Υλοποίηση αρχείου καταγραφής στο σύστημα αρχείων VFAT του Linux

Δημητριάδης Γεώργιος - ΑΜ: 5209
Δημητρίου Αριστοτέλης - ΑΜ: 5211
Τζιάσιος Κίμων - ΑΜ: 5365

ΜΥΥ601 Λειτουργικά Συστήματα
Εαρινό Εξάμηνο 2024

Πίνακας περιεχομένων

Εισαγωγή **3**

Περιγραφή της LKL 3

Σύστημα αρχείων VFAT **4**

Χαρακτηριστικά του VFAT: 5

Χρήσεις και εφαρμογές 6

Περιορισμοί 6

Δομές 6

Τοποθέτηση printk **10**

Αποτελέσματα **16**

Journaling **21**

Πλεονεκτήματα του journaling 21

Μειονεκτήματα του journaling 21

FAT και Journaling 22

Συμπεράσματα 22

Υλοποίηση Journaling **23**

Superblock 25

File allocate table 28

Directory 29

Επαλήθευση κώδικα **30**

ΣΗΜΕΙΩΣΗ: ΠΑΡΑΤΙΘΕΝΤΑΙ 2 ΦΑΚΕΛΟΙ.

1. **Ο ΦΑΚΕΛΟΣ (only printk) ΠΕΡΙΕΧΕΙ ΜΟΝΟ ΤΟ ΕΡΩΤΗΜΑ ΜΕ ΤΑ PRINTK.**
2. **Ο ΦΑΚΕΛΟΣ (journal try) ΠΕΡΙΕΧΕΙ ΚΑΙ ΤΟ Β ΜΕΡΟΣ ΠΟΥ ΔΕΝ ΤΡΕΧΕΙ ΣΩΣΤΑ**

ΕΙΣΑΓΩΓΗ

Σε αυτήν την εργαστηριακή άσκηση μας δίνεται ο πηγαίος κώδικας του πυρήνα του Linux με την μορφή βιβλιοθήκης. Διασυνδέοντας την βιβλιοθήκη με μια εφαρμογή μπορούμε να τρέξουμε τον πυρήνα του Linux σε επίπεδο χρήστη. Στην παρούσα άσκηση θα επεκτείνουμε το σύστημα αρχείων VFAT στην βιβλιοθήκη του πυρήνα Linux προκειμένου να παρέχουμε αρχείο καταγραφής (journaling) που αποθηκεύει τροποποιήσεις που συμβαίνουν στο σύστημα αρχείων κατά την εκτέλεση εφαρμογών.

Περιγραφή της LKL:

Η Linux Kernel Library (LKL) αποτελεί μια κρίσιμη πτυχή του πυρήνα του λειτουργικού συστήματος Linux. Αποτελείται από μια σειρά από βιβλιοθήκες, αρχεία κεφαλίδων, κώδικα και δομές δεδομένων που υποστηρίζουν τις βασικές λειτουργίες του πυρήνα. Ορισμένα από τα σημαντικότερα στοιχεία και τα οφέλη της LKL είναι τα εξής:

- Διαχείριση Μνήμης: Η LKL περιλαμβάνει μηχανισμούς διαχείρισης μνήμης, όπως το σύστημα διαχείρισης εικονικής μνήμης, που επιτρέπει τη διαμόρφωση της μνήμης του συστήματος σύμφωνα με τις ανάγκες των διαδικασιών. Περιλαμβάνει επίσης κώδικα για την κατανομή μνήμης (malloc, free) και τη διαχείριση σελίδων.
- Συστήματα Αρχείων: Η LKL περιλαμβάνει υποστηρικτικό κώδικα για διάφορα συστήματα αρχείων, όπως ext4, XFS, και άλλους τύπους συστημάτων. Προσφέρει επίσης διεπαφές για την πρόσβαση στα συστήματα αρχείων, συμπεριλαμβανομένων των εντολών για τη δημιουργία, διαγραφή και διαχείριση αρχείων.
- Διαχείριση Διεργασιών: Οι βιβλιοθήκες του πυρήνα παρέχουν τα εργαλεία για τη δημιουργία, διαχείριση και προγραμματισμό διεργασιών. Αυτό περιλαμβάνει τον προγραμματιστή εργασιών (scheduler), ο οποίος αποφασίζει πώς κατανέμονται οι διαθέσιμοι πόροι του συστήματος μεταξύ των διεργασιών.
- Συσκευές και Είσοδος/Εξοδος (I/O): Η LKL υποστηρίζει τη διαχείριση διαφόρων συσκευών υλικού, όπως δίσκους, κάρτες

δικτύου, και άλλες περιφερειακές συσκευές. Περιλαμβάνει επίσης κώδικα για τον χειρισμό λειτουργιών εισόδου και εξόδου μεταξύ του λειτουργικού συστήματος και των συσκευών αυτών.

- Συγχρονισμός και Πολυνηματικότητα: Η LKL περιλαμβάνει μηχανισμούς για τον συγχρονισμό των διεργασιών και των νημάτων που εκτελούνται ταυτόχρονα. Αυτό περιλαμβάνει mutexes, semaphores, και άλλες δομές για την πρόληψη ανταγωνιστικών καταστάσεων.
- Διεπαφές Προγραμματισμού Εφαρμογών (APIs): Προσφέρει APIs που επιτρέπουν στους προγραμματιστές να αλληλεπιδρούν με τον πυρήνα και τις λειτουργίες του. Αυτά τα APIs καθιστούν ευκολότερο να γράφονται προγράμματα που εκτελούνται σε επίπεδο συστήματος.
- Σταθερότητα και Ασφάλεια: Η LKL συμβάλλει στη σταθερότητα του συστήματος παρέχοντας δοκιμασμένες και αξιόπιστες ρουτίνες, καθώς και εργαλεία για την αντιμετώπιση σφαλμάτων και επιθέσεων ασφάλειας.
- Επεκτασιμότητα και Υποστήριξη Υλικού: Παρέχει υποδομές που επιτρέπουν την εύκολη προσθήκη νέων χαρακτηριστικών και την υποστήριξη νέου υλικού. Η επεκτασιμότητα αυτή διευκολύνει την προσαρμογή του Linux σε διάφορες συσκευές και περιβάλλοντα.

Συνολικά, η Linux Kernel Library είναι απαραίτητη για την αποτελεσματική λειτουργία του πυρήνα και του λειτουργικού συστήματος. Παρέχει τα εργαλεία και τους μηχανισμούς για να διασφαλιστεί η αποτελεσματική διαχείριση πόρων και η σωστή αλληλεπίδραση μεταξύ λογισμικού και υλικού, επιτρέποντας στο Linux να παρέχει μια σταθερή και αξιόπιστη εμπειρία χρήστη.

Σύστημα αρχείων VFAT

Το VFAT (Virtual File Allocation Table) είναι ένα σύστημα αρχείων που αποτελεί μια εξέλιξη του FAT (File Allocation Table) που χρησιμοποιείται στα λειτουργικά συστήματα της Microsoft. Το VFAT εισήχθη για πρώτη φορά στα Windows 95 και είναι μια επέκταση του συστήματος αρχείων FAT για την υποστήριξη εκτενή ονομάτων αρχείων.

Χαρακτηριστικά του VFAT

- Εκτενή Ονόματα Αρχείων: Ένα από τα βασικά χαρακτηριστικά του VFAT είναι η δυνατότητα να υποστηρίζει εκτενή ονόματα αρχείων, αντί για τα οκτώ χαρακτήρες (συν τριών χαρακτήρων επέκτασης) που ήταν χαρακτηριστικά του FAT.
- Υποστήριξη UTF-16: Για την αποθήκευση των εκτενών ονομάτων αρχείων, το VFAT χρησιμοποιεί κωδικοποίηση UTF-16, επιτρέποντας την υποστήριξη πολλών διαφορετικών γλωσσών και συμβόλων στα ονόματα αρχείων.
- Συμβατότητα με FAT: Το VFAT είναι συμβατό με τα παλαιότερα συστήματα αρχείων FAT12 και FAT16. Η συμβατότητα αυτή σημαίνει ότι οι συσκευές που υποστηρίζουν VFAT μπορούν να διαβάζουν και να γράφουν στα παλαιότερα συστήματα αρχείων FAT χωρίς προβλήματα.
- Αρχειοθέτηση (Attribute Bits): Όπως και το FAT, το VFAT χρησιμοποιεί αρχειοθέτηση για τα αρχεία, επιτρέποντας τη διαχείριση χαρακτηριστικών όπως το αν ένα αρχείο είναι μόνο για ανάγνωση, κρυφό, συστηματικό, κ.ά.
- Διάρθρωση Κομματιών (Clusters): Ο διαμερισμός των αρχείων στο VFAT εξακολουθεί να βασίζεται σε κομμάτια (clusters), όπως και στα παλαιότερα συστήματα αρχείων FAT. Αυτό επιτρέπει την αποδοτική κατανομή χώρου στον δίσκο.

Χρήσεις και Εφαρμογές

Το VFAT χρησιμοποιείται κυρίως σε αποθηκευτικά μέσα όπως κάρτες SD, USB drives, και άλλους φορητούς δίσκους, καθώς παρέχει μεγάλη συμβατότητα με μια ποικιλία συσκευών και λειτουργικών συστημάτων. Λόγω της ευρείας χρήσης του VFAT σε φορητά μέσα αποθήκευσης, έχει γίνει ένα από τα πιο κοινά συστήματα αρχείων για ανταλλαγή δεδομένων μεταξύ διαφορετικών λειτουργικών συστημάτων και συσκευών.

