

VE482 — Introduction to Operating Systems

Lab 9

VE482 — Introduction to Operating Systems Lab 9

[Tasks](#)

[Test Result](#)

[Reference](#)

Tasks

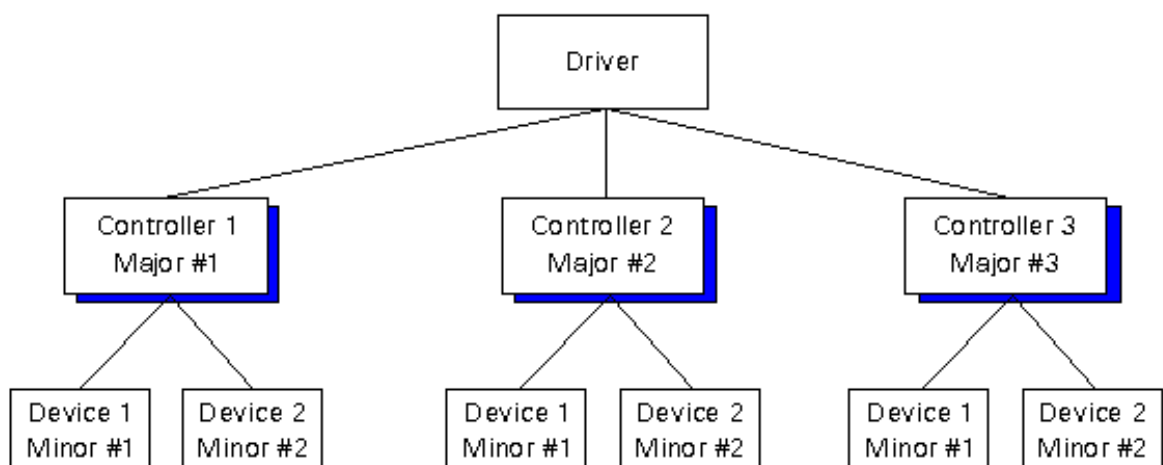
1. What needs to be returned by read and write file operations for a character device?

The value returned by read or write can be:

- the number of bytes transferred; if the returned value is less than the size parameter (the number of bytes requested), then it means that a partial transfer was made. Most of the time, the user-space app calls the system call (read or write) function until the required data number is transferred.
- 0 to mark the end of the file in the case of read ; if write returns the value 0 then it means that no byte has been written and that no error has occurred; In this case, the user-space application retries the write call.
- a negative value indicating an error code.

2. How are exactly those major and minor numbers working? You vaguely remember that you can display them using `ls -l /dev`.

Major and **minor** numbers are associated with the device special files in the `/dev` directory and are used by the operating system to determine the actual driver and device to be accessed by the user-level request for the special device file.



In `/dev`, we can see the major and minor numbers of different devices

```
bill@bill-virtual-machine:~$ ll /dev
total 4
drwxr-xr-x 20 root root 4220 12月 2 15:42 ./
drwxr-xr-x 20 root root 4096 11月 18 00:23 ../
crw-r--r-- 1 root root 10, 235 12月 2 15:37 autofs
drwxr-xr-x 2 root root 400 12月 2 15:42 block/
drwxr-xr-x 2 root root 80 12月 2 15:36 bsg/
crw----- 1 root root 10, 234 12月 2 15:36 btrfs-control
drwxr-xr-x 3 root root 60 12月 2 15:36 bus/
lrwxrwxrwx 1 root root 3 12月 2 15:37 cdrom -> sr0
lrwxrwxrwx 1 root root 3 12月 2 15:37 cdrw -> sr0
drwxr-xr-x 2 root root 3740 12月 2 15:36 char/
crw--w---- 1 root tty 5, 1 12月 2 15:37 console
lrwxrwxrwx 1 root root 11 12月 2 15:36 core -> /proc/kcore
drwxr-xr-x 4 root root 80 12月 2 15:36 cpu/
crw----- 1 root root 10, 123 12月 2 15:37 cpu_dma_latency
crw----- 1 root root 10, 203 12月 2 15:36 cuse
drwxr-xr-x 7 root root 140 12月 2 15:36 disk/
drwxr-xr-x 2 root root 60 12月 2 15:36 dma_heap/
crw-rw----+ 1 root audio 14, 9 12月 2 15:37 dmmidi
drwxr-xr-x 3 root root 100 12月 2 15:36 dri/
lrwxrwxrwx 1 root root 3 12月 2 15:37 dvd -> sr0
crw----- 1 root root 10, 126 12月 2 15:37 ecryptfs
crw-rw---- 1 root video 29, 0 12月 2 15:37 fb0
lrwxrwxrwx 1 root root 13 12月 2 15:36 fd -> /proc/self/fd/
crw-rw-rw- 1 root root 1, 7 12月 2 15:37 full
crw-rw-rw- 1 root root 10, 229 12月 2 15:37 fuse
crw----- 1 root root 241, 0 12月 2 15:37 hidraw0
```

