

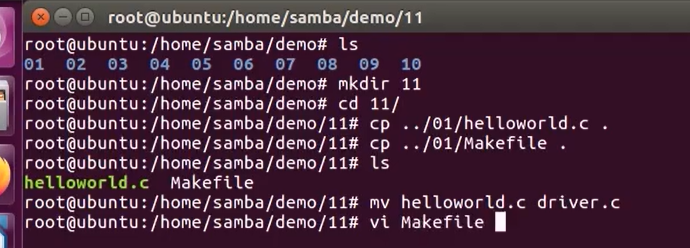
思路与上一节定义相似，只不过上一节定义的是platform\_device结构体变量。本节课是定义platform\_driver。

由于匹配成功之后就会去执行probe函数，所以重点是在编写probe函数。

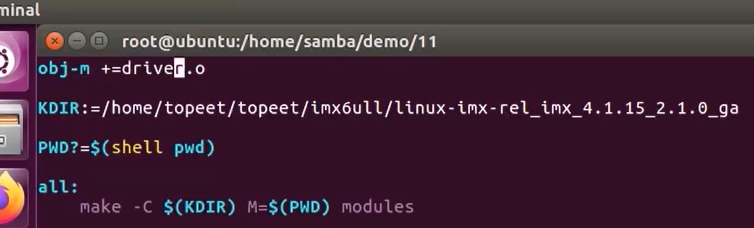
——————————————————————————————————

实验：

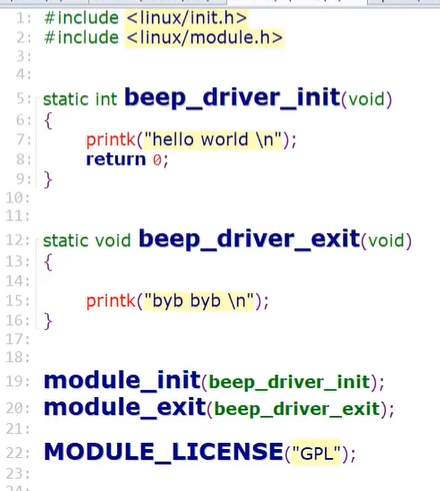
在01的基础上改写：



修改Makefile:

