Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Εργαστήριο Λειτουργικών Συστημάτων – 1η Άσκηση

Των Προπτυχιακών Φοιτητών
Αρώνη Παναγιώτη Α.Μ. 03109083
Δανασή Παναγιώτη Α.Μ. 03109004
e-mail: el09004@mail.ntua.gr
panaro32@hotmail.com

1. Οδηγός Ασύρματου Δικτύου Αισθητήρων

Η άσκηση αυτή αφορά την υλοποίηση ενός οδηγού συσκευής για το ΛΣ linux. Ο κώδικας που χρειάστηκε να υλοποιήσουμε, αφορούσε στο κομμάτι της δημιουργίας μιας συσκευής χαρακτήρων καθώς και στην υλοποίηση των file operations για την συσκευή αυτή. Με λίγα λόγια η δουλειά μας αφορούσε την ολοκλήρωση του “lunix-chrdev.c”.

- Αρχικά , στην lunix_chrdev_init, ζητάμε ένα εύρος από minor numbers, δημιουργούμε μία νέα συσκευή χαρακτήρων και την καταχωρούμε.

```
...
unsigned int lunix_minor_cnt = lunix_sensor_cnt << 3;

/* Allocate and register a cdevice */
cdev_init(&lunix_chrdev_cdev, &lunix_chrdev_fops);
lunix_chrdev_cdev.owner = THIS_MODULE;

/* Create dev_t from minor and major numbers */
dev_no = MKDEV(LUNIX_CHRDEV_MAJOR, 0);

lunix_chrdev_cdev.ops = &lunix_chrdev_fops;

/* Obtain device numbers to work with */
ret = register_chrdev_region(dev_no, lunix_minor_cnt, CDEV_NAME);

...
/* Inform kernel about the new device */
ret = cdev_add(&lunix_chrdev_cdev, dev_no, lunix_minor_cnt);

...
```

- Στην lunix_chrdev_open, βρίσκουμε από τα minor και major numbers τον αισθητήρα και ανοίγουμε το αντίστοιχο αρχείο. Κάνουμε allocate και αρχικοποίηση μία lunix_chrdev_state_struct και την τοποθετούμε στο private_data του file, για μελλοντική αναφορά.

```
minor_number = iminor(inode);
sensor_number = minor_number / 8;
minor_number = minor_number % 8;

sensor = &lunix_sensors[sensor_number];

/* Allocate a new Lunix character device private state structure */
state = kmalloc(sizeof (struct lunix_chrdev_state_struct), GFP_KERNEL);

...

state->sensor = sensor;
state->type = minor_number;
state->buf_timestamp = get_seconds();
state->buf_lim = 0;
sema_init(&(state->lock), 1);
debug("sema_init secceded\n");

filp->private_data = state;
filp->f_pos = 0;
```

- Στην lunix_chrdev_read, κλειδώνουμε γιατί έχουμε πρόσβαση σε κοινή δομή δεδομένων, ελέγχουμε αν έχουμε δεδομένα, αλλιώς κοιμόμαστε στην ουρά (χωρίς το κλείδωμα) και τέλος επιστρέφουμε τα δεδομένα.

```
/* Lock */
di = down_interruptible(&(state->lock));

...

/*
 * If the cached character device state needs to be
 * updated by actual sensor data (i.e. we need to report
 * on a "fresh" measurement, do so
 */
if (*f_pos == 0) {
    while (lunix_chrdev_state_update(state) == -EAGAIN) {
        /* The process needs to sleep */
        up(&(state->lock));
        if (wait_event_interruptible(sensor->wq, lunix_chrdev_state_needs_refresh(state)))
            return -ERESTARTSYS; /* signal: tell the fs layer to handle it */
        di = down_interruptible(&(state->lock));

        ...
    }
}

bytes = (cnt < state->buf_lim - *f_pos) ? cnt : state->buf_lim-*f_pos;

ret = copy_to_user(usrbuf, state->buf_data + *f_pos, bytes);

...

/* Auto-rewind on EOF mode? */
(*f_pos)+=bytes;
if(*f_pos >= state->buf_lim) *f_pos=0;
ret = bytes;

...
```

- Η lunix_chrdev_state_update, κάνει το update αφού πρώτα κάνει ένα γρήγορο έλεγχο για το αν υπάρχουν νέα δεδομένα καλώντας την lunix_chrdev_state_needs_refresh. Το κλείδωμα που χρησιμοποιούμε εδώ πέρα είναι ένα spinlock γιατί η δομή που πειράζουμε, αλλάζει και από interrupt context. Συγκεκριμένα, η ανανέωση γίνεται ασύγχρονα, από το κάτω επίπεδο, κάθε φορά που έρχονται νέα δεδομένα από τους αισθητήρες (και προκαλείται hardware interrupt).

```
/* Any new data available? */
need_refresh = lunix_chrdev_state_needs_refresh(state);
if (need_refresh) {
    spin_lock_irqsave(&(state->sensor->lock), flags);
    raw_data = state->sensor->msr_data[state->type]->values[0];
    raw_time = state->sensor->msr_data[state->type]->last_update;
    spin_unlock_irqrestore(&(state->sensor->lock), flags);
}
else {
    return -EAGAIN; /* No new data */
}
/* Now we can take our time to format them, holding only the private state semaphore */
switch(state->type) {
    debug("case 0\n");
    case BATT: data = lookup_voltage[raw_data]; break;
    debug("case 1\n");
    case TEMP: data = lookup_temperature[raw_data]; break;
    debug("case 2\n");
    case LIGHT: data = lookup_light[raw_data]; break;
    default: ;
    debug("case default\n");
}
c = (data >= 0) ? '+' : '-';
data = (data > 0) ? data : (-data);
n = sprintf(state->buf_data, "%c%ld.%ld\n", c, data/1000, data%1000 );
state->buf_timestamp = raw_time;
state->buf_lim = n;
```

Για τον έλεγχο της ορθής λειτουργίας του οδηγού μας και των κλειδωμάτων, γράψαμε ένα απλό πρόγραμμα (lunix_user.c) όπου ανοίγει το αρχείο της συσκευής, κάνει ένα fork και οι δύο διεργασίες διαβάζουν και γράφουν μετρήσεις στην οθόνη.

```
#include <stdio.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    FILE *fp = fopen(argv[1],"r");
    float x;
    fork();
    pid_t mypid = getpid();
    while(1)
    {
        fscanf(fp,"%f",&x);
        printf("(%ld) %.3f\n",(long)mypid,x);
    }
    return 0;
}
```