
195     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
196     //////////////////////////////////////////////////////////////////
197
198     fat_time_unix2fat(&fake_sb, &testcase->ts,
199                     &time, &date, &cs);
200     KUNIT_EXPECT_EQ_MSG(test,
201                         le16_to_cpu(testcase->time),
202                         le16_to_cpu(time),
203                         "Time mismatch\n");
204     KUNIT_EXPECT_EQ_MSG(test,
205                         le16_to_cpu(testcase->date),
206                         le16_to_cpu(date),
207                         "Date mismatch\n");
208     KUNIT_EXPECT_EQ_MSG(test,
209                         testcase->cs,
210                         cs,
211                         "Centisecond mismatch\n");
212     kfree(journal_buffer);
213 }
```

- ❖ Συνεχίζουμε ψάχνοντας με Ctrl+F μεταβλητές **msdos_sb_info** στην ***cache.c*** και βλέπουμε ότι δεν υπάρχουν αλλαγές.

- ❖ Με τον ίδιο τρόπο ψάχνω για μεταβλητές **msdos_sb_info** στην **inode.c** και βλέπουμε ότι υπάρχουν αλλαγές στη συνάρτηση **fat_fill_super** στις μεταβλητές **sec_per_clus**, **cluster_size**, **cluster_bits**, **fats**, **fat_bits**, **fat_start**, **fat_length**, **root_cluster**, **free_clusters**, **free_clus_valid**, **prev_free**, **myJournal**, **fsinfo_sector**, **vol_id**, **dir_per_block**, **dir_per_block_bits**, **dir_start**, **data_start**, **dirty**, **max_cluster**, **nls_disk**, **nls_io**, **fat_inode**, **fsinfo_inode**. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

1878     int fat_fill_super(struct super_block *sb, void *data, int silent, int isvfat,
1879                         void (*setup)(struct super_block *));
1880 {
1881     struct inode *root_inode = NULL, *fat_inode = NULL;
1882     struct inode *fsinfo_inode = NULL;
1883     struct buffer_head *bh;
1884     struct fat_bios_param_block bpb;
1885     struct msdos_sb_info *sbi;
1886     u16 logical_sector_size;
1887     u32 total_sectors, total_clusters, fat_clusters, rootdir_sectors;
1888     int debug;
1889     long error;
1890     char buf[50];
1891     struct timespec64 ts;
1892     char *journal_buffer; //dhlwsh buffer tou journal
1893     int file_descriptor;
1894
1895     printk(KERN_INFO " ~ ENTERING fat_fill_super (inode.c) ~ *");
1896     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
1897     if (!journal_buffer) {
1898         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
1899         return -ENOMEM;
1900     }
1901
1958     sbi->sec_per_clus = bpb.fat_sec_per_clus;
1959     //~~~~~ALLAGH~~~~~
1960     printk(KERN_INFO "-----\n");
1961     printk(KERN_INFO "-----We are in fat_fill_super of inode.c-----\n");
1962     file_descriptor = sbi->myJournal;
1963     sprintf(journal_buffer, "\n-----We are in fat_fill_super of inode.c----- sbi->sec_per_clus Changed: %u\n",
1964             sbi->sec_per_clus);
1965     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1966     //~~~~~ALLAGH~~~~~
1995     mutex_init(&sbi->s_lock);
1996     sbi->cluster_size = sb->s_blocksize * sbi->sec_per_clus;
1997     sbi->cluster_bits = ffs(sbi->cluster_size) - 1;
1998     //~~~~~ALLAGH~~~~~
1999     printk(KERN_INFO "-----\n");
2000     printk(KERN_INFO "-----We are in fat_fill_super of inode.c-----\n");
2001     file_descriptor = sbi->myJournal;
2002     sprintf(journal_buffer, " sbi->cluster_size Changed: %u\n sbi->cluster_bits Changed: %u\n",
2003             sbi->cluster_size, sbi->cluster_bits);
2004     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2005     //~~~~~ALLAGH~~~~~
2007     sbi->fats = bpb.fat_fats;
2008     sbi->fat_bits = 0; /* Don't know yet */
2009     //~~~~~ALLAGH~~~~~
2010     printk(KERN_INFO "-----\n");
2011     printk(KERN_INFO "-----We are in fat_fill_super of inode.c-----\n");
2012     file_descriptor = sbi->myJournal;
2013     sprintf(journal_buffer, " sbi->fats Changed: %u\n sbi->fat_bits Changed: %u\n",
2014             sbi->fats, sbi->fat_bits);
2015     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2016     //~~~~~ALLAGH~~~~~
2018     sbi->fat_start = bpb.fat_reserved;
2019     sbi->fat_length = bpb.fat_fat_length;
2020     //~~~~~ALLAGH~~~~~
2021     printk(KERN_INFO "-----\n");
2022     printk(KERN_INFO "-----We are in fat_fill_super of inode.c-----\n");
2023     file_descriptor = sbi->myJournal;
2024     sprintf(journal_buffer, " sbi->fat_start Changed: %u\n sbi->fat_length Changed: %lu\n",
2025             sbi->fat_start, sbi->fat_length);
2026     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2027     //~~~~~ALLAGH~~~~~

```

```

2029     sbi->root_cluster = 0;
2030     sbi->free_clusters = -1; /* Don't know yet */
2031 ////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
2032     printk(KERN_INFO "-----\n");
2033     printk(KERN_INFO "-----We are in fat_fill_super of inode.c.\n-----\n");
2034     file_descriptor = sbi->myJournal;
2035     sprintf(journal_buffer, " sbi->root_cluster Changed: %lu\n sbi->free_clusters Changed: %u\n",
2036             sbi->root_cluster, sbi->free_clusters);
2037     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2038 //////////////////////////////////////////////////
2039
2040     sbi->free_clus_valid = 0;
2041     sbi->prev_free = FAT_START_ENT;
2042 ////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
2043     printk(KERN_INFO "-----\n");
2044     printk(KERN_INFO "-----We are in fat_fill_super of inode.c.\n-----\n");
2045     file_descriptor = sbi->myJournal;
2046     sprintf(journal_buffer, " sbi->free_clus_valid Changed: %u\n sbi->prev_free Changed: %u\n",
2047             sbi->free_clus_valid, sbi->prev_free);
2048     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2049 //////////////////////////////////////////////////
2050
2051     if (!sbi->fat_length && bpbo.fat32_length) {
2052         struct fat_boot_fsinfo *fsinfo;
2053         struct buffer_head *fsinfo_bh;
2054
2055         /* Must be FAT32 */
2056         sbi->fat_bits = 32;
2057         sbi->fat_length = bpbo.fat32_length;
2058 ////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
2059         printk(KERN_INFO "-----\n");
2060         printk(KERN_INFO "-----We are in fat_fill_super of inode.c.\n-----\n");
2061         file_descriptor = sbi->myJournal;
2062         sprintf(journal_buffer, "\n-----We are in fat_fill_super of inode.c.\n-----\n sbi->fat_bits Changed: %u\n sbi->fat_length Changed: %lu\n",
2063                 sbi->fat_bits, sbi->fat_length);
2064         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2065 //////////////////////////////////////////////////
2066
2067         sbi->root_cluster = bpbo.fat32_root_cluster;
2068         /* MC - if info_sector is 0, don't multiply by 0 */
2069         sbi->fsinfo_sector = bpbo.fat32_info_sector;
2070 ////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
2071         printk(KERN_INFO "-----\n");
2072         printk(KERN_INFO "-----We are in fat_fill_super of inode.c.\n-----\n");
2073         file_descriptor = sbi->myJournal;
2074         sprintf(journal_buffer, " sbi->root_cluster Changed: %lu\n sbi->fsinfo_sector Changed: %lu\n",
2075                 sbi->root_cluster, sbi->fsinfo_sector);
2076         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2077 //////////////////////////////////////////////////
2078
2079     if (sbi->fsinfo_sector == 0){
2080         sbi->fsinfo_sector = 1;
2081         ////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
2082         printk(KERN_INFO "-----\n");
2083         printk(KERN_INFO "-----We are in fat_fill_super of inode.c.\n-----\n");
2084         file_descriptor = sbi->myJournal;
2085         sprintf(journal_buffer, "\n-----We are in fat_fill_super of inode.c.\n-----\n sbi->fsinfo_sector Changed: %lu\n", sbi->fsinfo_sector);
2086         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2087     }
2088
2089
2090     fsinfo = (struct fat_boot_fsinfo *)fsinfo_bh->b_data;
2091     if (!IS_FSINFO(fsinfo)) {
2092         fat_msg(sb, KERN_WARNING, "Invalid FSINFO signature: "
2093                 "0x%08x, 0x%08x (sector = %lu)",
2094                 le32_to_cpu(fsinfo->signature1),
2095                 le32_to_cpu(fsinfo->signature2),
2096                 sbi->fsinfo_sector);
2097     } else {
2098         if (sbi->options.usefree){
2099             sbi->free_clus_valid = 1;
2100             ////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
2101             printk(KERN_INFO "-----\n");
2102             printk(KERN_INFO "-----We are in fat_fill_super of inode.c.\n-----\n");
2103             file_descriptor = sbi->myJournal;
2104             sprintf(journal_buffer, "\n-----We are in fat_fill_super of inode.c.\n-----\n sbi->free_clus_valid Changed: %u\n", sbi->free_clus_valid);
2105             write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2106         }
2107         sbi->free_clusters = le32_to_cpu(fsinfo->free_clusters);
2108         sbi->prev_free = le32_to_cpu(fsinfo->next_cluster);
2109 ////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
2110         printk(KERN_INFO "-----\n");
2111         printk(KERN_INFO "-----We are in fat_fill_super of inode.c.\n-----\n");
2112         file_descriptor = sbi->myJournal;
2113         sprintf(journal_buffer, "\n-----We are in fat_fill_super of inode.c.\n-----\n sbi->free_clusters Changed: %u\n sbi->prev_free Changed: %u\n",
2114                 sbi->free_clusters, sbi->prev_free);
2115         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2116     }
2117
2118     brelse(fsinfo_bh);
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134

```

```

2136 /* interpret volume ID as a little endian 32 bit integer */
2137 if (is_fat32(sbi))
2138     sbi->vol_id = bp->fat32.vol_id;
2139 else /* fat 16 or 12 */
2140     sbi->vol_id = bp->fat16.vol_id;
2141 //~~~~~ALLAGH~~~~~
2142 printk(KERN_INFO "-----\n");
2143 printk(KERN_INFO "----We are in fat_fill_super of inode.c.----\n");
2144 file_descriptor = sbi->myJournal;
2145 sprintf(journal_buffer, "sbi->vol_id Changed: %u\n", sbi->vol_id);
2146 write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2147 //~~~~~ALLAGH~~~~~
2148
2149 sbi->dir_per_block = sb->s_blocksize / sizeof(struct msdos_dir_entry);
2150 sbi->dir_per_block_bits = ffs(sbi->dir_per_block) - 1;
2151 //~~~~~ALLAGH~~~~~
2152 printk(KERN_INFO "-----\n");
2153 printk(KERN_INFO "----We are in fat_fill_super of inode.c.----\n");
2154 file_descriptor = sbi->myJournal;
2155 sprintf(journal_buffer, "sbi->dir_per_block Changed: %u\n sbi->dir_per_block Bits Changed: %u\n",
2156         sbi->dir_per_block, sbi->dir_per_block_bits);
2157 write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2158 //~~~~~ALLAGH~~~~~
2159
2160 sbi->dir_start = sbi->fat_start + sbi->fats * sbi->fat_length;
2161 sbi->dir_entries = bp->fat_dir_entries;
2162 //~~~~~ALLAGH~~~~~
2163 printk(KERN_INFO "-----\n");
2164 printk(KERN_INFO "----We are in fat_fill_super of inode.c.----\n");
2165 file_descriptor = sbi->myJournal;
2166 sprintf(journal_buffer, "sbi->dir_start Changed: %lu\n sbi->dir_entries Changed: %u\n",
2167         sbi->dir_start, sbi->dir_entries);
2168 write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2169 //~~~~~ALLAGH~~~~~

2180 sbi->data_start = sbi->dir_start + rootdir_sectors;
2181 //~~~~~ALLAGH~~~~~
2182 printk(KERN_INFO "-----\n");
2183 printk(KERN_INFO "----We are in fat_fill_super of inode.c.----\n");
2184 file_descriptor = sbi->myJournal;
2185 sprintf(journal_buffer, "sbi->data_start Changed: %lu\n", sbi->data_start);
2186 write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2187 //~~~~~ALLAGH~~~~~

2195 if (!is_fat32(sbi)){
2196     sbi->fat_bits = (total_clusters > MAX_FAT12) ? 16 : 12;
2197 //~~~~~ALLAGH~~~~~
2198     printk(KERN_INFO "-----\n");
2199     printk(KERN_INFO "----We are in fat_fill_super of inode.c.----\n");
2200     file_descriptor = sbi->myJournal;
2201     sprintf(journal_buffer, "\n----We are in fat_fill_super of inode.c.\n sbi->fat_bits Changed: %u\n", sbi->fat_bits);
2202     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2203 }
2204 //~~~~~ALLAGH~~~~~

2206 if (is_fat32(sbi))
2207     sbi->dirty = bp->fat32.state & FAT_STATE_DIRTY;
2208 else /* fat 16 or 12 */
2209     sbi->dirty = bp->fat16.state & FAT_STATE_DIRTY;
2210 //~~~~~ALLAGH~~~~~
2211 printk(KERN_INFO "-----\n");
2212 printk(KERN_INFO "----We are in fat_fill_super of inode.c.----\n");
2213 file_descriptor = sbi->myJournal;
2214 sprintf(journal_buffer, "sbi->dirty Changed: %u\n", sbi->dirty);
2215 write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2216 //~~~~~ALLAGH~~~~~

2228 sbi->max_cluster = total_clusters + FAT_START_ENT;
2229 //~~~~~ALLAGH~~~~~
2230 printk(KERN_INFO "-----\n");
2231 printk(KERN_INFO "----We are in fat_fill_super of inode.c.----\n");
2232 file_descriptor = sbi->myJournal;
2233 sprintf(journal_buffer, "sbi->max_cluster Changed: %lu\n", sbi->max_cluster);
2234 write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2235 //~~~~~ALLAGH~~~~~
2236

