+------------------------------------------------+

| static dev\_t dev\_num = 0; | ← 你定义的设备号

+------------------------------------------------+

|

↓

+------------------------------------------------+

| struct cdev led\_cdev; | ← 你定义的设备对象

+------------------------------------------------+

|

↓

+------------------------------------------------+

dev\_num=MKDEV(240,0);

rt = register\_chrdev\_region(dev\_num,1,"myled");

//注册设备号（向内核申请访问）

+-----------------------------------------------------+

|

↓

+-----------------------------------------------------+

| cdev\_init(&led\_cdev, &fops); | ← 设置它的操作方法

+-----------------------------------------------------+

|

↓

+--------------------------------------------------------+

| cdev\_add(&led\_cdev, dev\_num, 1); | ← 注册到内核，绑定设备号

+--------------------------------------------------------+

|

┌────────────────────────┐

│ class\_create │

创建一个逻辑类

└─────────────┬─────────────┘

↓

┌────────────────────────┐

│ device\_create │

创建 sysfs 设备，触发 udev

└──────────┬─────────────┘

↓

用户空间 open("/dev/myled") →

内核根据 dev\_num 查到 led\_cdev →

调用你写的 open(), read(), write() 等函数

## 1.register\_chrdev\_region(dev\_num, 1, "myled")

### 功能：

### 向内核 注册一个固定的字符设备号。告诉内核：“我要使用主设备号 240，次设备号从 0 开始，一共注册 1 个设备”。它不会创建设备节点 /dev/xxx，只是登记信息。

### "myled" 在这里的作用：用于 cat /proc/devices 命令时展示：

### $ cat /proc/devices

### Character devices:

### ...

### 240 myled

### 所以这里只是 起个名字用于设备号描述而已，对设备节点名字没影响。

## 2. device\_create(led\_class, NULL, dev\_num, NULL, "myled")

## 功能：

## 向 udev 或 devtmpfs 通知：

## “我有一个设备，要创建设备节点了！”会在 /dev/ 目录下生成一个设备节点（如果有 devtmpfs 或 udev 支持）。生成的设备节点是：/dev/myled

## 2."myled" 在这里的作用：

## 是设备节点的名字，即文件名 myled。如果你写成 "myled0"，就会生成 /dev/myled0。

# Linux 驱动开发：设备注册核心概念笔记

## 一、/proc 是什么？

/proc 是内核提供的一个 **虚拟文件系统**，用于展示内核运行状态，不占用实际磁盘空间。

常用于查看系统信息，比如 CPU、内存、进程、驱动模块等。

**常用内容举例：**

/proc/cpuinfo → CPU 信息

/proc/meminfo → 内存信息

/proc/modules → 已加载的内核模块

/proc/devices → 已注册的设备号（字符设备/块设备）

## 二、/proc/devices 的作用

显示当前已注册的设备号（主设备号）及其对应的驱动名。主要来源是你在驱动中使用：

register\_chrdev\_region(dev\_num, 1, "myled");

"myled" 会显示在 /proc/devices：

Character devices:

240 myled

## 三、/dev 是什么？

/dev 是 Linux 系统中的 **设备节点目录**，包含设备文件（特殊文件）。

应用程序通过 /dev/xxx 访问对应的设备驱动。

每个 /dev/xxx 节点都与一个设备号（主设备号 + 次设备号）绑定。

**示例：**

/dev/sda → 硬盘设备

/dev/tty0 → 终端设备

/dev/myled → 用户自己驱动创建的 LED 设备节点

## 四、udev 是什么？

udev 是运行在用户空间的 **设备管理守护进程**。

会监听内核的设备创建事件，并在 /dev 中自动生成设备节点。

也支持规则匹配、自动挂载、执行脚本等操作。

没有 udev，设备节点可能不会自动生成，需要手动创建（用 mknod）。

## 五、devtmpfs 是什么？

devtmpfs 是内核提供的设备管理虚拟文件系统。

它会在内核空间自动创建设备节点，**不依赖 udev**。

常见于嵌入式系统中（如 busybox）。

**通常使用方式：**

devtmpfs 负责基础节点创建

udev 进一步管理节点属性和触发行为

## 六、对比总结表

| **项目** | **类型** | **功能** | **是否虚拟** | **示例/说明** |
| --- | --- | --- | --- | --- |
| /proc | 虚拟文件系统 | 内核运行信息 | ✅ | /proc/devices |
| /dev | 真实目录 | 存放设备节点 | ❌（依赖 devtmpfs） | /dev/myled |
| udev | 用户空间进程 | 监听设备事件并创建 /dev 节点 | ❌ | 自动创建设备文件 |
| devtmpfs | 内核虚拟文件系统 | 自动挂载 /dev、创建节点 | ✅ | 内核层自动生成 /dev 节点 |

## 七、补充说明：名字的区别

### register\_chrdev\_region(dev\_num, 1, "myled");

作用：注册设备号（告诉内核我要用 240 号）

"myled" → 会出现在 /proc/devices 中作为名称

### device\_create(led\_class, NULL, dev\_num, NULL, "myled");

作用：创建设备节点 /dev/myled

"myled" → 是设备文件名

## ✅ 总结一句话：

register\_chrdev\_region(...) 是 **在内核中注册设备号**，名字用于内核显示；

device\_create(...) 是 **创建设备节点文件** /dev/xxx，名字决定文件名；

两个 "myled" **互不干扰，可以不一样**，建议统一更清晰。