Περιορισμοί

Παρόλο που το VFAT παρέχει συμβατότητα και ευελιξία, δεν είναι τόσο ισχυρό όσο τα νεότερα συστήματα αρχείων, όπως το NTFS ή το ext4, όσον αφορά την αποδοτικότητα και την υποστήριξη σύγχρονων χαρακτηριστικών, όπως τα δικαιώματα πρόσβασης αρχείων. Συνολικά, το VFAT αποτελεί ένα πρακτικό και συμβατό σύστημα αρχείων που διευκολύνει την ανταλλαγή δεδομένων μεταξύ διαφορετικών συσκευών και λειτουργικών συστημάτων, ενώ παράλληλα υποστηρίζει εκτενή ονόματα αρχείων.

Δομές

Στο λειτουργικό σύστημα Linux, το Virtual File System (VFS) λειτουργεί ως ένα ενδιάμεσο στρώμα που επιτρέπει τη συμβατότητα μεταξύ εφαρμογών και διαφόρων τύπων συστημάτων αρχείων, παρέχοντας μια ενιαία διεπαφή. Οι κύριες δομές δεδομένων που χρησιμοποιούνται στο VFS είναι:

- Superblock: Αυτή η δομή ελέγχει και διαχειρίζεται ένα σύστημα αρχείων. Κάθε σύστημα αρχείων περιλαμβάνει το δικό του superblock, το οποίο δημιουργείται κατά τη μορφοποίηση. Περιλαμβάνει πληροφορίες όπως το μέγεθος του συστήματος αρχείων, τον τύπο του, τον αριθμό των inodes και των blocks, καθώς και άλλες σημαντικές λεπτομέρειες.
  - Η δομή super_block αποτελεί μια βασική δομή δεδομένων στο VFS του Linux και περιέχει πληροφορίες για ένα σύστημα αρχείων. Αναφέρεται σε όλες τις σημαντικές παραμέτρους που

αφορούν τη διαχείριση και τη λειτουργία του συστήματος αρχείων.

- Η δομή msdos_sb_info αναφέρεται σε ένα σύνολο δεδομένων που χρησιμοποιείται στον οδηγό FAT για να αποθηκεύει επιπρόσθετες πληροφορίες σχετικά με το σύστημα αρχείων FAT όπως cluster_bits (Ο αριθμός των bit που χρησιμοποιείται για να αντιπροσωπεύει ένα cluster) και fats (Ο αριθμός των FAT που υπάρχουν).

- Inode (Index Node): Οι inodes περιέχουν πληροφορίες για μεμονωμένα αρχεία και καταλόγους, όπως δικαιώματα πρόσβασης, ιδιοκτησία, σήματα χρόνου για τη δημιουργία, τροποποίηση και πρόσβαση στα αρχεία, καθώς και την τοποθεσία των δεδομένων στον δίσκο. Αυτές οι δομές παρέχουν μια λεπτομερή περιγραφή της δομής και των ιδιοτήτων κάθε αρχείου. Αυτή είναι η γενική δομή inode στο VFS του Linux:
  - struct inode: Αυτή είναι η γενική δομή inode στο VFS του Linux.
  - msdos_inode_info: Η συγκεκριμένη δομή αποτελεί μια εξειδικευμένη δομή δεδομένων που σχετίζεται με το σύστημα αρχείων FAT στο Linux. Είναι παράγωγη της δομής inode και προσφέρει επιπλέον πληροφορίες και λειτουργίες που είναι συγκεκριμένες για το σύστημα αρχείων FAT, όπως cache_lru (μια λίστα που διατηρεί τα αντικείμενα cache με βάση την πολιτική LRU) και i_start (το σημείο έναρξης των δεδομένων του αρχείου).

- Dentry (Directory Entry): Η δομή Dentry συνδέει τα ονόματα των αρχείων με τα αντίστοιχα inodes. Αυτή η δομή αντιπροσωπεύει ένα στοιχείο σε έναν κατάλογο, όπως αρχείο, υποκατάλογο ή συμβολικός σύνδεσμος. Κάθε Dentry συνδέεται με έναν κατάλογο και έναν αριθμό inode, και λειτουργεί ως γέφυρα μεταξύ του ονόματος του αρχείου και της εσωτερικής αναπαράστασης από το σύστημα αρχείων.

- struct dentry: Είναι η γενική δομή dentry στο VFS του Linux.
- fat_slot_info: Η δομή αυτή αναφέρεται σε μια δομή δεδομένων που χρησιμοποιείται για να διαχειριστεί τις καταχωρίσεις καταλόγου στο σύστημα αρχείων FAT. Συγκεκριμένα, αποθηκεύει πληροφορίες που σχετίζονται με ένα σύνολο καταχωρίσεων καταλόγου (slots) που αντιπροσωπεύουν ένα αρχείο ή κατάλογο.

- File: Αυτή η δομή αντιπροσωπεύει ένα αρχείο που είναι ανοιχτό και περιέχει πληροφορίες όπως η τρέχουσα θέση της κεφαλής ανάγνωσης/εγγραφής, τα flags ανοίγματος του αρχείου και δείκτες προς τη σχετική inode και dentry. Η δομή File χειρίζεται ενέργειες εισόδου/εξόδου (I/O) στο αρχείο.
  - Η δομή file είναι μια θεμελιώδης δομή στο Linux που αντιπροσωπεύει ένα ανοιχτό αρχείο. Περιέχει πληροφορίες σχετικά με την τρέχουσα κατάσταση του αρχείου, όπως η θέση ανάγνωσης/εγγραφής, τα δικαιώματα πρόσβασης και οι λειτουργίες που μπορούν να πραγματοποιηθούν στο αρχείο.
- Address Space Object: Αυτή η δομή αντιπροσωπεύει τον χώρο διευθύνσεων ενός αρχείου στη μνήμη. Χρησιμοποιείται για να διαχειρίζεται τις λειτουργίες I/O της μνήμης, συμπεριλαμβανομένης της προσωρινής αποθήκευσης και της χαρτογράφησης της μνήμης. Αυτή η δομή βοηθά στη διαχείριση του τρόπου με τον οποίο τα δεδομένα του αρχείου φορτώνονται στη μνήμη και γράφονται πίσω στον δίσκο.
- Filesystem Type: Αυτή η δομή καθορίζει τις λειτουργίες που είναι συγκεκριμένες για κάθε τύπο συστήματος αρχείων, όπως η δημιουργία ή η λήψη superblocks. Παρέχει τις λειτουργίες που είναι αναγκαίες για τη διαχείριση του συστήματος αρχείων και επιτρέπει στο VFS να υποστηρίζει πολλούς διαφορετικούς τύπους συστημάτων αρχείων.
  - Η δομή msdos_dir_entry αντιπροσωπεύει μια καταχώριση καταλόγου στο σύστημα αρχείων FAT (FAT12, FAT16 ή FAT32). Κάθε αρχείο ή κατάλογος σε έναν κατάλογο αντιπροσωπεύεται από μια τέτοια καταχώριση. Η δομή

περιλαμβάνει διάφορα πεδία που περιέχουν πληροφορίες σχετικά με το αρχείο ή τον κατάλογο

- Η δομή msdos_dir_slot αντιπροσωπεύει μια καταχώριση καταλόγου που σχετίζεται με εκτενή ονόματα αρχείων στο σύστημα αρχείων FAT. Τα εκτενή ονόματα επιτρέπουν ονόματα αρχείων μεγαλύτερα από το παραδοσιακό όριο των 11 χαρακτήρων (8 για το όνομα και 3 για την επέκταση)
- Η δομή fat_boot_sector αντιπροσωπεύει τον τομέα εκκίνησης (boot sector) στο σύστημα αρχείων FAT (FAT12, FAT16 ή FAT32). Ο τομέας εκκίνησης περιέχει κρίσιμες πληροφορίες για τη δομή του συστήματος αρχείων και τον τρόπο με τον οποίο το λειτουργικό σύστημα πρέπει να διαχειρίζεται τα δεδομένα στο δίσκο
- Η δομή fat_boot_fsinfo αντιπροσωπεύει τον τομέα πληροφοριών συστήματος αρχείων (FSInfo sector) σε ένα σύστημα αρχείων FAT32. Ο τομέας πληροφοριών παρέχει πρόσθετες πληροφορίες σχετικά με την κατάσταση του συστήματος αρχείων, όπως η κατανομή ελεύθερων clusters και το επόμενο διαθέσιμο cluster.

Τοποθέτηση printk

Αρχικά θα τοποθετήσουμε printk μέσα σε βασικές λειτουργίες του inode.c, στις λειτουργίες της βασικής δομής **superblock**.