```

```

2236
2237     /* check the free_clusters, it's not necessarily correct */
2238     if (sbi->free_clusters != -1 && sbi->free_clusters > total_clusters){
2239         sbi->free_clusters = -1;
2240         /////////////////////////////////////////////////////ALLAGH/////////////////////////////////////////////////
2241         printk(KERN_INFO "-----\n");
2242         printk(KERN_INFO "-----We are in fat_fill_super of inode.c.\n");
2243         file_descriptor = sbi->myJournal;
2244         sprintf(journal_buffer, "\n-----We are in fat_fill_super of inode.c.\n\n sbi->free_clusters Changed: %u\n",
2245                 sbi->free_clusters);
2246         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2247         /////////////////////////////////////////////////////
2248
2249     /* check the prev_free, it's not necessarily correct */
2250     sbi->prev_free += sbi->max_cluster;
2251     if (sbi->prev_free < FAT_START_ENT)
2252         sbi->prev_free = FAT_START_ENT;
2253     /////////////////////////////////////////////////////ALLAGH/////////////////////////////////////////////////
2254     printk(KERN_INFO "-----\n");
2255     printk(KERN_INFO "-----We are in fat_fill_super of inode.c.\n");
2256     file_descriptor = sbi->myJournal;
2257     sprintf(journal_buffer, " sbi->prev_free Changed: %u\n", sbi->prev_free);
2258     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2259     /////////////////////////////////////////////////////
2260
2261
2262     error = -EINVAL;
2263     sprintf(buf, "cp%d", sbi->options.codepage);
2264     sbi->nls_disk = load_nls(buf);
2265     if (!sbi->nls_disk) {
2266         fat_msg(sb, KERN_ERR, "codepage %s not found", buf);
2267         goto out_fail;
2268     }
2269     /////////////////////////////////////////////////////ALLAGH/////////////////////////////////////////////////
2270     printk(KERN_INFO "-----\n");
2271     printk(KERN_INFO "-----We are in fat_fill_super of inode.c.\n");
2272     file_descriptor = sbi->myJournal;
2273     sprintf(journal_buffer, " sbi->nls_disk Changed: %px\n", sbi->nls_disk);
2274     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2275     /////////////////////////////////////////////////////
2276
2277     /* FIXME: utf8 is using iocharset for upper/lower conversion */
2278     if (sbi->options.isvfat) {
2279         sbi->nls_io = load_nls(sbi->options.iocharset);
2280         if (!sbi->nls_io) {
2281             fat_msg(sb, KERN_ERR, "IO charset %s not found",
2282                     sbi->options.iocharset);
2283             goto out_fail;
2284         }
2285         /////////////////////////////////////////////////////ALLAGH/////////////////////////////////////////////////
2286         printk(KERN_INFO "-----\n");
2287         printk(KERN_INFO "-----We are in fat_fill_super of inode.c.\n");
2288         file_descriptor = sbi->myJournal;
2289         sprintf(journal_buffer, "\n-----We are in fat_fill_super of inode.c.\n\n sbi->nls_io Changed: %px\n",
2290                 sbi->nls_io);
2291         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2292         /////////////////////////////////////////////////////
2293
2294
2295     sbi->fat_inode = fat_inode;
2296     /////////////////////////////////////////////////////ALLAGH/////////////////////////////////////////////////
2297     printk(KERN_INFO "-----\n");
2298     printk(KERN_INFO "-----We are in fat_fill_super of inode.c.\n");
2299     file_descriptor = sbi->myJournal;
2300     sprintf(journal_buffer, "\n-----We are in fat_fill_super of inode.c.\n\n sbi->fat_inode Changed: %px\n", sbi->fat_inode);
2301     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2302     /////////////////////////////////////////////////////
2303
2304
2305     fsinfo_inode = new_inode(sb);
2306     if (!fsinfo_inode){
2307         goto out_fail;
2308     }
2309     fsinfo_inode->i_ino = MSDOS_FSINFO_INO;
2310     sbi->fsinfo_inode = fsinfo_inode;
2311     /////////////////////////////////////////////////////ALLAGH/////////////////////////////////////////////////
2312     printk(KERN_INFO "-----\n");
2313     printk(KERN_INFO "-----We are in fat_fill_super of inode.c.\n");
2314     file_descriptor = sbi->myJournal;
2315     sprintf(journal_buffer, "\n-----We are in fat_fill_super of inode.c.\n\n sbi->fsinfo_inode Changed: %px\n", sbi->fsinfo_inode);
2316     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
2317     /////////////////////////////////////////////////////

```

- ❖ Τέλος ψάχνω με Ctrl+F μεταβλητές **msdos_sb_info** στην ***namei_vfat.c*** και βλέπω ότι δεν υπάρχουν αλλαγές.

□ INODE

Η δομή inode ορίζεται στο **fs/fat/fat.h** στη δομή **msdos_inode_info**.

```
129 struct msdos_inode_info {
130     spinlock_t cache_lru_lock;
131     struct list_head cache_lru;
132     int nr_caches;
133     /* for avoiding the race between fat_free() and fat_get_cluster() */
134     unsigned int cache_valid_id;
135
136     /* NOTE: mmu_private is 64bits, so must hold ->i_mutex to access */
137     loff_t mmu_private; /* physically allocated size */
138
139     int i_start;      /* first cluster or 0 */
140     int i_logstart;   /* logical first cluster */
141     int i_attrs;      /* unused attribute bits */
142     loff_t i_pos;     /* on-disk position of directory entry or 0 */
143     struct hlist_node i_fat_hash; /* hash by i_location */
144     struct hlist_node i_dir_hash; /* hash by i_logstart */
145     struct rw_semaphore truncate_lock; /* protect bmap against truncate */
146     struct timespec64 i_ctime; /* File creation (birth) time */
147     struct inode vfs_inode;
148 };

```

Όπως νωρίτερα, θα χρησιμοποιήσουμε την εντολή **grep -r** για να κάνουμε αναζήτηση.

Παρακάτω έχουμε το screenshot από την εκτέλεση της εντολής **grep -r msdos_inode_info**.

```
myy601@myy601lab2:~/lkl-source$ grep -r msdos_inode_info
grep: vmlinux: binary file matches
grep: vmlinux.o: binary file matches
grep: tools/lkl/tests/disk: binary file matches
grep: tools/lkl/tests/boot: binary file matches
grep: tools/lkl/tests/net-test: binary file matches
grep: tools/lkl/tests/disk-vfio-pci: binary file matches
grep: tools/lkl/tests/test-dlmopen: binary file matches
grep: tools/lkl/liblkl.a: binary file matches
grep: tools/lkl/cptofs: binary file matches
grep: tools/lkl/lib/liblkl.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-zpoline.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-hijack.so: binary file matches
grep: tools/lkl/lib/lkl.o: binary file matches
grep: tools/lkl/fs2tar: binary file matches
grep: fs/fat/misc.o: binary file matches
grep: fs/fat/nfs.o: binary file matches
grep: fs/fat/inode.o: binary file matches
grep: fs/fat/cache.o: binary file matches
fs/fat/fat.h:struct msdos_inode_info {
fs/fat/fat.h:static inline struct msdos_inode_info *MSDOS_I(struct inode *inode)
fs/fat/fat.h:    return container_of(inode, struct msdos_inode_info, vfs_inode);
grep: fs/fat/dir.o: binary file matches
grep: fs/fat/file.o: binary file matches
grep: fs/fat/namei_vfat.o: binary file matches
fs/fat/nfs.c:    struct msdos_inode_info *i;
fs/fat/cache.c:    struct msdos_inode_info *i = MSDOS_I(inode);
fs/fat/inode.c:    struct msdos_inode_info *i;
fs/fat/inode.c:    struct msdos_inode_info *ei;
fs/fat/inode.c:    struct msdos_inode_info *ei = (struct msdos_inode_info *)foo;
fs/fat/inode.c:                                sizeof(struct msdos_inode_info),
grep: lkl.o: binary file matches
```

Μας κατευθύνει στις **fat.h**, **nfs.c**, **cache.c** και **inode.c**.

- ❖ Αρχικά ψάχνουμε με Ctrl+F στην **nfs.c** μεταβλητές **msdos_inode_info** και παρατηρούμε ότι δεν υπάρχουν αλλαγές.
- ❖ Στη συνέχεια ψάχνουμε με Ctrl+F μεταβλητές **msdos_inode_info** στην **cache.c** και βλέπουμε ότι υπάρχουν αλλαγές στην **_fat_cache_inval_inode** στις μεταβλητές **nr_caches** και **cache_valid_id**. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

187 static void __fat_cache_inval_inode(struct inode *inode)
188 {
189     struct msdos_inode_info *i = MSDOS_I(inode);
190     struct fat_cache *cache;
191     struct super_block *sb = inode->i_sb; //to prosthew gia na mporw na kalesw to MSDOS_SB(sb)
192     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthew gia na mporw na kalesw to sbi->myJournal
193     char *journal_buffer; //dhlwsh buffer tou journal
194     int file_descriptor;
195
196     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
197     if (!journal_buffer) {
198         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
199         return;
200     }
201
202     while (!list_empty(&i->cache_lru)) {
203         cache = list_entry(i->cache_lru.next,
204                             struct fat_cache, cache_list);
205         list_del_init(&cache->cache_list);
206         i->nr_caches--;
207         //ALLAGH
208         printk(KERN_INFO "-----\n");
209         printk(KERN_INFO "-----We are in __fat_cache_inval_inode of cache.c.-----\n");
210         file_descriptor = sbi->myJournal;
211         sprintf(journal_buffer, "\n-----We are in __fat_cache_inval_inode of cache.c.-----\n %u\n",
212                 i->nr_caches);
213         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
214         //ALLAGH
215         fat_cache_free(cache);
216     }
217     /* Update. The copy of caches before this id is discarded. */
218     i->cache_valid_id++;
219     if (i->cache_valid_id == FAT_CACHE_VALID)
220         i->cache_valid_id++;
221     //ALLAGH
222     printk(KERN_INFO "-----\n");
223     printk(KERN_INFO "-----We are in __fat_cache_inval_inode of cache.c.-----\n");
224     file_descriptor = sbi->myJournal;
225     sprintf(journal_buffer, "%u\n", i->cache_valid_id);
226     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
227     //ALLAGH
228
229     kfree(journal_buffer);
230 }

```

- ❖ Τέλος ψάχνουμε με Ctrl+F μεταβλητές msdos_inode_info στην inode.c και βλέπουμε ότι υπάρχουν αλλαγές στη fat_alloc_inode στις μεταβλητές mmu_private, i_start, i_logstart, i_attrs, i_pos, i_ctime.tv_sec, i_ctime.tv_nsec. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

906 static struct inode *fat_alloc_inode(struct super_block *sb)
907 {
908     struct msdos_inode_info *ei;
909     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthew gia na mporw na kalesw to sbi->myJournal
910     char *journal_buffer; //dhlwsh buffer tou journal
911     int file_descriptor;
912
913     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
914     if (!journal_buffer) {
915         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
916         return NULL;
917     }
918     printk(KERN_INFO "* ~ ENTERING fat_alloc_inode (inode.c/struct super_operations fat_sops) ~ *");
919     ei = alloc_inode_sb(sb, fat_inode_cachep, GFP_NOFS);
920     if (!ei)
921         return NULL;
922     init_rwsem(&ei->truncate_lock);
923     /* Zeroing to allow iput() even if partial initialized inode. */
924     ei->mmu_private = 0;
925     ei->i_start = 0;
926     //ALLAGH
927     printk(KERN_INFO "-----\n");
928     printk(KERN_INFO "-----We are in fat_alloc_inode of inode.c.-----\n");
929     file_descriptor = sbi->myJournal;
930     sprintf(journal_buffer, "\n-----We are in fat_alloc_inode of inode.c.-----\n ei->mmu_private Changed: %lu\n ei->i_start Changed: %u\n",
931             ei->mmu_private, ei->i_start);
932     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
933     //ALLAGH
934
935     ei->i_logstart = 0;
936     ei->i_attrs = 0;
937     //ALLAGH
938     printk(KERN_INFO "-----\n");
939     printk(KERN_INFO "-----We are in fat_alloc_inode of inode.c.-----\n");
940     file_descriptor = sbi->myJournal;
941     sprintf(journal_buffer, " ei->i_logstart Changed: %u\n ei->i_attrs Changed: %u\n",
942             ei->i_logstart, ei->i_attrs);
943     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
944     //ALLAGH

```

```

947     ei->i_pos = 0;
948     ei->i_crttime.tv_sec = 0;
949     ei->i_crttime.tv_nsec = 0;
950     //-----ALLAGH-----
951     printk(KERN_INFO "-----\n");
952     printk(KERN_INFO "-----We are in fat_alloc_inode of inode.c.-----\n");
953     file_descriptor = sbi->myJournal;
954     if (journal_buffer, "ei->i_pos Changed: %llu\n ei->i_crttime.tv_sec Changed: %lu\n ei->i_crttime.tv_nsec Changed: %ld\n",
955         ei->i_pos, ei->i_crttime.tv_sec, ei->i_crttime.tv_nsec);
956     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
957     //-----
958     kfree(journal_buffer);
959     return &ei->vfs_inode;
960 }

```

□ DIRECTORY ENTRIES

Για τη δομή directory entries παρατηρήσαμε πως οι διαφάνειες του μαθήματος αναφέρουν ότι ορίζεται στο **fs/fat/fat.h** στη δομή **msdos_slot_info**. Το αναζητήσαμε στο αρχείο fat.h, όμως δεν το βρήκαμε κάπου, οπότε σκεφτήκαμε ότι πρόκειται για τη δομή **fat_slot_info**.

```

150 struct fat_slot_info {
151     loff_t i_pos;          /* on-disk position of directory entry */
152     loff_t slot_off;       /* offset for slot or de start */
153     int nr_slots;          /* number of slots + 1(de) in filename */
154     struct msdos_dir_entry *de;
155     struct buffer_head *bh;
156 };

```

Όπως νωρίτερα, θα χρησιμοποιήσουμε την εντολή **grep -r** για να κάνουμε αναζήτηση.

