ΕΤΟΣ: 2020-2021

ΜΑΡΙΑ ΗΛΕΚΤΡΑ ΓΑΡΓΑΛΑ - Α.Μ.4330

ΔΗΜΟΣΘΕΝΗΣ ΖΑΓΚΑΣ – Α.Μ.4359

Email: cs04330@uoi.gr

Email: cs04359@uoi.gr

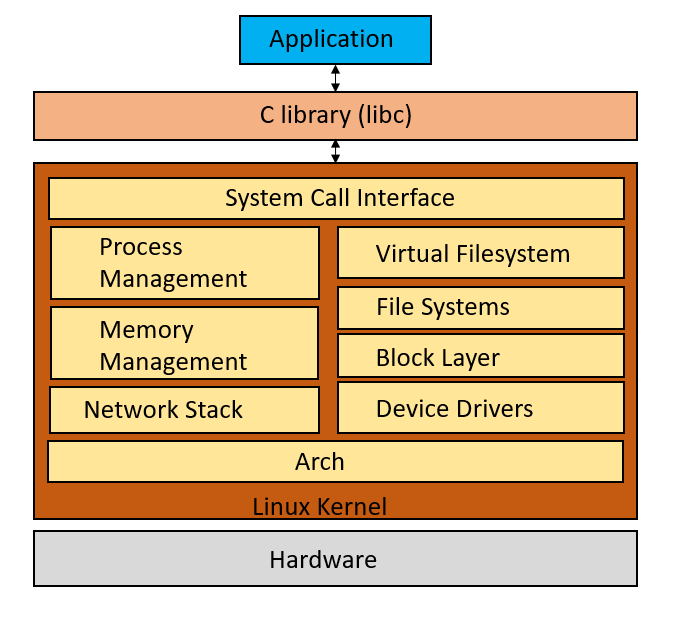
2Η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ  
-ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

Υλοποίηση αρχείου καταγραφής στο σύστημα αρχείων FAT του Linux

**Γενική περιγραφή δομή του Linux:**

Περιγραφή Εικόνας 1:

Οι εφαρμογές «Application» τρέχουν σε επίπεδο χρήστη. Οι εφαρμογές χρησιμοποιούν βιβλιοθήκες, ώστε να ζητήσουν από το σύστημα λειτουργικότητα.  
Η libc είναι βιβλιοθήκη που τρέχει σε επίπεδο χρήστη.



Το επίπεδο πυρήνα, εκεί δηλαδή που υπάρχει ο πυρήνας του Linux, βρίσκεται κάτω από το επίπεδο χρήστη. Αποτελείται από τα εξής μέρη:

*System Call Interface (διεπαφή αιτήσεων)*: Όταν η εφαρμογή θέλει να αποκτήσει προσβασιμότητα στον πυρήνα καλεί κλήση συστήματος

*Process Management (διαχείριση διεργασιών)*: Παραδείγματος χάριν, διαχειρίζεται τις διεργασίες, υλοποιεί τα νήματα, είναι υπεύθυνο για την χρονοδρομολόγηση.

*Memory Management (διαχείριση μνήμης)*: Διαχειρίζεται την φυσική και την εικονική μνήμη

*Network stack*: Υλοποίηση πρωτοκόλλων δικτύου

*Virtual Filesystem*: Διεπαφή συστημάτων αρχείων

*File System*: Υλοποίηση συστημάτων αρχείων

*Block Layer*: Επίπεδο μεταξύ δίσκου και συστήματος αρχείων

*Device Drivers*: Οδηγοί συσκευών

*Arch*: Υλοποίηση αρχιτεκτονικής

Ο πυρήνας δεν επιτρέπει στον προγραμματιστή να έχει πρόσβαση σε λειτουργίες βιβλιοθήκης, αφού ο πυρήνας είναι ανεξάρτητος από το επίπεδο χρήστη. Για να δουλέψουμε στον πυρήνα, θα πρέπει να χρησιμοποιούνται παρόμοιες συναρτήσεις.

