
# 2η Εργαστηριακή Άσκηση Λειτουργικών Συστημάτων

Γιώργος Ξυπόλιτος 03113629
Στέφανος-Ευστράτιος Πετρίδης 03113418

18-5-2017

# Οδηγός Ασύρματου Δικτύου Αισθητήρων στο ΛΣ Linux

## Σκοπός της άσκησης

Σκοπός της άσκησης είναι η δημιουργία μέρους ενός οδηγού συσκευής για μια συσκευή χαρακτήρων για το λειτουργικό σύστημα *Linux*.
Ο κώδικας που αναπτύχθηκε προορίζεται για εκτέλεση σε χώρο πυρήνα με σκοπό την ενσωμάτωσή του σε αυτόν. Αυτό απαιτεί πιο προσεκτικό προγραμματισμό και μια ¨τριβυτε' μηχανή πάνω στην οποία θα ελέγχουμε την λειτουργικότητα του κώδικά μας, καθώς σε περίπτωση σφάλματος υπάρχει η πιθανότητα κατέρρευσης του συστήματος.

### Λειτουργίες Οδηγού

Μερικές από τις λειτουργίες που πρέπει να υλοποιεί το $interface$ ενός οδηγού συγσκευής μιας συσκευής χαρακτήρων είναι τα $open()$, $read()$, $write()$, $lseek()$ και $close()$. Στον συγκεκριμένο οδηγό υλοποιούμε τα $open()$, $read()$, $close()$ όπως φαίνεται και στο αντίστοιχο $struct\ file_operations$ που αντιστοιχείται σε κάθε συσκευή του συγκεκριμένου οδηγού.

### Γενικά για τον οδηγό

Η ανάγκη για ύπαρξη του οδηγού οφείλεται στο γεγονός ότι η *pre-existing approach* δεν προσφέρει ταυτόχρονη πρόσβαση στην εικονική θύρα στην οποία κατευθύνονται οι μετρήσεις, καθώς και δεν διαχωρίζει από που προέρχονται, δηλαδή από ποιον αισθητήρα, αλλά και το είδος της μέτρησης. Επομένως, είναι αδύνατη η προσπέλασή τους από παραπάνω από μία διεργασίες ταυτόχρονα και η αξιοποίηση των δεδομένων τα οποία διαβάζονται.
Μέσω του οδηγού διαχωρίζουμε τις μετρήσεις και τους αισθητήρες, ώστε να μπορούμε να παρέχουμε διαφορετικά δικαιώματα σε κάθε αισθητήρα για τις διάφορες διεργασίες που μπορεί να προσπελαύνουν τις εικονικές συσκευές που δημιουργούνται.
Δεδομένου 16 συσκευών με 3 μετρήσεις η κάθε μία, έχουμε 48 εικονικές συσκευές. Για την προσπέλαση των μετρήσεων, δημιουργούνται ειδικά αρχεία υπό τον κατάλογο */dev*. Για τον διαχωρισμό των εικονικών συσκευών, αναθέτουμε διαφορετικό μινορ νυμβερ στα αρχεία, ανάλογα την πραγματική συσκευή και την μέτρηση στην οποία αντιστοιχεί.
Το κομμάτι που καλούμαστε να υλοποιήσουμε είναι αυτό της ανάκτησης των προωθημένων δεδομένων από τους αισθητήρες και της επεξεργασίας τους ανάλογα με τον αισθητήρα από τον οποίο προέρχονται. Οι συναρτήσεις για την μετατροπή των δεδομένων δίνονται από τον κατασκευαστή των αισθητήρων.

## Συναρτήσεις Οδηγού

Οι συναρτήσεις που χρειάστηκε να υλοποιηθούν στο αρχείο *lunix_chrdev.c* είναι οι εξής:

**1.init**

Η συνάρτηση *lunix_chrdev_init* αρχικοποιεί τις συσκευές στον kernel. Δηλαδή ορίζει τον *major number* των συσκευών και έπειτα δεσμεύει μια περιοχή *minor numbers* η οποία θα χρησιμοποιηθεί για τους αισθητήρες. Έπειτα προσθέτει τις συσκευές αυτές στις ενεργές συσκευές. Επίσης, αυτή η συνάρτηση ορίζει και τις συναρτήσεις διεπαφής των συσκευών με τον πυρήνα.

**2.open**

Η συνάρτηση *lunix_chrdev_open* παίρνει το inode του αρχείου και ένα *struct file* το οποίο περιέχει τις συναρτήσεις που υλοποιούνται για τον συγκεκριμένο οδηγό και αρχικοποιεί το *lunix_chrdev_state_struct*, το οποίο είναι η δομή που περιέχει τα δεδομένα που έχουν φτάσει από τον αισθητήρα και προορίζονται για ανάγνωση από κάποιο πρόγραμμα χρήστη, δείχνει τον τύπο του αισθητήρα που αντιπροσωπεύει και το πότε εισήλθαν τελευταία φορά δεδομένα.

**3.read**

Η συνάρτηση *lunix_chrdev_read* έχει την λειτουργία να προσπαθεί να περάσει *cnt bytes* από τον buffer του αισθητήρα, ο οποίος περιέχεται στο *private_data* της δομής $file$, στον buffer που παρέχει το πρόγραμμα χρήστη($usrbuf$) που προσπάθησε να διαβάσει από τον αισθήτηρα κάνοντας ένα *system call* στο $file\ descriptor$ του αντίστοιχου αισθητήρα. Όσο δεν υπάρχουν καινούρια δεδομένα, η διεργασία που κάλεσε την συνάρτηση τοποθετείται σε μια ουρά αναμονής εώς ούτε γίνεται *interrupt* από το *hardware* του αντίστοιχου αισθητήρα. Η υλοποίηση αυτής της συνάρτησης βασίζεται σε άλλες δύο συναρτήσεις, η μία εκ των οποίων ελέγχει αν έχουν έρθει καινούρια δεδομένα και η δεύτερη ανανεώνει και φέρνει σε κατάλληλο $format$ τα δεδομένα που ήρθαν από τον αισθητήρα.

Η συνάρτηση $lunix\_chrdev\_state\_needs\_refresh$ ελέγχει άμα ο χρόνος τελευταίας ανανέωσης των δεδομένων του buffer είναι διαφορετικός από τον χρόνο τελευταίας ανανέωσης των δεδομένων που υπάρχουν στον αισθητήρα. Η κλήση αυτής της συνάρτησης γίνεται πάντα με τον σημαφόρο της αντίστοιχης δομής κατηλειμμένο, καθώς τροποποιούμε τον αντίστοιχο buffer.

Η συνάρτηση *lunix_chrdev_state_update* ανανεώνει τα δεδομένα του buffer τοποθετόντας σε αυτόν τα δεδομένα που υπάρχουν στον αισθητήρα. Σε αυτή την περίπτωση απαιτείται η χρήση *spinlock*(*busy wait*) καθώς η ανανέωση γίνεται σε *interrupt context*, δηλαδή δεν επιτρέπεται η διεργασία να κοιμηθεί ή να καλέσει συναρτήσεις που θα την κοίμιζαν. Αφού πάρουμε τα δεδομένα τα τροποποιούμε κατάληλλα και τα τοποθετούμε έτοιμα στον buffer για μελλοντική κατανάλωση.

**4.close**

Η συνάρτηση *lunix_chrdev_release* αντιστρέφει την λειτουργία της αντίστοιχης *open*, αποδεσμεύοντας από την μνήμη τον χώρο που έπιανε το *lunix_chrdev_state_struct*, το οποίο έχει αποθηκευτεί στο *private_data* της δομής $file$.