Λειτουργικά Συστήματα Δραστηριότητα 5

2024-2025

ΑΜ: 1096060

Ονοματεπώνυμο: Μαρία-Νίκη Ζωγράφου

**Περιεχόμενα**

[Ερώτημα 1 3](#_Toc188191629)

[Ερώτημα 2 6](#_Toc188191630)

[**Δομή των δεδομένων από το /dev/input/mice:** 7](#_Toc188191631)

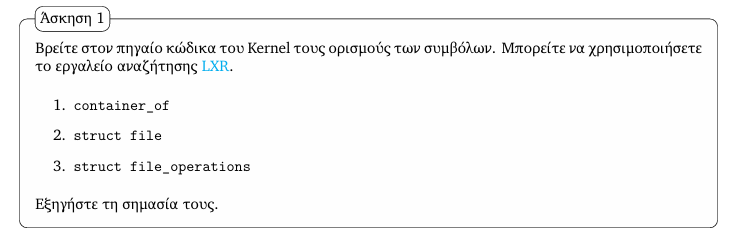
[Ερώτημα 3 8](#_Toc188191632)

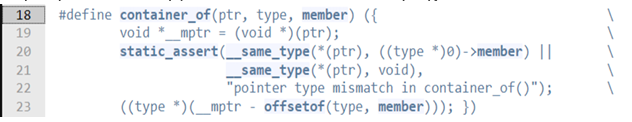
[Ερώτημα 4 8](#_Toc188191633)

[**Συμπληρώσεις στον κώδικα του chardev.c:** 9](#_Toc188191634)

[**Δημιουργία εφαρμογής χρήστη:** 11](#_Toc188191635)

# Ερώτημα 1



1.  Το container\_of είναι ένα macro που στον πυρήνα του Linux με σκοπό να υπολογίσει τη διεύθυνση μιας δομής (structure) που "περιέχει" ένα μέλος της, όταν δίνουμε τη διεύθυνση του μέλους αυτού ως input.
2. Η **struct file** είναι μια δομή που χρησιμοποιείται στον πυρήνα του Linux για να αντιπροσωπεύει ένα **ανοικτό αρχείο**. Αυτή η δομή περιέχει όλες τις απαραίτητες πληροφορίες για τη διαχείριση ενός αρχείου που έχει ανοιχτεί από έναν χρήστη ή ένα σύστημα.

struct file {

    atomic\_long\_t           f\_count;

    spinlock\_t          f\_lock;

    fmode\_t             f\_mode;

    const struct file\_operations    \*f\_op;

    struct address\_space        \*f\_mapping;

    void                \*private\_data;

    struct inode            \*f\_inode;

    unsigned int            f\_flags;

    unsigned int            f\_iocb\_flags;

    const struct cred       \*f\_cred;

    /\* --- cacheline 1 boundary (64 bytes) --- \*/

    struct path         f\_path;

    union {

        /\* regular files (with FMODE\_ATOMIC\_POS) and directories \*/

        struct mutex        f\_pos\_lock;

        /\* pipes \*/

        u64         f\_pipe;

    };

    loff\_t              f\_pos;

#ifdef CONFIG\_SECURITY

    void                \*f\_security;

#endif

    /\* --- cacheline 2 boundary (128 bytes) --- \*/

    struct fown\_struct      \*f\_owner;

    errseq\_t            f\_wb\_err;

    errseq\_t            f\_sb\_err;

#ifdef CONFIG\_EPOLL

    struct hlist\_head       \*f\_ep;

#endif

    union {

        struct callback\_head    f\_task\_work;

        struct llist\_node   f\_llist;

        struct file\_ra\_state    f\_ra;

        freeptr\_t       f\_freeptr;

    };

    /\* --- cacheline 3 boundary (192 bytes) --- \*/

} \_\_randomize\_layout

  \_\_attribute\_\_((aligned(4)));  /\* lest something weird decides that 2 is OK \*/

Η **struct file** περιγράφει ένα συγκεκριμένο άνοιγμα του αρχείου (open instance) και δεν αναφέρεται στο ίδιο το αρχείο στο σύστημα αρχείων. Περιλαμβάνει το struct inode το οποίο αναφέρεται στο ίδιο το αρχείο στο σύστημα αρχείων. Περιέχει πληροφορίες όπως το μέγεθος, τα δικαιώματα, και το αναγνωριστικό (inode number).

