**📙 二、ioctl 总结**

* 1. **什么是 ioctl？**

ioctl 是 “input/output control” 的缩写，是用户空间对设备进行控制操作（非读写）的接口，比如控制 LED 开关、设置串口参数等。

**1.2 函数原型**

long (\*unlocked\_ioctl)(struct file \*filp, unsigned int cmd, unsigned long arg);

* cmd：命令号，通常由 \_IO()、\_IOW() 等宏生成；
* arg：参数，可以是值、指针（如结构体、数组）等。

**1.3 ioctl 宏的使用**

#define CMD\_LED\_ON \_IOW('L', 0, int)

#define CMD\_LED\_OFF \_IOW('L', 1, int)

#define CMD\_LED\_READ \_IOR('L', 2, int)

#define CMD\_LED\_RDWR \_IOWR('L', 3, struct led\_cfg)

| **宏** | **含义** |
| --- | --- |
| \_IO | 无参数 |
| \_IOR | 从内核读数据 |
| \_IOW | 向内核写数据 |
| \_IOWR | 双向传输 |

**1.4 ioctl 参数传递**

| **类型** | **用法** |
| --- | --- |
| 值传递（int、unsigned long） | 直接使用 arg |
| 指针传递（结构体、数组） | 用 copy\_from\_user() / copy\_to\_user() |

**1.5 用户态调用方式**

int fd = open("/dev/myled", O\_RDWR);

int arg = 1;

ioctl(fd, CMD\_LED\_ON, arg);

或

struct led\_cfg cfg = {0, 1}; // 开第 0 个灯

ioctl(fd, CMD\_LED\_RDWR, &cfg);

**2.6 示例：驱动中实现**

long myled\_unlocked\_ioctl(struct file \*filp, unsigned int cmd, unsigned long arg) {

void \_\_user \*argp = (void \_\_user \*)arg;

struct led\_cfg cfg;

switch (cmd) {

case CMD\_LED\_ON:

gpio\_set\_value(leds[arg - 7].gpio, 0);

break;

case CMD\_LED\_RDWR:

copy\_from\_user(&cfg, argp, sizeof(cfg));

gpio\_set\_value(leds[cfg.index].gpio, cfg.state);

break;

}

return 0;

}

**2.7 面试常考点（ioctl）**

| **问题** | **解答** |
| --- | --- |
| ioctl 和 read/write 区别？ | ioctl 用于控制类操作，非数据流式 |
| ioctl 中如何区分指令和数据？ | 通过 cmd 中的 \_IOC\_TYPE、\_IOC\_NR |
| 为什么要用 copy\_from\_user？ | 内核不能直接访问用户空间指针 |
| struct file 中的 unlocked\_ioctl 和 ioctl 有什么区别？ | ioctl 是旧接口，需持锁；新内核使用 unlocked\_ioctl |

**📌 三、知识导图（推荐思维结构）**

Linux 驱动开发

├── 字符设备

│ └── 普通字符设备

│ ├── cdev\_init

│ ├── register\_chrdev\_region

│ └── device\_create

├── 混杂设备

│ ├── miscdevice 结构体

│ ├── misc\_register/misc\_deregister

│ └── 自动生成 /dev 设备

└── ioctl 控制接口

├── ioctl 宏

├── unlocked\_ioctl 函数

├── 参数传值 / 指针

├── 用户层调用 ioctl()

└── copy\_from\_user / copy\_to\_user

**🎯 四、项目建议（实战）**

* **实验建议：**
  + 用 ioctl 控制 4 个 LED 的开关
  + 实现数组传递、结构体传递
  + 添加错误处理，打印调试信息
* **提升建议：**
  + 将 ioctl 改为支持多种命令分类（如 MAGIC 字段区分）
  + 学习 compat\_ioctl 兼容 32/64 位 ioctl 调用

**🧠 五、总结一句话记忆法：**

混杂设备让你不操心主设备号，ioctl 则是设备控制的万能遥控器。