

# Λειτουργικά Συστήματα

## 3<sup>η</sup> Εργασία

Τσάμπρας Κωνσταντίνος

υρ1083865

➔ 1)

- 1. `container_of`: ένα macro του linux kernel το οποίο μας επιτρέπει να πάρουμε τον δείκτη για ένα struct το οποίο περιέχει το δοθέν πεδίο

```
18 #define container_of(ptr, type, member) ({  
19     void *__mptr = (void *) (ptr);  
20     static_assert(__same_type(*(ptr), ((type *)0)->member) ||  
21                   __same_type(*(ptr), void),  
22                   "pointer type mismatch in container_of()");  
23     ((type *) (__mptr - offsetof(type, member))); })
```

- 2. `struct file` : αντιπροσωπεύει ένα ανοικτό αρχείο στο σύστημα. Περιέχει πληροφορίες για το αρχείο όπως την κατάσταση του και την θέση του

```
992 struct file {  
993     union {  
994         struct llist_node    f_llist;  
995         struct rcu_head      f_rcuhead;  
996         unsigned int         f_iocb_flags;  
997     };  
998  
999     /*  
1000      * Protects f_ep, f_flags.  
1001      * Must not be taken from IRQ context.  
1002      */  
1003     spinlock_t                f_lock;  
1004     fmode_t                   f_mode;  
1005     atomic_long_t              f_count;  
1006     struct mutex               f_pos_lock;  
1007     loff_t                     f_pos;  
1008     unsigned int               f_flags;  
1009     struct fown_struct         f_owner;  
1010     const struct cred          *f_cred;  
1011     struct file_ra_state       f_ra;  
1012     struct path                f_path;  
1013     struct inode               *f_inode; /* cached value */  
1014     const struct file_operations *f_op;  
1015  
1016     u64                        f_version;  
1017 #ifdef CONFIG_SECURITY  
1018     void                       *f_security;  
1019 #endif  
1020     /* needed for tty driver, and maybe others */  
1021     void                       *private_data;  
1022  
1023 #ifdef CONFIG_EPOLL  
1024     /* Used by fs/eventpoll.c to link all the hooks to this file */  
1025     struct hlist_head          *f_ep;  
1026 #endif /* #ifdef CONFIG_EPOLL */  
1027     struct address_space       *f_mapping;  
1028     errseq_t                   f_wb_err;  
1029     errseq_t                   f_sb_err; /* for syncfs */  
1030 } __randomize_layout  
1031 __attribute__((aligned(4))); /* lest something weird decides that 2 is OK */  
1032
```

