



Πανεπιστήμιο Ιωαννίνων
Τμήμα Μηχανικών Η/Υ & Πληροφορικής
Ακαδημαϊκό Έτος 2022-2023

ΜΥΥ601 Λειτουργικά Συστήματα

2η Εργαστηριακή Άσκηση

Υλοποίηση αρχείου καταγραφής στο σύστημα
αρχείων FAT του Linux

ΟΜΑΔΑ

Αρχοντής Νέστωρας - 4747

Σπυρίδων Χαλιδιάς - 4830

Περιεχόμενα

Τρόπος Σκέψεις	3
----------------------	---

- ❖ Εισαγωγή

1ο στάδιο	3
-----------------	---

- ❖ GDB

2ο στάδιο	4
-----------------	---

3ο στάδιο	6
-----------------	---

4ο στάδιο	7
-----------------	---

- ❖ Εισαγωγή
- ❖ Ιδέα για υλοποίηση
- ❖ Υλοποίηση

Τρόπος Σκέψεις

Εισαγωγή

Η δεύτερη εργαστηριακή άσκηση ασχολείται με τη βιβλιοθήκη lkl που περιέχει τον πυρήνα των linux και το σύστημα αρχείων Fat. Στόχος αυτής της άσκησης είναι να προσθέσουμε ένα αρχείο καταγραφής journaling στο οποίο θα υπάρχουν όλες οι αλλαγές που θα γίνονται στο σύστημα. Αυτό το κάνουμε έτσι ώστε να μπορέσουμε να επαναφέρουμε το σύστημα σε περίπτωση κάποιου προβλήματος. Την προσπάθεια μας να υλοποιήσουμε αυτό το αρχείο τη χωρίζουμε σε στάδια.

1ο στάδιο

Στην αρχή θέλουμε να τρέξουμε τη βιβλιοθήκη και το σύστημα αρχείων. Αυτό θα γίνει με τις παρακάτω εντολές

- `make -j8 -C tools/lkl`
για τη βιβλιοθήκη lkl
- `make -C tools/lkl test`
για τη βιβλιοθήκη lkl

Το αρχείο που εμφανίζεται έπειτα είναι `/tmp/vfatfile`

GDB

Χρησιμοποιήσαμε τον debugger gdb για την κατανόηση του κώδικα. Σταματήσαμε τον κώδικα σε διάφορα σημεία για να καταλάβουμε τις κλήσεις. Για την κατανόηση του κώδικα χρησιμοποιήσαμε και το lkl-dox έτσι ώστε να πάρουμε μία πιο γενική ιδέα για τη δομή του προγράμματος

2ο στάδιο

Στη συνέχεια βάλουμε στον κώδικα τα `printk` έτσι ώστε να καταλάβουμε ακόμα καλύτερα τη δομή του κώδικα. Αυτό που βάζουμε να εμφανίζεται είναι το αρχείο στο οποίο βρίσκεται μέσα η μέθοδος που καλείτε εκείνη την στιγμή και έπειτα εμφανίζουμε και την ίδια την μέθοδο.

Ένα τυχαίο παράδειγμα σύνταξης της εντολής `printk`.

```
printk(KERN_INFO "IN THE _____ FILE: fs/msdos/namei_msdos.c _____ METHOD: msdos_rename\n");
```

Μέσα σε διάφορες μεθόδους των παρακάτω αρχείων γράφτηκε οι εντολή `printk`. Τα αρχεία είναι:

- `fatent.c`
- `file.c`
- `inode.c`
- `namei_msdos.c`
- `namei_vfat.c`

με αυτήν την εντολή:

```
myy601@myy601lab2:~/lkl/lkl-source/tools/lkl$ tests/boot -t vfat -d /tmp/vfatfile -p -P 0
```