3. Knowing the major number and minor numbers of a device, how to add a character device to `/dev`?

```
# (Create a new device directly)
mknod /dev/$device_name c $major_id $minor_id

# (add a character device to /dev)
cdev_add(struct cdev *dev, dev_t num, unsigned int count);
```

4. **Where are the following terms located in Linux source code?**

term	location
module_init	include/linux/module.h
module_exit	include/linux/module.h
printk	include/linux/printk.h
container_of	include/linux/kernel.h
dev_t	include/linux/types.h
MAJOR	include/linux/kdev_t.h
MINOR	include/linux/kdev_t.h
MKDEV	include/linux/kdev_t.h
alloc_chrdev_region	include/linux/fs.h
module_param	include/linux/moduleparam.h
cdev_init	include/linux/cdev.h
cdev_add	include/linux/cdev.h
cdev_del	include/linux/cdev.h
THIS_MODULE	include/linux/export.h

5. **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.**

The primary kernel interface is

```
void get_random_bytes(void *buf, int nbytes)
```

This interface will return the requested number of random bytes, and place it in the requested buffer. This is equivalent to a read from `/dev/urandom`.

For less critical applications, there are the functions:

```
u32 get_random_u32()
```

```
u64 get_random_u64()
```

```
unsigned int get_random_int()
```

```
unsigned long get_random_long()
```

These are produced by a cryptographic RNG seeded from `get_random_bytes`, and so do not deplete the entropy pool as much. These are recommended for most in-kernel operations *if the result is going to be stored in the kernel*.

6. How to define and specify module options?

We can use the function `module_param(name, type, perm)` to pass a parameter to the module. In `linux/moduleparam.h`, the usage of it is described as follows:

- `name`: The variable to alter, and exposed parameter name.
- `type`: The type of the parameter. Standard types include `byte`, `hexint`, `short`, `ushort`, `int`, `uint`, `long`, and `ulong`.
- `perm`: The visibility in sysfs. It is 0 if the variable is not to appear in sysfs, or 0444 for world-readable, 0644 for root-writable, etc.

Example:

```
// In dice.c
#include <linux/moduleparam.h>

int gen_sides = 20;
module_param(gen_sides, int, 0644);
```

```
# During installation
insmod dice.ko gen_sides = 8
```

Test Result

```

bill@kali:~/dice$ sudo insmod ./dicedevice.ko
bill@kali:~/dice$ cat /proc/modules | grep "dicedevice"
dicedevice 16384 0 - Live 0x0000000000000000 (OE)
bill@kali:~/dice$ cat /proc/devices | grep "Dice"
237 Dice
bill@kali:~/dice$ sudo mknod /dev/dice0 c 237 0
bill@kali:~/dice$ sudo mknod /dev/dice1 c 237 1
bill@kali:~/dice$ sudo mknod /dev/dice2 c 237 2
bill@kali:~/dice$ sudo chmod 777 /dev/dice0
bill@kali:~/dice$ sudo chmod 777 /dev/dice1
bill@kali:~/dice$ sudo chmod 777 /dev/dice2
bill@kali:~/dice$ cat /dev/dice1
8 8

bill@kali:~/dice$ echo 1 > /dev/dice1
bill@kali:~/dice$ cat /dev/dice0
-----
| o | | o |
|   | |   |
| o o | | o o |
-----

bill@kali:~/dice$ cat /dev/dice0
-----
|   | |   |
| o o | | o |
|   | |   |
-----

bill@kali:~/dice$ sudo rm -f /dev/dice0
bill@kali:~/dice$ sudo rm -f /dev/dice1
bill@kali:~/dice$ sudo rm -f /dev/dice2
bill@kali:~/dice$ sudo rmmmod dicedevice.ko
bill@kali:~/dice$

```

Reference

- [1] return value of read and write https://linux-kernel-labs.github.io/refs/heads/master/labs/device_drivers.html
- [2] major and minor numbers http://osr600doc.sco.com/en/HDK_concepts/ddT_majmin.html
- [3] Linux source code <https://elixir.bootlin.com/linux/v5.15.6/source>