Προβλήματα: Όταν προγραμματίζουμε σε επίπεδο πυρήνα, δεν υπάρχει προστασία μνήμης (πχ σε περίπτωση σφάλματος στην μνήμη ο πυρήνας κρασάρει και χρειάζεται reboot), το μέγεθος της στοίβας είναι πολύ μικρό και κατ’ επέκταση, δε πρέπει να έχουμε πολλές μεταβλητές στην στοίβα, ούτε και αναδρομή. Επίσης δεν υποστηρίζεται το swapping, δηλαδή η λειτουργία κατά την οποία αν γεμίσει η μνήμη μεταφέρει τα αρχεία στον δίσκο.

**Virtual Filesystem:**

*Αρχεία*: περιλαμβάνουν τα δεδομένα του χρήστη.

*Φάκελοι*: περιλαμβάνουν αρχεία και φακέλους και βοηθούν στην οργάνωση των δεδομένων.

Το σύστημα αρχείων είναι μέθοδος αποθήκευσης δεδομένων(περιεχόμενα αρχείου) και μεταδεδομένων(πληροφορίες αρχείου όπως όνομα, μέγεθος, δικαιώματα, δείκτες σε block κλπ.) σε αρχεία και φακέλους. Τα δεδομένα αυτά αποθηκεύονται σε block δεδομένων.

*Βασικές Δομές Συστήματος αρχείων(VF):*

*Block δεδομένων*: αποθήκευση δεδομένων των αρχείων

*Block μεταδεδομένων*: αποθήκευση μεταδεδομένων όπου είναι οργανωμένα σε inodes

*Bitmaps*: πίνακας όπου δείχνει τα ελεύθερα block

*Superblock*: πληροφορίες για το σύστημα αρχείων. Βρίσκεται στην συνήθως στην αρχή του δίσκου

*Inodes:*

Κάθε inode σχετίζεται με ένα διαφορετικό αρχείο και περιλαμβάνει πληροφορίες μεταδεδομένων. Διατηρούνται σε πίνακα και κάθε inode έχει και έναν αριθμό όπου δείχνει στον πίνακα.

*Block δεδομένων:*

Αποθηκεύονται τα δεδομένα των αρχείων. Επειδή τα block μπορεί να βρίσκονται διάσπαρτα στον δίσκο υπάρχουν δύο τρόποι ώστε να μπορούμε να βρούμε την θέση τους στον δίσκο:

*Άμεσοι-έμμεσοι δείκτες*: στο inode όπου υπάρχει ένας αριθμός σε δείκτη που δείχνει στα block

*Πίνακας καταχώρησης αρχείων(File Allocation Table):*  το inode έχει έναν δείκτη στο πρώτο block του αρχείου και στην συνέχεια υπάρχει δείκτης στο επόμενο block. Το τελευταίο block δείχνει στο NULL ώστε να σηματοδοτείται το τέλος.

*Superblocks*: περιέχει πληροφορίες για το σύστημα αρχείων.

*Dentry*: καταχωρήσεις φακέλων.

*File*: η δομή χαρακτηρίζει κάθε φορά το αρχείο. Δημιουργείται με το open και διαλύεται με το close. Κάθε διεργασία έχει δικό της αντικείμενο file για την προσπέλαση του αρχείου.

**Προσάρτηση Συστήματος Αρχείων:**

Με το mount ενώνεται ένα σύστημα αρχείων σε έναν φάκελο. Με την διαδικασία αυτή, το σύστημα γίνεται ορατό.

**Σφάλματα**:

Σε περίπτωση σφάλματος όπου πχ ο υπολογιστής κρασάρει το σύστημα μένει σε ασυνεπή μορφή.

1η λύση για να διατηρηθούν τα δεδομένα είναι να έχουμε μεταβλητή στο superblock, όπου όταν προσαρτίζεται το σύστημα τού προσδίδει την τιμή dirty. Εφόσον το σύστημα δεν προσαρτηθεί, θα έχει ακόμη την μεταβλητή dirty και θα λειτουργήσει σαν flag ώστε να ξεκινήσει μια διαδικασία ελέγχων επιδιόρθωσης. Η λύση αυτή δεν είναι βέλτιστη, αφού είναι χρονοβόρα.