- Τοποθέτηση printk (γραμμή 656) στην συνάρτηση fat_evict_inode, η οποία είναι μέρος του driver συστήματος αρχείων FAT και χειρίζεται την απομάκρυνση (εκκαθάριση) ενός inode από το σύστημα αρχείων
- Τοποθέτηση printk (γραμμή 738) στη συνάρτηση fat_put_super, η οποία είναι μέρος του driver του συστήματος αρχείων FAT στο Linux και χειρίζεται την απελευθέρωση του superblock όταν το σύστημα αρχείων απεγκαθίσταται ή αποσυνδέεται.
- Τοποθέτηση printk (γραμμή 751) στη συνάρτηση fat_alloc_inode, η οποία είναι μέρος του driver του συστήματος αρχείων FAT στο Linux και είναι υπεύθυνη για τη δημιουργία ενός νέου inode για το σύστημα αρχείων FAT.
- Τοποθέτηση printk (γραμμή 813) στη συνάρτηση fat_remount, η οποία είναι μια λειτουργία που ανήκει στο driver του συστήματος αρχείων FAT στο Linux. Αυτή η συνάρτηση είναι υπεύθυνη για τον επαναπροσανατολισμό του συστήματος αρχείων και την ενημέρωση των σημείων πρόσδεσης με βάση νέα δεδομένα που παρέχονται.
- Τοποθέτηση printk (γραμμή 832) στη συνάρτηση fat_statfs, η οποία είναι μια λειτουργία που ανήκει στο driver του συστήματος αρχείων FAT στο Linux. Αυτή η συνάρτηση είναι υπεύθυνη για την επιστροφή πληροφοριών σχετικά με το σύστημα αρχείων, συμπεριλαμβανομένων των μετρήσεων ελεύθερου χώρου, του μεγέθους του συστήματος αρχείων και άλλων πληροφοριών.
- Τοποθέτηση printk (γραμμή 912) στη συνάρτηση fat_write_inode, η οποία είναι υπεύθυνη για τη διαδικασία εγγραφής ενός κόμβου inode του συστήματος αρχείων FAT. Αυτή η διαδικασία μπορεί να περιλαμβάνει είτε τη συγγραφή δεδομένων σε συγκεκριμένο inode είτε τη συγγραφή σε ένα ειδικό inode που περιέχει πληροφορίες σχετικά με το σύστημα αρχείων.
- Τοποθέτηση printk (γραμμή 950) στη συνάρτηση fat_show_options παρουσιάζει τις επιλογές που χρησιμοποιήθηκαν κατά το μοντάρισμα ενός συστήματος αρχείων FAT σε μια δομή

seq_file. Η δομή seq_file χρησιμοποιείται για να επιτρέπει τη δημιουργία αρχείων σε μορφή ακολουθίας. Η συνάρτηση παράγει μια συμβολοσειρά που περιέχει διάφορες επιλογές του συστήματος αρχείων.

Στη συνέχεια θα τοποθετήσουμε printk μέσα σε βασικές λειτουργίες του inode.c, στις **λειτουργίες της μνήμης**.

- Τοποθέτηση printk (γραμμή 189) στη συνάρτηση fat_get_block, η οποία είναι υπεύθυνη για τη χαρτογράφηση ενός iblock ενός inode σε έναν πραγματικό τομέα του συστήματος αρχείων FAT
- Τοποθέτηση printk (γραμμή 198) στη συνάρτηση fat_writepage, η οποία είναι υπεύθυνη για την εγγραφή των δεδομένων μιας σελίδας στο σύστημα αρχείων FAT. Συγκεκριμένα, η συνάρτηση χρησιμοποιείται για να γράψει δεδομένα μιας σελίδας στον δίσκο, χρησιμοποιώντας τη συνάρτηση block_write_full_page.
- Τοποθέτηση printk (γραμμή 205) στη συνάρτηση fat_writepages, η οποία είναι υπεύθυνη για τη διαχείριση της εγγραφής πολλών σελίδων από το σύστημα αρχείων FAT στον δίσκο. Πρόκειται για μια λειτουργία που χρησιμοποιείται για να γράψει ένα σύνολο σελίδων που σχετίζονται με μια συγκεκριμένη περιοχή μνήμης (mapping) στον δίσκο.
- Τοποθέτηση printk (γραμμή 236) στη συνάρτηση fat_write_begin, η οποία είναι υπεύθυνη για την έναρξη της εγγραφής δεδομένων σε μια σελίδα μνήμης που σχετίζεται με μια συγκεκριμένη θέση (pos) και μήκος (len) στον χώρο μνήμης (mapping).
- Τοποθέτηση printk (γραμμή 251) στη συνάρτηση fat_write_end, η οποία είναι υπεύθυνη για τον χειρισμό της ολοκλήρωσης μιας εγγραφής δεδομένων στο σύστημα αρχείων FAT. Αυτή η συνάρτηση εκτελείται μετά την προσπάθεια εγγραφής δεδομένων στη μνήμη και ελέγχει το αποτέλεσμα αυτής της εγγραφής.
- Τοποθέτηση printk (γραμμή 305) στη συνάρτηση fat_get_block_bmap, η οποία είναι υπεύθυνη για τη χαρτογράφηση iblock σε έναν αριθμό τομέα στο σύστημα αρχείων FAT, όταν καλείται από την λειτουργία bmap. Η συνάρτηση αυτή επιστρέφει τον χαρτογραφημένο τομέα στο bh_result buffer head. Αυτή είναι

μια συνάρτηση για τον χειρισμό χαρτογράφησης μπλοκ για λειτουργίες bmap στο σύστημα αρχείων FAT.

Έπειτα θα τοποθετήσουμε printk μέσα σε βασικές λειτουργίες του fatent.c, στις **λειτουργίες των εγγραφών FAT**.

- Τοποθέτηση printk (γραμμή 40) στη συνάρτηση fat_ent_blocknr, η οποία υπολογίζει την τοποθεσία ενός συγκεκριμένου entry (καταχώρισης) στο σύστημα αρχείων FAT και επιστρέφει τον αριθμό του τομέα (blocknr) και το offset εντός αυτού του τομέα. Η συνάρτηση χρησιμοποιείται για να χαρτογραφήσει μία καταχώριση στο σύστημα αρχείων σε έναν φυσικό τομέα.
- Τοποθέτηση printk (γραμμή 69) στη συνάρτηση fat32_ent_set_ptr, η οποία έχει ως στόχο να ρυθμίσει τον δείκτη (ent32_p) της δομής fat_entry (fatent) ώστε να δείχνει σε ένα συγκεκριμένο σημείο εντός ενός buffer. Αυτό το σημείο καθορίζεται από το offset, το οποίο είναι το offset εντός των δεδομένων (b_data) του buffer (bhs[0]) της δομής fat_entry. Η δομή fat_entry χρησιμοποιείται στο σύστημα αρχείων FAT32 για να διαχειρίζεται τους δείκτες των καταχωρίσεων των αρχείων.
- Τοποθέτηση printk (γραμμή 110) στη συνάρτηση fat_ent_bread, η οποία είναι υπεύθυνη για την απόκτηση της επόμενης καταχώρησης στον πίνακα κατανομής αρχείων (FAT) για έναν δεδομένο τομέα FAT32
- Τοποθέτηση printk (γραμμή 153) στη συνάρτηση fat32_ent_get, η οποία είναι υπεύθυνη για την απόκτηση της επόμενης καταχώρησης στον πίνακα κατανομής αρχείων (FAT) για έναν δεδομένο τομέα FAT32
- Τοποθέτηση printk (γραμμή 192) στη συνάρτηση fat32_ent_put, η οποία είναι υπεύθυνη για την ενημέρωση μιας καταχώρησης στον πίνακα κατανομής αρχείων (FAT) για ένα δεδομένο τομέα FAT32
- Τοποθέτηση printk (γραμμή 248) στη συνάρτηση fat32_ent_next, η οποία εκτελεί την πρόοδο στον πίνακα κατανομής αρχείων (FAT) κατά τη διάρκεια της επεξεργασίας των καταχωρίσεων FAT32

Όμοια για fat12 και fat16:

- Γραμμή 29 printk στην συνάρτηση fat12_ent_blocknr
- Γραμμή 49 printk στην συνάρτηση fat12_set_ptr
- Γραμμή 62 printk στην συνάρτηση fat16_ent_set_ptr
- Γραμμή 78 printk στην συνάρτηση fat12_ent_bread
- Γραμμή 128 printk στην συνάρτηση fat12_ent_get
- Γραμμή 144 printk στην συνάρτηση fat16_ent_get
- Γραμμή 162 printk στην συνάρτηση fat12_ent_put
- Γραμμή 182 printk στην συνάρτηση fat16_ent_put
- Γραμμή 205 printk στην συνάρτηση fat12_ent_next
- Γραμμή 236 printk στην συνάρτηση fat16_ent_next

Έπειτα θα τοποθετήσουμε printk μέσα σε βασικές λειτουργίες του namei_vfat.c, για **τις λειτουργίες των καταλόγων**.