Παρακάτω έχουμε το screenshot από την εκτέλεση της εντολής **grep -r fat_slot_info**.

```

myy601@myy601lab2:~/lkl-source$ grep -r fat_slot_info
grep: vmlinux: binary file matches
grep: vmlinux.o: binary file matches
grep: tools/lkl/tests/disk: binary file matches
grep: tools/lkl/tests/boot: binary file matches
grep: tools/lkl/tests/net-test: binary file matches
grep: tools/lkl/tests/disk-vfio-pci: binary file matches
grep: tools/lkl/tests/test-dlmopen: binary file matches
grep: tools/lkl/liblkl.a: binary file matches
grep: tools/lkl/cptofs: binary file matches
grep: tools/lkl/lib/liblkl.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-zpooline.so: binary file matches
grep: tools/lkl/lib/hijack/liblkl-hijack.so: binary file matches
grep: tools/lkl/lib/lkl.o: binary file matches
grep: tools/lkl/fs2tar: binary file matches
grep: fs/fat/nfs.o: binary file matches
fs/fat/dir.c:           int name_len, struct fat_slot_info *sinfo)
fs/fat/dir.c: * The ".." entry can not provide the "struct fat_slot_info" information
fs/fat/dir.c:           struct fat_slot_info *sinfo)
fs/fat/dir.c:           struct fat_slot_info *sinfo)
fs/fat/dir.c: int fat_remove_entries(struct inode *dir, struct fat_slot_info *sinfo)
fs/fat/dir.c:           struct fat_slot_info *sinfo)
fs/fat/fat.h:struct fat_slot_info {
fs/fat/fat.h:           int name_len, struct fat_slot_info *sinfo);
fs/fat/fat.h:           struct fat_slot_info *sinfo);
fs/fat/fat.h:           struct fat_slot_info *sinfo);
fs/fat/fat.h:           struct fat_slot_info *sinfo);
fs/fat/fat.h:extern int fat_remove_entries(struct inode *dir, struct fat_slot_info *sinfo);
fs/fat/namei_msdos.c:   struct fat_slot_info sinfo;
fs/fat/namei_msdos.c:   struct timespec64 *ts, struct fat_slot_info *sinfo)
fs/fat/namei_msdos.c:   struct fat_slot_info sinfo;
fs/fat/namei_msdos.c:   struct fat_slot_info sinfo;
fs/fat/namei_msdos.c:   struct fat_slot_info sinfo;
fs/fat/namei_msdos.c:   struct fat_slot_info old_sinfo, sinfo;

grep: fs/fat/dir.o: binary file matches
grep: fs/fat/namei_vfat.o: binary file matches
fs/fat/nfs.c:   struct fat_slot_info sinfo;
fs/fat/namei_vfat.c:   struct fat_slot_info sinfo;
fs/fat/namei_vfat.c:           struct fat_slot_info *sinfo)
fs/fat/namei_vfat.c:           struct fat_slot_info *sinfo)
fs/fat/namei_vfat.c:   struct fat_slot_info sinfo;
fs/fat/namei_vfat.c:   struct fat_slot_info old_sinfo, sinfo;
grep: lkl.o: binary file matches

```

Μας κατευθύνει στις ***dir.c***, ***fat.h***, ***namei_msdos.c***, ***nfs.c*** και ***namei_vfat.c***.

- ❖ Αρχικά ψάχνουμε με Ctrl+F στην ***dir.c*** μεταβλητές ***fat_slot_info*** και παρατηρούμε ότι υπάρχουν αλλαγές σε κάποιες συναρτήσεις:

→ Στη ***fat_search_long*** στις μεταβλητές ***slot_off***, ***nr_slots***, ***de***, ***bh***, ***i_pos***. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

534     found:
535         nr_slots++; /* include the de */
536         sinfo->slot_off = cpos - nr_slots * sizeof(*de);
537         sinfo->nr_slots = nr_slots;
538         //~~~~~ALLAGH~~~~~
539         printk(KERN_INFO "-----\n");
540         printk(KERN_INFO "-----We are in fat_search_long of dir.c.-----\n");
541         file_descriptor = sbi->myJournal;
542         sprintf(journal_buffer, "\n-----We are in fat_search_long of dir.c.-----\n %llu\n %u\n",
543                 sinfo->slot_off, sinfo->nr_slots);
544         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
545         //~~~~~ALLAGH~~~~~
546         sinfo->de = de;
547         sinfo->bh = bh;
548         sinfo->i_pos = fat_make_i_pos(sb, sinfo->bh, sinfo->de);
549         //~~~~~ALLAGH~~~~~
550         printk(KERN_INFO "-----\n");
551         printk(KERN_INFO "-----We are in fat_search_long of dir.c.-----\n");
552         file_descriptor = sbi->myJournal;
553         sprintf(journal_buffer, "%u\n %u\n %u\n",
554                 sinfo->de, sinfo->bh, sinfo->i_pos);
555         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
556         //~~~~~ALLAGH~~~~~
557         err = 0;
558     end_of_dir:
559         if (unicode)
560             __putname(unicode);
561         kfree(journal_buffer);
562         return err;
563 }

```

→ Στη ***fat_scan*** στις μεταβλητές ***slot_off***, ***bh***, ***nr_slots*** και ***i_pos***. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

991 int fat_scan(struct inode *dir, const unsigned char *name,
992             struct fat_slot_info *sinfo)
993 {
994     struct super_block *sb = dir->i_sb;
995     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthetw gia na mpow na kalesw to sbi->myJournal
996     char *journal_buffer; //dhlwsh buffer tou journal
997     int file_descriptor;
998
999     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
1000     if (!journal_buffer) {
1001         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
1002         return -ENOMEM;
1003     }
1004     sinfo->slot_off = 0;
1005     sinfo->bh = NULL;
1006     //~~~~~ALLAGH~~~~~
1007     printk(KERN_INFO "-----\n");
1008     printk(KERN_INFO "-----We are in fat_scan of dir.c.-----\n");
1009     file_descriptor = sbi->myJournal;
1010     sprintf(journal_buffer, "\n-----We are in fat_scan of dir.c.-----\n %llu\n %u\n",
1011             sinfo->slot_off, sinfo->bh);
1012     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1013
1014     while (fat_get_short_entry(dir, &sinfo->slot_off, &sinfo->bh,
1015                                &sinfo->de) >= 0) {
1016         if (!strcmp(sinfo->de->name, name, MSDOS_NAME)) {
1017             sinfo->slot_off -= sizeof(*sinfo->de);
1018             sinfo->nr_slots = 1;
1019             sinfo->i_pos = fat_make_i_pos(sb, sinfo->bh, sinfo->de);
1020             //~~~~~ALLAGH~~~~~
1021             printk(KERN_INFO "-----\n");
1022             printk(KERN_INFO "-----We are in fat_scan of dir.c.-----\n");
1023             file_descriptor = sbi->myJournal;
1024             sprintf(journal_buffer, "%u\n %u\n %u\n",
1025                     sinfo->slot_off, sinfo->nr_slots, sinfo->i_pos);
1026             write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1027             //~~~~~ALLAGH~~~~~
1028             kfree(journal_buffer);
1029             return 0;
1030         }
1031     }
1032     kfree(journal_buffer);
1033     return -ENOENT;
1034 }

```

→ Στη `fat_scan_logstart` στις μεταβλητές `slot_off`, `bh`, `nr_slots` και `i_pos`. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

1041 int fat_scan_logstart(struct inode *dir, int i_logstart,
1042                      struct fat_slot_info *sinfo)
1043 {
1044     struct super_block *sb = dir->i_sb;
1045     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthetw gia na mporw na kalesw to sbi->myJournal
1046     char *journal_buffer; //dhlwsh buffer tou journal
1047     int file_descriptor;
1048
1049     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
1050     if (!journal_buffer) {
1051         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
1052         return -ENOMEM;
1053     }
1054     sinfo->slot_off = 0;
1055     sinfo->bh = NULL;
1056     ////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
1057     printk(KERN_INFO "-----\n");
1058     printk(KERN_INFO "-----We are in fat_scan_logstart of dir.c.-----\n");
1059     file_descriptor = sbi->myJournal;
1060     sprintf(journal_buffer, "-----We are in fat scan_logstart of dir.c.\n sinfo->slot_off Changed: %llu\n sinfo->bh Changed: %px\n",
1061             sinfo->slot_off, sinfo->bh);
1062     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1063     //////////////////////////////////////////////////
1064     while (fat_get_short_entry(dir, &sinfo->slot_off, &sinfo->bh,
1065                               &sinfo->de) >= 0) {
1066         if (fat_get_start(MSDOS_SB(sb), sinfo->de) == i_logstart) {
1067             sinfo->slot_off -= sizeof(*sinfo->de);
1068             sinfo->nr_slots = 1;
1069             sinfo->i_pos = fat_make_i_pos(sb, sinfo->bh, sinfo->de);
1070             ////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
1071             printk(KERN_INFO "-----\n");
1072             printk(KERN_INFO "-----We are in fat_scan_logstart of dir.c.-----\n");
1073             file_descriptor = sbi->myJournal;
1074             sprintf(journal_buffer, " sinfo->slot_off Changed: %llu\n sinfo->nr_slots Changed: %u\n sinfo->i_pos Changed: %llu\n",
1075                     sinfo->slot_off, sinfo->nr_slots, sinfo->i_pos);
1076             write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1077             //////////////////////////////////////////////////
1078             kfree(journal_buffer);
1079             return 0;
1080         }
1081     }
1082     kfree(journal_buffer);
1083     return -ENOENT;
1084 }
```

→ Στη `fat_remove_entries` στις μεταβλητές `de` και `bh`. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

1136 int fat_remove_entries(struct inode *dir, struct fat_slot_info *sinfo)
1137 {
1138     struct super_block *sb = dir->i_sb;
1139     struct msdos_dir_entry *de;
1140     struct buffer_head *bh;
1141     int err = 0, nr_slots;
1142     char *journal_buffer; //dhlwsh buffer tou journal
1143     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthetw gia na mporw na kalesw to sbi->myJournal
1144     int file_descriptor;
1145     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
1146     if (!journal_buffer) {
1147         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
1148         return -ENOMEM;
1149     }
1150     /*
1151      * First stage: Remove the shortname. By this, the directory
1152      * entry is removed.
1153      */
1154     nr_slots = sinfo->nr_slots;
1155     de = sinfo->de;
1156     sinfo->de = NULL;
1157     ////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
1158     printk(KERN_INFO "-----\n");
1159     printk(KERN_INFO "-----We are in fat_remove_entries of dir.c.-----\n");
1160     file_descriptor = sbi->myJournal;
1161     sprintf(journal_buffer, "-----We are in fat_remove_entries of dir.c.\n sinfo->de Changed: %px\n",
1162             sinfo->de);
1163     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1164     //////////////////////////////////////////////////
1165     bh = sinfo->bh;
1166     sinfo->bh = NULL;
1167     ////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
1168     printk(KERN_INFO "-----\n");
1169     printk(KERN_INFO "-----We are in fat_remove_entries of dir.c.-----\n");
1170     file_descriptor = sbi->myJournal;
1171     sprintf(journal_buffer, " sinfo->bh Changed: %px\n", sinfo->bh);
1172     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1173     //////////////////////////////////////////////////
```

→ Στη `fat_add_entries` στις μεταβλητές `nr_slots`, `slot_off`, `bh`, `de`, `bh` και `i_pos`.

Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

1412 int fat_add_entries(struct inode *dir, void *slots, int nr_slots,
1413     struct fat_slot_info *sinfo)
1414 {
1415     struct super_block *sb = dir->i_sb;
1416     struct msdos_sb_info *sbi = MSDOS_SB(sb);
1417     struct buffer_head *bh, *prev, *bns[3]; /* 32*slots (672bytes) */
1418     struct msdos_dir_entry *de;
1419     int err, free_slots, i, nr_bhs;
1420     char *journal_buffer; //dhlwsh buffer tou journal
1421     int file_descriptor;
1422     loff_t pos, i_pos;
1423
1424     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
1425     if (!journal_buffer) {
1426         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
1427         return -ENOMEM;
1428     }
1429     sinfo->nr_slots = nr_slots;
1430     ////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
1431     printk(KERN_INFO "-----\n");
1432     printk(KERN_INFO "-----We are in fat_add_entries of dir.c.-----\n");
1433     file_descriptor = sbi->myJournal;
1434     sprintf(journal_buffer, "\n-----We are in fat_add_entries of dir.c.-----\n sinfo->nr_slots Changed: %u\n",
1435             sinfo->nr_slots);
1436     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1437     //////////////////////////////////////////////////

```

```

1541     sinfo->slot_off = pos;
1542     sinfo->de = de;
1543     ////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
1544     printk(KERN_INFO "-----\n");
1545     printk(KERN_INFO "-----We are in fat_add_entries of dir.c.-----\n");
1546     file_descriptor = sbi->myJournal;
1547     sprintf(journal_buffer, " sinfo->slot_off Changed: %llu\n sinfo->de Changed: %px\n",
1548             sinfo->slot_off, sinfo->de);
1549     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1550     //////////////////////////////////////////////////
1551     sinfo->bh = bh;
1552     sinfo->i_pos = fat_make_i_pos(sb, sinfo->bh, sinfo->de);
1553     ////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
1554     printk(KERN_INFO "-----\n");
1555     printk(KERN_INFO "-----We are in fat_add_entries of dir.c.-----\n");
1556     file_descriptor = sbi->myJournal;
1557     sprintf(journal_buffer, " sinfo->bh Changed: %px\n sinfo->i_pos Changed: %llu\n",
1558             sinfo->bh, sinfo->i_pos);
1559     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1560     //////////////////////////////////////////////////
1561     kfree(journal_buffer);
1562     return 0;

```

❖ Στη συνέχεια ψάχνουμε με Ctrl+F μεταβλητές `fat_slot_info` στην `namei_msdos.c` και βλέπουμε ότι υπάρχουν αλλαγές στην `do_msdos_rename` στις μεταβλητές `old_sinfo.bh` και `sinfo.bh`. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

497 static int do_msdos_rename(struct inode *old_dir, unsigned char *old_name,
498     struct dentry *old_dentry,
499     struct inode *new_dir, unsigned char *new_name,
500     struct dentry *new_dentry, int is_hid)
501 {
502     struct buffer_head *dotdot_bh;
503     struct msdos_dir_entry *dotdot_de;
504     struct inode *old_inode, *new_inode;
505     struct fat_slot_info old_sinfo, sinfo;
506     struct timespec64 ts;
507     loff_t new_i_pos;
508     int err, oldAttrs, is_dir, update_dotdot, corrupt = 0;
509     char *journal_buffer; //dhlwsh buffer tou journal
510     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthetai gia na mporw na kalesw to sbi->myJournal
511     int file_descriptor;
512     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
513     if (!journal_buffer) {
514         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
515         return;
516     }
517     old_sinfo.bh = sinfo.bh = dotdot_bh = NULL;
518     ////////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
519     printk(KERN_INFO "-----\n");
520     printk(KERN_INFO "-----We are in do_msdos_rename of namei_msdos.c.-----\n");
521     file_descriptor = sbi->myJournal;
522     sprintf(journal_buffer, "\n-----We are in do_msdos_rename of namei_msdos.c.-----\n old_sinfo.bh Changed: %u\n",
523             old_sinfo.bh);
524     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
525     //////////////////////////////////////////////////