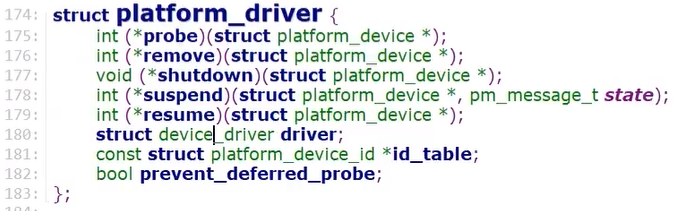


基础结构：



/\*

Platform\_driver结构体同样位于 include/linux/platform\_device.h文件里面：



Probe函数：当driver和device匹配成功之后，就会去执行probe函数。

我们可以在这里面实现注册字符/杂项设备，点灯等操作。

Remove函数:当设备或驱动任意一个或全部remove的时候就会执行这个函 数。

Shutdown函数：当设备收到Shutdown命令时就会执行该函数。

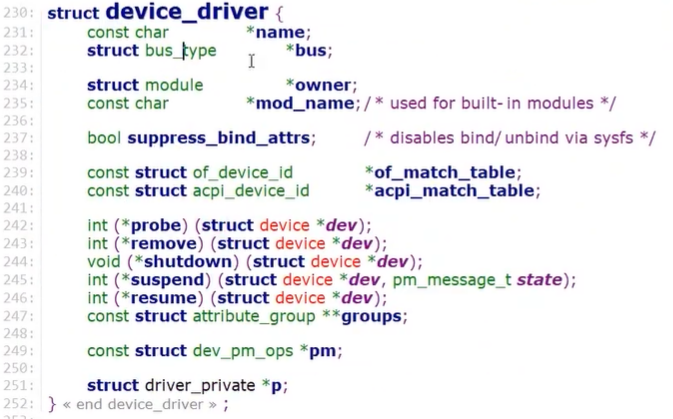
Suspend函数：休眠，当设备收到休眠的命令时就会执行这个函数。

Resume函数：唤醒，当设备收到唤醒的命令时就会执行这个函数。

Device\_driver：这里只用到了两个：

.name成员：匹配时的名字

.owner成员：THIS\_MODULE



Id\_table：

位于 include/linux/mod\_devicetable.h文件。

里面也有一个name成员：当device\_driver和id\_table里面都使用了.name成员时，优先匹配id\_table的.name成员。



\*/

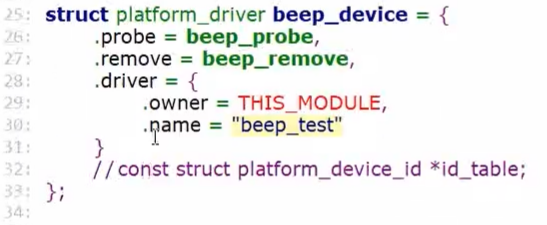
包含头文件：

IMG_256

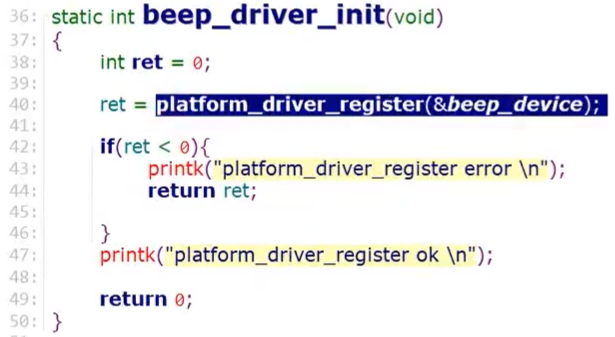
实现probe和remove函数：



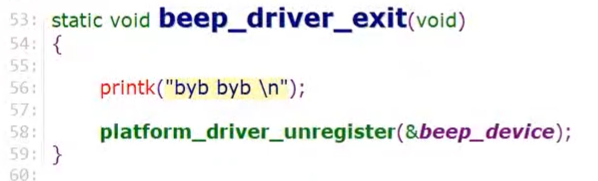
实现platform\_driver:



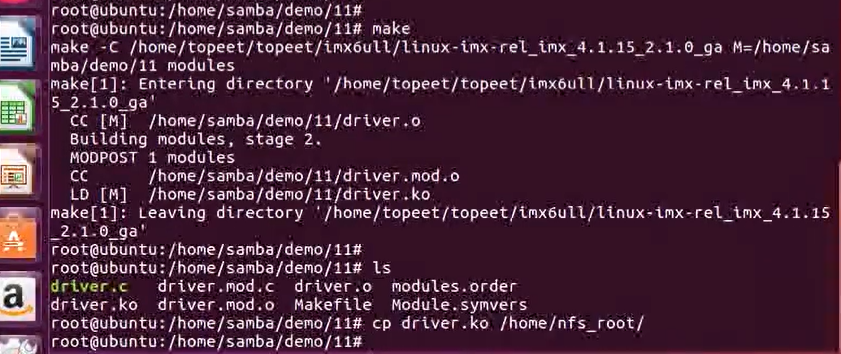
注册到系统内核：



实现注销：

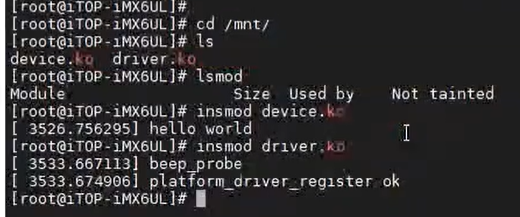


写完之后进行编译拷贝：



现在开发板就有两个ko文件：一个device.ko，另一个是driver.ko：

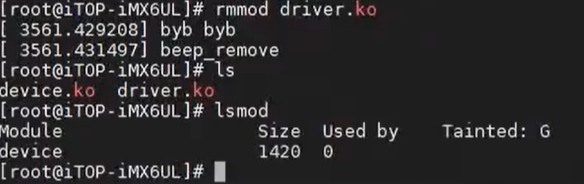
注册：



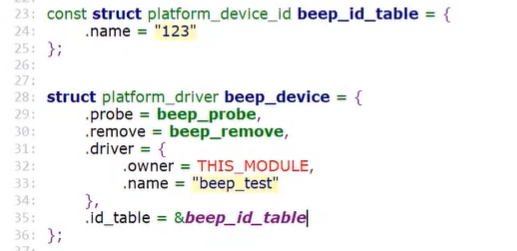
可见它打印了beep\_probe，此时在该probe函数里面我们就可以去实现我们想要的操作了。

——————————————————————————————————

现在将driver模块删除，进行实验id\_table：

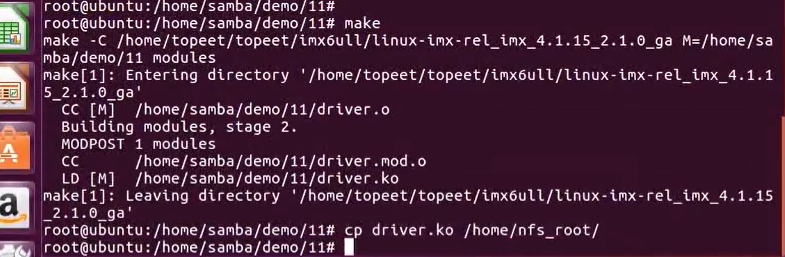


实现platform\_device\_id结构体：



当匹配的时候由于id\_table的优先级高于driver，并且其匹配名称为123，所以会匹配失败。

编译拷贝：



重新加载两个驱动：



可以看到，它没有实现peobe函数，所以并没有匹配成功。