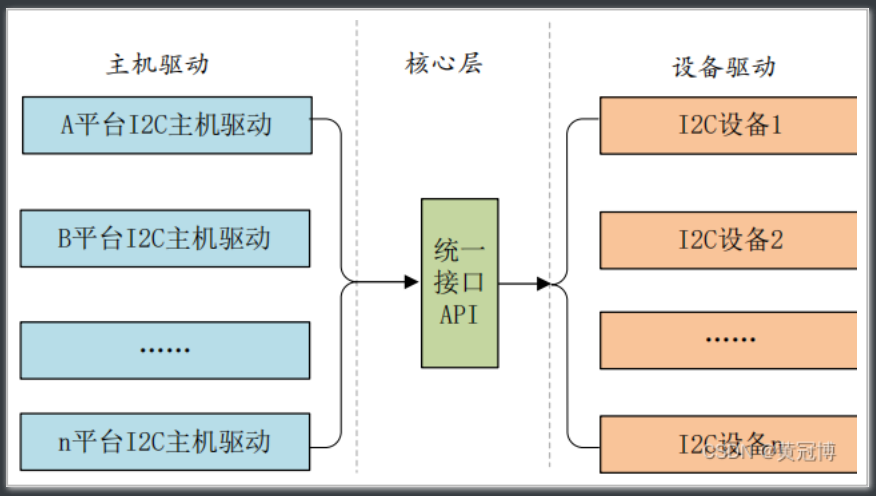
驱动开发 platfrom 总线驱动的三种方式 plat comm -CSDN 博客

驱动的分隔与分离：

对于 Linux 这样一个成熟、庞大、复杂的操作系统，代码的重用性非常重要，在驱动程序，因为驱动程序占用了 Linux 内核代码量的大头，如果不对驱动程序加以管理，任由重复的代码肆意增加，那么用不了多久 Linux 内核的文件数量就庞大到无法接受的地步。

例如：现在有三个 SOC A、B 和 C 上都有 MPU6050 这个 I2C 接口的六轴传感器，按照我们写裸机 I2C 驱动的时候的思路，每个平台都有一个 MPU6050 的驱动，那么设备端的驱动将会重复的编写好几次。显然在 Linux 驱动程序中这种写法是不推荐的，最好的做法就是每个 SOC 的 I2C 控制器都提供一个统一的接口 (也叫做主机驱动)，每个设备的话也只提供一个驱动程序 (设备驱动)，每个设备通过统一的 I2C 接口驱动来访问，这样就可以大大简化驱动文件。



当我们向系统注册一个驱动的时候，总线就会在右侧的设备中查找，看看有没有与之匹配

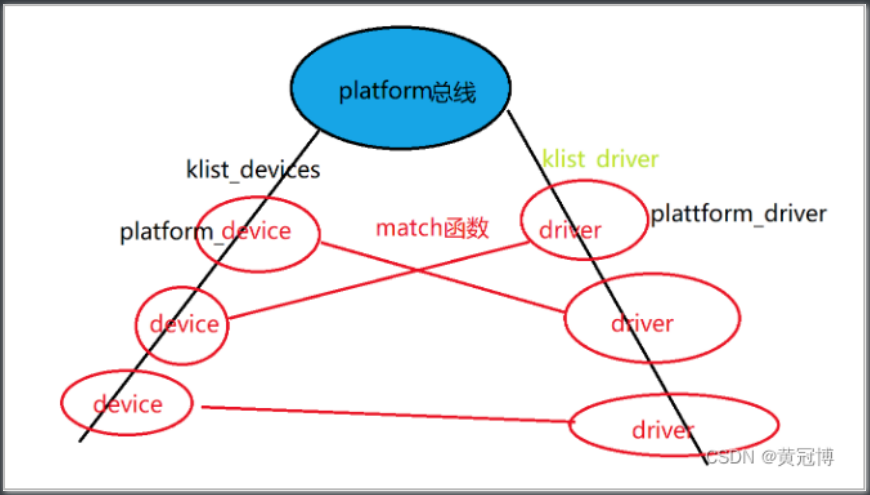
的设备，如果有的话就将两者联系起来。同样的，当向系统中注册一个设备的时候，总线就会

在左侧的驱动中查找看有没有与之匹配的设备，有的话也联系起来。

在 linux 内核中的驱动程序都采用总线、驱动和设备这样的模式。

platform 驱动就是这一思想下的产物。

总线驱动模型：



platform 总线遵从总线模型，platform 是 linux 内阁抽象出来的软件代码，没有真实的总线和它对应（不存在）

platfor 总线去驱动的思想：是将设备信息和驱动进行分离。platform_device 和 platform_driver 通过总线进行匹配，匹配成功后会执行驱动中的 probe 函数，在 probe 函数中可以获取到 device 中的硬件设备信息。

