

# Assignment 5

## 1. insmod 之后 /proc/devices 中可以看到增加的驱动

```
./build_image.sh x _e1000/rootfs
cat /proc/devices
Character devices:
 1 mem
 4 /dev/vc/0
 4 tty
 4 ttyS
 5 /dev/tty
 5 /dev/console
 5 /dev/ptmx
 7 vcs
10 misc
13 input
21 sg
116 alsa
128 ptm
136 pts
180 usb
189 usb_device
202 cpu/msr
203 cpu/cpuid
226 drm
248 rust_chrdev
249 hidraw
250 usbmon
251 bsg
252 ptp
253 pps
254 rtc
Block devices:
```

## 2. 对字符设备进行读写

```
./build_image.sh x _e1000/rootfs
[ 14.051265] modprobe (67) used greatest stack depth: 14272 bytes left
[ 14.060153] cfg80211: Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
[ 14.061605] platform regulatory.0: Direct firmware load for regulatory.db failed with error -2
[ 14.062123] cfg80211: failed to load regulatory.db
[ 14.063235] ALSA device list:
[ 14.063522]   No soundcards found.
[ 14.104327] Freeing unused kernel image (initmem) memory: 1328K
[ 14.105013] Write protecting the kernel read-only data: 24576k
[ 14.108546] Freeing unused kernel image (text/rodata gap) memory: 2032K
[ 14.109660] Freeing unused kernel image (rodata/data gap) memory: 840K
[ 14.226760] x86/mm: Checked W+X mappings: passed, no W+X pages found.
[ 14.227086] Run sbinit/init as init process
[ 14.253825] mount (72) used greatest stack depth: 13920 bytes left

Please press Enter to activate this console.
~ # insmod ./rust_chrdev.ko
[ 32.455349] rust_chrdev: Rust character device sample (init)
[ 32.456732] insmod (80) used greatest stack depth: 13816 bytes left
~ # echo "hello" > /dev/cicv
~ # cat /dev/cicv
[ 44.107886] rust_chrdev: got write request 6
hello
[ 52.574497] rust_chrdev: got read request 6
~ # echo "world" > /dev/cicv
[ 52.574976] rust_chrdev: got read request 0
~ # cat /dev/cicv
[ 115.822893] rust_chrdev: got write request 6
world
[ 123.857136] rust_chrdev: got read request 6
~ # _
```

## 问答题

Q: 作业5中的字符设备/dev/cicv是怎么创建的？它的设备号是多少？它是如何与我们写的字符设备驱动关联上的？

- 是通过 `build_image.sh` 中的 `echo "mknod /dev/cicv c 248 0" >> etc/init.d/rcS` 指定，在系统启动时创建名为 `/dev/cicv`、主设备号为 `248`、次设备号为 `0` 的字符设备。

- 通过载入内核模块 `rust_chrdev.rs`，在初始化模块时调用了 `chrdev::Registration::register`，其中又调用了 `bindings::alloc_chrdev_region`，将设备号 248 关联到 `rust_chrdev` 驱动
- 正好分配到 248 是因为字符设备的可动态分配的设备号从 254 开始往下按序分配，前面 249~254 都已经被占用了。