1. Η **struct file\_operations** είναι μια δομή δεδομένων στον πυρήνα του Linux που ορίζει ένα σύνολο λειτουργιών/συναρτήσεων που μπορούν να εκτελεστούν σε ένα αρχείο. Περιέχει δείκτες σε συναρτήσεις, οι οποίες χρησιμοποιούνται για τη διαχείριση του αρχείου, όπως το άνοιγμα, το κλείσιμο, η ανάγνωση και η εγγραφή.

struct file\_operations {

    struct module \*owner;

    fop\_flags\_t fop\_flags;

    loff\_t (\*llseek) (struct file \*, loff\_t, int);

    ssize\_t (\*read) (struct file \*, char \_\_user \*, size\_t, loff\_t \*);

    ssize\_t (\*write) (struct file \*, const char \_\_user \*, size\_t, loff\_t \*);

    ssize\_t (\*read\_iter) (struct kiocb \*, struct iov\_iter \*);

    ssize\_t (\*write\_iter) (struct kiocb \*, struct iov\_iter \*);

    int (\*iopoll)(struct kiocb \*kiocb, struct io\_comp\_batch \*,

            unsigned int flags);

    int (\*iterate\_shared) (struct file \*, struct dir\_context \*);

    \_\_poll\_t (\*poll) (struct file \*, struct poll\_table\_struct \*);

    long (\*unlocked\_ioctl) (struct file \*, unsigned int, unsigned long);

    long (\*compat\_ioctl) (struct file \*, unsigned int, unsigned long);

    int (\*mmap) (struct file \*, struct vm\_area\_struct \*);

    int (\*open) (struct inode \*, struct file \*);

    int (\*flush) (struct file \*, fl\_owner\_t id);

    int (\*release) (struct inode \*, struct file \*);

    int (\*fsync) (struct file \*, loff\_t, loff\_t, int datasync);

    int (\*fasync) (int, struct file \*, int);

    int (\*lock) (struct file \*, int, struct file\_lock \*);

    unsigned long (\*get\_unmapped\_area)(struct file \*, unsigned long, unsigned long, unsigned long, unsigned long);

    int (\*check\_flags)(int);

    int (\*flock) (struct file \*, int, struct file\_lock \*);

    ssize\_t (\*splice\_write)(struct pipe\_inode\_info \*, struct file \*, loff\_t \*, size\_t, unsigned int);

    ssize\_t (\*splice\_read)(struct file \*, loff\_t \*, struct pipe\_inode\_info \*, size\_t, unsigned int);

    void (\*splice\_eof)(struct file \*file);

    int (\*setlease)(struct file \*, int, struct file\_lease \*\*, void \*\*);

    long (\*fallocate)(struct file \*file, int mode, loff\_t offset,

              loff\_t len);

    void (\*show\_fdinfo)(struct seq\_file \*m, struct file \*f);

#ifndef CONFIG\_MMU

    unsigned (\*mmap\_capabilities)(struct file \*);

#endif

    ssize\_t (\*copy\_file\_range)(struct file \*, loff\_t, struct file \*,

            loff\_t, size\_t, unsigned int);

    loff\_t (\*remap\_file\_range)(struct file \*file\_in, loff\_t pos\_in,

                   struct file \*file\_out, loff\_t pos\_out,

                   loff\_t len, unsigned int remap\_flags);

    int (\*fadvise)(struct file \*, loff\_t, loff\_t, int);

    int (\*uring\_cmd)(struct io\_uring\_cmd \*ioucmd, unsigned int issue\_flags);

    int (\*uring\_cmd\_iopoll)(struct io\_uring\_cmd \*, struct io\_comp\_batch \*,

                unsigned int poll\_flags);

} \_\_randomize\_layout;