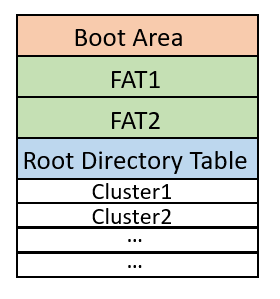
Η 2η λύση είναι η δημιουργία αρχείου log (καταγραφής) ώστε να αποθηκεύονται όλες οι λειτουργίες. Στην περίπτωση σφάλματος, οι λειτουργίες έχουν γραφεί στο αρχείο και μπορούν να ανακτηθούν από εκεί.

**Σύστημα Αρχείων FAT**

Το σύστημα αρχείων FAT οργανώνει τον δίσκο σε τομείς(sectors). Πολλοί sectors μαζί δημιουργούν ένα cluster. Τα δεδομένα ενός αρχείου γράφονται σε cluster. Εφόσον υπερβαίνουν το μέγεθος του cluster, μπορεί τα δεδομένα να γράφονται σε πολλαπλά clusters.  
To File Allocation Table είναι ένας πίνακας ευρετήριο όπου δημιουργείται κατά το format του συστήματος. Κατά την εκκίνηση δημιουργούνται δύο File Allocation Tables, όπου το 2ο λειτουργεί ως αντίγραφο του 1ου σε περίπτωση που χρειαστεί. Κάθε στοιχείο που βρίσκεται στον πίνακα είναι ένας δείκτης που δείχνει στο επόμενο cluster που αφορά ένα αρχείο. Στο τέλος του αρχείου, ο δείκτης δείχνει σε NULL, ώστε να ξέρουμε πως έφτασε στο τέλος.

*Δομή FAT*:

* *Reserved Sectors*: δεσμευμένη περιοχή, όπου αποθηκεύονται πληροφορίες
* *Fat Region*: αποθηκεύονται οι File Allocation Table
* *Root Directory Region*: περιλαμβάνονται αριθμοί αρχείων σε cluster και υπάρχει μόνο για FAT12 και FAT16
* *Data Region*: υπάρχουν εκεί τα clusters που περιέχουν δεδομένα αρχείων ή φακέλους

*Οργάνωση FAT*:

Στην αρχή του δίσκου υπάρχει η boot area όπου υπάρχουν πληροφορίες για το σύστημα αρχείων. Στην συνέχεια υπάρχει η περιοχή FAT Region1 και FAT Region2 που περιλαμβάνονται οι File Allocation Tables και μετά για τα συστήματα FAT12 και FAT16 υπάρχει το Root Directory Table. Τέλος, υπάρχουν τα clusters.

**Linux Kernel Library (LKL)**

Το LKL είναι μία βιβλιοθήκη όπου την διασυνδέουμε σε μορφή πυρήνα με την εφαρμογές και έτσι τρέχει σαν διεργασία και η εφαρμογή λειτουργεί χωρίς να επικοινωνεί με τον πυρήνα του Linux. Ουσιαστικά, είναι ένας τρόπος για να τροποποιήσουμε τον πυρήνα του Linux ώστε να λειτουργεί σαν βιβλιοθήκη σε επίπεδο χρήστη.

*LKL Environments*: μας βοηθάει να το τρέχουμε σε τρέχουμε σε διάφορα συστήματα. (Φάκελος tools/lkl)

*Generic LKL architecture*: εκεί υλοποιούνται νήματα, οι διαχειρίσεις των κλήσεων συστήματος κλπ. (Φάκελος arch/lkl)

*Linux Kernel*: πυρήνας Linux όπου περιλαμβάνει το VFS, NET, Process Manager κλπ.