- 3. `struct file_operations`: Μια δομή δεδομένων που ορίζει ένα σύνολο συναρτήσεων/λειτουργιών ενός αρχείου. Δηλαδή περιέχει δείκτες

```

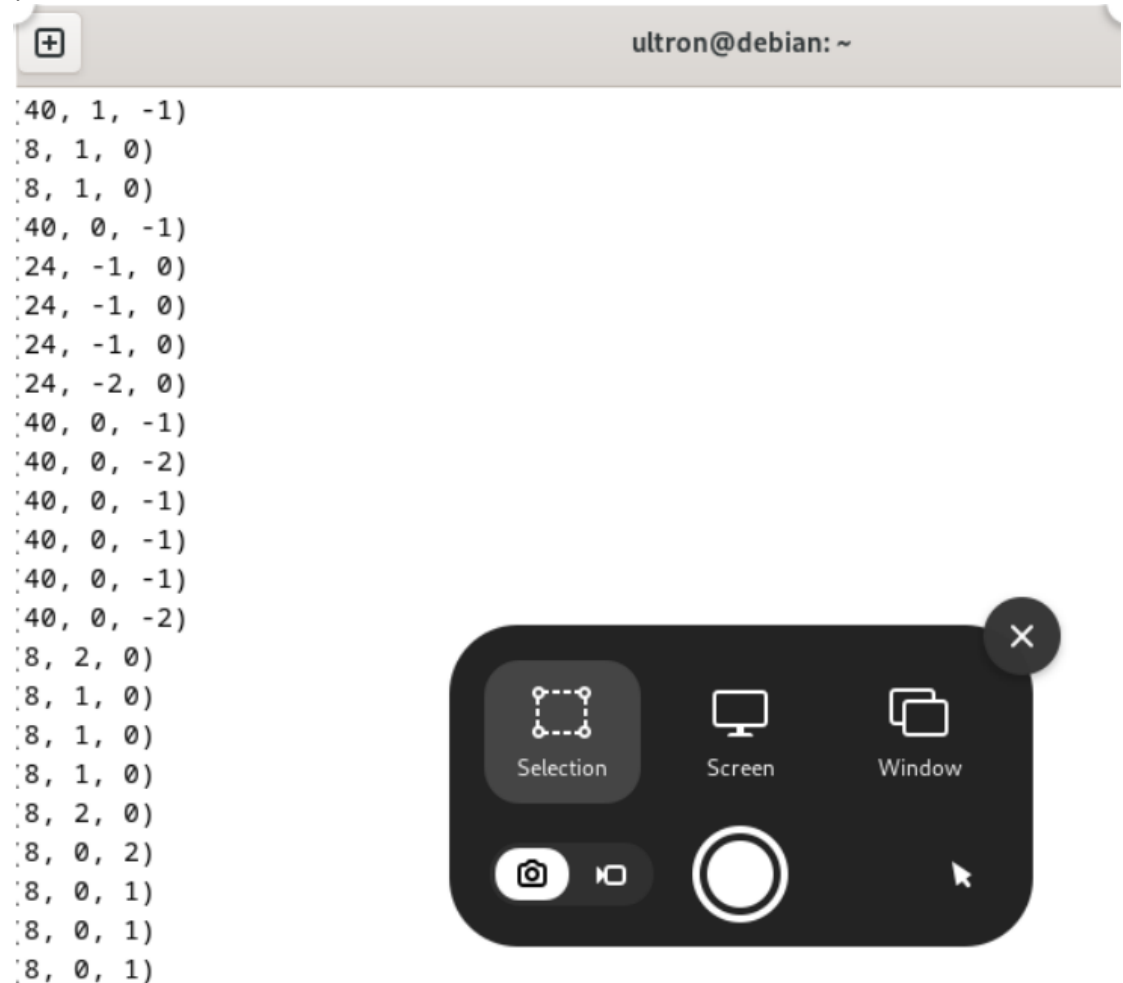
1916 struct file_operations {
1917     struct module *owner;
1918     loff_t (*llseek) (struct file *, loff_t, int);
1919     ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
1920     ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
1921     ssize_t (*read_iter) (struct kiocb *, struct iov_iter *);
1922     ssize_t (*write_iter) (struct kiocb *, struct iov_iter *);
1923     int (*iopoll) (struct kiocb *kiocb, struct io_comp_batch *,
1924         unsigned int flags);
1925     int (*iterate_shared) (struct file *, struct dir_context *);
1926     __poll_t (*poll) (struct file *, struct poll_table_struct *);
1927     long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
1928     long (*compat_ioctl) (struct file *, unsigned int, unsigned long);
1929     int (*mmap) (struct file *, struct vm_area_struct *);
1930     unsigned long mmap_supported_flags;
1931     int (*open) (struct inode *, struct file *);
1932     int (*flush) (struct file *, fl_owner_t id);
1933     int (*release) (struct inode *, struct file *);
1934     int (*fsync) (struct file *, loff_t, loff_t, int datasync);
1935     int (*fasync) (int, struct file *, int);
1936     int (*lock) (struct file *, int, struct file_lock *);
1937     unsigned long (*get_unmapped_area) (struct file *, unsigned long, unsigned long, unsigned long);
1938     int (*check_flags) (int);
1939     int (*flock) (struct file *, int, struct file_lock *);
1940     ssize_t (*splice_write) (struct pipe_inode_info *, struct file *, loff_t *, size_t, unsigned int);
1941     ssize_t (*splice_read) (struct file *, loff_t *, struct pipe_inode_info *, size_t, unsigned int);
1942     void (*splice_eof) (struct file *file);
1943     int (*setlease) (struct file *, int, struct file_lock **, void **);
1944     long (*fallocate) (struct file *file, int mode, loff_t offset,
1945         loff_t len);
1946     void (*show_fdinfo) (struct seq_file *m, struct file *f);
1947 #ifndef CONFIG_MMU
1948     unsigned (*mmap_capabilities) (struct file *);
1949 #endif
1950     ssize_t (*copy_file_range) (struct file *, loff_t, struct file *,
1951         loff_t, size_t, unsigned int);
1952     loff_t (*remap_file_range) (struct file *file_in, loff_t pos_in,
1953         struct file *file_out, loff_t pos_out,
1954         loff_t len, unsigned int remap_flags);
1955     int (*fadvise) (struct file *, loff_t, loff_t, int);
1956     int (*uring_cmd) (struct io_uring_cmd *iucmd, unsigned int issue_flags);
1957     int (*uring_cmd_iopoll) (struct io_uring_cmd *, struct io_comp_batch *,
1958         unsigned int poll_flags);
1959 } __randomize_layout;
1960