βγαίνει αυτό:

```
mount_fs      passed [proc: 0]
chdir         passed [0]
opendir       passed [4]
getdents64    passed [4 . .. fs bus irq net sys tty kmsg maps misc stat iomem cry]
umount_fs     passed [proc: 0 0 0]
[ 0.138475] IN THE _____ FILE: fs/fat/inode.c _____ STRUCT: fat_alloc_inode
[ 0.138497] IN THE _____ FILE: fs/fat/inode.c _____ STRUCT: fat_alloc_inode
[ 0.138501] IN THE _____ FILE: fs/fat/inode.c _____ STRUCT: fat_alloc_inode
mount_dev     passed [0]
chdir         passed [0]
opendir       passed [4]
getdents64    passed [4 . .. ]
[ 0.139921] IN THE _____ FILE: fs/fat/inode.c _____ METHOD: fat_evict_inode
[ 0.139943] IN THE _____ FILE: fs/fat/inode.c _____ METHOD: fat_destroy_inode
[ 0.139949] IN THE _____ FILE: fs/fat/inode.c _____ METHOD: fat_put_super
[ 0.139989] IN THE _____ FILE: fs/fat/inode.c _____ METHOD: fat_evict_inode
[ 0.139995] IN THE _____ FILE: fs/fat/inode.c _____ METHOD: fat_destroy_inode
[ 0.140000] IN THE _____ FILE: fs/fat/inode.c _____ METHOD: fat_evict_inode
[ 0.140004] IN THE _____ FILE: fs/fat/inode.c _____ METHOD: fat_destroy_inode
umount_dev    passed [0 0 0]
lo_ifup       passed [0]
gettid        passed [13350]
syscall_thread passed []
many_syscall_threads passed []
[ 0.161691] reboot: Restarting system
myy601@myy601lab2:~/lkl/lkl-source/tools/lkl$
```

Άλλη εντολή:

```
myy601@myy601lab2:~/lkl/lkl-source/tools/lkl$ ./cptofs -i /tmp/vfatfile -p -t vfat lklfuse.c /
```

Βγαίνει αυτό:

```
0.022981] this architecture does not have kernel memory protection.
[ 0.023415] IN THE      FILE: fs/fat/inode.c      STRUCT: fat_alloc_inode
[ 0.023428] IN THE      FILE: fs/fat/inode.c      STRUCT: fat_alloc_inode
[ 0.023430] IN THE      FILE: fs/fat/inode.c      STRUCT: fat_alloc_inode
[ 0.023771] IN THE      FILE: fs/vfat/fatent.c      STRUCT: vfat_lookup
[ 0.023785] IN THE      FILE: fs/vfat/fatent.c      METHOD: vfat_add_entry
[ 0.023798] IN THE      FILE: fs/fat/inode.c      STRUCT: fat_alloc_inode
[ 0.023808] IN THE      FILE: fs/fat/inode.c      METHOD: fat_write_begin
[ 0.023813] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_blocknr
[ 0.023815] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_bread
[ 0.023826] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_set_ptr
[ 0.023829] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_get
[ 0.023831] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_put
[ 0.023837] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_blocknr
[ 0.023839] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_bread
[ 0.023841] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_set_ptr
[ 0.023843] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_get
[ 0.023845] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_put
[ 0.023847] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_blocknr
[ 0.023849] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_bread
[ 0.023851] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_set_ptr
[ 0.023860] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_get
[ 0.023863] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_blocknr
[ 0.023865] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_bread
[ 0.023866] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_set_ptr
[ 0.023868] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_get
[ 0.023870] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_put
[ 0.023873] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_blocknr
[ 0.023875] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_bread
[ 0.023877] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_set_ptr
[ 0.023879] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_get
[ 0.023884] IN THE      FILE: fs/fat/inode.c      METHOD: fat_write_end
[ 0.023891] IN THE      FILE: fs/fat/inode.c      METHOD: fat_write_begin
[ 0.023896] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_blocknr
[ 0.023898] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_bread
[ 0.023900] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_set_ptr
[ 0.023902] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_get
[ 0.023904] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_put
[ 0.023906] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_blocknr
[ 0.023908] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_bread
[ 0.023910] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_set_ptr
[ 0.023912] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_get
[ 0.023914] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_blocknr
[ 0.023916] IN THE      FILE: fs/fat/fatent.c      METHOD: fat_ent_bread
[ 0.023918] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_set_ptr
[ 0.023920] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_get
[ 0.023921] IN THE      FILE: fs/fat/fatent.c      METHOD: fat16_ent_put
```

και συνεχίζεται

Θέλουμε να καταλάβουμε κυρίως τις δομές του συστήματος fat που είναι:

- Superblock:(fs/fat/inode.c)
- Μνήμη: (fs/fat/inode.c)
- Εγγραφές FAT:(fs/fat/fatent.c)