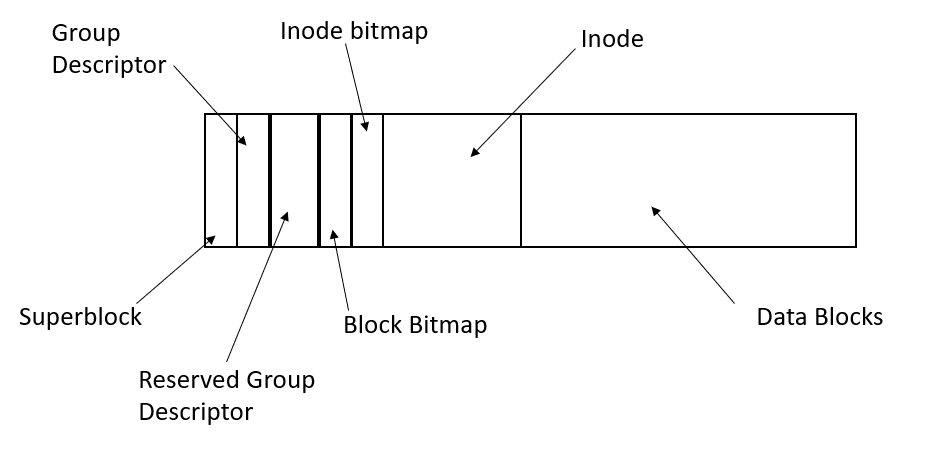
*LKL System call API*: διασυνδέει το LKL με την εφαρμογή. Μπορεί να διασυνδεθεί είτε με lkl\_ μπροστά από τις κλήσεις συστήματος, είτε με προφόρτωση της εφαρμογής LKL στην εφαρμογή είτε με χρήση τροποποιημένης libc.

**Περιγραφή 2ης Εργαστηριακής Άσκησης**

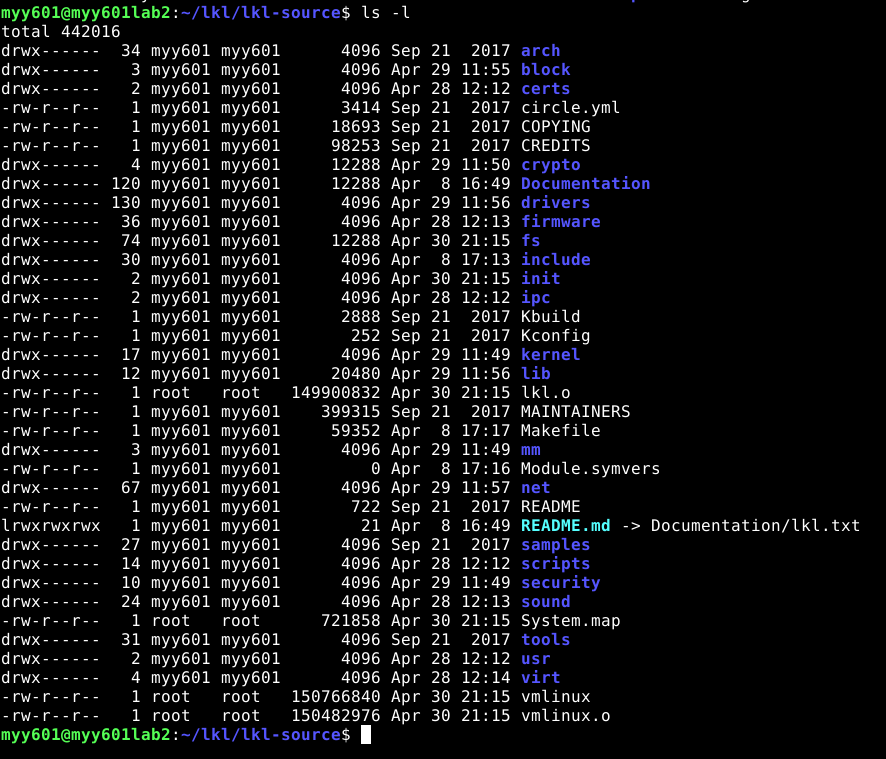
Για την Υλοποίηση της **2ης Εργαστηριακής Άσκησης**, πρέπει να χρησιμοποιήσουμε την βιβλιοθήκη Linux Kernel Library και σκοπός είναι η προσθήκη αρχείου καταγραφής στο σύστημα αρχείων FAT του Linux.

Το σύστημα αρχείων ext4 είναι το default σύστημα του Linux από το 2010 και αποτελεί βελτιωμένη έκδοση του ext3. Έχει περισσότερη χωρητικότητα και καλύτερη λειτουργία.