```

```

671     sinfo.bh = NULL;
672     //~~~~~ALLAGH~~~~~\n";
673     printk(KERN_INFO "-----\n");
674     printk(KERN_INFO "-----We are in do_msdos_rename of namei_msdos.c.-----\n");
675     file_descriptor = sbi->myJournal;
676     sprintf(journal_buffer, "\n-----We are in do_msdos_rename of namei_msdos.c.-----\n sinfo.bh Changed: %u\n",
677             sinfo.bh);
678     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
679     //~~~~~ALLAGH~~~~~\n";

```

- ❖ Έπειτα ψάχνουμε με Ctrl+F μεταβλητές **fat_slot_info** στην **nfs.c** και βλέπουμε ότι δεν υπάρχουν αλλαγές.
- ❖ Τέλος ψάχνουμε με Ctrl+F μεταβλητές **fat_slot_info** στην **namei_vfat.c** και βλέπουμε ότι υπάρχουν αλλαγές στην **vfat_rename** στις μεταβλητές **old_sinfo.bh** και **sinfo.bh**. Προσθέτουμε κώδικα για να εγγράφουμε τις αλλαγές στο journal.

```

1016 static int vfat_rename(struct inode *old_dir, struct dentry *old_dentry,
1017                      struct inode *new_dir, struct dentry *new_dentry)
1018 {
1019     struct buffer_head *dotdot_bh;
1020     struct msdos_dir_entry *dotdot_de = NULL;
1021     struct inode *old_inode, *new_inode;
1022     struct fat_slot_info old_sinfo, sinfo;
1023     struct timespec64 ts;
1024     loff_t new_i_pos;
1025     int err, is_dir, corrupt = 0;
1026     struct super_block *sb = old_dir->i_sb;
1027     char *journal_buffer; //dhlwsh buffer tou journal
1028     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthetw gia na mporw na kalesw to sbi->myJournal
1029     int file_descriptor;
1030
1031     old_sinfo.bh = sinfo.bh = dotdot_bh = NULL;
1032     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
1033     if (!journal_buffer)
1034         return -ENOMEM;
1035
1036     //~~~~~ALLAGH~~~~~\n";
1037     printk(KERN_INFO "-----\n");
1038     printk(KERN_INFO "-----We are in vfat_rename of namei_vfat.c.-----\n");
1039     file_descriptor = sbi->myJournal;
1040     sprintf(journal_buffer, "\n-----We are in vfat_rename of namei_vfat.c.-----\n old_sinfo.bh Changed: %px\n",
1041             old_sinfo.bh);
1042     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1043     //~~~~~ALLAGH~~~~~\n";

```

```

1128 } else {
1129     /*
1130     * If new entry was not sharing the data cluster, it
1131     * shouldn't be serious corruption.
1132     */
1133     int err2 = fat_remove_entries(new_dir, &sinfo);
1134     if (corrupt)
1135         corrupt |= err2;
1136     sinfo.bh = NULL;
1137     //~~~~~ALLAGH~~~~~\n";
1138     printk(KERN_INFO "-----\n");
1139     printk(KERN_INFO "-----We are in vfat_rename of namei_vfat.c.-----\n");
1140     file_descriptor = sbi->myJournal;
1141     sprintf(journal_buffer, "\n-----We are in vfat_rename of namei_vfat.c.-----\n sinfo.bh Changed: %px\n", sinfo.bh);
1142     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1143     //~~~~~ALLAGH~~~~~\n";
1144 }
1145 if (corrupt < 0) {
1146     fat_fs_error(new_dir->i_sb,
1147                  "%s: Filesystem corrupted (i_pos %lld)",
1148                  __func__, sinfo.i_pos);
1149     goto out;
1150 }

```

□ FILES

Η δομή files ορίζεται στο **include/linux/fs.h** στη δομή **file**.

```
940 struct file {
941     union {
942         struct llist_node    f_llist;
943         struct rcu_head      f_rcuhead;
944         unsigned int          f_iocb_flags;
945     };
946     struct path        f_path;
947     struct inode       *f_inode; /* cached value */
948     const struct file_operations *f_op;
949
950     /*
951      * Protects f_ep, f_flags.
952      * Must not be taken from IRQ context.
953      */
954     spinlock_t        f_lock;
955     atomic_long_t     f_count;
956     unsigned int      f_flags;
957     fmode_t           f_mode;
958     struct mutex      f_pos_lock;
959     loff_t            f_pos;
960     struct fown_struct f_owner;
961     const struct cred *f_cred;
962     struct file_ra_state f_ra;
963
964     u64               f_version;
965 #ifdef CONFIG_SECURITY
966     void              *f_security;
967 #endif
968     /* needed for tty driver, and maybe others */
969     void              *private_data;
970
971 #ifdef CONFIG_EPOLL
972     /* Used by fs/eventpoll.c to link all the hooks to this file */
973     struct hlist_head *f_ep;
974 #endif /* ifdef CONFIG_EPOLL */
975     struct address_space *f_mapping;
976     errseq_t           f_wb_err;
977     errseq_t           f_sb_err; /* for syncfs */
978 } __randomize_layout
979 __attribute__((aligned(4))); /* lest something weird decides that 2 is OK */
980
981 struct file_handle {
982     __u32 handle_bytes;
983     int handle_type;
984     /* file identifier */
985     unsigned char f_handle[];
986 }
```

Θα χρησιμοποιήσουμε την εντολή **grep -r** για να κάνουμε αναζήτηση στο fs/fat.

Παρακάτω έχουμε κομμάτια από screenshot από την εκτέλεση της εντολής **grep -r file**.

```
my601@myy601lab2:~/lkl-source/fs/fat$ grep -r file
Makefile:# Makefile for the Linux fat filesystem support.
Makefile:fat-y := cache.o dir.o fatent.o file.o inode.o misc.o nfs.o
grep: fatent.o: binary file matches
grep: misc.o: binary file matches
fatent.c:      int file_descriptor;
fatent.c:      file_descriptor = sbi->myJournal;
fatent.c:      write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);

.file.o.o.cmd:source_fs/fat/file.o := fs/fat/file.c
.file.o.o.cmd:deps_fs/fat/file.o := \
.file.o.o.cmd:  include/linux/kernel_read_file.h \
.file.o.o.cmd:  include/linux/file.h \
.file.o.o.cmd:fs/fat/file.o: $(deps_fs/fat/file.o)
.file.o.o.cmd:$(deps_fs/fat/file.o):
grep: nfs.o: binary file matches
dir.c: * directory handling functions for fat-based filesystems
dir.c: * Hidden files 1995 by Albert Cahalan <albert@ccs.neu.edu> <adc@coe.neu.edu>
dir.c: * filesystem. The following four characters are the hexadecimal digits

grep: inode.o: binary file matches
Kconfig:           If you want to use one of the FAT-based file systems (the MS-DOS and
Kconfig:           VFAT (Windows 95) file systems), then you must say Y or M here

grep: cache.o: binary file matches
fat.h:ssize_t write(int file_descriptor, const void *buf, size_t length);
fat.h:static void write_to_journal(int file_descriptor, char buf[2048],size_t length) {
fat.h:    write(file_descriptor, buf, length);
```

```

misc.c:         file_descriptor = sbi->myJournal;
misc.c:         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
misc.c: * We must locate the last cluster of the file to add this new
misc.c:           * Since generic write_sync() synchronizes regular files later,
namei_msdos.c: * Hidden files 1995 by Albert Cahalan <albert@ccs.neu.edu> <adc@coe.neu.edu>
namei_msdos.c:/* Characters that are undesirable in an MS-DOS file name */
namei_msdos.c:***** Formats an MS-DOS file name. Rejects invalid names. */

```

```

grep: dir.o: binary file matches
grep: file.o: binary file matches
.built-in.a.cmd:cmd_fs/fat/built-in.a := rm -f fs/fat/built-in.a; printf "fs/fat/%s "
file.c: * linux/fs/fat/file.c
file.c: * regular file handling primitives for fat-based filesystems
file.c:static long fat_fallocate(struct file *file, int mode,
file.c:static int fat_ioctl_set_attributes(struct file *file, u32 __user *user_attr)

```

```

grep: namei_vfat.o: binary file matches
nfs.c:           /* If a file is deleted on server and client is not updated
nfs.c: * Map a NFS file handle to a corresponding dentry.
nfs.c: * The dentry may or may not be connected to the filesystem root.
nfs.c: * Find the parent for a file specified by NFS handle.
nfs.c: * to the filesystem root.
nfs.c: * the filesystem root.
fat_test.c: * KUnit tests for FAT filesystems.
fat_test.c:     int file_descriptor;

```

```

fat_test.c:     int file_descriptor;
fat_test.c:     file_descriptor = sbi->myJournal;
fat_test.c:     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
cache.c:       int fcluster; /* cluster number in the file. */
cache.c:       int file_descriptor;

```

```

inode.c:#include <linux/seq_file.h>
inode.c:#include <linux/file.h>
inode.c:int myJournal; // arxikopoish tou journal file descriptor

```

```

namei_vfat.c:   file_descriptor = sbi->myJournal;
namei_vfat.c:   write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
namei_vfat.c:   file_descriptor = sbi->myJournal;
namei_vfat.c:   write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);

```

Μας κατευθύνει στις ***fatent.c***, ***dir.c***, ***fat.h***, ***misc.c***, ***namei_msdos.c***, ***file.c***, ***nfs.c***, ***fat_test.c***, ***cache.c***, ***inode.c*** και ***namei_vfat.c***.

- ❖ Αρχικά ψάχνουμε με Ctrl+F στην ***fatent.c*** μεταβλητές ***file*** και παρατηρούμε ότι δεν υπάρχουν αλλαγές.
- ❖ Έπειτα ψάχνουμε με Ctrl+F στην ***dir.c*** μεταβλητές ***file*** και παρατηρούμε ότι υπάρχουν αλλαγές στη ***fat_ioctl_readdir*** στη μεταβλητή ***f_pos***. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

785 static int fat_ioctl_readdir(struct inode *inode, struct file *file,
786                             void __user *dirent, filldir_t filldir,
787                             int short_only, int both)
788 {
789     struct fat_ioctl_filldir_callback buf = {
790         .ctx.actor = filldir,
791         .dirent = dirent
792     };
793     int ret;
794     char *journal_buffer; //dhlwsh buffer tou journal
795     struct super_block *sb = inode->i_sb; //to prosthew gia na mporw na kalesw to MSDOS_SB(sb)
796     struct msdos_sb_info *sbi = MSDOS_SB(sb); //to prosthew gia na mporw na kalesw to sbi->myJournal
797     int file_descriptor;
798     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
799     if (!journal_buffer) {
800         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
801         return -ENOMEM;
802     }
803     buf.dirent = dirent;
804     buf.result = 0;
805     inode_lock_shared(inode);
806     buf.ctx.pos = file->f_pos;
807     ret = -ENOENT;
808     if (!IS_DEADDIR(inode)) {
809         ret = __fat_readdir(inode, file, &buf.ctx,
810                             short_only, both ? &buf : NULL);
811         file->f_pos = buf.ctx.pos;
812         //-----ALLAGH-----
813         printk(KERN_INFO "-----\n");
814         printk(KERN_INFO "-----We are in fat_ioctl_readdir of dir.c-----\n");
815         file_descriptor = sbi->myJournal;
816         sprintf(journal_buffer, "\n-----We are in fat_ioctl_readdir of dir.c-----\n %llu\n", file->f_pos Changed: %llu\n");
817         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
818     }

```

- ❖ Στη συνέχεια ψάχνουμε με Ctrl+F στις `misc.c`, `namei_msdos.c`, `file.c`, `nfs.c`, `fat_test.c`, `cache.c`, `inode.c` και `namei_vfat.c` μεταβλητές file και παρατηρούμε ότι δεν υπάρχουν αλλαγές.

□ ΜΠΛΟΚ ΜΕΤΑΔΕΛΟΜΕΝΩΝ – INODES

Τα μεταδεδομένα ενός αρχείου αποθηκεύονται σε μια δομή που ονομάζεται **inode**.

Η δομή **inode** βρίσκεται στο `lklsource/include/linux/fs.h` και έχει την εξής μορφή:

```
struct inode {
    umode_t          i_mode;
    unsigned short   i_opflags;
    kuid_t           i_uid;
    kgid_t           i_gid;
    unsigned int     i_flags;

#ifndef CONFIG_FS_POSIX_ACL
    struct posix_acl *i_acl;
    struct posix_acl *i_default_acl;
#endif

    const struct inode_operations *i_op;
    struct super_block *i_sb;
    struct address_space *i_mapping;

#ifndef CONFIG_SECURITY
    void             *i_security;
#endif

    /* Stat data, not accessed from path walking */
    unsigned long     i_ino;
    /*

     * Filesystems may only read i_nlink directly. They shall use the
     * following functions for modification:
     *
     *      (set|clear|inc|drop) nlink
     *      inode_(inc|dec)_link_count
     */
    union {
        const unsigned int i_nlink;
        unsigned int __i_nlink;
    };
    dev_t            i_rdev;
    loff_t           i_size;
    struct timespec64 i_atime;
    struct timespec64 i_mtime;
    struct timespec64 i_ctime;
    spinlock_t       i_lock; /* i_blocks, i_bytes, maybe i_size */
    unsigned short   i_bytes;
    u8               i_blkbits;
    u8               i_write_hint;

    struct file_lock_context *i_flctx;
    struct address_space i_data;
    struct list_head   i_devices;
    union {
        struct pipe_inode_info *i_pipe;
        struct cdev        *i_cdev;
        char              *i_link;
        unsigned          i_dir_seq;
    };

    __u32            i_generation;

#ifndef CONFIG_FSNOTIFY
    __u32            i_fsnotify_mask; /* all events this inode cares about */
    struct fsnotify_mark_connector __rcu *i_fsnotify_marks;
#endif

#ifndef CONFIG_FS_ENCRYPTION
    struct fscrypt_info *i_crypt_info;
#endif

#ifndef CONFIG_FS_VERITY
    struct fsverity_info *i_verity_info;
#endif

    void             *i_private; /* fs or device private pointer */
} __randomize_layout;
```