- Αρχείο:(fs/fat/file.c)
- Inode:(fs/fat/file.c)
- Κατάλογοι

3ο στάδιο

Εισαγωγή

Το journal είναι ένα αρχείο που κρατάει τις πληροφορίες του συστήματος σε περίπτωση βλάβης ώστε να μπορούμε να επαναφέρουμε την τελική του κατάσταση πριν το σφάλμα. Συγκεκριμένα σε αυτό το αρχείο γράφονται οι αλλαγές χωρίς κάποια συγκεκριμένη ταξινόμηση ή ιδιαίτερη τοποθέτηση αλλά όπως τις λαμβάνουμε. Δεν έχει νοημα να γίνει κάποια ταξινόμηση γιατί το κάνει ήδη το σύστημα αρχείων fat. Το αρχείο journal βρίσκεται εκεί μόνο στη περίπτωση σοβαρού σφάλματος. Έπειτα αυτή η αποθήκευση γίνεται γρήγορα. Αυτό είναι ιδιαίτερα θετικό γιατί αυτή η υπηρεσία είναι προληπτική και υπάρχει πιθανότητα να μη χρησιμοποιηθεί.

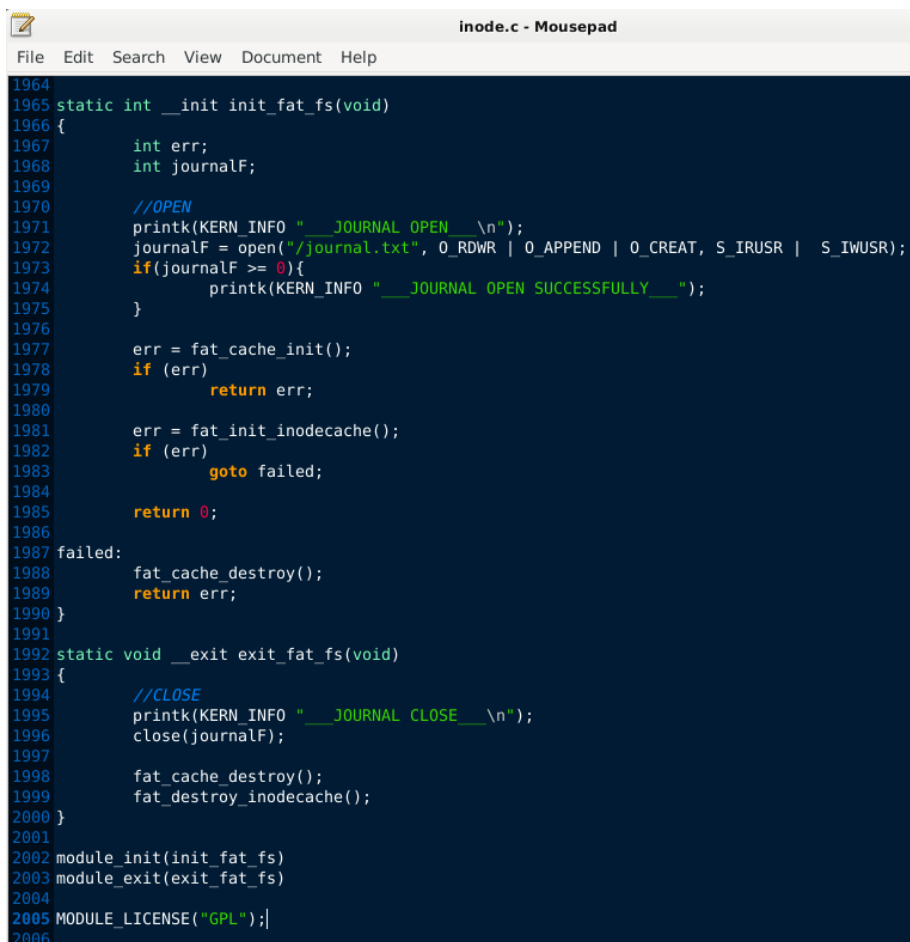
Ιδέα για την υλοποίηση

Πριν ξεκινήσουμε να φτιάχνουμε το αρχείο κάναμε ένα σχεδιασμό και ακολουθήσαμε κάποια βήματα. Αρχικά έπρεπε να αποφασίσουμε για τη μορφή που θα έχει κάθε προσθήκη του αρχείου. Κάθε εγγραφή πρέπει να κρατάει στοιχεία για το που έγινε στο δίσκο η αλλαγή, τι είδους αλλαγή ήταν αυτή καθώς και κάποια πληροφορία για το πως θα γίνουν τα δεδομένα μετά την αλλαγή. Στη συνέχεια πρέπει να δούμε σε ποιά σημεία του συστήματος πρέπει να γίνουν η εγγραφές στο αρχείο. Καταλάβαμε ότι πρέπει να βρίσκονται σε κάθε σημείο που γίνονται τροποποιήσεις στο σύστημα αρχείων. Αφού λοιπόν φτιαχτεί το αρχείο πρέπει να φτιάξουμε το μηχανισμό ανάκτησης των δεδομένων σε περίπτωση σημαντικού λάθους στο σύστημα. Αυτός ο μηχανισμός θα διαβάζει τις εγγραφές από το αρχείο και θα ξαναφτιαχνει το σύστημα από την αρχή. Τελική σκέψη είναι ότι επειδή το αρχείο αυτό θα γίνεται πολύ μεγάλο να φτιάξουμε μία στρατηγική

που θα διαγράφει εγγραφές. Μία τέτοια στρατηγική θα μπορούσε να είναι να σβήνονται οι εγγραφές που δεν επηρεάζουν πλέον το σύστημα.