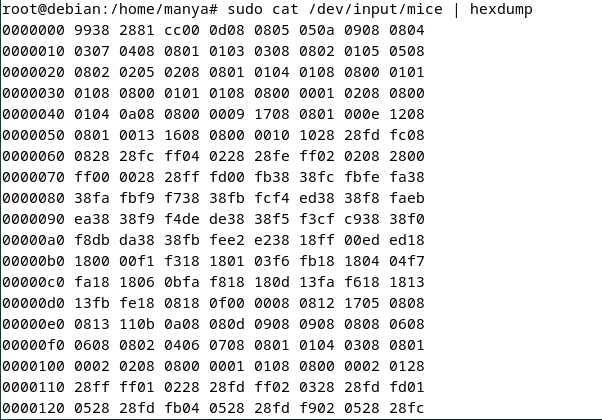
**Σημείωση:** Οι εντολές **#ifndef** και **#endif** στη C είναι μέρος του preprocessor, which processes the source code before it is compiled. ο οποίος επεξεργάζεται τον πηγαίο κώδικα πριν από τη μεταγλώττιση. Αυτές οι εντολές ελέγχουν το **conditional compilation** (υπό όρους μεταγλώττιση), επιτρέποντας να συμπεριληφθούν ή να αποκλειστούν συγκεκριμένα τμήματα του κώδικα ανάλογα με συγκεκριμένες συνθήκες. Η εντολή **#ifndef** σημαίνει "αν δεν έχει οριστεί" και ελέγχει αν μια συγκεκριμένη μακροεντολή ή σταθερά δεν έχει οριστεί. Αν η μακροεντολή δεν έχει οριστεί, ο κώδικας μεταξύ των **#ifndef** και **#endif** περιλαμβάνεται στη μεταγλώττιση. Αντίθετα, αν έχει οριστεί, ο κώδικας παραλείπεται. Η εντολή #endif σηματοδοτεί το τέλος του μπλοκ που ξεκινά με **#ifndef, #ifdef** ή **#if** .

# Ερώτημα 2

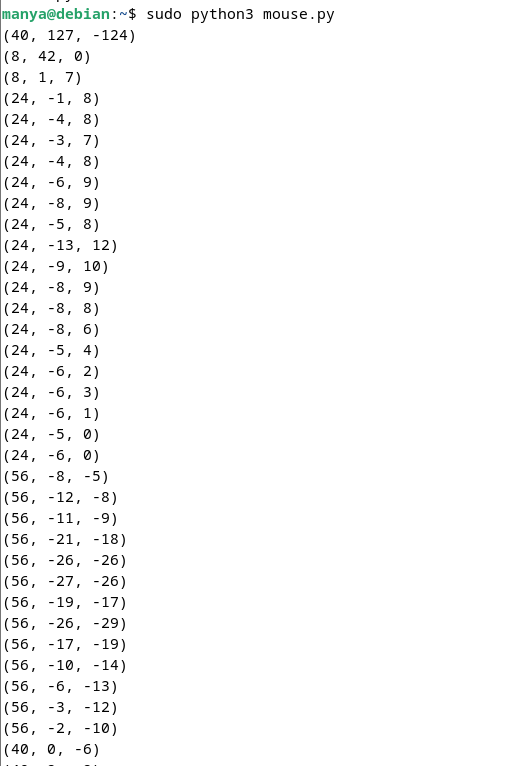
To sudo apt install hexdump: **E:** Unable to locate package hexdump. Επομένως χρησιμοποιήθηκε το sudo apt-get install bsdmainutils.

Εκτελούμε την εντολή sudo cat /dev/input/mice | hexdump

Παίρνουμε σαν αποτέλεσμα το stream από bytes από την κίνηση του ποντικιού!!



**Python script:**

****Το f.read(3) διαβάζει **3 bytes κάθε φορά** από τη συσκευή ποντικιού. Αυτά τα bytes περιέχουν πληροφορίες για τις κινήσεις και τις ενέργειες του ποντικιού.

Το struct.unpack('bbb', data) μετατρέπει τα 3 bytes σε υπογεγραμμένους ακέραιους (signed integers). Το αποτέλεσμα εκτυπώνεται ως τριάδα ακέραιων αριθμών.

## **Δομή των δεδομένων από το /dev/input/mice:**

**Πρώτο byte (flags)**:

Περιέχει πληροφορίες για τα κουμπιά του ποντικιού:

Bit 0: Αριστερό κουμπί (1 = πατημένο).

Bit 1: Δεξί κουμπί (1 = πατημένο).

Bit 2: Μεσαίο κουμπί (1 = πατημένο).

Περιλαμβάνει επίσης πληροφορίες για την κατεύθυνση της κίνησης.