```
blkcnt_t          i_blocks;

#ifndef _NEED_I_SIZE_ORDERED
    seqcount_t     i_size_seqcount;
#endif

/* Misc */
unsigned long     i_state;
struct rw_semaphore i_rwsem;

unsigned long     dirtied_when; /* jiffies of first dirtying */
unsigned long     dirtied_time_when;

struct hlist_node i_hash;
struct list_head  i_io_list; /* backing dev IO list */

#ifndef CONFIG_CGROUP_WRITEBACK
    struct bdi_writeback *i_wb; /* the associated cgroup wb */
#endif

/* foreign inode detection, see wbc_detach_inode() */
int              i_wb_frn_winner;
u16              i_wb_frn_avg_time;
u16              i_wb_frn_history;

union {
    struct hlist_head i_lru; /* inode LRU list */
    struct list_head  i_sb_list;
    struct list_head  i_wb_list; /* backing dev writeback list */
};

union {
    struct hlist_head i_dentry;
    struct rcu_head   i_rcu;
};

atomic64_t        i_version;
atomic64_t        i_sequence; /* see futex */
atomic_t          i_count;
atomic_t          i_dio_count;
atomic_t          i_writecount;
#ifndef defined(CONFIG_IMA) || defined(CONFIG_FILE_LOCKING)
    atomic_t        i_readcount; /* struct files open RO */
#endif

union {
    const struct file_operations *i_fop; /* former ->i_op->default_file_ops */
    void (*free_inode)(struct inode *);
};
```

Μεταδεδομένα ενός αρχείου είναι:

- Τα χαρακτηριστικά που αφορούν κάθε αρχείο, π.χ. το όνομα, οι δείκτες στα block του δίσκου που βρίσκονται τα δεδομένα, το μέγεθος, τα δικαιώματα πρόσβασης, ο ιδιοκτήτης, ο χρόνος δημιουργίας, ο χρόνος τελευταίας προσπέλασης/τροποποίησης. Επίσης είναι
- Τα χαρακτηριστικά που αφορούν συνολικά το σύστημα αρχείων, π.χ. τύπος συστήματος αρχείων, αριθμός block/ inode, μέγεθος block.

Βλέπουμε λοιπόν ότι η δομή inode αντιπροσωπεύει μια εγγραφή inode σε ένα σύστημα αρχείων του Linux. Περιέχει όλα τα απαραίτητα δεδομένα για να διαχειριστεί τις ιδιότητες και τις λειτουργίες ενός αρχείου ή καταλόγου στο σύστημα αρχείων. Θα ξεχωρίσουμε κάποια πολύ σημαντικά πεδία με τα χαρακτηριστικά που αφορούν κάθε αρχείο και συνολικά το σύστημα αρχείων, τα οποία θα ψάξουμε με Ctrl + F μέσα στα αρχεία του **fs/fat**. Όπου παρατηρούμε αλλαγές στις μεταβλητές θα προσθέσουμε κώδικα για να εγγράψουμε τις αλλαγές αυτές στο journal, όπως κάναμε μέχρι τώρα. Θα λάβουμε υπόψιν μας τα πεδία:

1. **i_mode**: Τύπος του αρχείου (π.χ. κανονικό αρχείο, κατάλογος).
2. **i_uid**: To User ID του κατόχου του αρχείου.
3. **i_gid**: To Group ID του κατόχου του αρχείου.
4. **i_flags**: Διάφορες σημαίες που καθορίζουν συμπεριφορές του αρχείου.
5. **i_op**: Δείκτης σε δομή που περιέχει λειτουργίες που μπορούν να γίνουν στο inode.
6. **i_sb**: Δείκτης στο superblock που ανήκει το inode.
7. **i_mapping**: Δείκτης στον address space που συνδέεται με το αρχείο.
8. **i_security**: Δείκτης σε δεδομένα ασφαλείας.
9. **i_ino**: Ο αριθμός του inode (μοναδικός ταυτοποιητής).
10. **i_size**: Μέγεθος του αρχείου σε bytes.
11. **i_atime**: Χρόνος τελευταίας πρόσβασης.
12. **i_mtime**: Χρόνος τελευταίας τροποποίησης.
13. **i_ctime**: Χρόνος τελευταίας αλλαγής στα μεταδεδομένα.
14. **i_blocks**: Αριθμός των δίσκων blocks που καταλαμβάνει το αρχείο.
15. **i_state**: Κατάσταση του inode (π.χ. αν είναι ενεργό).
16. **i_version**: Έκδοση του inode.

Στα αρχεία **cache.c**, **dir.c**, **fat.h**, **fatent.c**, **namei_vfat.c** και **namei_msdos.c** παρατηρούμε ότι δεν υπάρχουν αλλαγές στις μεταβλητές τύπου inode.

Στο αρχείο **inode.c** παρατηρούμε ότι υπάρχουν αλλαγές στις συναρτήσεις:

- ❖ Στη **fat_calc_dir_size** στη μεταβλητή **inode->i_size**.
Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

495 static int fat_calc_dir_size(struct inode *inode)
496 {
497     struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
498     int ret, fcclus, dclus;
499     char *journal_buffer; //dhlwsh buffer tou journal
500     int file_descriptor;
501
502     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
503     if (!journal_buffer) {
504         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
505         return -ENOMEM;
506     }
507     inode->i_size = 0;
508     if (MSDOS_I(inode)->i_start == 0){
509         kfree(journal_buffer);
510         return 0;
511     }
512     ret = fat_get_cluster(inode, FAT_ENT_EOF, &fcclus, &dclus);
513     if (ret < 0){
514         kfree(journal_buffer);
515         return ret;
516     }
517     inode->i_size = (fcclus + 1) << sbi->cluster_bits;
518     //-----ALLAGH-----
519     printk(KERN_INFO "-----\n");
520     printk(KERN_INFO "-----We are in fat_calc_dir_size of inode.c.\n");
521     file_descriptor = sbi->myJournal;
522     sprintf(journal_buffer, "\n-----We are in fat_calc_dir_size of inode.c.\n inode->i_size Changed: %lld\n",
523             (long long) inode->i_size);
524     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
525     //-----
526     kfree(journal_buffer);
527     return 0;
528 }

```

- ❖ Στη **fat_fill_inode** στις μεταβλητές **inode->i_uid**, **inode->i_gid**, **inode->i_mode**, **inode->i_op**, **inode->i_mapping**, **inode->i_flags**, **inode->i_blocks**, **inode->i_ctime**, **inode->i_atime**. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

549 int fat_fill_inode(struct inode *inode, struct msdos_dir_entry *de)
550 {
551     struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
552     int error;
553     char *journal_buffer; //dhlwsh buffer tou journal
554     int file_descriptor;
555
556     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
557     if (!journal_buffer) {
558         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
559         return -ENOMEM;
560     }
561     MSDOS_I(inode)->i_pos = 0;
562     inode->i_uid = sbi->options.fs_uid;
563     inode->i_gid = sbi->options.fs_gid;
564     inode_inc_iversion(inode);
565     inode->i_generation = get_random_u32();
566     //-----ALLAGH-----
567     printk(KERN_INFO "-----\n");
568     printk(KERN_INFO "-----We are in fat_fill_inode of inode.c.\n");
569     file_descriptor = sbi->myJournal;
570     sprintf(journal_buffer, "\n-----We are in fat_fill_inode of inode.c.\n inode->i_uid Changed: %u\n inode->i_gid Changed: %u\n",
571             __uid_val(inode->i_uid), __kgid_val(inode->i_gid));
572     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
573     //-----

```

```

575     if ((de->attr & ATTR_DIR) && !IS_FREE(de->name)) {
576         inode->i_generation += ~1;
577         inode->i_mode = fat_make_mode(sbi, de->attr, S_IRWXUGO);
578         inode->i_op = sbi->dir_ops;
579         inode->i_fop = &fat_dir_operations;
580         //-----ALLAGH-----
581         printk(KERN_INFO "-----\n");
582         printk(KERN_INFO "-----We are in fat_fill_inode of inode.c.\n");
583         file_descriptor = sbi->myJournal;
584         sprintf(journal_buffer, " inode->i_mode Changed: %u\n inode->i_op Changed: %px\n",
585                 inode->i_mode, (void *) inode->i_op);
586         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
587         //-----

```

```

604     } else { /* not a directory */
605         inode->i_generation += 1;
606         inode->i_mode = fat_make_mode(sbi, de->attr,
607             ((sbi->options.showexec && !is_exec(de->name + 8))
608             ? S_IRUGO|S_IWUGO : S_IRWXUGO));
609         MSDOS_I(inode)->i_start = fat_get_start(sbi, de);
610
611         MSDOS_I(inode)->i_logstart = MSDOS_I(inode)->i_start;
612         inode->i_size = le32_to_cpu(de->size);
613         inode->i_op = &fat_file_inode_operations;
614         inode->i_fop = &fat_file_operations;
615         inode->i_mapping->a_ops = &fat_aops;
616         MSDOS_I(inode)->mmu_private = inode->i_size;
617         //-----ALLAGH-----
618         printk(KERN_INFO "-----\n");
619         printk(KERN_INFO "-----We are in fat_fill_inode of inode.c.\n");
620         file_descriptor = sbi->myJournal;
621         sprintf(journal_buffer, " inode->i_size Changed: %lld\n inode->i_op Changed: %px\n inode->i_mapping Changed: %px\n",
622                 inode->i_size, (void *) inode->i_op, (void *) inode->i_mapping);
623         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
624     }

```

```

626     if (de->attr & ATTR_SYS) {
627         if (sbi->options.sys_immutable){
628             inode->i_flags |= S_IMMUTABLE;
629             //////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
630             printk(KERN_INFO "-----\n");
631             printk(KERN_INFO "-----We are in fat_fill_inode of inode.c.\n");
632             file_descriptor = sbi->myJournal;
633             sprintf(journal_buffer, " inode->i_flags Changed: %u\n", inode->i_flags);
634             write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
635             //////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
636         }
637     }
638     fat_save_attrs(inode, de->attr);
639
640     inode->i_blocks = ((inode->i_size + (sbi->cluster_size - 1))
641                         & ~((loff_t)sbi->cluster_size - 1)) >> 9;
642
643     fat_time_fat2unix(sbi, &inode->i_mtime, de->time, de->date, 0);
644     inode->i_ctime = inode->i_mtime;
645     //////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
646     printk(KERN_INFO "-----\n");
647     printk(KERN_INFO "-----We are in fat_fill_inode of inode.c.\n");
648     file_descriptor = sbi->myJournal;
649     sprintf(journal_buffer, " inode->i_blocks Changed: %llu\n inode->i_ctime Changed: %lld\n",
650             inode->i_blocks, (long long)inode->i_ctime.tv_sec);
651     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
652     //////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////

```

```

653     if (sbi->options.isvfat) {
654         fat_time_fat2unix(sbi, &inode->i_atime, 0, de->adate, 0);
655         fat_time_fat2unix(sbi, &MSDOS_I(inode)->i_crtimetime, de->ctime,
656                            de->cdate, de->ctime_cs);
657     } else{
658         inode->i_atime = fat_truncate_atime(sbi, &inode->i_mtime);
659         //////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
660         printk(KERN_INFO "-----\n");
661         printk(KERN_INFO "-----We are in fat_fill_inode of inode.c.\n");
662         file_descriptor = sbi->myJournal;
663         sprintf(journal_buffer, " inode->i_atime Changed: %lld\n", (long long)inode->i_atime.tv_sec);
664         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
665         //////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
666     }
667     kfree(journal_buffer);
668     return 0;
669 }

```

❖ Στη **fat_build_inode** στη μεταβλητή **inode->i_ino**. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

683 struct inode *fat_build_inode(struct super_block *sb,
684                               struct msdos_dir_entry *de, loff_t i_pos)
685 {
686     struct inode *inode;
687     struct msdos_sb_info *sbi; //to prosthetw gia na mporw na kalesw to sbi->myJournal
688     char *journal_buffer; //dhlwsh buffer tou journal
689     int file_descriptor;
690     int err;
691     inode = new_inode(sb);
692
693     if (!inode) {
694         printk(KERN_ERR "Failed to allocate inode\n");
695         return ERR_PTR(-ENOMEM);
696     }
697     sbi = MSDOS_SB(inode->i_sb);
698
699     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
700     if (!journal_buffer) {
701         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
702         return ERR_PTR(-ENOMEM);
703     }

```

```

713     inode->i_ino = iunique(sb, MSDOS_ROOT_INO);
714     //////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
715     printk(KERN_INFO "-----\n");
716     printk(KERN_INFO "-----We are in fat_build_inode of inode.c.\n");
717     file_descriptor = sbi->myJournal;
718     sprintf(journal_buffer, "\n-----We are in fat_build_inode of inode.c.\n\n inode->i_ino Changed: %lu\n",
719             inode->i_ino);
720     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
721     //////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////

```

- ❖ Στη **fat_evict_inode** στη μεταβλητή **inode->i_size**. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

