Πανεπιστήμιο Ιωαννίνων

Τμήμα Μηχανικών Η/Υ και Πληροφορικής

ΜΥΥ601

Λειτουργικά Συστήματα

-2023-

Εργαστήριο 2:Υλοποίηση αρχείου καταγραφής στο σύστημα αρχείων FAT του Linux

ΘΩΜΑ ΕΙΡΗΝΗ 4292 , ΤΣΙΝΟΡΕΜΑΣ ΗΛΙΑΣ 2151, ΜΑΗ ΧΡΙΣΤΙΝΑ 4612

Εισαγωγή:

Ξεκινώντας την αναφορά μας θα θέλαμε να επισημάνουμε ότι δεν καταφέραμε να ολοκληρώσουμε τις υλοποιήσεις που απαιτούνται για την πραγματοποίηση της 2ης άσκησης με τίτλο «Υλοποίηση αρχείου καταγραφής στο σύστημα αρχείων FAT του Linux» , πέραν του πρώτου ερωτήματος στο οποίο μας ζητήθηκε η τύπωση μηνυμάτων στην κονσόλα προσθέτοντας την συνάρτηση printk(KERN_INFO ”…”, …)

Παρόλα αυτά, δημιουργήσαμε αυτό το PDF αρχείο έτσι ώστε να σας παρουσιάσουμε το πως αντιληφθήκαμε την επίλυση της άσκησης, και τις γνώσεις που προσκομίσαμε όλο αυτό το διάστημα.

Προετοιμασία – Κατανόηση της λειτουργίας του συστήματος :

Ακολουθώντας τις οδηγίες που μας παρείχε ο καθηγητής για την εγκατάσταση του λειτουργικού συστήματος «Debian» , αρχικά περιηγηθήκαμε μέσω terminal στον κατάλογο /lkl-source ακολουθώντας την διαδρομή /home/bin/lkl/lkl-source και πραγματοποιήσαμε compile με την εντολή:

```
make -j8 -C tools/lkl
```

Και στη συνέχεια:

```
make -C tools/lkl test
```

Έπειτα στον κατάλογο /tools/lkl:

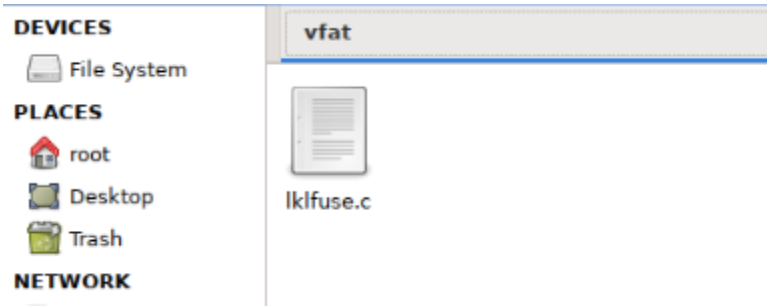
mount -t vfat -o loop /tmp/vfatfile /vfat, για να βεβαιωθούμε ότι το σύστημα αρχείων είναι mounted.

Επίσης παρατηρήσαμε ότι εκτελώντας το πρόγραμμα cptofs την εντολή που μας δίνεται ως παράδειγμα

./cptofs -i /tmp/vfatfile -p -t vfat lklfuse.c /, το αρχείο lklfuse.c θα αντιγραφεί στον κατάλογο / του συστήματος αρχείων tmp/vfatfile.

Παρατηρήσαμε τις αλλαγές που έκανε στο σύστημα αρχείων η συγκεκριμένη εντολή με umount /vfat και μετά mount -t vfat -o loop /tmp/vfatfile /vfat.

Έτσι στον κατάλογο /vfat μπορούσαμε να δούμε πλέον το lklfuse.c και ως εικονίδιο.



Στη συνέχεια τρέξαμε το cptofs.c με το gdb για να δούμε πως λειτουργεί το συγκεκριμένο εργαλείο.

Παρατηρήσαμε ότι στην main, συγκεκριμένα στην γραμμή 564 βρίσκεται η lkl_start_kernel από την οποία καλείται η setup.c μέσα στον φάκελο arch και ξεκινάει ο πυρήνας της βιβλιοθήκης lkl ενώ στη συνέχεια καλείται η lkl_cpu_init που βρίσκεται στο cpu.c δημιουργώντας έτσι νήματα που εκτελούν διεργασίες παράλληλα, πιο απλά «ξυπνά» τον πυρήνα δημιουργώντας νήματα που εκτελούν διάφορες λειτουργίες.

Έπειτα είδαμε πως η εντολή lkl_mount_dev κάνει mount το σύστημα αρχείων, στη συγκεκριμένη περίπτωση κάνει mount το fat.

Η συνάρτηση umask αλλάζει τις άδειες ενός αρχείου ή φακέλου.