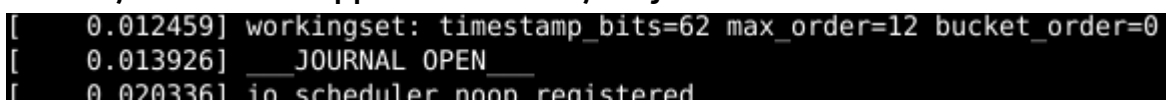
4ο στάδιο Υλοποίηση

Εδώ σε αυτές τις 2 τελευταίες μεθόδους του αρχείου inode.c ανοίγουμε και κλείνουμε το journal. Γραμμές 1970-1975 και 1994-1996.



```
1964
1965 static int __init init_fat_fs(void)
1966 {
1967     int err;
1968     int journalF;
1969
1970     //OPEN
1971     printk(KERN_INFO "___JOURNAL OPEN___\n");
1972     journalF = open("/journal.txt", O_RDWR | O_APPEND | O_CREAT, S_IRUSR | S_IWUSR);
1973     if(journalF >= 0){
1974         printk(KERN_INFO "___JOURNAL OPEN SUCCESSFULLY___");
1975     }
1976
1977     err = fat_cache_init();
1978     if (err)
1979         return err;
1980
1981     err = fat_init_inodecache();
1982     if (err)
1983         goto failed;
1984
1985     return 0;
1986
1987 failed:
1988     fat_cache_destroy();
1989     return err;
1990 }
1991
1992 static void __exit exit_fat_fs(void)
1993 {
1994     //CLOSE
1995     printk(KERN_INFO "___JOURNAL CLOSE___\n");
1996     close(journalF);
1997
1998     fat_cache_destroy();
1999     fat_destroy_inodecache();
2000 }
2001
2002 module_init(init_fat_fs)
2003 module_exit(exit_fat_fs)
2004
2005 MODULE_LICENSE("GPL");
2006
```

Εντοπίζεται και στο τερματικό ότι άνοιξε το journal.



```
[ 0.012459] workingset: timestamp_bits=62 max_order=12 bucket_order=0
[ 0.013926] ___JOURNAL OPEN___
[ 0.020336] io scheduler noop registered
```

Έπειτα, αφού έχουμε ανοίξει το journal, μπορούμε να κάνουμε τις απαραίτητες εγγραφές/προσθήκες στο αρχείο journal, στις κατάλληλες μεθόδους. Σε γενικές γραμμές, οι εγγραφές θα χρειαστεί να γίνουν στις περισσότερες μεθόδους που αποφασίσαμε στην προηγούμενη ενότητα να βάλουμε την εντολή printk.

Οι ιδέες που αναφέρονται στο τρίτο στάδιο από αυτό το σημείο και μετά δεν έχουν υλοποιηθεί και το journal είναι ακόμα σε πρωταρχικό επίπεδο.