Το σύστημα ext4 έχει δομημένα τα δεδομένα όπως φαίνεται στην παρακάτω εικόνα:

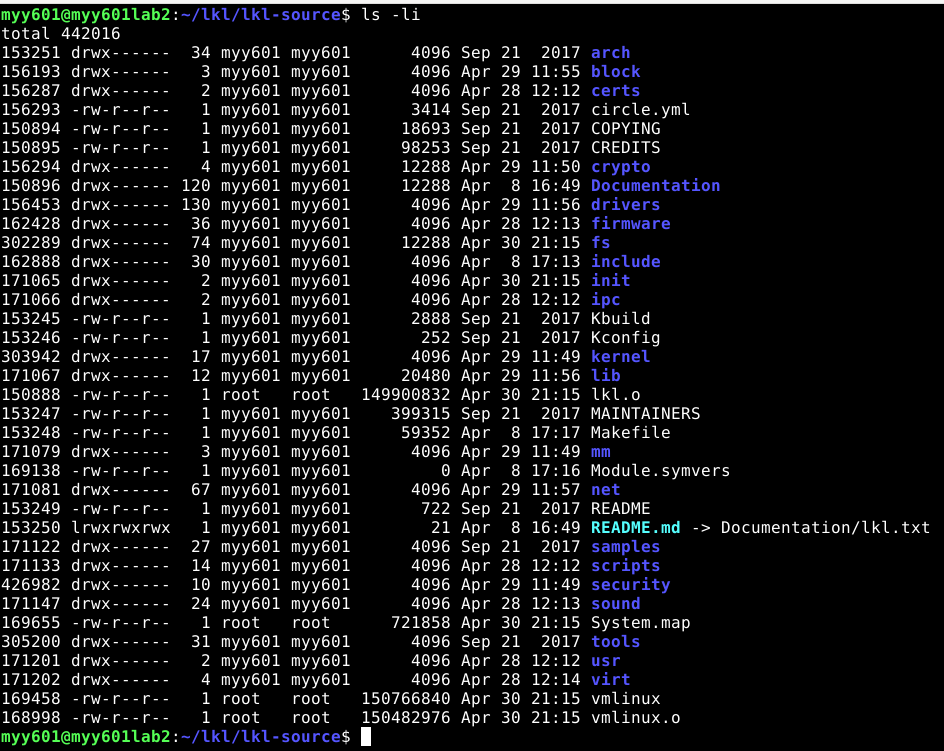


Το superblock βρίσκεται μπροστά και είναι μία περιοχή του δίσκου και περιλαμβάνει πληροφορίες για το σύστημα (θέσεις των δεδομένων κλπ).  
Το group descriptor βρίσκεται μετά το superblock και περιέχει τα block δεδομένων στον δίσκο. Περιέχει τη θέση όπου βρίσκονται τα δεδομένα μπλοκ δεδομένων bitmap, inode Bitmap και inode Table σε ομάδα block.  
Το reserved group descriptor αφορά επεκτάσεις που ενδέχεται να γίνουν μελλοντικά στο σύστημα. Βρίσκεται μετά από το descriptor group και έχουν παρόμοια λειτουργία.  
Το block bitmap αναφέρεται στην χαρτογράφηση του διαθέσιμου χώρου στον δίσκο. Το bitmap αποτελείται από block όπου αναπαριστώνται με bit ∙ αν το bit είναι 1 τότε το block είναι σε χρήση, ενώ αν είναι 0 τότε δεν είναι σε χρήση.  
Το inode table είναι μία δομή δεδομένων που αποθηκεύει δεδομένα για τα αρχεία. Το ext4 έχει συγκεκριμένο αριθμό inodes.  
Τα data blocks είναι σύνολο από block όπου περιέχουν τα δεδομένα των αρχείων.

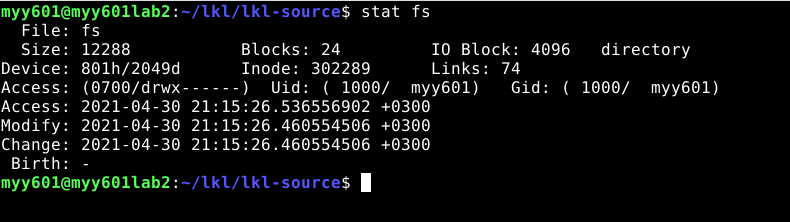


Εκτελώντας στον φάκελο lkl-source την εντολή **ls -l**, εκτυπώνονται τα ονόματα των αρχείων, τα δικαιώματα, ο χρόνος κλπ.