- Τοποθέτηση printk (γραμμή 709) στη συνάρτηση vfat_lookup, η οποία είναι υπεύθυνη για τον εντοπισμό και την επεξεργασία ενός στοιχείου σε έναν κατάλογο στο σύστημα αρχείων VFAT. Η συνάρτηση αυτή λαμβάνει ως είσοδο έναν δείκτη στον κατάλογο dir, μια καταχώριση dentry, και μια μεταβλητή σημαίας flags.
- Τοποθέτηση printk (γραμμή 767) στη συνάρτηση vfat_create είναι υπεύθυνη για τη δημιουργία ενός νέου αρχείου ή καταλόγου σε ένα κατάλογο στο σύστημα αρχείων VFAT. Δέχεται ως είσοδο έναν δείκτη στον κατάλογο dir, μια καταχώριση dentry που αντιπροσωπεύει το νέο αρχείο/κατάλογο προς δημιουργία, έναν umode_t mode που προσδιορίζει τα δικαιώματα πρόσβασης, και μια τιμή bool excl που υποδεικνύει αν το αρχείο πρέπει να δημιουργηθεί αποκλειστικά.
- Τοποθέτηση printk (γραμμή 796) στη συνάρτηση vfat_rmdir, η οποία είναι υπεύθυνη για την κατάργηση ενός καταλόγου (dentry) από ένα κατάλογο (dir) σε ένα σύστημα αρχείων VFAT. Έχει ως είσοδο έναν δείκτη στον κατάλογο dir και μια καταχώριση dentry που αντιπροσωπεύει τον κατάλογο προς διαγραφή.
- Τοποθέτηση printk (γραμμή 827) στη συνάρτηση vfat_unlink, η οποία είναι υπεύθυνη για την διαγραφή ενός στοιχείου από έναν

κατάλογο στο σύστημα αρχείων VFAT. Η συνάρτηση δέχεται ως είσοδο έναν δείκτη στον κατάλογο dir και μια καταχώριση dentry που αντιπροσωπεύει το στοιχείο προς διαγραφή.

- Τοποθέτηση printk (γραμμή 855) στη συνάρτηση vfat_mkdir, η οποία είναι υπεύθυνη για τη δημιουργία ενός νέου καταλόγου (dentry) μέσα σε έναν υπάρχοντα κατάλογο (dir) στο σύστημα αρχείων VFAT. Λαμβάνει τις παραμέτρους mnt_userns (αντιπροσωπεύει τον χώρο χρήστη), dir (τον υπάρχοντα κατάλογο), dentry (τον νέο κατάλογο προς δημιουργία), και mode (τις άδειες πρόσβασης).

Αξίζει να παρατηρήσει κανείς προσεκτικά τη δομή struct file_operations fat_file_operations καθώς εκεί μέσα ορίζονται λειτουργίες αρχείου. Πιο συγκεκριμένα:

- Τοποθέτηση printk (γραμμή 147) στη συνάρτηση generic_file_llseek στο αρχείο *read_write.c*, η οποία είναι υπεύθυνη για την επανατοποθέτηση του δείκτη αρχείου (file) στο νέο σημείο (offset) που καθορίζεται από τη λειτουργία whence (αρχή αρχείου, τρέχουσα θέση, ή τέλος αρχείου). Ονομάζει μια άλλη συνάρτηση generic_file_llseek_size και της περνάει το μέγιστο επιτρεπτό μέγεθος (s_maxbytes) για το αρχείο στο σύστημα αρχείων και το τρέχον μέγεθος του αρχείου (i_size_read).
- Τοποθέτηση printk (γραμμή 160) στη συνάρτηση fat_generic_ioctl στο αρχείο file.c είναι μια υλοποίηση της λειτουργίας ioctl για αρχεία στο σύστημα αρχείων FAT. Αυτή η συνάρτηση επιτρέπει την εκτέλεση διαφόρων εντολών που σχετίζονται με αρχεία σε αυτό το σύστημα αρχείων μέσω του παραμέτρου cmd και των αντίστοιχων δεδομένων arg.
- Τοποθέτηση printk (γραμμή 177) στη συνάρτηση fat_file_release στο αρχείο *file.c*. Πρόκειται για μια λειτουργία απελευθέρωσης αρχείου για αρχεία στο σύστημα αρχείων FAT. Όταν καλείται, αυτή η συνάρτηση εκτελούνται τις απαραίτητες ενέργειες για να απελευθερωθεί το αρχείο μετά τη χρήση του.
- Τοποθέτηση printk (γραμμή 191) στη συνάρτηση fat_file_fsync στο file.c είναι μια λειτουργία συγχρονισμού αρχείου (fsync) για

αρχεία στο σύστημα αρχείων FAT. Αυτή η συνάρτηση είναι υπεύθυνη για τη διασφάλιση ότι τα δεδομένα ενός αρχείου έχουν συγχρονιστεί με τον δίσκο και ότι οι αλλαγές στα δεδομένα έχουν εφαρμοστεί.

- Τοποθέτηση printk (γραμμή 271) στη συνάρτηση fat_fallocate στο file.c η οποία είναι υπεύθυνη για την κατανομή χώρου αρχείων στο σύστημα αρχείων FAT. Εξασφαλίζει ότι το αρχείο έχει επαρκή χώρο δίσκου για να καλύψει την προδιαγραφόμενη περιοχή (offset και len) στο αρχείο.
- Τοποθέτηση printk (γραμμή 2775) στο filemap.c στη συνάρτηση generic_file_read_iter, η οποία είναι υπεύθυνη για την ανάγνωση δεδομένων από ένα αρχείο σε ένα σύστημα αρχείων.
- Τοποθέτηση printk (γραμμή 3450) στο filemap.c στη συνάρτηση generic_file_mmap, η οποία είναι υπεύθυνη για το χειρισμό της χαρτογράφησης ενός αρχείου σε μια εικονική περιοχή μνήμης (VM) στο σύστημα αρχείων.
- Τοποθέτηση printk (γραμμή 3908) στο filemap.c στη συνάρτηση generic_file_write_iter είναι υπεύθυνη για την εγγραφή δεδομένων σε ένα αρχείο στο σύστημα αρχείων

Τέλος θα τοποθετήσουμε printk μέσα σε βασικές συναρτήσεις του file.c, για **τις λειτουργίες inode**.

- Τοποθέτηση printk (γραμμή 404) στη συνάρτηση fat_getattr, η οποία είναι υπεύθυνη για την απόκτηση των χαρακτηριστικών ενός αρχείου σε ένα σύστημα αρχείων FAT και την επιστροφή των πληροφοριών στην δομή kstat.
- Τοποθέτηση printk (γραμμή 488) στη συνάρτηση fat_setattr, η οποία είναι υπεύθυνη για την ρύθμιση των χαρακτηριστικών ενός αρχείου σε ένα σύστημα αρχείων FAT, όπως το μέγεθος, το δικαίωμα πρόσβασης, ο χρόνος τελευταίας πρόσβασης, δημιουργίας και τροποποίησης, και το UID/GID του ιδιοκτήτη.

ΑΠΟΤΕΛΕΣΜΑΤΑ

Αφού ακολουθήσαμε πιστά τις οδηγίες της άσκησης και μεταγλωττίσαμε με -j8, μπήκαμε στον κατάλογο lkl/tools/tests και εκτελέσαμε απευθείας την εφαρμογή ./disk.sh -t vfat. Ουσιαστικά δημιουργείται το αρχείο δίσκου /tmp/disk και το κάνει mount πριν κάνει δοκιμαστικές λειτουργίες. Επίσης εκτελέσαμε για έλεγχο την εντολή ./disk -t vfat -d /tmp/disk και μας εμφανίστηκαν τα αποτελέσματα που φαίνονται παρακάτω.