以下是 platfrom 的三种匹配方式：

一：设备名

pdrv:

```
#include<linux/init.h>
#include<linux/module.h>
#include<linux/platform_device.h>
#include<linux/mod_devicetable.h>
struct resource *res;
int irqno;
int pdrv_probe(struct platform_device *pdev)
{
    res=platform_get_resource(pdev,IORESOURCE_MEM,0);
    if(res==NULL)
    {
        return ENODATA;
    }
```

```
    irqno=platform_get_irq(pdev,0);
    if(irqno<0)
    {
        return ENODATA;
    }
    printk("addr:%#llx,irqno:%d\n",res->start,irqno);
    return 0;
}
int pdrv_remove(struct platform_device *pdev)
{
    printk("%s:%d\n",__func__,__LINE__);
    return 0;
}


struct platform_driver pdrv={
    .probe=pdrv_probe,
    .remove=pdrv_remove,
    .driver={
        .name="aaaaa",
    },
};
module_platform_driver(pdrv);
MODULE_LICENSE("GPL");
```

pdev:

```
#include<linux/init.h>
#include<linux/module.h>
#include<linux/platform_device.h>

struct resource res[]={
    [0]={
        .start=0x12345678,
        .end=0x12345678+49,
        .flags=IORESOURCE_MEM,
    },
    [1]={
        .start=71,
        .end=71,
        .flags=IORESOURCE_IRQ,
    },
};
void pdev_release(struct device *dev)
{
    printk("%s:%d\n",__func__,__LINE__);
}
struct platform_device pdev=
{
    .name="aaaaa",
    .id=PLATFORM_DEVID_AUTO,
    .dev={
        .release=pdev_release,
    },
    .resource=res,
    .num_resources=ARRAY_SIZE(res),
};


static int __init demo_init(void)
{
```

```
    platform_device_register(&pdev);
    return 0;
}

static void __exit demo_exit(void)
{
    platform_device_unregister(&pdev);
}

module_init(demo_init);
module_exit(demo_exit);
MODULE_LICENSE("GPL");
```

```
ubuntu@ubuntu:platform$ sudo dmesg -C
ubuntu@ubuntu:platform$ sudo insmod pdrv.ko
ubuntu@ubuntu:platform$ sudo insmod pdev.ko
ubuntu@ubuntu:platform$ dmesg
[ 1282.370310] pdrv: loading out-of-tree module taints kernel.
[ 1282.370348] pdrv: module verification failed: signature and/or requ
ired key missing - tainting kernel
[ 1288.571056] addr:0x12345678,irqno:71
ubuntu@ubuntu:platform$
```

CSDN @黄冠博

二：设备名列表

pdev:

```
#include<linux/init.h>
#include<linux/module.h>
#include<linux/platform_device.h>


struct resource res[]={
    [0]={
        .start=0x12345678,
        .end=0x12345678+49,
        .flags=IORESOURCE_MEM,
    },
    [1]={
        .start=71,
        .end=71,
        .flags=IORESOURCE_IRQ,
    },
};
void pdev_release(struct device *dev)
{
    printk("%s:%d\n",__func__,__LINE__);
}
struct platform_device pdev=
{
    .name="hello1",
    .id=PLATFORM_DEVID_AUTO,
    .dev={
        .release=pdev_release,
    },
    .resource=res,
    .num_resources=ARRAY_SIZE(res),
};
```

```
static int __init demo_init(void)
{
    platform_device_register(&pdev);
    return 0;
}

static void __exit demo_exit(void)
{
    platform_device_unregister(&pdev);
}

module_init(demo_init);
module_exit(demo_exit);
MODULE_LICENSE("GPL");
```

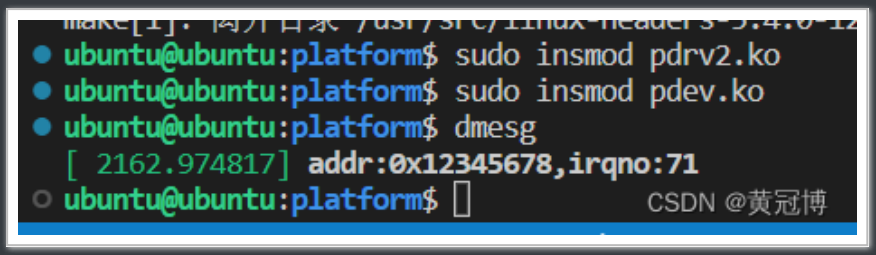
pdrv2:

```
#include<linux/init.h>
#include<linux/module.h>
#include<linux/platform_device.h>
#include<linux/mod_devicetable.h>
struct resource *res;
int irqno;
int pdrv_probe(struct platform_device *pdev)
{
    res=platform_get_resource(pdev,IORESOURCE_MEM,0);
    if(res==NULL)
    {
        return ENODATA;
    }
    irqno=platform_get_irq(pdev,0);
    if(irqno<0)
    {
        return ENODATA;
    }
    printk("addr:%#llx,irqno:%d\n",res->start,irqno);
    return 0;
}
int pdrv_remove(struct platform_device *pdev)
{
    printk("%s:%d\n",__func__,__LINE__);
    return 0;
}

struct  platform_device_id idtable[]={
    {"hello1",0},
    {"hello2",1},
    {"hello3",2},
    {}
};

struct platform_driver pdrv={
    .probe=pdrv_probe,
    .remove=pdrv_remove,
    .driver={
        .name="aaaaa",
    },
    .id_table=idtable,
};
MODULE_DEVICE_TABLE(platform,idtable);
```

```
module_platform_driver(pdrv);
MODULE_LICENSE("GPL");
```