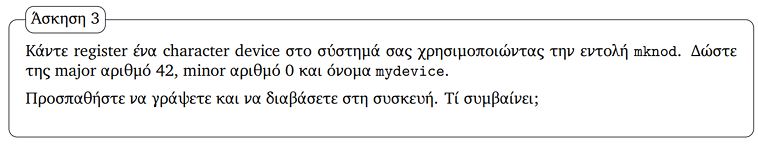
**Δεύτερο byte (x):**

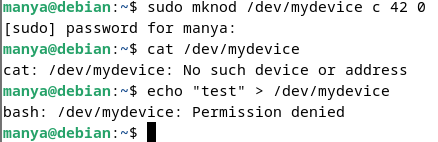
**Δείχνει τη μετατόπιση στον άξονα X (αριστερά ή δεξιά) και είναι προσημασμένος ακέραιος αριθμός (π.χ., θετικός για δεξιά, αρνητικός για αριστερά).**

**Τρίτο byte (y):**

**Δείχνει τη μετατόπιση στον άξονα Y (πάνω ή κάτω) και είναι προσημασμένος ακέραιος αριθμός (π.χ., θετικός για κάτω, αρνητικός για πάνω).**

# Ερώτημα 3

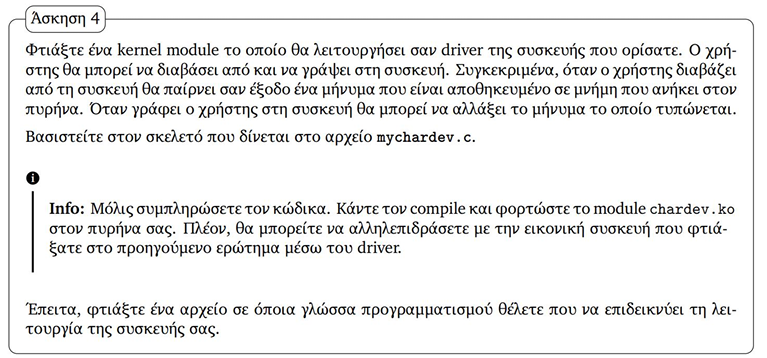
****

****

Επειδή δεν έχουμε υλοποιήσει ή συνδέσει κάποιον driver για το major αριθμό 42, το σύστημα δεν ξέρει πώς να διαχειριστεί αυτή τη συσκευή και εμφανίζει σφάλμα.

Το αρχείο συσκευής που δημιουργήσαμε με την εντολή mknod δεν έχει συσχετιστεί με κάποιον υπαρκτό driver. Όταν εκτελούμε την εντολή mknod /dev/mydevice c 42 0, δημιουργούμε ένα **character device file** με major αριθμό 42 και minor αριθμό 0. Ωστόσο, το λειτουργικό σύστημα αναμένει ότι υπάρχει ένας driver καταχωρημένος στον πυρήνα για τη διαχείριση συσκευών με τον συγκεκριμένο major αριθμό. Επειδή δεν έχουμε υλοποιήσει και φορτώσει έναν τέτοιο driver, οι λειτουργίες ανάγνωσης (read) και εγγραφής (write) δεν μπορούν να πραγματοποιηθούν, με αποτέλεσμα να εμφανίζονται σφάλματα όπως Permission Denied ή No such device or address.

# Ερώτημα 4



Ο κώδικας που παρέχεται στο αρχείο dev.c είναι ο σκελετός για τη δημιουργία ενός character device driver. Ο driver αυτός θα επιτρέπει στον χρήστη να διαβάζει και να γράφει δεδομένα στη συσκευή, αποθηκεύοντας το μήνυμα στον buffer που ανήκει στη δομή του driver.

## **Συμπληρώσεις στον κώδικα του chardev.c:**

**1. Προσθήκη του μέλους cdev και του buffer στη δομή my\_device\_data:** Στη δομή my\_device\_data, προσθέτουμε:

struct cdev cdev; // Χειριστής character device

char buffer[BUFFER\_SIZE]; // Buffer αποθήκευσης δεδομένων

**2.** **Χρήση του container\_of για απόκτηση του δείκτη στη δομή:** Στη συνάρτηση my\_cdev\_open, συμπληρώνουμε:

data = container\_of(inode->i\_cdev, struct my\_device\_data, cdev);

**3. Διαχείριση της μεταβλητής access:** Στη συνάρτηση my\_cdev\_open, ελέγχουμε την πρόσβαση:

if (atomic\_cmpxchg(&data->access, 0, 1) != 0) {

    return -EBUSY; // Συσκευή είναι ήδη σε χρήση

}

τη συνάρτηση my\_cdev\_release, επαναφέρουμε την πρόσβαση:

atomic\_set(&data->access, 0);