767 static void fat_evict_inode(struct inode *inode)
768 {
769     struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb); //to prosthetw gia na mpow na kalesw to sbi->myJournal
770     char *journal_buffer; //dhlwsh buffer tou journal
771     int file_descriptor;
772
773     printk(KERN_INFO "/* ~ ENTERING fat_evict_inode (inode.c/struct super_operations fat_sops) ~ */");
774     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
775     if (!journal_buffer) {
776         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
777         return;
778     }
779     truncate_inode_pages_final(&inode->i_data);
780     if (!inode->i_nlink) {
781         inode->i_size = 0;
782         //////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
783         printk(KERN_INFO "-----\n");
784         printk(KERN_INFO "-----We are in fat_evict_inode of inode.c.-----\n");
785         file_descriptor = sbi->myJournal;
786         sprintf(journal_buffer, "\n-----We are in fat_evict_inode of inode.c.-----\n inode->i_size Changed: %lld\n",
787                 inode->i_size);
788         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
789         //////////////////////////////////////////////////
790         fat_truncate_blocks(inode, 0);
791     } else
792         fat_free_eofblocks(inode);
793
794     invalidate_inode_buffers(inode);
795     clear_inode(inode);
796     fat_cache_inval_inode(inode);
797     fat_detach(inode);
798     kfree(journal_buffer);
799 }

```

- ❖ Στη **fat_read_root** στις μεταβλητές **inode->i_uid**, **inode->i_gid**, **inode->i_mode**, **inode->i_op**, **inode->i_size**, **inode->i_blocks**, **inode->i_mtime**, **inode->i_atime**, **inode->i_ctime**. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

1613 static int fat_read_root(struct inode *inode)
1614 {
1615     struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
1616     int error;
1617     char *journal_buffer; //dhlwsh buffer tou journal
1618     int file_descriptor;
1619
1620     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
1621     if (!journal_buffer) {
1622         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
1623         return -ENOMEM;
1624     }
1625     MSDOS_I(inode)->i_pos = MSDOS_ROOT_INO;
1626     inode->i_uid = sbi->options.fs_uid;
1627     inode->i_gid = sbi->options.fs_gid;
1628     inode_inc_inversion(inode);
1629     inode->i_generation = 0;
1630     inode->i_mode = fat_make_mode(sbi, ATTR_DIR, S_IRWXUGO);
1631     inode->i_op = sbi->dir_ops;
1632     inode->i_fop = fat_dir_operations;
1633     //////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
1634     printk(KERN_INFO "-----\n");
1635     printk(KERN_INFO "-----We are in fat_read_root of inode.c.-----\n");
1636     file_descriptor = sbi->myJournal;
1637     sprintf(journal_buffer, "\n-----We are in fat_read_root of inode.c.-----\n inode->i_uid Changed: %u\n inode->i_gid Changed: %u\n inode->i_mode Changed: %u\n inode->i_op Changed: %px\n",
1638             __kuid_val(inode->i_uid), __kgid_val(inode->i_gid), inode->i_mode, (void *)inode->i_op);
1639     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1640 }

```

```

1641     if (is_fat32(sbi)) {
1642         MSDOS_I(inode)->i_start = sbi->root_cluster;
1643         error = fat_calc_dir_size(inode);
1644         if (error < 0){
1645             kfree(journal_buffer);
1646             return error;
1647         }
1648     } else {
1649         MSDOS_I(inode)->i_start = 0;
1650         inode->i_size = sbi->dir_entries * sizeof(struct msdos_dir_entry);
1651         //////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
1652         printk(KERN_INFO "-----\n");
1653         printk(KERN_INFO "-----We are in fat12_ent_next of fatent.c.-----\n");
1654         file_descriptor = sbi->myJournal;
1655         sprintf(journal_buffer, " inode->i_size Changed: %lld\n", (long long) inode->i_size);
1656         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1657         //////////////////////////////////////////////////
1658     }
1659     inode->i_blocks = ((inode->i_size + (sbi->cluster_size - 1))
1660                         & ~((loff_t)sbi->cluster_size - 1)) >> 9;
1661     //////////////////////////////////////////////////ALLAGH////////////////////////////////////////////////
1662     printk(KERN_INFO "-----\n");
1663     printk(KERN_INFO "-----We are in fat12_ent_next of fatent.c.-----\n");
1664     file_descriptor = sbi->myJournal;
1665     sprintf(journal_buffer, " inode->i_blocks Changed: %llu\n", inode->i_blocks);
1666     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1667     //////////////////////////////////////////////////
1668     MSDOS_I(inode)->i_logstart = 0;
1669     MSDOS_I(inode)->mmu_private = inode->i_size;

```

```

1672     inode->i_mtime.tv_sec = inode->i_atime.tv_sec = inode->i_ctime.tv_sec = 0;
1673     //-----ALLAGH-----\n";
1674     printk(KERN_INFO "-----We are in fat_read_root of inode.c.\n");
1675     file_descriptor = sbi->myJournal;
1676     sprintf(journal_buffer, "inode->i_mtime.tv_sec Changed: %lld\n inode->i_atime.tv_sec Changed: %lld\n inode->i_ctime.tv_sec Changed: %lld\n",
1677             (long long)inode->i_mtime.tv_sec, (long long)inode->i_atime.tv_sec, (long long)inode->i_ctime.tv_sec);
1678     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1679     //-----\n";
1680     inode->i_mtime.tv_nsec = inode->i_atime.tv_nsec = inode->i_ctime.tv_nsec = 0;
1681     //-----ALLAGH-----\n";
1682     printk(KERN_INFO "-----We are in fat_read_root of inode.c.\n");
1683     file_descriptor = sbi->myJournal;
1684     sprintf(journal_buffer, "inode->i_mtime.tv_nsec Changed: %ld\n inode->i_atime.tv_nsec Changed: %ld\n inode->i_ctime.tv_nsec Changed: %ld\n",
1685             inode->i_mtime.tv_nsec, inode->i_atime.tv_nsec, inode->i_ctime.tv_nsec);
1686     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
1687     //-----\n";
1688     set_nlink(inode, fat_subdirs(inode)+2);
1689
1690     kfree(journal_buffer);
1691
1692     return 0;
1693 }

```

Στο αρχείο **misc.c** παρατηρούμε ότι υπάρχουν αλλαγές στις συναρτήσεις:

- ❖ Στη **fat_chain_add** στη μεταβλητή **inode->i_blocks**.

Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

131     int fat_chain_add(struct inode *inode, int new_dclus, int nr_cluster)
132 {
133     struct super_block *sb = inode->i_sb;
134     struct msdos_sb_info *sbi = MSDOS_SB(sb);
135     int ret, new_fclus, last;
136     char *journal_buffer; //dhlwsh buffer tou journal
137     int file_descriptor;
138
139     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
140     if (!journal_buffer) {
141         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
142         return -ENOMEM;
143     }
144
203     inode->i_blocks += nr_cluster << (sbi->cluster_bits - 9);
204     //-----ALLAGH-----\n";
205     printk(KERN_INFO "-----We are in fat_chain_add of misc.c.\n");
206     file_descriptor = sbi->myJournal;
207     sprintf(journal_buffer, "\n-----We are in fat_chain_add of misc.c.\n inode->i_blocks Changed: %llu\n",
208             inode->i_blocks);
209     write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
210     //-----\n";
211     kfree(journal_buffer);
212     return 0;
213 }

```

- ❖ Στη **fat_truncate_time** στις μεταβλητές **inode->i_atime**, **inode->i_mtime**, **inode->i_ctime**. Προσθέτουμε κώδικα για να εγγράψουμε τις αλλαγές στο journal.

```

359     int fat_truncate_time(struct inode *inode, struct timespec64 *now, int flags)
360 {
361     struct msdos_sb_info *sbi = MSDOS_SB(inode->i_sb);
362     struct timespec64 ts;
363     char *journal_buffer; //dhlwsh buffer tou journal
364     int file_descriptor;
365
366     journal_buffer = kmalloc(JOURNAL_BUFFER_SIZE, GFP_KERNEL);
367     if (!journal_buffer) {
368         printk(KERN_ERR "Failed to allocate memory for journal_buffer\n");
369         return -ENOMEM;
370     }
371     if (inode->i_ino == MSDOS_ROOT_INO){
372         kfree(journal_buffer);
373         return 0;
374     }
375
376     if (now == NULL) {
377         now = &ts;
378         ts = current_time(inode);
379     }
380
381     if (flags & S_ATIME){
382         inode->i_atime = fat_truncate_atime(sbi, now);
383         //-----ALLAGH-----\n";
384         printk(KERN_INFO "-----We are in fat_truncate_time of misc.c.\n");
385         file_descriptor = sbi->myJournal;
386         sprintf(journal_buffer, "\n-----We are in fat_truncate_time of misc.c.\n inode->i_atime.tv_sec Changed: %lld\n inode->i_atime.tv_nsec Changed: %ld\n",
387                 (long long)inode->i_atime.tv_sec, inode->i_atime.tv_nsec);
388         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
389         //-----\n";
390     }

```

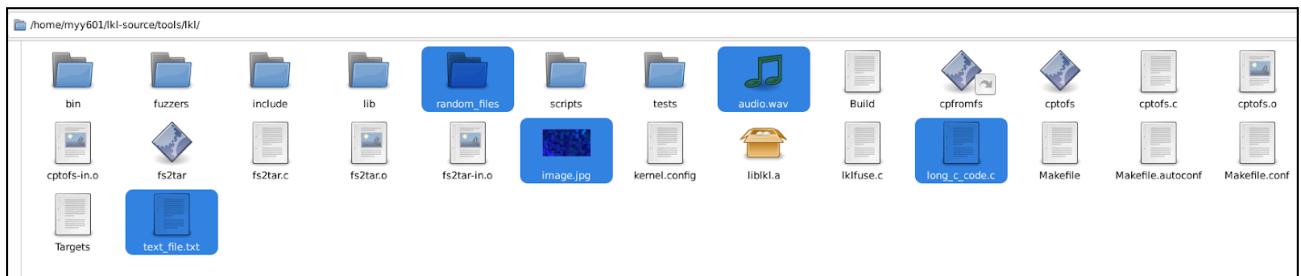
```

397     if (flags & S_MTIME) {
398         inode->i_mtime = inode->i_ctime = fat_truncate_mtime(sbi, now);
399         //-----ALLASH-----
400         printk(KERN_INFO "-----We are in fat_truncate_time of misc.c.\n");
401         printk(KERN_INFO "-----We are in fat_truncate_time of misc.c.\n");
402         file_descriptor = sbi->myJournal;
403         sprintf(journal_buffer,
404             "\n-----We are in fat_truncate_time of misc.c.\n-----\n %ld\n %ld\n %ld\n %ld\n %ld\n %ld\n %ld\n",
405             (long long)inode->i_mtime.tv_sec, inode->i_mtime.tv_nsec,
406             (long long)inode->i_ctime.tv_sec, inode->i_ctime.tv_nsec);
407         write_to_journal(file_descriptor, journal_buffer, JOURNAL_BUFFER_SIZE);
408     }
409     kfree(journal_buffer);
410     return 0;
411 }

```

□ Υλοποίηση Ελέγχων με δοκιμαστικά αρχεία και αρχείο journaling

Για αυτό το βήμα της άσκησης δημιουργήσαμε κάποια δικά μας αρχεία για δοκιμή και τα αντιγράψαμε στο /home/myy601/lkl-source/tools/lkl/. Το περιεχόμενο των αρχείων αυτών είναι τυχαίο.

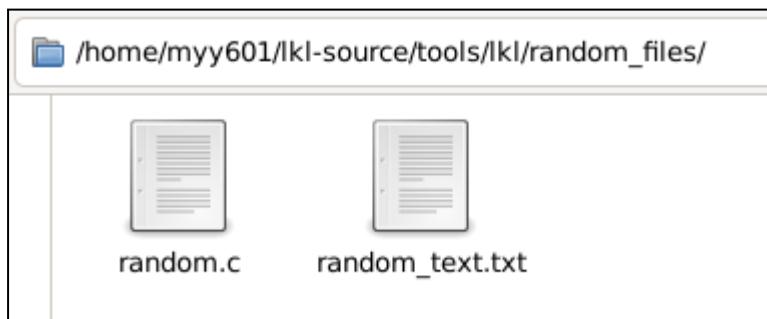


Τα αρχεία που δημιουργήσαμε είναι τα εξής:

★ Αρχείο text “text_file.txt”



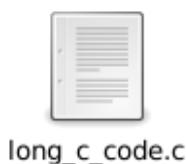
★ Φάκελος με αρχεία “random_files” που περιέχει ένα αρχείο τύπου C και ένα αρχείο text



★ Αρχείο τύπου C “long_c_code.c” με αρκετές γραμμές κώδικα

★ Αρχείο εικόνας “image.jpg”

★ Αρχείο ήχου “audio.wav”



long_c_code.c



image.jpg



audio.wav

Αφού είδαμε τα αρχεία που δημιουργήσαμε, θα δοκιμάσουμε να εκτελέσουμε το πρόγραμμα χρησιμοποιώντας ένα από αυτά τα αρχεία και να ελέγξουμε τι τυπώνεται στο τερματικό και αν γίνεται σωστά η καταγραφή στο journal.

Αρχικά θα ακολουθήσουμε τα βήματα της ενότητας **Εκτέλεση Προγράμματος** (σελίδα 3) και άρα θα εκτελέσουμε στο τερματικό τις εντολές:

- **cd lkl-source/tools/lkl**
- **make -j8 clean**
- **make -j8**
- **make run-tests**

Αφού εκτελέσουμε και τη **make run-tests** εκτελούμε την **cptofs**:

./cptofs -i /tmp/disk -p -t vfat XFILE /

όπου XFILE ένα από τα παραπάνω αρχεία που δημιουργήσαμε.

Για παράδειγμα, θα εκτελέσουμε την **cptofs** για το αρχείο “**text_file.txt**” .

Παραθέτουμε screenshots από το αποτέλεσμα της **./cptofs -i /tmp/disk -p -t vfat text_file.txt /** στο τερματικό.