```

Βλέπουμε ότι όταν κουνάμε το ποντίκι μας, στέλνονται δεδομένα από αυτήν την συσκευή στο παραπάνω path.

```
ultron@debian: ~  
00000b0 0801 0001 0208 0800 0001 0108 0800 0200  
00000c0 0108 0800 0102 0108 0800 0104 0108 0802  
00000d0 0001 0208 0800 0001 0108 0801 0001 0208  
00000e0 0801 0001 0108 0800 0002 0008 0801 0001  
00000f0 0108 0800 0001 0008 0802 0002 0108 0800  
0000100 0001 0208 0800 0001 0108 0800 0001 0208  
0000110 0800 0001 0108 0800 0002 0108 1800 00ff  
0000120 fe18 1801 00ff ff18 1800 00fe ff18 1800  
0000130 00ff ff18 1800 00fe ff18 1800 00ff fe18  
0000140 1800 00ff ff18 1801 01ff fe18 0800 0200  
0000150 ff18 1800 00fd ff18 1801 00ff ff18 1801  
0000160 00fe ff18 1800 02ff 0008 1801 00ff fe18  
0000170 1800 00ff ff18 0800 0100 fe18 0800 0002  
0000180 0108 2800 ff00 0028 28ff fe00 0028 18ff  
0000190 00ff 0028 18ff 00fe ff18 1800 00ff ff18  
00001a0 1800 00fe ff18 0800 0100 ff18 1800 00fe  
00001b0 ff18 1800 00ff ff18 1800 00fe ff18 1800  
00001c0 00ff 0008 1801 00fe ff18 1800 00ff ff18  
00001d0 1800 00fe ff18 0800 0200 ff18 1800 00ff  
00001e0 fe18 0800 0100 ff18 1800 00ff 0008 1801  
00001f0 00fe ff18 1800 00ff 0008 1801 00ff fe18  
0000200 1800 02ff ff18 1801 00fe ff18 1800 00ff  
0000210 fd18 1801 02ff ff18 1800 00fe fe18 0801
```

Οργανώνοντας αυτά τα δεδομένα φαίνεται ότι κάθε νέα «είσοδος» αποτελείται από 3 bytes, ένα που (μάλλον) δηλώνει κατεύθυνση, ένα που δηλώνει μεταβολή στον άξονα x και ένα που δηλώνει μεταβολή στον άξονα y:



The screenshot shows a terminal window titled "ultron@debian: ~". The terminal displays a list of 3D coordinates (x, y, z) in a columnar format. To the right of the terminal, a dark floating control panel is visible, featuring icons for "Selection" (a dashed box), "Screen" (a monitor), "Window" (two overlapping windows), a camera icon, a video icon, a large circular button, and a mouse cursor icon. A close button (X) is in the top right corner of the panel.

```
ultron@debian: ~  
40, 1, -1)  
8, 1, 0)  
8, 1, 0)  
40, 0, -1)  
24, -1, 0)  
24, -1, 0)  
24, -1, 0)  
24, -2, 0)  
40, 0, -1)  
40, 0, -2)  
40, 0, -1)  
40, 0, -1)  
40, 0, -1)  
40, 0, -2)  
8, 2, 0)  
8, 1, 0)  
8, 1, 0)  
8, 1, 0)  
8, 2, 0)  
8, 0, 2)  
8, 0, 1)  
8, 0, 1)  
8, 0, 1)
```

➔ 3)

Φαίνεται ότι δεν έχουμε πρόσβαση σε αυτό το path. Θα πρέπει να φτιάξουμε ένα module και να το φορτώσουμε για να έχουμε πρόσβαση στα δεδομένα της συσκευής.

```
root@debian:/home/ultron/task5# sudo mknod /dev/mydevice c 42 0  
mknod: /dev/mydevice: File exists  
root@debian:/home/ultron/task5# nano test.c  
root@debian:/home/ultron/task5# gcc test.c  
root@debian:/home/ultron/task5# ls  
a.out  mouse.py  test.c  
root@debian:/home/ultron/task5# ./a.out  
open: No such device or address
```

➔ 4)

Αρχικά συμπληρώνουμε τα σημεία TODO στον κώδικα του αρχείου dev.c.

- Προσθήκη του μέλους cdev (που αντιπροσωπεύει την character συσκευή) στην δομή my\_device\_data, καθώς και του buffer για ανάγνωση/διάβασμα δεδομένων:

```
struct my_device_data {  
    struct cdev cdev; /* TODO Add the cdev member variable. */  
  
    char buffer[BUFFER_SIZE]; /* TODO Add a buffer with BUFFER_SIZE elements member variable. */  
  
    size_t size; /* "size" variable holds the size of the buffer. */  
    atomic_t access; /* "access" variable is used so that this device can only be accessed by a single process */  
};
```

- Παίρνουμε τον δείκτη για τα δεδομένα χρησιμοποιώντας το container\_of macro που είδαμε προηγουμένως

```
/* TODO inode->i_cdev contains our cdev struct, use container_of to obtain a pointer to my_device_data */  
data = container_of(inode->i_cdev, struct my_device_data, cdev);
```

- Επιστρέφουμε το κατάλληλο error code αν δεν έχουμε πρόσβαση

```
50      /* TODO Return immediately if access isn't 0, use atomic_cmpxchg */  
51      if (atomic_cmpxchg(&data->access, 0, 1) != 0)  
52          return -EBUSY;
```

- Μηδενίζουμε το access αφού ελευθερώνουμε την συσκευή

```
50      /* TODO Return immediately if access isn't 0, use atomic_cmpxchg */  
51      if (atomic_cmpxchg(&data->access, 0, 1) != 0)  
52          return -EBUSY;
```

- Στην συνάρτηση my\_cdev\_read αντιγράφουμε τα δεδομένα από τον buffer της συσκευής στον buffer του user

```
76      pr_info( "READ: %i\n", size, *offset);  
77  
78      /* TODO : Copy data from data->buffer to user buffer. Use copy_to_user */  
79      if (copy_to_user(user_buffer, data->buffer + *offset, to_read))  
80          return -EFAULT;  
81
```

- Ενώ στην συνάρτηση `my_cdev_write` αντιγράφουμε τα δεδομένα του χρήστη στον `buffer` της συσκευής

```

94     size = (offset + size > BUFFER_SIZE) ? (BUFFER_SIZE - offset) : size;
95     /* TODO Copy user_buffer to data->buffer. Use copy_from_user. Make
96        if (copy_from_user(data->buffer + *offset, user_buffer, size))
97            return -EFAULT;
98