Η copy_one αντιγράφει το αρχείο από τον έναν φάκελο στον άλλον

Η lkl_umount_dev κάνει unmount το σύστημα αρχείων

Τέλος η lkl_sys_halt σταματάει τη λειτουργία του πυρήνα της βιβλιοθήκης, υπό κανονικές προϋποθέσεις ο πυρήνας μπαίνει σε αναμονή αλλά στη συγκεκριμένη περίπτωση είναι βιβλιοθήκη, και για αυτό σταματά την λειτουργία.

Βήμα 1ο - Τοποθέτηση των printk:

Αφού ψάξαμε μέσω του gdb μέσα στον κατάλογο fat ξεκινήσαμε να βάζουμε printk σε συγκεκριμένες συναρτήσεις για να καταλάβουμε πως λειτουργεί το σύστημα αρχείων fat. Υποθέσαμε από το όνομα της κάθε συνάρτησης ποια λειτουργία επιτελεί.

<u>Βάλαμε τα εξής printk:</u>

Στον κατάλογο lkl-source/fs/fat:

- Στο inode.c προσθέσαμε printk στις συναρτήσεις:
  - fat_get_cluster
  - fat_get_block
  - fat_iget
  - fat_read_root
  - fat_build_inode -> δημιουργεί το inode
  - fat_alloc_inode -> δεσμεύει το inode
  - fat_write_inode -> γράφει στον δίσκο το inode

- init_fat_fs -> αρχικοποιεί το fat

Τα print γράφτηκαν στις εξής γραμμές:

101	450	851	1434
124	592	904	1952
178	738	1397	

- Στο faten.c προσθέσαμε printk στις συναρτήσεις:
  - fat_ent_read
  - fat_ent_write
  - fat_ent_read_block -> διαβάζει το block στο δίσκο
  - fat_alloc_clusters -> προσθέτει στον δίσκο ένα cluster

Τα print γράφτηκαν στις εξής γραμμές:

356	415	442	476

- Στο file.c προσθέσαμε printk στις συναρτήσεις:
  - fat_free

Τα print γράφτηκαν στις εξής γραμμές:

284

- Στο namei_vfat.c προσθέσαμε printk στις συναρτήσεις:
  - setup
  - vfat_mount -> κάνει mount το σύστημα vfat

Τα print γράφτηκαν στις εξής γραμμές:

1061	1077

Στον κατάλογο lkl-source/mm:

- Στο page_writeback_init.c προσθέσαμε printk στις συναρτήσεις:
  - _wb_writeout_inc
  - page_writeback_init

Τα print γράφτηκαν στις εξής γραμμές:

603	2083

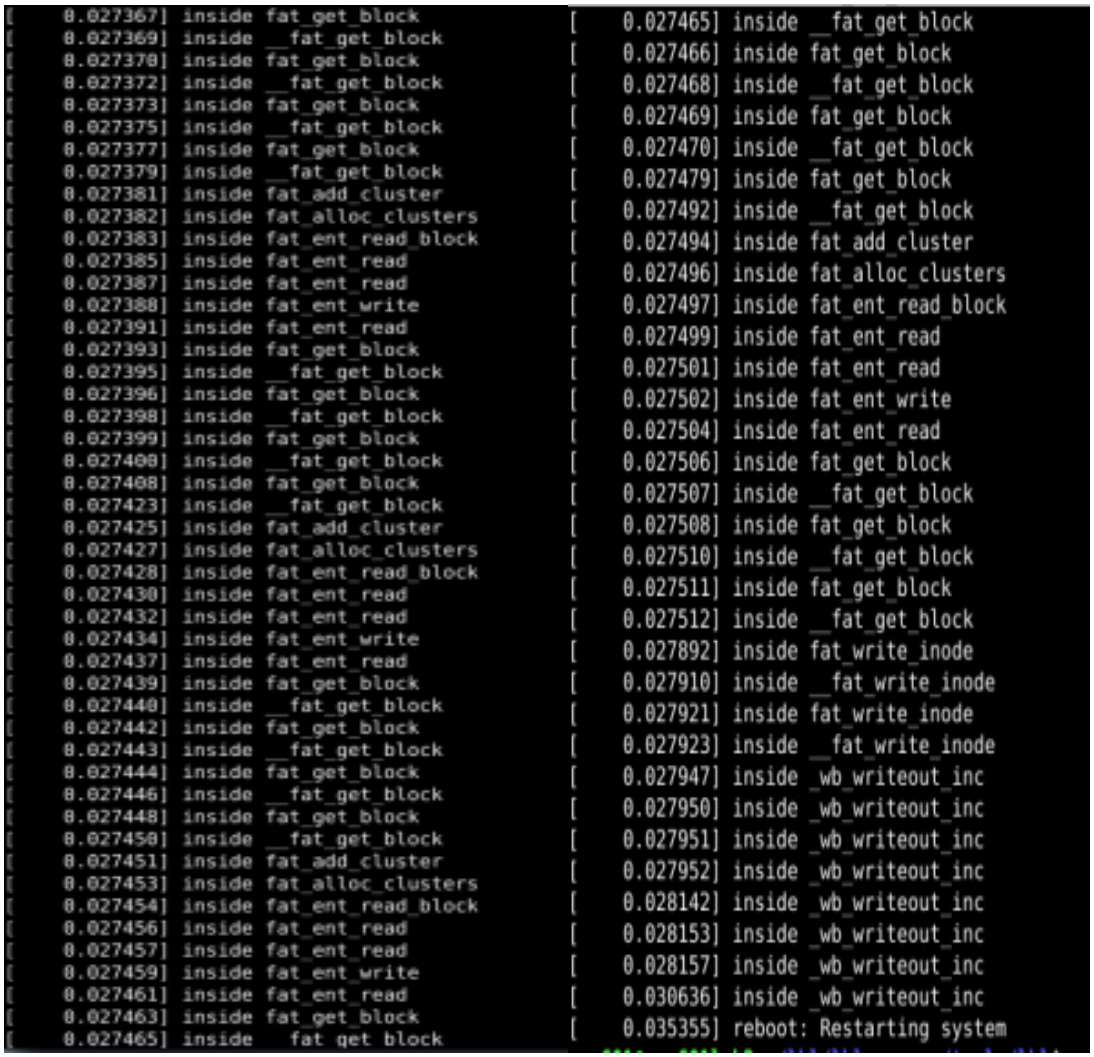
Για να δούμε τα print στο terminal , θα πλοηγηθούμε στο κατάλογο lkl/lkl-source/tools/lkl και θα τρέξουμε την εντολή **./cptofs -i /tmp/vfatfile -p -t vfat lklfuse.c /**