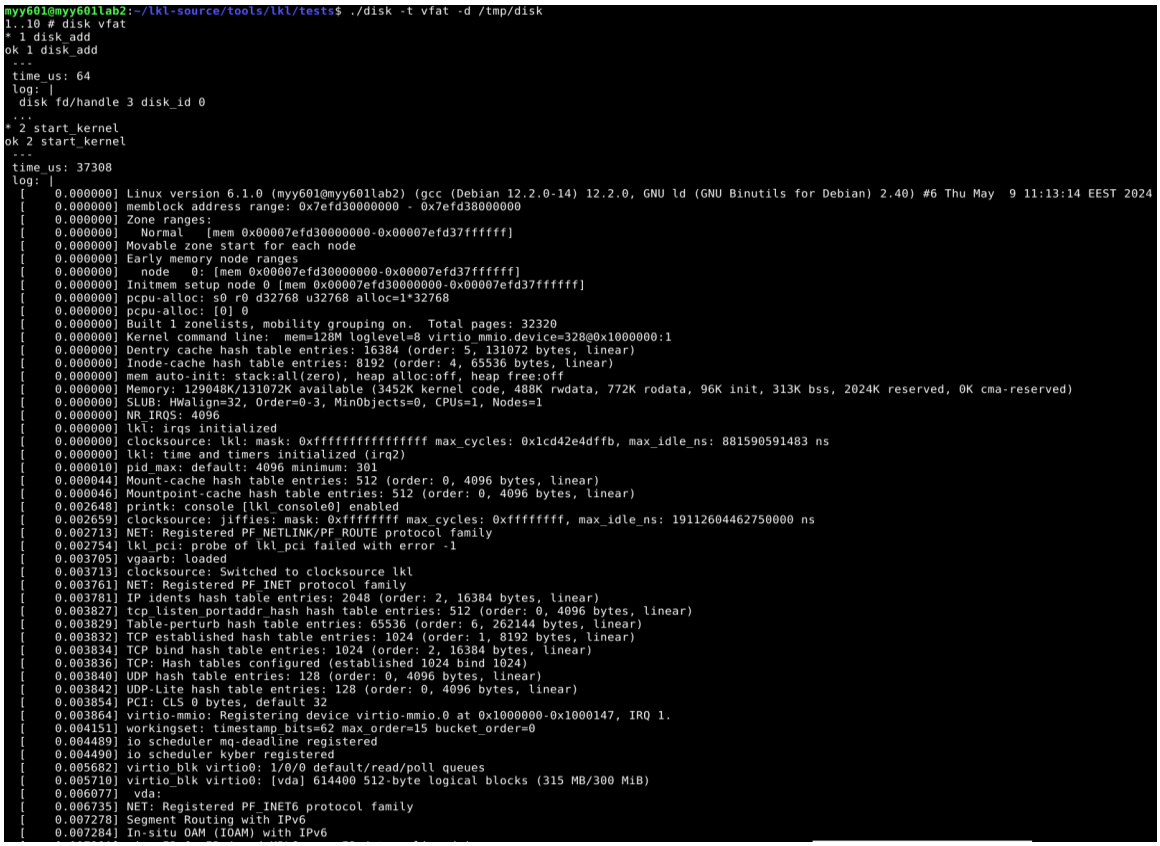

Figure 1: Έλεγχος εντολής ./disk -t vfat -d /tmp/disk

```
  [    0.007686]   with arguments:
  [    0.007687]     /init
  [    0.007687]   with environment:
  [    0.007688]     HOME=/
  [    0.007688]     TERM=linux
  [    0.007692] FILES_FUNCTION:generic_file_read_iter
  lkl_start_kernel("mem=128M loglevel=8") = 0
 ...
* 3 mount_dev
ok 3 mount_dev
 ---
 time_us: 2159
 log: |
  [    0.008665] SUPERBLOCK_FUNCTION:fat_alloc_inode
  [    0.008669] SUPERBLOCK_FUNCTION:fat_alloc_inode
  [    0.008669] SUPERBLOCK_FUNCTION:fat_alloc_inode
  lkl_mount_dev(disk_id, cla.partition, cla.fstype, 0, NULL, mnt_point, sizeof(mnt_point)) = 0
 ...
* 4 chdir_mnt_point
ok 4 chdir_mnt_point
 ---
 time_us: 4
 log: |
  lkl_sys_chdir(mnt_point) = 0
 ...
* 5 opendir
ok 5 opendir
 ---
 time_us: 4
 log: |
  lkl_opendir(/mnt/0000fe00) = 0 Success
 ...
* 6 readdir
ok 6 readdir
 ---
 time_us: 7
 log: |
  . ..
 ...
* 7 closedir
ok 7 closedir
 ---
 time_us: 2
 log: |
  lkl_closedir(dir) = 0
 ...
* 8 umount_dev
ok 8 umount_dev
 ---
 time_us: 333
 log: |
  0 0
 ...
* 9 stop_kernel
ok 9 stop_kernel
 ---
 time_us: 1940
 log: |
  [    0.023025] reboot: Restarting system
  lkl_sys_halt() = 0
 ...
* 10 disk_remove
ok 10 disk_remove
 ---
 time_us: 8
 log: |
```

Figure 2: Έλεγχος εντολής ./disk -t vfat -d /tmp/disk

Στη συνέχεια μεταβαίνουμε στον κατάλογο lkl-source/tools/lkl που εμπεριέχονται οι εφαρμογές cptofs και cpfromfs, οι οποίες μας επιτρέπουν την μετακίνηση των αρχείων ή και καταλόγων μεταξύ του συστήματος αρχείων της εικονικής μηχανής και του συστήματος αρχείων της lkl. Έπειτα εκτελούμε την εντολή ./cptofs -i /tmp/disk -t vfat lklfuse.o /, που θα αντιγράψει το αρχείο lklfuse.c στον κατάλογο / του /tmp/disk. Τέλος για να δούμε τα printk που βάλαμε στις βασικές

λειτουργίες εκτελούμε ./cptofs -i /tmp/disk -p -t vfat lklfuse.c / και παρατηρούμε όλα τα superblock function, fat write function.

```
myy601@myy601lab2:~/lkl-source/tools/lkl$ ./cptofs -i /tmp/disk -p -t vfat lklfuse.c /
[    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) #6 Thu May  9 11:13:14 EEST 2024
[    0.000000] memblock address range: 0x7ff661c00000 - 0x7ff668000000
[    0.000000] Zone ranges:
[    0.000000]   Normal   [mem 0x00007ff661c00000-0x00007ff667ffffff]
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x00007ff661c00000-0x00007ff667ffffff]
[    0.000000] Initmem setup node 0 [mem 0x00007ff661c00000-0x00007ff667ffffff]
[    0.000000] Built 1 zonelists, mobility grouping on.  Total pages: 25250
[    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 (3452K kernel code, 488K rwdata, 771K rodata, 99K 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: 0x1cd42e4dffb, max_idle_ns: 881590591483 ns
[    0.000000] lkl: time and timers initialized (irq2)
[    0.000011] pid_max: default: 4096 minimum: 301
[    0.000105] Mount-cache hash table entries: 512 (order: 0, 4096 bytes, linear)
[    0.000107] Mountpoint-cache hash table entries: 512 (order: 0, 4096 bytes, linear)
[    0.003075] printk: console [lkl_console0] enabled
[    0.003087] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 19112604462750000 ns
[    0.003143] NET: Registered PF_NETLINK/PF_ROUTE protocol family
[    0.003193] lkl_pci: probe of lkl_pci failed with error -1
[    0.003925] vgaarb: loaded
[    0.003953] clocksource: Switched to clocksource lkl
[    0.004013] NET: Registered PF_INET protocol family
[    0.004057] IP idents hash table entries: 2048 (order: 2, 16384 bytes, linear)
[    0.004127] tcp_listen_portaddr_hash hash table entries: 512 (order: 0, 4096 bytes, linear)
[    0.004144] Table-perturb hash table entries: 65536 (order: 6, 262144 bytes, linear)
[    0.004148] TCP established hash table entries: 1024 (order: 1, 8192 bytes, linear)
[    0.004151] TCP bind hash table entries: 1024 (order: 2, 16384 bytes, linear)
[    0.004154] TCP: Hash tables configured (established 1024 bind 1024)
[    0.004161] UDP hash table entries: 128 (order: 0, 4096 bytes, linear)
[    0.004164] UDP-Lite hash table entries: 128 (order: 0, 4096 bytes, linear)
[    0.004180] PCI: CLS 0 bytes, default 32
[    0.004208] virtio-mmio: Registering device virtio-mmio.0 at 0x1000000-0x1000147, IRQ 1.
[    0.004529] workingset: timestamp_bits=62 max_order=15 bucket_order=0
[    0.004882] io scheduler mq-deadline registered
[    0.004897] io scheduler kyber registered
[    0.006117] virtio_blk virtio0: 1/0/0 default/read/poll queues
[    0.006165] virtio_blk virtio0: [vda] 614400 512-byte logical blocks (315 MB/300 MiB)
[    0.006494]  vda:
[    0.007100] NET: Registered PF_INET6 protocol family
[    0.007618] Segment Routing with IPv6
[    0.007640] In-situ OAM (IOAM) with IPv6
[    0.007650] sit: IPv6, IPv4 and MPLS over IPv4 tunneling driver
[    0.008036] Warning: unable to open an initial console.
[    0.008054] This architecture does not have kernel memory protection.
[    0.008057] Run /init as init process
[    0.008070] FILES_FUNCTION:generic_file_read_iter
[    0.008416] SUPERBLOCK_FUNCTION:fat_alloc_inode
[    0.008434] SUPERBLOCK_FUNCTION:fat_alloc_inode
[    0.008436] SUPERBLOCK_FUNCTION:fat_alloc_inode
[    0.012697] DIRECTORIES_FUNCTION:vfat_lookup
[    0.012729] DIRECTORIES_FUNCTION:vfat_create
[    0.012740] SUPERBLOCK_FUNCTION:fat_alloc_inode
[    0.016962] FILES_FUNCTION:generic_file_write_iter
[    0.016995] SUPERBLOCK_FUNCTION:fat_write_begin
[    0.017011] SUPERBLOCK_FUNCTION:fat_get_block
[    0.017021] FAT_WRITE_FUNCTION:fat_ent_blocknr
[    0.017030] FAT_WRITE_FUNCTION:fat_ent_bread
[    0.017228] FAT_WRITE_FUNCTION:fat16_set_ptr
```