**4. Αντιγραφή δεδομένων από και προς τον buffer:**

Στη my\_cdev\_read:

if (copy\_to\_user(user\_buffer, data->buffer + \*offset, to\_read)) {

    return -EFAULT; // Σφάλμα κατά την αντιγραφή

}

Στη my\_cdev\_write:

if (copy\_from\_user(data->buffer + \*offset, user\_buffer, size)) {

    return -EFAULT; // Σφάλμα κατά την αντιγραφή

}

**5. Προσθήκη συναρτήσεων στη δομή my\_fops:**

.open = my\_cdev\_open,

.release = my\_cdev\_release,

.read = my\_cdev\_read,

.write = my\_cdev\_write,

**6. Προσθήκη συναρτήσεων στη δομή my\_fops:**

Στη συνάρτηση my\_init:

err = register\_chrdev\_region(MKDEV(MY\_MAJOR, MY\_MINOR), NUM\_MINORS, MODULE\_NAME);

if (err != 0) {

    pr\_info("Register character device failed.");

    return err;

}

for (i = 0; i < NUM\_MINORS; i++) {

    cdev\_init(&devs[i].cdev, &my\_fops);

    devs[i].cdev.owner = THIS\_MODULE;

    err = cdev\_add(&devs[i].cdev, MKDEV(MY\_MAJOR, MY\_MINOR + i), 1);

    if (err) {

        pr\_info("Add cdev failed.");

        return err;

    }

}

Στη συνάρτηση my\_exit:

for (i = 0; i < NUM\_MINORS; i++) {

    cdev\_del(&devs[i].cdev);

}

unregister\_chrdev\_region(MKDEV(MY\_MAJOR, MY\_MINOR), NUM\_MINORS);

Επιβεβαίωση ότι το module μας λειτουργεί ορθά:

root@debian:/media/sf\_OS3# sudo insmod dev.ko

root@debian:/media/sf\_OS3# lsmod | grep dev

dev 24576 0

snd\_seq\_device 16384 1 snd\_seq

snd 126976 10 snd\_seq,snd\_seq\_device,snd\_intel8x0,snd\_timer,snd\_ac97\_codec,snd\_pcm

joydev 28672 0

evdev 28672 10

ppdev 24576 0

parport 73728 3 parport\_pc,lp,ppdev

root@debian:/media/sf\_OS3# sudo mknod /dev/mydevice c 42 0

mknod: /dev/mydevice: File exists

root@debian:/media/sf\_OS3# sudo chmod 666 /dev/mydevice

root@debian:/media/sf\_OS3# echo "Νέο μήνυμα!" > /dev/mydevice

root@debian:/media/sf\_OS3# cat /dev/mydevice

Νέο μήνυμα!

root@debian:/media/sf\_OS3# sudo rmmod dev

Ο driver μας είναι λειτουργικός και μπορούμε να γράψουμε και να διαβάσουμε από την συσκευή. Θα το επιβεβαιώσουμε και με εφαρμογή σε python.

## **Δημιουργία εφαρμογής χρήστη:**

import os

DEVICE\_PATH = "/dev/mydevice"

def write\_to\_device(message):

    """Γράφει ένα μήνυμα στη συσκευή."""

    try:

        with open(DEVICE\_PATH, "w") as device:

            device.write(message)

        print(f"Γράφτηκε το μήνυμα: '{message}' στη συσκευή.")

    except Exception as e:

        print(f"Σφάλμα κατά την εγγραφή: {e}")

def read\_from\_device():

    """Διαβάζει δεδομένα από τη συσκευή."""

    try:

        with open(DEVICE\_PATH, "r") as device:

            data = device.read()

        print(f"Διαβάστηκε από τη συσκευή: '{data}'")

    except Exception as e:

        print(f"Σφάλμα κατά την ανάγνωση: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

    print("1. Γράψτε δεδομένα στη συσκευή.")

    print("2. Διαβάστε δεδομένα από τη συσκευή.")

    choice = input("Δώστε επιλογή (1/2): ")

    if choice == "1":

        message = input("Δώστε το μήνυμα που θέλετε να γράψετε στη συσκευή: ")

        write\_to\_device(message)

    elif choice == "2":

        read\_from\_device()

    else:

        print("Μη έγκυρη επιλογή.")

Τρέχουμε την εφαρμογή και επιβεβαιώνουμε ότι λειτουργεί σωστά:

****