*Προϋπόθεση είναι να έχουμε καθαρίσει την τρέχουσα μεταγλώττιση με make -j8 clean; make -j8 -C tools/lkl clean και να έχουμε κάνει compile στον lkl/lkl-source όπως αναφέρεται και στην σελίδα 1**

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

Αφού έχουμε κάνει τα απαραίτητα βήματα, μπορούμε να δούμε να εκτυπώνονται τα εξής:

```
[    0.026412] This architecture does not have kernel memory protection.
[    0.026745] inside vfat_mount
[    0.026772] inside setup
[    0.026789] inside calc_fat_clusters
[    0.026793] inside fat_alloc_inode
[    0.026798] inside fat_alloc_inode
[    0.026800] inside fat_alloc_inode
[    0.026801] inside fat_read_root
[    0.027243] inside fat_build_inode
[    0.027262] inside fat_iget
[    0.027264] inside fat_alloc_inode
[    0.027275] inside fat_get_block
[    0.027276] inside __fat_get_block
[    0.027278] inside fat_add_cluster
[    0.027279] inside fat_alloc_clusters
[    0.027284] inside fat_ent_read_block
[    0.027299] inside fat_get_block
[    0.027301] inside __fat_get_block
[    0.027302] inside fat_get_block
[    0.027304] inside __fat_get_block
[    0.027305] inside fat_get_block
[    0.027309] inside __fat_get_block
[    0.027313] inside fat_get_block
[    0.027314] inside __fat_get_block
[    0.027316] inside fat_add_cluster
[    0.027317] inside fat_alloc_clusters
[    0.027318] inside fat_ent_read_block
[    0.027321] inside fat_ent_read
[    0.027323] inside fat_ent_read
[    0.027324] inside fat_ent_write
[    0.027327] inside fat_ent_read
[    0.027330] inside fat_get_block
[    0.027331] inside __fat_get_block
[    0.027333] inside fat_get_block
[    0.027334] inside __fat_get_block
[    0.027336] inside fat_get_block
[    0.027337] inside __fat_get_block
[    0.027348] inside fat_get_block
[    0.027350] inside __fat_get_block
[    0.027353] inside fat_add_cluster
[    0.027354] inside fat_alloc_clusters
[    0.027356] inside fat_ent_read_block
[    0.027358] inside fat_ent_read
[    0.027359] inside fat_ent_read
[    0.027361] inside fat_ent_write
[    0.027362] inside fat_ent_read
```