```

- Προσθέτουμε τις συναρτήσεις που «γράψαμε» στην δομή `file_operations` που όπως είπαμε αποθηκεύει τις λειτουργίες/συναρτήσεις που καθορίζουν την συμπεριφορά της συσκευής.

```

107
108     static const struct file_operations my_fops = {
109         .owner = THIS_MODULE,
110         .open = my_cdev_open,      /* TODO Add open function. */
111         .release = my_cdev_release, /* TODO Add release function. */
112         .read = my_cdev_read,      /* TODO Add read function */
113         .write = my_cdev_write,    /* TODO Add write function */
114     };
115

```

- Ορίζουμε την περιοχή της συσκευής με την συνάρτηση `register_chrdev_region`

```

120
121     /* TODO register char device region for MY_MAJOR and NUM_MINORS starting at MY_MINOR
122        Use the register_chrdev_region function */
123     err = register_chrdev_region(MKDEV(MY_MAJOR, MY_MINOR), NUM_MINORS, MODULE_NAME);
124     if (err != 0) {
125         pr_info("Register character device failed.\n");
126         return err;
127     }
128

```

- Αρχικοποιούμε και προσθέτουμε την συσκευή στον πυρήνα εντός της `my_init`

```

136
137     /* TODO init and add cdev to kernel core. Use cdev_init and cdev_add */
138     cdev_init(&devs[i].cdev, &my_fops);
139     devs[i].cdev.owner = THIS_MODULE;
140     devs[i].cdev.ops = &my_fops;
141

```

- Διαγράφουμε από τον πυρήνα και κάνουμε unregister την συσκευή όταν το module μας εκφορτώνεται (my\_exit)

```
3 static void my_exit(void)
4 {
5     int i;
6
7     for (i = 0; i < NUM_MINORS; i++) {
8         /* TODO delete cdev from kernel core using cdev_del */
9         cdev_del(&devs[i].cdev);
10    }
11
12    /* TODO Unregister char device region, for MY_MAJOR and NUM_MINORS s
13    unregister_chrdev_region(MKDEV(MY_MAJOR, MY_MINOR), NUM_MINORS);
14 }
```

Στην συνέχεια έχουμε την εφαρμογή μας της οποίας σκοπός είναι να διαβάζει το μήνυμα που έχουμε αποθηκευμένο μέσα στην συσκευή αλλά και να το αλλάζει.

(Αρχείο my\_app.c)

```

C my_app.c
1  #include <stdio.h>
2  #include <fcntl.h>
3  #include <unistd.h>
4  #include <string.h>
5
6  #define DEVICE_PATH "/dev/mydevice"
7
8  int main() {
9      int fd = open(DEVICE_PATH, O_RDWR);
10     if (fd == -1) {
11         perror("Failed to open the device");
12         return -1;
13     }
14
15     // Reading from the kernel module
16     char read_buffer[256];
17     ssize_t bytes_read = read(fd, read_buffer, sizeof(read_buffer));
18     if (bytes_read == -1) {
19         perror("Failed to read from the device");
20         close(fd);
21         return -1;
22     }
23
24     printf("Read from the device: %s\n", read_buffer);
25
26     // Writing to the kernel module
27     const char* write_data = "New Message";
28     ssize_t bytes_written = write(fd, write_data, strlen(write_data));
29     if (bytes_written == -1) {
30         perror("Failed to write to the device");
31         close(fd);
32         return -1;
33     }
34
35     printf("Write to the device successful\n");
36
37     close(fd);
38     return 0;
39 }

```

Σε αυτό το αρχείο ανοίγουμε το αρχείο στο path της συσκευής, έπειτα διαβάζουμε και τυπώνουμε το περιεχόμενο και τέλος γράφουμε ένα νέο μήνυμα.

```
root@debian:~# gcc my_app.c
```

```
root@debian:~# ./a.out
```

```
Read from the device: hello
```

```
Write to the device successful
```

```
root@debian:~# █
```