Figure 3: Έλεγχος εντολής ./cptofs -i /tmp/disk -p -t vfat lklfuse.c /

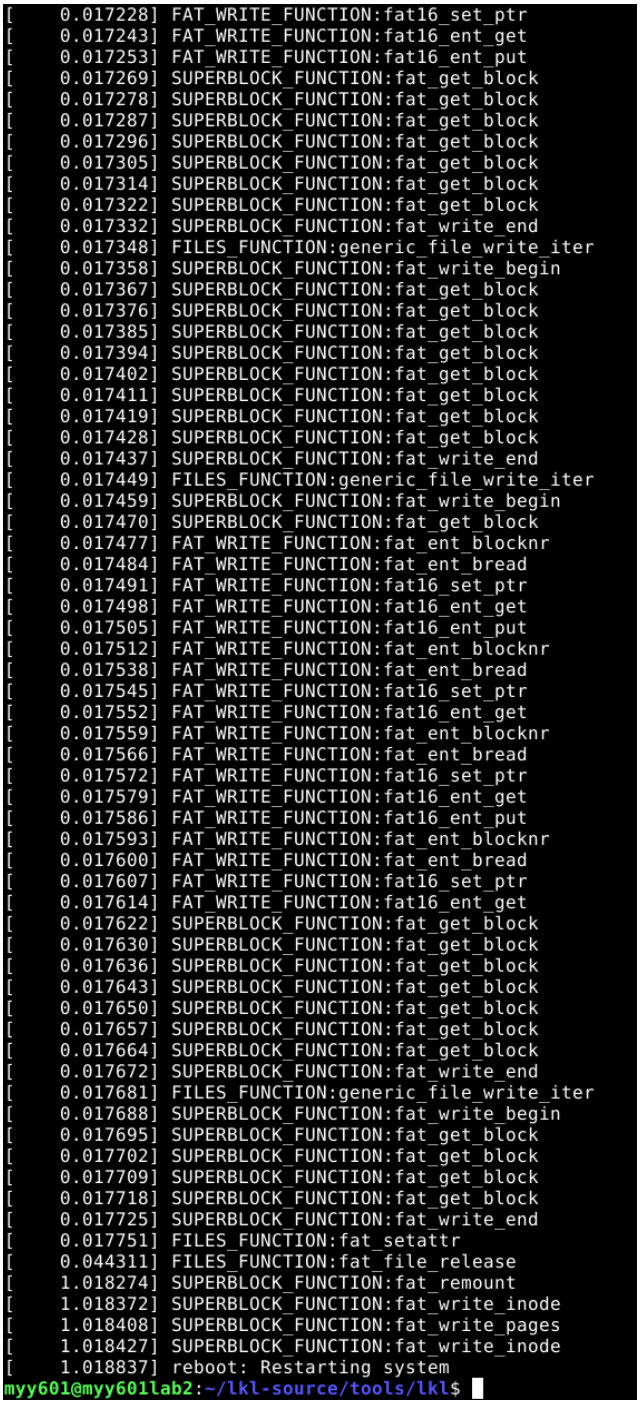

Figure 4: Έλεγχος εντολής

Δεν θα ήταν πλεονασμός να σχολιάζαμε την εντολή ./cptofs -i /tmp/disk -p -t vfat lklfuse.c / καθώς το τελευταίο / καθορίζει τον προορισμό της λειτουργίας αντιγραφής, δηλαδή τον κατάλογο root του συστήματος vfat, το οποίο το καθορίζουμε πάλι πάνω στην εντολή με το -t vfat lklfuse.c /.

Αφού δημιουργήσαμε έναν κενό φάκελο στο / home/myy601 με όνομα testfolder, εκτελέσαμε την εντολή ./cptofs -i /tmp/disk -p -t vfat testfolder / και παρατηρήσαμε και το files function από τα printk που έχουμε τοποθετήσει.

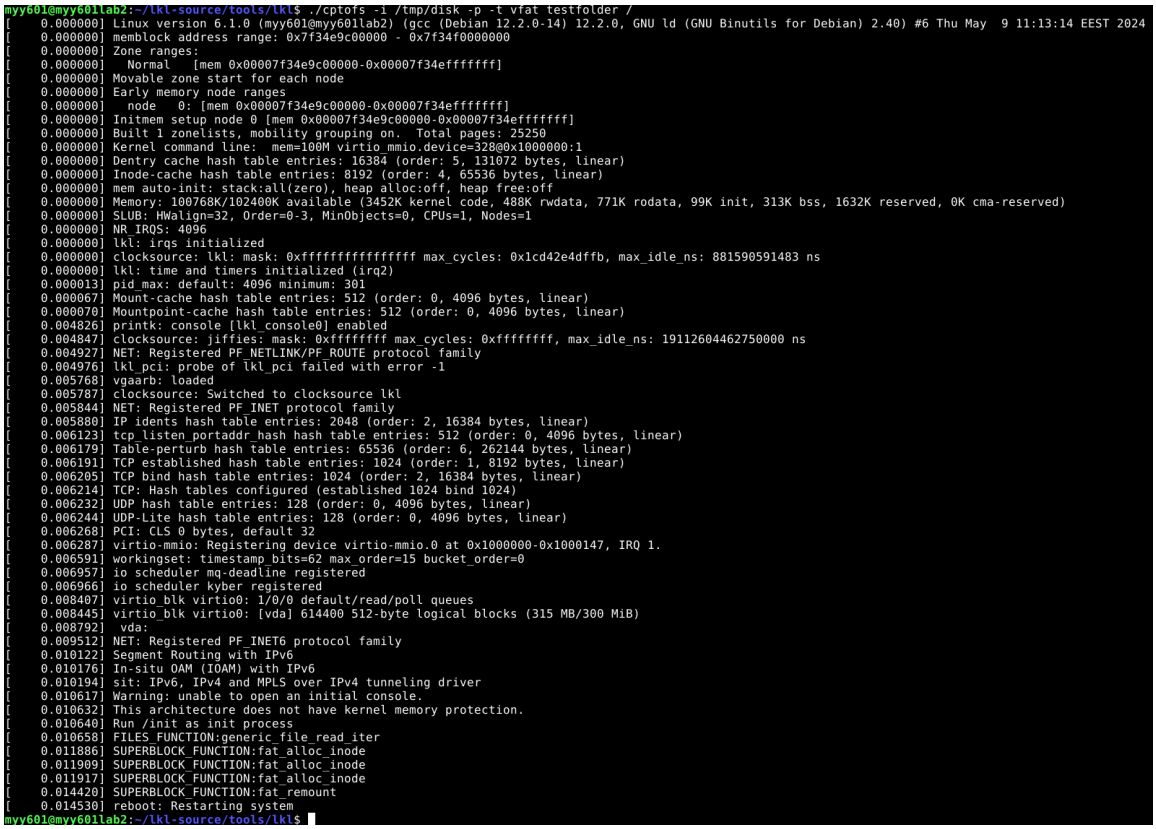

Figure 5: Έλεγχος εντολής ./cptofs -i /tmp/disk -p -t vfat testfolder/

JOURNALING

Τα συστήματα αρχείων με journaling είναι μια τεχνολογία που χρησιμοποιείται σε συστήματα αρχείων για να διασφαλίζει την ακεραιότητα των δεδομένων σε περιπτώσεις απροσδόκητων διακοπών λειτουργίας, όπως διακοπή ρεύματος ή σφάλματα λογισμικού. Το journaling λειτουργεί καταγράφοντας τις αλλαγές που πρόκειται να γίνουν σε ένα αρχείο ή σε μια δομή δεδομένων πριν αυτές εφαρμοστούν. Αυτό το "ημερολόγιο" (journal) διατηρείται στο δίσκο και ενημερώνεται συνεχώς. Αφού οι αλλαγές καταγραφούν στο ημερολόγιο, εφαρμόζονται στο σύστημα αρχείων.

Πλεονεκτήματα του journaling:

- Ακεραιότητα Δεδομένων: Σε περίπτωση διακοπής λειτουργίας, το σύστημα αρχείων μπορεί να αποκαταστήσει τις αλλαγές από το journal και να διατηρήσει τα δεδομένα ακριβή.
- Ταχεία Ανάκτηση: Η διαδικασία ανάκτησης μετά από διακοπή λειτουργίας είναι γρήγορη, καθώς το σύστημα αρχείων μπορεί να επαναφέρει τα δεδομένα χρησιμοποιώντας το journal.
- Αποτελεσματική Διαχείριση Σφαλμάτων: Οι περισσότερες ενημερώσεις είναι σταδιακές και οι πιθανότητες να προκληθεί σφάλμα λόγω διακοπής μειώνονται.
- Βελτιωμένη Απόδοση: Τα συστήματα αρχείων με journaling μπορούν να αυξήσουν την απόδοση, καθώς οι ενημερώσεις μπορούν να συγκεντρωθούν πριν από την εφαρμογή τους.

Μειονεκτήματα του journaling:

- Πρόσθετος Χώρος: Η καταγραφή των αλλαγών στο journal απαιτεί επιπλέον χώρο στο δίσκο.
- Επιπλέον Επιδόσεις: Η διαδικασία της καταγραφής και εφαρμογής των αλλαγών μπορεί να επιβαρύνει το σύστημα και να μειώσει τις επιδόσεις.
- Πολυπλοκότητα: Η προσθήκη ενός journaling συστήματος αρχείων αυξάνει την πολυπλοκότητα της διαχείρισης των αρχείων.

- Πιθανότητα Διαφθοράς: Αν και το journaling μειώνει σημαντικά την πιθανότητα διαφθοράς των δεδομένων, σε σπάνιες περιπτώσεις μπορεί να προκαλέσει προβλήματα εάν το journal δεν ενημερωθεί σωστά.

FAT και Journaling:

Το σύστημα αρχείων FAT (File Allocation Table) είναι ένα απλό σύστημα αρχείων που δεν υποστηρίζει journaling. Το FAT σχεδιάστηκε αρχικά για παλαιότερα λειτουργικά συστήματα και για αποθηκευτικά μέσα με περιορισμένη χωρητικότητα. Δεν διαθέτει μηχανισμό journaling, με αποτέλεσμα να μην παρέχει τα προαναφερόμενα πλεονεκτήματα που προσφέρει το journaling.