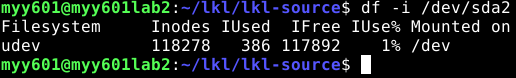
Εκτελώντας στην συνέχεια την εντολή **lkl -li**, εκτός λαμβάνω περισσότερες πληροφορίες. Πριν από τα δικαιώματα, εκτυπώνεται στην οθόνη ένας αριθμός, όπου αντιστοιχεί στον αριθμό του inode του αρχείου και φανερώνει το σημείο του table όπου βρίσκεται το αρχείο.



Για να μάθουμε πληροφορίες για ένα αρχείο, εκτελούμε την εντολή stat «όνομα». Τρέχοντας την εντολή για τον φάκελο πχ fs (**stat fs**), λαμβάνουμε το εξής:



Λαμβάνουμε πληροφορίες όπως τον αριθμό του inode, το μέγεθος, τα δικαιώματα, τα block που καταλαμβάνει, σε ποια συσκευή βρίσκεται, πόσα links υπάρχουν στο αρχείο αυτό κλπ.



Στην παραπάνω εικόνα, εκτελώντας την εντολή **df -i /dev/sda1**, παρατηρούμε τα δύο διαμερίσματα του δίσκου. Οι πληροφορίες που μας δίνονται είναι οι εξής:

* Το σύστημα αρχείων στο οποίο αναφέρεται (Filesystem)
* Τον συνολικό αριθμό των inode σε αυτό το σύστημα αρχείων (Inodes)
* Τον αριθμό των inode που χρησιμοποιούνται (IUse)
* Τον αριθμό των υπόλοιπων inode διαθέσιμων για χρήση (IFree)
* Το ποσοστό των χρησιμοποιημένων inode (IUse%)
* Το σημείο προσάρτησης για αυτό το σύστημα αρχείων (Mounted on)

***Mount (προσάρτηση)***

Για να μπορέσουμε να χρησιμοποιήσουμε ένα σύστημα αρχείων Linux, θα πρέπει να το προσαρτήσουμε (mount) τα μέρη του δίσκου. Για να γίνει αυτό θα πρέπει να χρησιμοποιήσουμε την εντολή **mount -t vfat -o loop /tmp/vfatfile /vfat** (μορφή: **mount -t** *‘σύστημα αρχείων που θέλω να χρησιμοποιήσω’ ‘μέρος που θέλω να κάνω mount’ ‘το σημείο που θέλουμε να γίνει mount’*).  
Για να κάνω umount το σύστημα (πχ τον φάκελο vfat) χρησιμοποιώ την εντολή **umount /vfat**.

Όταν κάνω αλλαγές στο σύστημα αρχείων, πολλές φορές χρειάζεται να κάνουμε umount και ξανά mount για να γίνουν ορατές οι αλλαγές.

To mountpoint είναι ένας κατάλογος όπου συνδέεται με ένα μέρος αποθήκευσης δεδομένων εκτός του δίσκου και των διαμερισμάτων του συστήματος.

Με την εντολή **./cptofs -i /tmp/vfatfile -p -t vfat lklfuse.c /** αντιγράφεται το αρχείο lklfuse.c στο vfatfile. Εκτελώντας την εντολή **cat /tmp/vfatfile** μπορούμε να δούμε ότι πράγματι τα περιεχόμενα του lklfuse.c βρίσκονται στο vfatfile. Μετά από αυτήν την ενέργεια, θα περιμέναμε να έχει δημιουργηθεί ένα αντίγραφο στον φάκελο /vfat. Για να γίνει ορατή η αλλαγή αυτή, όπως προαναφέρθηκε, θα πρέπει να εκτελέσουμε **umount /vfat** και στην συνέχεια **mount -t vfat -o loop /tmp/vfatfile /vfat** ώστε να γίνουν ορατές οι αλλαγές.