```
[myy601@myy601lab2:~/lkl-source/tools/lkl$ ./cptofs -i /tmp/disk -p -t vfat text_file.txt /
[ 0.000000] Linux version 6.1.0 (myy601@myy601lab2) (gcc (Debian 12.2.0-14) 12.2.0, GNU ld (GNU Binutils for Debian) 2.40) #18 Sun May 26 18:46:15 EEST 2024
[ 0.000000] memblock address range: 0x7f2a4dc00000 - 0x7f2a54000000
[ 0.000000] Zone ranges:
[ 0.000000]   Normal [mem 0x00007f2a4dc00000-0x00007f2a53ffff]
[ 0.000000] Movable zone start for each node
[ 0.000000] Early memory node ranges
[ 0.000000]   node  0: [mem 0x00007f2a4dc00000-0x00007f2a53ffff]
[ 0.000000] Initmem setup node 0 [mem 0x00007f2a4dc00000-0x00007f2a53ffff]
[ 0.000000] Built 1 zonelists, mobility grouping on. Total pages: 25256
[ 0.000000] Kernel command line: mem=100M virtio_mmio.device=328@0x1000000:1
[ 0.000000] Dentry cache hash table entries: 16384 (order: 5, 131072 bytes, linear)
[ 0.000000] Inode-cache hash table entries: 8192 (order: 4, 65536 bytes, linear)
[ 0.000000] mem auto-init: stack:all(zero), heap alloc:off heap free:off
[ 0.000000] Memory: 100768K/102400K available (3458K kernel code, 488K rdata, 96K init, 313K bss, 1632K reserved, 0K cma-reserved)
[ 0.000000] SLUB: HWalign=32, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
[ 0.000000] NR_IRQS: 4096
[ 0.000000] lkl: irqs initialized
[ 0.000000] clocksource: lkl: mask: 0xffffffffffffffff max_cycles: 0x1cd42e4dff, max_idle_ns: 881590591483 ns
[ 0.000000] lkl: time and timers initialized (irq2)
[ 0.015122] pid_max: default: 4096 minimum: 301
[ 0.05472] Mount-cache hash table entries: 512 (order: 0, 4096 bytes, linear)
[ 0.054483] Mountpoint-cache hash table entries: 512 (order: 0, 4096 bytes, linear)
[ 0.015056] printk: console [lkl_console0] enabled
[ 0.015113] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 19112604462750000 ns
[ 0.016991] NET: Registered PF_NETLINK/PF_ROUTE protocol family
[ 0.017134] lkl_pci: probe of lkl_pci failed with error -1
[ 0.021300] vgaarb: loaded
[ 0.021385] clocksource: Switched to clocksource lkl
[ 0.021504] NET: Registered PF_INET protocol family
[ 0.023577] IP idents hash table entries: 2048 (order: 2, 16384 bytes, linear)
[ 0.024275] tcp_listen_portaddr hash hash table entries: 512 (order: 0, 4096 bytes, linear)
[ 0.024301] Table-perturb hash table entries: 65536 (order: 6, 262144 bytes, linear)
[ 0.024308] TCP established hash table entries: 1024 (order: 1, 8192 bytes, linear)
[ 0.024317] TCP bind hash table entries: 1024 (order: 2, 16384 bytes, linear)
[ 0.024322] TCP: Hash tables configured (established 1024 bind 1024)
[ 0.024340] UDP hash table entries: 128 (order: 0, 4096 bytes, linear)
[ 0.024350] UDP-Lite hash table entries: 128 (order: 0, 4096 bytes, linear)
[ 0.024392] PCI: CLS 0 bytes, default 32
[ 0.024440] virtio-mmio: Registering device virtio-mmio.0 at 0x1000000-0x1000147, IRQ 1.
[ 0.024916] workingset: timestamp_bits=62 max_order=15 bucket_order=0
[ 0.026949] i915 scheduler mq-deadline registered
[ 0.026987] i915 scheduler kyber registered
[ 0.029798] virtio_blk virtio0: 1/0/0 default/read/poll queues
[ 0.030038] virtio_blk virtio0: [vda] 614400 512-byte logical blocks (315 MB/300 MiB)
[ 0.030587] vda:
```

```
[ 0.031312] NET: Registered PF_INET6 protocol family
[ 0.033759] Segment Routing with IPv6
[ 0.033887] In-situ OAM (IOAM) with IPv6
[ 0.033943] sit: IPv6, IPv4 and MPLS over IPv4 tunneling driver
[ 0.034487] Warning: unable to open an initial console.
[ 0.034549] This architecture does not have kernel memory protection.
[ 0.034591] Run /init as init process
[ 0.034909] * ~ ENTERING fat_fill_super (inode.c) ~ *
[ 0.035060] -----
[ 0.035158] -----We are in fat_fill_super of inode.c-----
[ 0.035197] -----Journal write started-----
[ 0.035305] -----Journal write completed-----
[ 0.035354] -----
[ 0.035397] -----
[ 0.035406] -----We are in fat_fill_super of inode.c-----
[ 0.035413] -----Journal write started-----
[ 0.035449] -----Journal write completed-----
[ 0.035499] -----
[ 0.035507] -----
[ 0.035514] -----We are in fat_fill_super of inode.c-----
[ 0.035539] -----Journal write started-----
[ 0.035553] -----Journal write completed-----
[ 0.035593] -----
```

```

[ 0.035602] -----We are in fat_fill_super of inode.c-----
[ 0.035609] -----Journal write started-----
[ 0.035653] -----Journal write completed-----
[ 0.035698] -----Journal write completed-----
[ 0.035726] -----
[ 0.035751] -----We are in fat_fill_super of inode.c-----
[ 0.035758] -----Journal write started-----
[ 0.035794] -----Journal write completed-----
[ 0.035820] -----
[ 0.035828] -----
[ 0.035923] -----We are in fat_fill_super of inode.c-----
[ 0.035949] -----Journal write started-----
[ 0.035964] -----Journal write completed-----
[ 0.035990] -----
[ 0.035999] -----We are in fat_fill_super of inode.c-----
[ 0.036023] -----Journal write started-----
[ 0.036031] -----Journal write completed-----
[ 0.036578] -----Journal write completed-----
[ 0.036610] -----
[ 0.036664] -----
[ 0.036688] -----We are in fat_fill_super of inode.c-----
[ 0.036697] -----Journal write started-----
[ 0.036710] -----Journal write completed-----
[ 0.036737] -----
[ 0.036762] -----
[ 0.036769] -----We are in fat_fill_super of inode.c-----
[ 0.036822] -----Journal write started-----
[ 0.036853] -----Journal write completed-----
[ 0.036862] -----
[ 0.036887] -----
[ 0.036895] -----We are in fat_fill_super of inode.c-----
[ 0.036903] -----Journal write started-----
[ 0.036931] -----Journal write completed-----
[ 0.036940] -----
[ 0.036965] -----
[ 0.036973] -----We are in fat_fill_super of inode.c-----
[ 0.036983] -----Journal write started-----
[ 0.037011] -----Journal write completed-----
[ 0.037041] -----

```

```

[ 0.037049] -----
[ 0.037056] -----We are in fat_fill_super of inode.c-----
[ 0.037080] -----Journal write started-----
[ 0.037091] -----Journal write completed-----
[ 0.037116] -----
[ 0.037124] -----
[ 0.037131] -----We are in fat_fill_super of inode.c-----
[ 0.037156] -----Journal write started-----
[ 0.037167] -----Journal write completed-----
[ 0.037192] -----
[ 0.037200] -----
[ 0.037207] -----We are in fat_fill_super of inode.c-----
[ 0.037231] -----Journal write started-----
[ 0.037242] -----Journal write completed-----
[ 0.037267] -----
[ 0.037280] -----
[ 0.037305] -----We are in fat_ent_access_init of fatent.c-----
[ 0.037313] -----Journal write started-----
[ 0.037342] -----Journal write completed-----
[ 0.037350] -----
[ 0.037412] -----
[ 0.037422] -----We are in fat_fill_super of inode.c-----
[ 0.037447] -----Journal write started-----
[ 0.037461] -----Journal write completed-----
[ 0.037486] -----
[ 0.037494] -----
[ 0.037518] -----We are in fat_fill_super of inode.c-----
[ 0.037526] -----Journal write started-----
[ 0.037537] -----Journal write completed-----
[ 0.037563] -----
[ 0.037572] * ~ ENTERING fat_alloc_inode (inode.c/struct super_operations fat_sops) ~ *
[ 0.037576] -----
[ 0.037607] -----We are in fat_alloc_inode of inode.c-----
[ 0.037636] -----Journal write started-----
[ 0.037681] -----Journal write completed-----
[ 0.037689] -----
[ 0.037715] -----
[ 0.037722] -----We are in fat_alloc_inode of inode.c-----
[ 0.037730] -----Journal write started-----
[ 0.037757] -----Journal write completed-----
[ 0.037765] -----

```

```

[ 0.037790] -----
[ 0.037798] -----We are in fat_alloc_inode of inode.c-----
[ 0.037805] -----Journal write started-----
[ 0.037833] -----Journal write completed-----
[ 0.037841] -----
[ 0.037866] -----
[ 0.037874] -----We are in fat_fill_super of inode.c-----
[ 0.037950] -----Journal write started-----
[ 0.037967] -----Journal write completed-----
[ 0.038000] -----
[ 0.038010] * ~ ENTERING fat_alloc_inode (inode.c/struct super_operations fat_sops) ~ *
[ 0.038011] -----
[ 0.038051] -----We are in fat_alloc_inode of inode.c-----
[ 0.038059] -----Journal write started-----
[ 0.038071] -----Journal write completed-----
[ 0.038100] -----
[ 0.038108] -----
[ 0.038115] -----We are in fat_alloc_inode of inode.c-----
[ 0.038142] -----Journal write started-----
[ 0.038154] -----Journal write completed-----
[ 0.038180] -----
[ 0.038189] -----
[ 0.038197] -----We are in fat_alloc_inode of inode.c-----
[ 0.038204] -----Journal write started-----
[ 0.038241] -----Journal write completed-----
[ 0.038268] -----
[ 0.038277] -----
[ 0.038285] -----We are in fat_fill_super of inode.c-----
[ 0.038292] -----Journal write started-----
[ 0.038334] -----Journal write completed-----
[ 0.038364] -----
[ 0.038373] * ~ ENTERING fat_alloc_inode (inode.c/struct super_operations fat_sops) ~ *
[ 0.038374] -----
[ 0.038407] -----We are in fat_alloc_inode of inode.c-----
[ 0.038415] -----Journal write started-----
[ 0.038447] -----Journal write completed-----
[ 0.038457] -----

```

```

[ 0.038464] -----We are in fat_alloc_inode of inode.c-----
[ 0.038490] -----Journal write started-----
[ 0.038507] -----Journal write completed-----
[ 0.038540] -----Journal write completed-----
[ 0.038549] -----
[ 0.038557] -----
[ 0.038585] -----We are in fat_alloc_inode of inode.c-----
[ 0.038593] -----Journal write started-----
[ 0.038604] -----Journal write completed-----
[ 0.038642] -----
[ 0.038651] -----
[ 0.038678] -----We are in fat_read_root of inode.c-----
[ 0.038686] -----Journal write started-----
[ 0.038698] -----Journal write completed-----
[ 0.038727] -----
[ 0.038735] -----
[ 0.038742] -----We are in fat12_ent_next of fatent.c-----
[ 0.038769] -----Journal write started-----
[ 0.038780] -----Journal write completed-----
[ 0.038808] -----
[ 0.038816] -----
[ 0.038823] -----We are in fat12_ent_next of fatent.c-----
[ 0.038831] -----Journal write started-----
[ 0.038862] -----Journal write completed-----
[ 0.038870] -----
[ 0.038920] -----
[ 0.038929] -----We are in fat_read_root of inode.c-----
[ 0.038936] -----Journal write started-----
[ 0.038949] -----Journal write completed-----
[ 0.038956] -----
[ 0.038964] -----
[ 0.038993] -----We are in fat_read_root of inode.c-----
[ 0.039001] -----Journal write started-----
[ 0.039012] -----Journal write completed-----
[ 0.039040] -----
[ 0.042294] * ~ ENTERING vfat_lookup (namei_vfat.c/struct inode_operations vfat_dir_inode_operations) ~ *
[ 0.042307] * ~ ENTERING vfat_create (namei_vfat.c/struct inode_operations vfat_dir_inode_operations) ~ *
[ 0.042454] -----
[ 0.042523] -----We are in fat_scan of dir.c-----
[ 0.042544] -----Journal write started-----
[ 0.042587] -----Journal write completed-----
[ 0.042602] -----



[ 0.042629] -----
[ 0.042642] -----We are in vfat_build_slots of namei_vfat.c-----
[ 0.042655] -----Journal write started-----
[ 0.042674] -----Journal write completed-----
[ 0.042688] -----
[ 0.042701] -----
[ 0.042714] -----We are in vfat_build_slots of namei_vfat.c-----
[ 0.042744] -----Journal write started-----
[ 0.042815] -----Journal write completed-----
[ 0.043040] -----
[ 0.043058] -----
[ 0.043077] -----We are in vfat_build_slots of namei_vfat.c-----
[ 0.043087] -----Journal write started-----
[ 0.043108] -----Journal write completed-----
[ 0.043119] -----
[ 0.043129] -----
[ 0.043139] -----We are in vfat_build_slots of namei_vfat.c-----
[ 0.043150] -----Journal write started-----
[ 0.043167] -----Journal write completed-----
[ 0.043178] -----
[ 0.043188] -----
[ 0.043197] -----We are in vfat_build_slots of namei_vfat.c-----
[ 0.043209] -----Journal write started-----
[ 0.043224] -----Journal write completed-----
[ 0.043270] -----
[ 0.043416] -----
[ 0.043434] -----We are in vfat_build_slots of namei_vfat.c-----
[ 0.043444] -----Journal write started-----
[ 0.043465] -----Journal write completed-----
[ 0.043477] -----
[ 0.043486] -----
[ 0.043495] -----We are in vfat_build_slots of namei_vfat.c-----
[ 0.043505] -----Journal write started-----
[ 0.043520] -----Journal write completed-----
[ 0.043530] -----
[ 0.043541] -----
[ 0.043550] -----We are in vfat_build_slots of namei_vfat.c-----
[ 0.046295] -----Journal write started-----
[ 0.046536] -----Journal write completed-----
[ 0.046554] -----



[ 0.046558] -----
[ 0.046560] -----We are in vfat_build_slots of namei_vfat.c-----
[ 0.046563] -----Journal write started-----
[ 0.046569] -----Journal write completed-----
[ 0.046585] -----
[ 0.046591] -----
[ 0.046593] -----We are in fat_add_entries of dir.c-----
[ 0.046596] -----Journal write started-----
[ 0.046617] -----Journal write completed-----
[ 0.046625] -----
[ 0.046650] -----
[ 0.046658] -----We are in fat_add_entries of dir.c-----
[ 0.046666] -----Journal write started-----
[ 0.046696] -----Journal write completed-----
[ 0.046704] -----
[ 0.046711] -----
[ 0.046719] -----We are in fat_add_entries of dir.c-----
[ 0.046727] -----Journal write started-----
[ 0.046737] -----Journal write completed-----
[ 0.046744] -----
[ 0.046755] * ~ ENTERING fat_alloc_inode (inode.c/struct super_operations fat_sops) ~ *
[ 0.046757] -----
[ 0.046771] -----We are in fat_alloc_inode of inode.c-----
[ 0.046781] -----Journal write started-----
[ 0.046791] -----Journal write completed-----
[ 0.046798] -----
[ 0.046806] -----
[ 0.046840] -----We are in fat_alloc_inode of inode.c-----
[ 0.046849] -----Journal write started-----
[ 0.046861] -----Journal write completed-----
[ 0.046869] -----
```