Συμπεράσματα:

Εν ολίγοις, τα συστήματα αρχείων με journaling προσφέρουν αυξημένη ακεραιότητα και ασφάλεια των δεδομένων, αλλά μπορεί να προκαλούν πρόσθετο φόρτο εργασίας και κατανάλωση πόρων. Παρόλο που το FAT είναι ένα απλό σύστημα αρχείων, η έλλειψη υποστήριξης για journaling μπορεί να το κάνει λιγότερο κατάλληλο για σύγχρονες εφαρμογές που απαιτούν αξιόπιστη αποθήκευση δεδομένων.

Υλοποίηση Journaling

Όπως αναφέραμε και πριν, το σύστημα αρχείων FAT (File Allocation Table) δεν υποστηρίζει εγγενώς journaling, καθώς είναι ένα παλιό σύστημα αρχείων που σχεδιάστηκε για να είναι απλό και ελαφρύ. Ωστόσο, σε ένα μοντέρνο λειτουργικό σύστημα όπως το Linux, μπορούμε να προσθέσουμε κάποια μορφή journaling λειτουργικότητας στην κορυφή ενός συστήματος αρχείων FAT, μέσω εξωτερικής διαχείρισης ή σύνθεσης.

Για να επιτευχθεί αυτό, θα μπορούσαμε να δημιουργήσουμε ένα στρώμα συστήματος αρχείων πάνω από το FAT, το οποίο θα διαχειρίζεται το journaling, καταγράφοντας αλλαγές πριν εφαρμοστούν. Αυτό το στρώμα θα μπορούσε να είναι υπεύθυνο για:

- Καταγραφή αλλαγών: Όταν γίνει αλλαγή σε ένα αρχείο ή έναν κατάλογο στο FAT, το στρώμα journaling θα καταγράφει την αλλαγή σε ένα ξεχωριστό αρχείο journal.
- Εφαρμογή αλλαγών: Όταν οι αλλαγές έχουν καταγραφεί στο journal, θα εφαρμόζονται στο FAT.
- Ανάκτηση: Σε περίπτωση διακοπής λειτουργίας, το στρώμα journaling μπορεί να χρησιμοποιήσει το journal για να αποκαταστήσει τυχόν απώλειες δεδομένων στο FAT.

Καταρχάς η υλοποίηση του journaling απαιτεί εσωτερικές κλήσεις του lkl, όπως sys_open και sys_write, ώστε να ανοίξουμε και να γράψουμε ένα αρχείο journal, το οποίο το δημιουργήσαμε εκ των προτέρων (journal.txt) και βρίσκεται στο fs/fat. Έπειτα πριν καταφύγουμε στην γραφή κώδικα στις απαραίτητες συναρτήσεις, προσθέτουμε τα εξής:

```
#include <linux/syscalls.h>//prosthiki to include
```

Figure 6: Προσθήκη include

```
int fdiscriptor_journal;
```

Figure 7: Προσθήκη file descriptor

Το #include <linux/syscalls.h> περιλαμβάνει το κατάλληλο header file, που περιέχει δηλώσεις και ορισμούς για τη χρήση συστημικών κλήσεων στον πυρήνα του Linux.

Παρακάτω θα εξηγήσουμε διεξοδικά τον κώδικα που προσθέσαμε στις συναρτήσεις:

```
char buffer[1024];
sbi->fdiscriptor_journal=sys_open("journal.txt",O_CREAT|O_WRONLY|O_APPEND,0);
sprintf(buffer,"inode.txt:SUPERBLOCK");
sys_write(sbi->fdiscriptor_journal,buffer,sizeof(buffer));
sys_close(sbi->fdiscriptor_journal);
```

Figure 8: Κώδικας Journaling

- char buffer[2024] : Δημιουργείται ένας πίνακας χαρακτήρων (buffer) με μέγεθος 2024 bytes. Αυτός ο buffer χρησιμοποιείται για την αποθήκευση δεδομένων που θα γραφτούν στο αρχείο.
- sbi->fdiscriptor_journal = sys_open("journal.txt", O_CREAT | O_WRONLY | O_APPEND, 0): Χρησιμοποιείται η συστημική κλήση sys_open για να ανοίξει το αρχείο journal.txt.

Παράμετροι:

- "journal.txt": Το όνομα του αρχείου που ανοίγεται.
- O_CREAT | O_WRONLY | O_APPEND: Σημαίες που καθορίζουν ότι το αρχείο θα δημιουργηθεί αν δεν υπάρχει (O_CREAT), θα ανοιχτεί για εγγραφή μόνο (O_WRONLY), και η εγγραφή θα γίνει στο τέλος του αρχείου (O_APPEND).

Το αποτέλεσμα της κλήσης αποθηκεύεται στο sbi->fdiscriptor_journal, το οποίο είναι ένας file descriptor.

- sprintf(buffer, "inode.txt:SUPERBLOCK");:
- Χρησιμοποιείται η sprintf για να γράψει τη συμβολοσειρά "inode.txt:SUPERBLOCK" στον buffer.

Αυτό σημαίνει ότι ο buffer τώρα περιέχει τα δεδομένα που θα γραφτούν στο αρχείο.

- sys_write(sbi->fdiscriptor_journal, buffer, sizeof(buffer)): Χρησιμοποιείται η συστημική κλήση sys_write για να γράψει τα δεδομένα από τον buffer στο αρχείο που ανοίχτηκε.

Παράμετροι:

- sbi->fdiscriptor_journal: Ο file descriptor του αρχείου.
- buffer: Τα δεδομένα που θα γραφτούν στο αρχείο.
- sizeof(buffer): Το μέγεθος των δεδομένων που θα γραφτούν (όλο το μέγεθος του buffer).

sys_close(sbi->fdiscriptor_journal): Χρησιμοποιείται η συστημική κλήση sys_close για να κλείσει το αρχείο που ανοίχτηκε.

Παράμετρος:

- sbi->fdiscriptor_journal: Ο file descriptor του αρχείου που θα κλείσει.

Σημείωση: Όλα τα παραπάνω απαιτούν το αρχείο fat.h να υπάρχει σε όλα τα παρακάτω αρχεία που θα αναφερθούν, αλλά και το struct msdos_sb_info να είναι αρχικοποιημένο (ώστε να είναι ορατός ο file descriptor).

SUPERBLOCK

Οι συναρτήσεις στις οποίες συμπληρώθηκε κώδικας είναι οι παρακάτω:

Πιο συγκεκριμένα:

Στο αρχείο inode.c

Η συνάρτηση fat_read_root διαβάζει τον ριζικό κατάλογο (root directory) ενός συστήματος αρχείων FAT και εκτελεί τις ακόλουθες λειτουργίες:

- Ορίζονται οι βασικές πληροφορίες του inode, όπως θέση (i_pos), χρήστη (i_uid), ομάδα (i_gid), έκδοση (i_generation), δικαιώματα (i_mode), και λειτουργίες (i_op, i_fop).
- Ανοίγει το αρχείο journal.txt για εγγραφή ή δημιουργία αν δεν υπάρχει.

- Γράφεται τη συμβολοσειρά "inode.txt:SUPERBLOCK" στο αρχείο.
- Κλείνει το αρχείο.
- Αν το σύστημα αρχείων είναι FAT32, υπολογίζει το μέγεθος του directory και ορίζει την αρχική θέση (i_start) στο root_cluster.
- Διαφορετικά, ορίζει το i_start στο 0 και το μέγεθος του inode ανάλογα με τον αριθμό των καταχωρίσεων του directory (dir_entries).
- Υπολογίζονται και ορίζεται ο αριθμός των blocks (i_blocks) που καταλαμβάνει το inode.
- Αποθηκεύονται τα χαρακτηριστικά του inode και ορίζει τους χρόνους τελευταίας τροποποίησης, πρόσβασης και δημιουργίας σε μηδέν.
- Ορίζεται ο αριθμός των συνδέσμων (nlink) του inode.

Η συνάρτηση fat_fill_inode έχει ως σκοπό να:

- Διαβάσει τις πληροφορίες από ένα συγκεκριμένο msdos_dir_entry (δηλαδή μια καταχώριση καταλόγου) και να ενημερώσει την αντίστοιχη δομή inode του πυρήνα.
- Αρχικοποιήσει το inode με τις κατάλληλες τιμές που αντιστοιχούν στα δεδομένα του αρχείου ή καταλόγου που αντιπροσωπεύει αυτή η καταχώριση καταλόγου.
- Ορίσει τις λειτουργίες και τα χαρακτηριστικά του inode ανάλογα με το αν πρόκειται για αρχείο ή κατάλογο.

Λειτουργίες της Συνάρτησης fat_fill_inode

- Άνοιγμα, Εγγραφή και Κλείσιμο Αρχείου Καταγραφής:
- Η συνάρτηση ανοίγει ένα αρχείο καταγραφής journal.txt, γράφει μια συμβολοσειρά και στη συνέχεια κλείνει το αρχείο.
- Αυτό μπορεί να χρησιμοποιείται για καταγραφή ενεργειών ή για σκοπούς εντοπισμού σφαλμάτων (debugging).