Πράγματι, μετά από αυτές τις ενέργειες, στο /vfat έχει δημιουργηθεί αντίγραφο του αρχείου lklfuse.c.

Με την χρήση της **./cptofs -i /tmp/vfatfile -p -t vfat tests /** αντιγράφεται όλος ο φάκελος tests στο vfatfile, παρόλα αυτά λόγω έλλειψης χώρου κάποια αρχεία δεν αντιγράφονται.

Graphical user interface, text

Description automatically generated

*Σκοπός της άσκησης είναι η δημιουργία αρχείου καταγραφής.*Θέλουμε να δημιουργήσουμε ένα αρχείο όπου θα αποθηκεύουμε εκεί τις αλλαγές που γίνονται στο σύστημα, έτσι ώστε σε περίπτωση σφάλματος (πχ να κρασάρει το σύστημα), να μπορούμε να ανακτήσουμε τις ενέργειες από το αρχείο αυτό.

Αρχικά, μπορούμε να ανοίξουμε το αρχείο με την εντολή open(…), σε επίπεδο χρήστη και να αποθηκεύσουμε το αρχείο σε έναν φάκελο της εικονικής μας μηχανής VMware. Θα έχουμε μέσα σε ένα αρχείο ένα file descriptor και θα προσθέτουμε κάθε φορά τα δεδομένα και μεταδεδομένα. Για να τα προσθέσουμε όμως, θα πρέπει να βρούμε τα κατάλληλα σημεία στον κώδικα, ώστε να γράψουμε τις πληροφορίες αυτές στο αρχείο.

Εφόσον θέλουμε να δημιουργήσουμε το αρχείο σε επίπεδο πυρήνα, μπορούμε να προσθέσουμε στον struct του superblock τον file descriptor όπου θα δείχνει στο αρχείο, για να καταλάβουμε χώρο σε αυτήν την περιοχή για το αρχείο μας. Η δημιουργία του αρχείου θα μπορούσε να γίνει και σε άλλα σημεία.  
Στην συνέχεια, εκεί όπου γίνεται mount το σύστημα (προσαρτίζεται) θα δημιουργήσουμε το αρχείο με την κλήση συστήματος sys\_open.

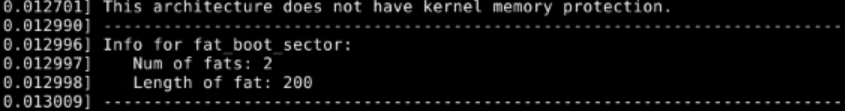
( Στον κώδικα του Linux, το inode είναι ορισμένο ως ένα struct και περιέχει διάφορες μεταβλητές όπως:

το **struct hlist\_node** (hash list, όπου περιέχει πληροφορίες για το αρχείο), **struct list head** (λίστα με inodes, superblocks), **uid\_t** (περιέχει το id του κατόχου του αρχείου/φακέλου), **gid\_t** (περιέχει το id του group), **loff\_t** (μέγεθος αρχείου σε bytes), **struct timespec** (τελευταίος χρόνος τροποποίησης/αλλαγής), **umode\_t i\_mode** (αφορά τα δικαιώματα πρόσβασης). )

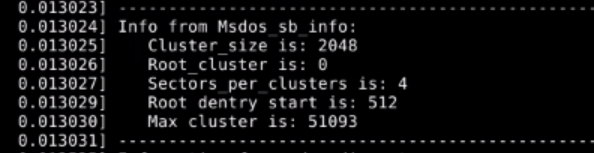
*Για το πρώτο ερώτημα:* Χρησιμοποιήσαμε το printk καθώς και το gdb, για να ελέγξουμε τα δεδομένα που βρίσκονται στις δομές αποθήκευσης στον δίσκο του FAT.

Προσθέτουμε printk στο αρχείο /lkl-source/fs/fat/inode.c στην συνάρτηση \_\_fat\_write\_inode Γραμμές 862-866, στην συνάρτηση fat\_read\_root Γραμμές 1687-1402, στην συνάρτηση fat\_read\_bpb Γραμμές 1.