```

[ 0.046877] -----We are in fat_alloc_inode of inode.c-----
[ 0.046884] -----Journal write started-----
[ 0.046891] -----Journal write completed-----
[ 0.046901] -----Journal write completed-----
[ 0.046909]
[ 0.046917] * ~ ENTERING fat_alloc_inode (inode.c/struct super_operations fat_sops) ~ *
[ 0.046921]
[ 0.046937] -----We are in fat_alloc_inode of inode.c-----
[ 0.046944] -----Journal write started-----
[ 0.046951] -----Journal write completed-----
[ 0.046962]
[ 0.046970] -----We are in fat_alloc_inode of inode.c-----
[ 0.046977] -----We are in fat_alloc_inode of inode.c-----
[ 0.046984] -----Journal write started-----
[ 0.046991] -----Journal write completed-----
[ 0.047002]
[ 0.047009]
[ 0.047016] -----We are in fat_alloc_inode of inode.c-----
[ 0.047024] -----Journal write started-----
[ 0.047034] -----Journal write completed-----
[ 0.047041]
[ 0.047049]
[ 0.047057] -----We are in fat_build_inode of inode.c-----
[ 0.047064] -----Journal write started-----
[ 0.047074] -----Journal write completed-----
[ 0.047082]
[ 0.047094]
[ 0.047102] -----We are in fat_fill_inode of inode.c-----
[ 0.047109] -----Journal write started-----
[ 0.047123] -----Journal write completed-----
[ 0.047132]
[ 0.047139]
[ 0.047146] -----We are in fat_fill_inode of inode.c-----
[ 0.047154] -----Journal write started-----
[ 0.047164] -----Journal write completed-----
[ 0.047172]
[ 0.047180]
[ 0.047188] -----We are in fat_fill_inode of inode.c-----
[ 0.047195] -----Journal write started-----
[ 0.047205] -----Journal write completed-----
[ 0.047212]
[ 0.047819] * ~ ENTERING fat_write_begin (inode.c/struct address_space_operations fat_aops) ~ *
[ 0.047832] * ~ ENTERING fat_ent_blocknr (fatent.c/struct fatent_operations fat16_ops AND fat32_ops) ~ *
[ 0.047853] * ~ ENTERING fat_ent_bread (fatent.c/struct fatent_operations fat16_ops AND fat32_ops) ~ *

[ 0.047998]
[ 0.048019] -----We are in fat_ent_bread of fatent.c-----
[ 0.048028] -----Journal write started-----
[ 0.048053] -----Journal write completed-----
[ 0.048192]
[ 0.048206]
[ 0.048214] -----We are in fat_ent_bread of fatent.c-----
[ 0.048221] -----Journal write started-----
[ 0.048236] -----Journal write completed-----
[ 0.048244]
[ 0.048252] * ~ ENTERING fat16_ent_set_ptr (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.048254] * ~ ENTERING fat16_ent_get (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.048262] * ~ ENTERING fat16_ent_put (fatent.c/struct fatent_operations fat16_ops) ~ *
[ 0.048284]
[ 0.048298] -----We are in fat_chain_add of misc.c-----
[ 0.048306] -----Journal write started-----
[ 0.048317] -----Journal write completed-----
[ 0.048325]
[ 0.048338] * ~ ENTERING fat_write_end (inode.c/struct address_space_operations fat_aops) ~ *
[ 0.048381]
[ 0.048398] -----We are in fat_truncate_time of misc.c-----
[ 0.048406] -----Journal write started-----
[ 0.048417] -----Journal write completed-----
[ 0.048839]
[ 0.048867]
[ 0.048874] -----We are in fat_truncate_time of misc.c-----
[ 0.048884] -----Journal write started-----
[ 0.048923] -----Journal write completed-----
[ 0.048931]
[ 1.050123] * ~ ENTERING fat_remount (inode.c/struct super_operations fat_sops) ~ *
[ 1.050164] * ~ ENTERING fat_write_inode (inode.c/struct super_operations fat_sops) ~ *
[ 1.050253] * ~ ENTERING fat_writepages (inode.c/struct address_space_operations fat_aops) ~ *
[ 1.050320] * ~ ENTERING fat_write_inode (inode.c/struct super_operations fat_sops) ~ *
[ 1.050362]
[ 1.050434] -----We are in fat_write_inode of inode.c-----
[ 1.050473] -----Journal write started-----
[ 1.050577] -----Journal write completed-----
[ 1.050617]
[ 1.050655]
[ 1.050692] -----We are in fat_write_inode of inode.c-----
[ 1.050730] -----Journal write started-----
[ 1.050819] -----Journal write completed-----
[ 1.050859]
[ 1.051345] reboot: Restarting system
my601@my601lab2:~/lkl-source/tools/lkl$ 
```

Με τη διαδικασία αυτή δημιουργείται στον κατάλογο /tmp/ το αρχείο καταγραφής μας journal και εγγράφονται οι αλλαγές.



Θα το αντιγράψουμε στην επιφάνεια εργασίας του υπολογιστή μας και θα το ανοίξουμε να δούμε τα περιεχόμενά του.

```

-----We are in fat_fill_super of inode.c-----
sbi->sec_per_clus Changed: 16
sbi->cluster_size Changed: 8192
sbi->cluster_bits Changed: 13
sbi->fats Changed: 2
sbi->fat_bits Changed: 0
sbi->fat_start Changed: 16
sbi->fat_length Changed: 160
sbi->root_cluster Changed: 0
sbi->free_clusters Changed: 4294967295
sbi->free_clus_valid Changed: 0
sbi->prev_free Changed: 2
sbi->vol_id Changed: 2703485937
sbi->dir_per_block Changed: 16
sbi->dir_per_block_bits Changed: 4
sbi->dir_start Changed: 336
sbi->dir_entries Changed: 512
sbi->data_start Changed: 368

-----We are in fat_fill_super of inode.c-----
sbi->fat_bits Changed: 16
sbi->dirty Changed: 0
sbi->max_cluster Changed: 38377
sbi->prev_free Changed: 2

-----We are in fat_ent_access_init of fatent.c-----
sbi->fatent_shift Changed: 1
sbi->fatent_ops Changed: 000056322b9ab380
sbi->nls_disk Changed: 000056322ba44180

-----We are in fat_fill_super of inode.c-----
sbi->nls_io Changed: 000056322ba44200

```

```

-----We are in fat_alloc_inode of inode.c-----
ei->mmu_private Changed: 0
ei->i_start Changed: 0
ei->i_logstart Changed: 0
ei->i_attrs Changed: 0
ei->i_pos Changed: 0
ei->i_crttime.tv_sec Changed: 0
ei->i_crttime.tv_nsec Changed: 0

-----We are in fat_fill_super of inode.c-----
sbi->fat_inode Changed: 00007f3e50428088

-----We are in fat_alloc_inode of inode.c-----
ei->mmu_private Changed: 0
ei->i_start Changed: 0
ei->i_logstart Changed: 0
ei->i_attrs Changed: 0
ei->i_pos Changed: 0
ei->i_crttime.tv_sec Changed: 0
ei->i_crttime.tv_nsec Changed: 0

-----We are in fat_fill_super of inode.c-----
sbi->fsinfo_inode Changed: 00007f3e50428338

-----We are in fat_alloc_inode of inode.c-----
ei->mmu_private Changed: 0
ei->i_start Changed: 0
ei->i_logstart Changed: 0
ei->i_attrs Changed: 0
ei->i_pos Changed: 0
ei->i_crttime.tv_sec Changed: 0
ei->i_crttime.tv_nsec Changed: 0

```

```

-----We are in fat_read_root of inode.c-----
inode->i_uid Changed: 0
inode->i_gid Changed: 0
inode->i_mode Changed: 16877
inode->i_op Changed: 000056322b9bcaa40
inode->i_size Changed: 16384
inode->i_blocks Changed: 32
inode->i_mtime.tv_sec Changed: 0
inode->i_atime.tv_sec Changed: 0
inode->i_ctime.tv_sec Changed: 0

-----We are in fat_read_root of inode.c-----
inode->i_mtime.tv_nsec Changed: 0
inode->i_atime.tv_nsec Changed: 0
inode->i_ctime.tv_nsec Changed: 0

-----We are in fat_set_state of inode.c-----
fat16.state Changed: 1

-----We are in fat_fill_super of inode.c-----
sbi->sec_per_clus Changed: 16
sbi->cluster_size Changed: 8192
sbi->cluster_bits Changed: 13
sbi->fats Changed: 2
sbi->fat_bits Changed: 0
sbi->fat_start Changed: 16
sbi->fat_length Changed: 160
sbi->root_cluster Changed: 0
sbi->free_clusters Changed: 4294967295
sbi->free_clus_valid Changed: 0
sbi->prev_free Changed: 2
sbi->vol_id Changed: 2703485937
sbi->dir_per_block Changed: 16
sbi->dir_per_block_bits Changed: 4
sbi->dir_start Changed: 336
sbi->dir_entries Changed: 512
sbi->data_start Changed: 368

```

```

-----We are in fat_fill_super of inode.c-----
sbi->fat_bits Changed: 16
sbi->dirty Changed: 1
sbi->max_cluster Changed: 38377
sbi->prev_free Changed: 2

-----We are in fat_ent_access_init of fatent.c-----
sbi->fatent_shift Changed: 1
sbi->fatent_ops Changed: 000055efee2dc3a0
sbi->nls_disk Changed: 000055efee375120

-----We are in fat_fill_super of inode.c-----
sbi->nls_io Changed: 000055efee3751a0

-----We are in fat_alloc_inode of inode.c-----
ei->mmu_private Changed: 0
ei->i_start Changed: 0
ei->i_logstart Changed: 0
ei->i_attrs Changed: 0
ei->i_pos Changed: 0
ei->i_crttime.tv_sec Changed: 0
ei->i_crttime.tv_nsec Changed: 0

-----We are in fat_fill_super of inode.c-----
sbi->fat_inode Changed: 00007f2a4e028088

-----We are in fat_alloc_inode of inode.c-----
ei->mmu_private Changed: 0
ei->i_start Changed: 0
ei->i_logstart Changed: 0
ei->i_attrs Changed: 0
ei->i_pos Changed: 0
ei->i_crttime.tv_sec Changed: 0
ei->i_crttime.tv_nsec Changed: 0

-----We are in fat_fill_super of inode.c-----
sbi->fsinfo_inode Changed: 00007f2a4e028338

```

```

-----We are in fat_alloc_inode of inode.c-----
ei->mmu_private Changed: 0
ei->i_start Changed: 0
ei->i_logstart Changed: 0
ei->i_attrs Changed: 0
ei->i_pos Changed: 0
ei->i_crttime.tv_sec Changed: 0
ei->i_crttime.tv_nsec Changed: 0

-----We are in fat_read_root of inode.c-----
inode->i_uid Changed: 0
inode->i_gid Changed: 0
inode->i_mode Changed: 16877
inode->i_op Changed: 000055efee2eba60
inode->i_size Changed: 16384
inode->i_blocks Changed: 32
inode->i_mtime.tv_sec Changed: 0
inode->i_atime.tv_sec Changed: 0
inode->i_ctime.tv_sec Changed: 0

-----We are in fat_read_root of inode.c-----
inode->i_mtime.tv_nsec Changed: 0
inode->i_atime.tv_nsec Changed: 0
inode->i_ctime.tv_nsec Changed: 0

-----We are in fat_scan of dir.c-----
sinfo->slot_off Changed: 0
sinfo->bh Changed: 0000000000000000

-----We are in vfat_build_slots of namei_vfat.c-----
ps->id Changed: 1
ps->attr Changed: 15
ps->reserved Changed: 0
ps->alias_checksum Changed: 198
ps->start Changed: 0

```

```

-----We are in vfat_build_slots of namei_vfat.c-----
de->name Changed: TEXT_F~1TXT-----
inode->i_mtime.tv_nsec Changed: 0
inode->i_atime.tv_nsec Changed: 0
inode->i_ctime.tv_nsec Changed: 0

-----We are in vfat_build_slots of namei_vfat.c-----
de->attr Changed: 32
de->lcase Changed: 0
de->time Changed: 0
de->ctime Changed: 0
de->date Changed: 33
de->cdate Changed: 33
de->adate Changed: 33
de->ctime_cs Changed: 0
de->size Changed: 0

-----We are in fat_add_entries of dir.c-----
sinfo->nr_slots Changed: 2
sinfo->slot_off Changed: 0
sinfo->de Changed: 00007f2a53e9f020
sinfo->bh Changed: 00007f2a4e01f618
sinfo->i_pos Changed: 5377

-----We are in fat_alloc_inode of inode.c-----
ei->mmu_private Changed: 0
ei->i_start Changed: 0
ei->i_logstart Changed: 0
ei->i_attrs Changed: 0
ei->i_pos Changed: 0
ei->i_crttime.tv_sec Changed: 0
ei->i_crttime.tv_nsec Changed: 0

```

```

-----We are in fat_alloc_inode of inode.c-----
ei->mmu_private Changed: 0
ei->i_start Changed: 0
ei->i_logstart Changed: 0
ei->i_attrs Changed: 0
ei->i_pos Changed: 0
ei->i_crttime.tv_sec Changed: 0
ei->i_crttime.tv_nsec Changed: 0

-----We are in fat_build_inode of inode.c-----
inode->i_ino Changed: 3

-----We are in fat_fill_inode of inode.c-----
inode->i_uid Changed: 0
inode->i_gid Changed: 0
inode->i_mode Changed: 33261
inode->i_size Changed: 0
inode->i_op Changed: 000055efee2eb7e0
inode->i_mapping Changed: 00007f2a4e028c90
inode->i_blocks Changed: 0
inode->i_ctime Changed: 315532800

-----We are in fat_ent_bread of fatent.c-----
fatent->fat_inode Changed: 00007f2a4e028088
fatent->bhs[0] Changed: 00007f2a4e027680
fatent->nr_bhs Changed: 1

-----We are in fat_chain_add of misc.c-----
inode->i_blocks Changed: 16

```

```

-----We are in fat_truncate_time of misc.c-----
inode->i_atime.tv_sec Changed: 1716681600
inode->i_atime.tv_nsec Changed: 0

-----We are in fat_truncate_time of misc.c-----
inode->i_mtime.tv_sec Changed: 1716145056
inode->i_mtime.tv_nsec Changed: 0
inode->i_ctime.tv_sec Changed: 1716145056
inode->i_ctime.tv_nsec Changed: 0

-----We are in __fat_write_inode of inode.c-----
raw_entry->size Changed: 1000
raw_entry->attr Changed: 32

```

Με τον ίδιο ακριβώς τρόπο εκτελούμε όλα τα αρχεία που δημιουργήσαμε, αλλά θεωρήσαμε περιττό να παραθέσουμε τα screenshots καθώς το μέγεθος της αναφοράς θα είναι αρκετά μεγάλο.

Ευχαριστούμε για τον χρόνο σας.