- Αρχικοποίηση inode: Ορίζει τα βασικά πεδία του inode, όπως ο χρήστης (i_uid), η ομάδα (i_gid), η έκδοση (i_version), και η γενιά (i_generation).
- Αναλόγως αν το de αντιπροσωπεύει directory (ATTR_DIR) ή όχι, ορίζει τις αντίστοιχες λειτουργίες και χαρακτηριστικά για το inode.
- Διαχείριση Directory: Αν η καταχώριση είναι directory, ενημερώνει τις δομές και υπολογίζει το μέγεθος του καταλόγου.
- Διαχείριση Απλού Αρχείου: Αν δεν είναι directory, ενημερώνει τις δομές και ορίζει τα πεδία όπως το μέγεθος αρχείου και τις λειτουργίες αρχείου.
- Αντιμετώπιση Συστήματος Αρχείων: Αν η καταχώριση είναι αρχείο συστήματος (ATTR_SYS), ενδέχεται να το ορίσει ως immutable, ανάλογα με τις επιλογές του συστήματος αρχείων.
- Υπολογισμός Χώρου και Χρόνων: Υπολογίζει τον αριθμό των blocks που καταλαμβάνει το inode.
- Μετατρέπει και ορίζει τους χρόνους τελευταίας μεταβολής από τη μορφή του FAT συστήματος σε μορφή Unix.

Στη συνάρτηση fat_fill_super:

- Ανοίγει ένα αρχείο καταγραφής journal.txt, γράφει μια συμβολοσειρά και στη συνέχεια κλείνει το αρχείο.
- Δεσμεύεται μνήμη για τη δομή msdos_sb_info που περιέχει πληροφορίες για το σύστημα αρχείων FAT.
- Ορίζονται τα βασικά πεδία του superblock, όπως οι συναρτήσεις λειτουργίας (s_op) και εξαγωγής (s_export_op), η μαγική σύμβολο (s_magic) και οι δομές χρόνου (s_time_min και s_time_max).
- Διαχειρίζεται τις επιλογές του χρήστη και εφαρμόζει τις ανάλογες ρυθμίσεις.
- Διαβάζεται το boot sector του συστήματος αρχείων FAT για να πάρει πληροφορίες σχετικά με το μέγεθος των clusters και άλλες παραμέτρους του συστήματος αρχείων.
- Αρχικοποιούνται οι παράμετροι του FAT όπως το μέγεθος του, τον αριθμό των FATs και τη θέση τους.
- Υπολογίζεται το μέγεθος των clusters και τον αριθμό των bits που απαιτούνται για την αναπαράστασή τους.

- Δημιουργούνται και αρχικοποιούνται τα inodes για το FAT filesystem, συμπεριλαμβανομένων του root inode και του inode του FAT.
- Ορίζεται ο ριζικός κατάλογος ως το κεντρικό σημείο εκκίνησης του filesystem.
- Φορτώνεται ο κατάλληλος πίνακας κωδικοποίησης για τη διαχείριση χαρακτήρων στο filesystem.

FILE ALLOCATION TABLE

Στο αρχείο fatent.c:

Η συνάρτηση fat_ent_update_ptr είναι υπεύθυνη για την ενημέρωση ενός καταχωρητή (entry) FAT στο superblock του συστήματος αρχείων FAT στον πυρήνα του Linux. Ας αναλύσουμε τις λειτουργίες της:

- Όπως και σε προηγούμενες συναρτήσεις, η συνάρτηση ανοίγει ένα αρχείο καταγραφής journal.txt, εγγράφει μια συμβολοσειρά και στη συνέχεια κλείνει το αρχείο.
- Ελέγχεται αν ο καταχωρητής FAT που πρόκειται να ενημερωθεί ανήκει στον ίδιο τομέα διακοπής με τον τομέα που δίνεται ως όρισμα.
- Καλούνται οι αντίστοιχες λειτουργίες που παρέχονται από τη δομή fatent_operations για να ενημερωθεί ο καταχωρητής FAT στον superblock.

Η συνάρτηση fat_free_clusters χρησιμοποιείται για την ελευθέρωση καταχωρητών (clusters) στο σύστημα αρχείων FAT. Ας δούμε τη λειτουργία της:

- Αρχικά, η συνάρτηση ανοίγει ένα αρχείο καταγραφής journal.txt, εγγράφει μια συμβολοσειρά για καταγραφή της δραστηριότητας και στη συνέχεια κλείνει το αρχείο.
- Η συνάρτηση αναλαμβάνει να διαβάσει τους καταχωρητές FAT, αναζητώντας τους ανάλογα με το δείκτη cluster που περνάει ως παράμετρος.

- Αφού εντοπιστεί ο καταχωρητής FAT που αντιστοιχεί στον δείκτη cluster, η συνάρτηση ενημερώνει τον καταχωρητή FAT ώστε να δείχνει στη θέση του στον πίνακα καταχωρητών FAT ότι το συγκεκριμένο cluster είναι ελεύθερο.
- Οι ενέργειες που γίνονται καταγράφονται σε ένα αρχείο καταγραφής για λόγους αποκατάστασης.
- Εάν ο διαχειριστής ελεύθερων cluster δεν είναι -1, τότε αυξάνεται ο αριθμός των ελεύθερων cluster και ο διαχειριστής επισημαίνεται ως "βρώμικος" (dirty), δηλώνοντας ότι το FSINFO πρέπει να ενημερωθεί.
- Η συνάρτηση επιστρέφει τυχόν σφάλματα που μπορεί να προέκυψαν κατά τη διαδικασία της ελευθέρωσης των cluster.

DIRECTORY

Στο αρχείο dir.c:

Η συνάρτηση fat_add_entries χρησιμοποιείται για την προσθήκη νέων καταχωρήσεων σε έναν κατάλογο στο σύστημα αρχείων FAT. Ας δούμε τη λειτουργία της:

- Αρχικά, η συνάρτηση ανοίγει ένα αρχείο καταγραφής journal.txt, εγγράφει μια συμβολοσειρά για καταγραφή της δραστηριότητας και στη συνέχεια κλείνει το αρχείο.
- Η συνάρτηση αναζητά τον αριθμό των διαθέσιμων καταχωρήσεων στον κατάλογο. Χρησιμοποιείται για να βρει ποιες θέσεις είναι διαθέσιμες για τις νέες καταχωρήσεις που θα προστεθούν.
- Αν υπάρχουν διαθέσιμες θέσεις στον κατάλογο, η συνάρτηση προσθέτει τις νέες καταχωρήσεις στις διαθέσιμες θέσεις αυτές.
- Αν οι διαθέσιμες θέσεις δεν είναι αρκετές, τότε η συνάρτηση διατρέχει τα στάδια για να προσαρτήσει νέα cluster στον κατάλογο και να τοποθετήσει εκεί τις νέες καταχωρήσεις.
- Οι ενέργειες που γίνονται καταγράφονται σε ένα αρχείο καταγραφής για λόγους αποκατάστασης.

- Η συνάρτηση επιστρέφει τυχόν σφάλματα που μπορεί να προέκυψαν κατά τη διαδικασία της προσθήκης των νέων καταχωρήσεων.

Στο αρχείο namei_msdos:

- Η συνάρτηση msdos_find παίρνει ένα όνομα αρχείου σε μορφή Unicode και το μετατρέπει σε μορφή που ταιριάζει με τη μορφή ονοματοδοσίας του συστήματος αρχείων FAT (MS-DOS). Αυτή η διαδικασία περιλαμβάνει τη μετατροπή των χαρακτήρων σε κεφαλαία, την αγνόηση κενών διαστημάτων και την αντικατάσταση κάποιων ειδικών χαρακτήρων.
- Η συνάρτηση αναζητά το αρχείο με το συγκεκριμένο όνομα στον κατάλογο dir. Χρησιμοποιεί τη συνάρτηση fat_scan για να εξετάσει τις καταχωρήσεις στον κατάλογο και να βρει αυτή που αντιστοιχεί στο συγκεκριμένο όνομα αρχείου.
- Αν το όνομα αρχείου ξεκινάει με τελεία, ελέγχει αν η αντίστοιχη καταχώρηση είναι κρυφή. Αν το όνομα αρχείου δεν ξεκινά με τελεία, ελέγχει αν η αντίστοιχη καταχώρηση είναι ορατή. Αν η καταχώρηση δεν είναι ορατή, επιστρέφει το αντίστοιχο σφάλμα.
- Οι ενέργειες που γίνονται καταγράφονται σε ένα αρχείο καταγραφής για λόγους αποκατάστασης.
- Η συνάρτηση επιστρέφει έναν κωδικό σφάλματος ή επιστρέφει μηδενικό κωδικό αν δεν υπάρχει σφάλμα και το αρχείο βρεθεί επιτυχώς.

Επαλήθευση κώδικα

Είμαστε στην δυσάρεστη θέση να ανακοινώσουμε πως δεν καταφέραμε να επαληθεύσουμε την λειτουργία του κώδικα, καθώς μας εμφάνιζε το παρακάτω segmentation fault και λόγω περιορισμένου χρόνου και υψηλού φόρτου εργασίας δεν μπορέσαμε να βρούμε τι φταίει.

```
root@myy601lab2:/home/myy601/lkl-source/tools/lkl# ./cptofs -i /tmp/disk -t vfat lklfuse.c /
Segmentation fault
```

Figure 9: Segmentation Fault