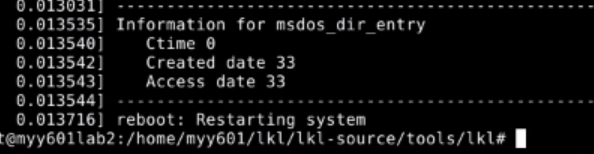
* Για το fat\_boot\_sector: Ο αριθμός των fat ισούται με 2 και το μήκος του Fat ισούται με 200, όπως φαίνεται στην Εικόνα:



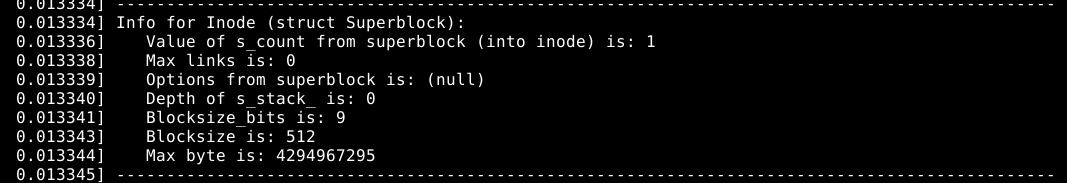
* Για το msdos\_sb\_info: Το μέγεθος του cluster είναι 2048, το root\_cluster είναι 0, τα sectors ανά cluster είναι 4, η αρχή του root dentry είναι στο 512 και το μέγιστο cluster είναι 51093.



* Για το msdos\_dir\_entry: Ο χρόνος δημιουργίας είναι 0, η ημερομηνία δημιουργίας είναι 33 και η ημερομηνία τελευταίας πρόσβασης είναι 33.



* Για το inode: Η τιμή του s\_count ισούται με 1, ο μέγιστος αριθμός των links είναι ίσος με 0, το options είναι NULL, το βάθος της στοίβας είναι 0, το μέγεθος του block σε bits είναι 9, το μέγεθος του block είναι 512 και ο μέγιστος αριθμός byte είναι 4294967295.

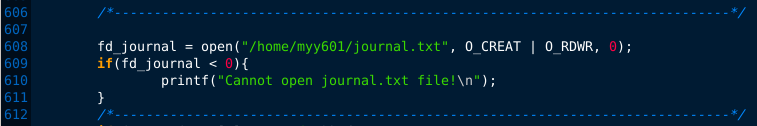


Τα αποτελέσματα των printk, εκτυπώνονται κατά την κλήση της **./cptofs -i /tmp/vfatfile -p -t vfat lklfuse.c /**.

*Για το δεύτερο ερώτημα:* Δημιουργία journal αρχείο σε επίπεδο χρήστη.

Προσπαθήσαμε να δημιουργήσουμε το αρχείο στο **/tools/lkl/tests/boot.c**, με το σκεπτικό ότι κάθε φορά θα θέλουμε να χρησιμοποιήσουμε τις εφαρμογές (**cptofs, cpfromfs**) θα πρέπει να κάνουμε boot το σύστημα, άρα πάντα θα δημιουργείται το αρχείο journal στην αρχή.  
Το πρόβλημα που προκύπτει όμως, είναι ότι θέλουμε να κάνουμε write στην εφαρμογή cptofs, η οποία εκτελείται μετά την ολοκλήρωση της εφαρμογής boot και αυτό μας εμποδίζει να γράψουμε στο αρχείο.

Μία λύση θα ήταν να δημιουργούμε το αρχείο στην cptofs, μέσα στην οποία θα κάνουμε και write. Αρχικοποιούμε τον file descriptor στο αρχείο κεφαλίδας **/tools/lkl/include/lkl\_host.c**, **Γραμμή 56** (**int fd\_journal;**).



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Κάτι θέλει να βάλουμε εδώ

Βέβαια, η λύση αυτή δεν είναι η ιδανική λύση, καθώς για να δημιουργηθεί το αρχείο, είναι αναγκαίο να εκτελέσουμε την **cptofs**, η οποία καλεί λειτουργίες αντιγραφής.

Θα θέλαμε η open να πραγματοποιείται σε μία !!!!!!!!!!!!!!!!!!!!