[0.027367] inside fat_get_block
[0.027369] inside __fat_get_block
[0.027370] inside fat_get_block
[0.027372] inside __fat_get_block
[0.027373] inside fat_get_block
[0.027375] inside __fat_get_block
[0.027377] inside fat_get_block
[0.027379] inside __fat_get_block
[0.027381] inside fat_add_cluster
[0.027382] inside fat_alloc_clusters
[0.027383] inside fat_ent_read_block
[0.027385] inside fat_ent_read
[0.027387] inside fat_ent_read
[0.027388] inside fat_ent_write
[0.027391] inside fat_ent_read
[0.027393] inside fat_get_block
[0.027395] inside __fat_get_block
[0.027396] inside fat_get_block
[0.027398] inside __fat_get_block
[0.027399] inside fat_get_block
[0.027408] inside __fat_get_block
[0.027408] inside fat_get_block
[0.027423] inside __fat_get_block
[0.027425] inside fat_add_cluster
[0.027427] inside fat_alloc_clusters
[0.027428] inside fat_ent_read_block
[0.027430] inside fat_ent_read
[0.027432] inside fat_ent_read
[0.027434] inside fat_ent_write
[0.027437] inside fat_ent_read
[0.027439] inside fat_get_block
[0.027440] inside __fat_get_block
[0.027442] inside fat_get_block
[0.027443] inside __fat_get_block
[0.027444] inside fat_get_block
[0.027446] inside __fat_get_block
[0.027448] inside fat_get_block
[0.027450] inside __fat_get_block
[0.027451] inside fat_add_cluster
[0.027453] inside fat_alloc_clusters
[0.027454] inside fat_ent_read_block
[0.027456] inside fat_ent_read
[0.027457] inside fat_ent_read
[0.027459] inside fat_ent_write
[0.027461] inside fat_ent_read
[0.027463] inside fat_get_block
[0.027465] inside __fat_get_block
[0.027465] inside __fat_get_block
[0.027466] inside fat_get_block
[0.027468] inside __fat_get_block
[0.027469] inside fat_get_block
[0.027470] inside __fat_get_block
[0.027479] inside fat_get_block
[0.027492] inside __fat_get_block
[0.027494] inside fat_add_cluster
[0.027496] inside fat_alloc_clusters
[0.027497] inside fat_ent_read_block
[0.027499] inside fat_ent_read
[0.027501] inside fat_ent_read
[0.027502] inside fat_ent_write
[0.027504] inside fat_ent_read
[0.027506] inside fat_get_block
[0.027507] inside __fat_get_block
[0.027508] inside fat_get_block
[0.027510] inside __fat_get_block
[0.027511] inside fat_get_block
[0.027512] inside __fat_get_block
[0.027892] inside fat_write_inode
[0.027910] inside __fat_write_inode
[0.027921] inside fat_write_inode
[0.027923] inside __fat_write_inode
[0.027947] inside _wb_writeout_inc
[0.027950] inside _wb_writeout_inc
[0.027951] inside _wb_writeout_inc
[0.027952] inside _wb_writeout_inc
[0.028142] inside _wb_writeout_inc
[0.028153] inside _wb_writeout_inc
[0.028157] inside _wb_writeout_inc
[0.030636] inside _wb_writeout_inc
[0.035355] reboot: Restarting system

Βήμα 2ο – Αποθήκευση εγγραφών των τροποποιήσεων του FAT:

Για το βήμα 2 το σκεπτικό μας ήταν να φτιάξουμε ένα αρχείο journal στον φάκελο tmp, την open() την γράψαμε στο namei_vfat.c και βάλαμε και έναν file descriptor στο fat.h.

Στη συνέχεια θα κάναμε write τα structs του inode, file allocation table, directory entries από τα αρχεία inode.c, fat.c βάζοντας write σε συναρτήσεις των αρχείων που τροποποιούν τα συγκεκριμένα structs, όπως για παράδειγμα fat_write_inode(). Όμως κατά το compile είχαμε error “sys/types.h no such file or directory” με την open() και δοκιμάσαμε διάφορες άλλες βιβλιοθήκες (π.χ. linux/fcntl.h) όμως και πάλι δεν αναγνωριζόταν η συνάρτηση open(). Επίσης μια ιδέα ήταν να γράψουμε τις συναρτήσεις open(), write() στα cptofs.c, boot.c όμως σε αυτή την περίπτωση δεν μπορέσαμε να βρούμε πρόσβαση στα structs του συστήματος αρχείων fat.

Επίσης δοκιμάσαμε όπως ζητάει στο ερώτημα iii) με τη sys_open(). Παρατηρήσαμε ότι η sys_open() παίρνει 3 ορίσματα εν αντιθέσει με την open(). Γράψαμε την sys_open στο αρχείο namei_vfat.c και χρειάστηκαν οι βιβλιοθήκες linux/kernel.h, linux/syscalls.h. Σε αυτή την περίπτωση κατά το compile δεν είχαμε κάποιο error όμως ενώ γράφαμε «fs/fat/journal» δεν φτιαχνόταν κανένα αρχείο σε αυτόν τον φάκελο όπως και σε άλλους που δοκιμάσαμε.