三：设备树

添加设备树节点：

pdrv3:

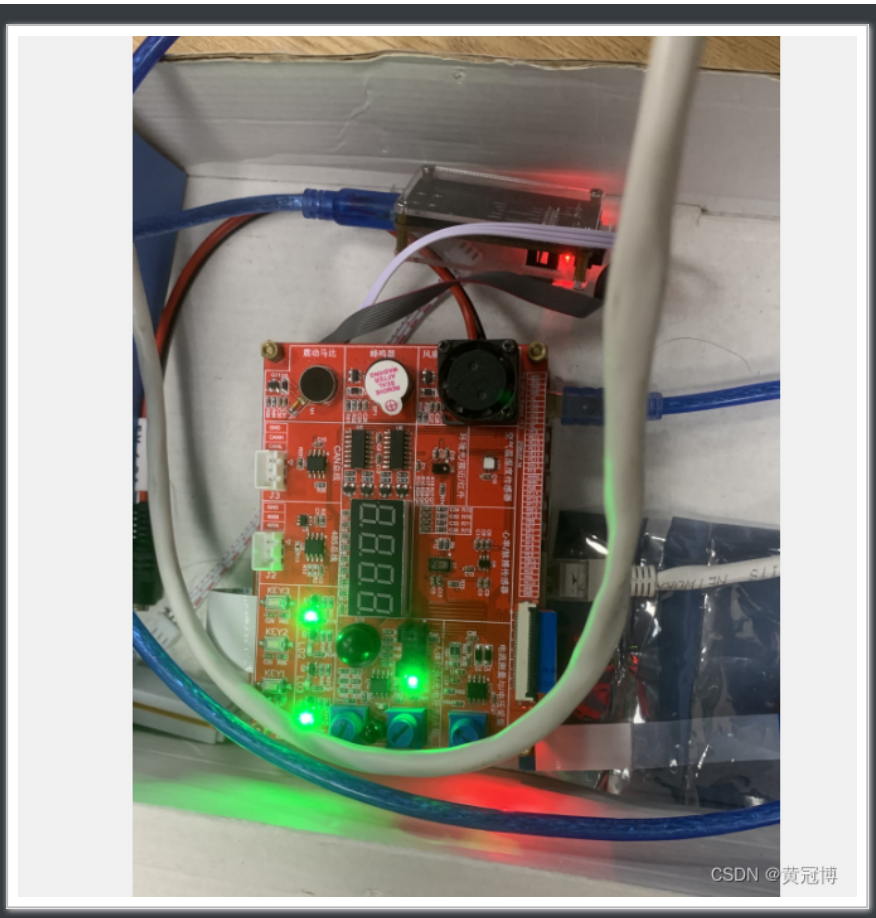
```
#include<linux/init.h>
#include<linux/module.h>
#include<linux/platform_device.h>
#include<linux/mod_devicetable.h>
#include<linux/of.h>
#include<linux/of_gpio.h>
struct resource *res;
int irqno;
struct gpio_desc *gpiono;
int pdrv_probe(struct platform_device *pdev)
{
    res=platform_get_resource(pdev,IORESOURCE_MEM,0);
    if(res==NULL)
    {
        return ENODATA;
    }
    irqno=platform_get_irq(pdev,0);
    if(irqno<0)
    {
        return ENODATA;
    }
    printk("addr:%#x,irqno:%d\n",res->start,irqno);
```

```c
    gpiono=gpiod_get_from_of_node(pdev->dev.of_node,"myled1",0,GPIOD_OUT_HIGH,0);
    if(IS_ERR(gpiono))
    {
        printk("获取gpio编号失败\n");
        return PTR_ERR(gpiono);
    }
    gpiod_set_value(gpiono,1);
    return 0;
}
int pdrv_remove(struct platform_device *pdev)
{
    gpiod_set_value(gpiono,0);
    gpiod_put(gpiono);
    printk("%s:%d\n",__func__,__LINE__);
    return 0;
}
struct of_device_id oftable[]={
    {.compatible="hqyj,platform",},
    {}
};
struct platform_driver pdrv={
    .probe=pdrv_probe,
    .remove=pdrv_remove,
    .driver={
        .name="aaaaa",
        .of_match_table=oftable,
    },

};
module_platform_driver(pdrv);
MODULE_LICENSE("GPL");
```

```
[root@fsmp1a /]# insmod pdrv3.ko
[  233.817600] Module has invalid ELF structur
[  233.825213] Module has invalid ELF structur
insmod: can't insert 'pdrv3.ko': invalid modul
[root@fsmp1a /]# insmod pdrv3.ko
[  464.986560] addr:0x12345678,irqno:71
[root@fsmp1a /]#
```


Serial: COM6, 115200 52, 18 52 Rows, 66



全文完