# VE482 — Introduction to Operating Systems

*Lab 9*

Manuel — UM-JI (Fall 2020)

**Goals of the lab**

- Understand the basics about memory
- Compare two page replacement algorithms
- Work with Minix 3 kernel source code

## 1 Introduction

You are pretty exhausted after a long day debugging your multi-threaded LemonDB project. All you want is some quiet time and a good night sleep. When you arrive you immediately notice that something is wrong: your grandpa looks very upset. You walk to him and quietly seat beside him in the sofa. After a few minutes of silence he speaks: "You know the last thing your mum did? She confiscated my phone!" Indeed your grandpa is always the nose on his phone discussing on Wechat with his friends and playing gambling games, while today he is seating doing nothing.

As you do not know how to react, you compassionately say that you know how it feels as in the past more than once did your mother take away all your devices pretending you were not focused enough on your studies. As you pronounce the word "studies" your grandpa has a sparkle in the eye: "You studied computers! and i can still access the family computer, even if you mum re-enabled the parental control software she had for you. So maybe you could figure out a simple way to allow me to play with my friends when they come here?" Then he adds on a sad and desperate tone, "I am not even allowed to go the park in case I play a money game with them".

Deeply moved by his story which is so similar to yours, you decide to help him. Knowing your mum, you are sure she uses some tool like `tripewire` to keep track of all the files on the computer, so you cannot write a simple program to roll dice as she would quickly discover it. Then you would also be in trouble... who knows what she could plot with Mr. Frown?

As you think of the best solution for your grandpa, he gives you an nice incentive: you if can solve his problem then he will give you 10% of his gain. However lost in your thoughts you do not react making him think this is not enough, so he continues, "20%... 25% but this is my last word!" That was an easy bargain, especially as you would have done it for free.

"Eureka!" You have the idea to trick mum! You will write a module to create a character device which simulates dice rolls. With everything you have learnt about Linux kernel and modules while working on dadfs it should not be too hard of a job.

## 2 A dice module

You remember that for what you want to implement, a character device should at least match the `open`, `close`, `read`, and `write` system calls. You will also have to decide on a major and a minor for your dice device. By definition the major must correspond to the driver associated to the device so it cannot "freely" choose it. However the minor is more flexible and you can pass it to the driver. You therefore decide to use it to create more than one type of dice device: each type would feature a different minor corresponding to a different type of dice. The displayed result of a roll would then depend on the minor.

That being set, you now want to specify more precisely how to implement your devices, but for that you need grandpa's help. What kind of game does he want to play? How many players there would be? Based on his answers it appears that you need to implement the following options:

- Change the number of dice by a `write` on the device;
- Display the result of a roll by a `read` on the device;
- Use the minor to specify the type of dice when creating the device;
- Depending on the type of dice display a different output;
- Use a module option to set set the number of sides for a generic type of dice;

Based on his answers you quickly get a sketch of what you will do, and show it to him to ensure you match all his expectations. You will have three main types of dice: (i) regular, (ii) backgammon, and (iii) generic featuring an arbitrary number of sides. For this latter dice the side number should be specified as a module option.

```
$ #regular dice
$ echo 2 > /dev/dice0
$ cat /dev/dice0
-------  -------
| o o |  | o o |
| o o |  |  o  |
| o o |  | o o |
'-----'  '-----'
```

```
$ #backgammon die
$ echo 1 > /dev/dice1
$ cat /dev/dice1
16
```

```
$ #generic dice, gen_sides=20
$ # gen_sides: module option
$ echo 3 > /dev/dice2
$ cat /dev/dice2
12 19 8
```

As grandpa is perfectly happy with your idea you decide to carry on with this idea.

## 3 Tasks

First you want to make sure you are fully familiar with the kernel API for creating character devices. So you ask yourself the following questions.

- What needs to be returned by `read` and `write` file operations for a character device?
- How are exactly those major and minor numbers working? You vaguely remember that you can display them using `ls -l /dev`.
- Knowing the major number and minor numbers of a device, how to add a character device to `/dev`?
- Where are the following terms located in linux source code?
  - `module_init`
  - `module_exit`
  - `printk`
  - `container_of`
  - `dev_t`
  - `MAJOR`
  - `MINOR`
  - `MKDEV`
  - `alloc_chrdev_region`
  - `module_param`
  - `cdev_init`
  - `cdev_add`
  - `cdev_del`
  - `THIS_MODULE`
- How to generate random numbers when working inside the Linux kernel? You think that a while back you read something about getting the current time.
- How to define and specify module options?

As you can answer all those questions it feels that you are almost ready to get started. Your grandpa is very impressed by everything you have just done and is really eager to play. Just as he thanks you for your good work, you both hear the noise of keys unlocking the door. It must be mum! Your grandpa rushes to the kitchen, pretending to prepare dinner while you switch terminal and run your last version on LemonDB. All safe for now! As soon as you will be alone again you will complete this dice device and make grandpa smile again...

A few useful references:

- `https://github.com/starpos/scull`
- `https://www.oreilly.com/library/view/linux-device-drivers/0596005903/ch03.html`
- `https://elixir.bootlin.com/linux/latest/source`