在linux-2.6以后的驱动模型中引入了总线(bus)、设备(device)、设备驱动(drivers)三个实体,

总线将设备和驱动绑定，在系统每注册一个设备的时候，会寻找与之匹配的驱动。相反，在系统每注册一个驱动的时候，寻找与之匹配的设备，匹配是由总线来完成的。一个现实的Linux设备和驱动通常都需要挂接在一种总线上，对于本身依附于PCI、USB、I2C、SPI、UART等总线的设备而言，这自然不是问题，但是在嵌入式系统里面，SoC 系统中集成的独立的外设控制器、挂接在SoC内存空间的外设等却不依附于此类总线（外设控制器和直接挂接在内存空间的外设不依附于此类总线）。基于这一背景，Linux 发明了一种虚拟的总线，称为platform总线。SOC系统中集成的独立外设控制器单元(I2C，LCD，SPI，RTC等)、直接挂接在内存空间的外设等都被当作平台设备来处理，而它们本身是字符型设备。从Linux2.6内核起，引入一套新的驱动管理和注册机制：platform\_device和 platform\_driver 。Linux 中大部分的设备驱动，都可以使用这套机制，设备用 platform\_device表示;驱动用platform\_driver 进行注册。

CPU

i2c外设

usb外设

i2c控制器

usb控制器

uart外设

uart控制器

gpio外设

嵌入式系统SoC中基本都是挂接外设控制器，直接挂接在bus（内存）上的不多

i2c bus

platform bus

platform bus

内存空间

usb bus

i2c外设

usb外设

例子1

led设备驱动

先看Makefile：

# LED Core

**obj-$(CONFIG\_NEW\_LEDS) +**= led-core.o

**obj-$(CONFIG\_LEDS\_CLASS) +**= led-class.o

**obj-$(CONFIG\_LEDS\_CLASS\_FLASH) +**= led-class-flash.o

**obj-$(CONFIG\_LEDS\_TRIGGERS) +**= led-triggers.o

# LED Platform Drivers

**obj-$(CONFIG\_LEDS\_88PM860X) +**= leds-88pm860x.o

**obj-$(CONFIG\_LEDS\_AAT1290) +**= leds-aat1290.o

**obj-$(CONFIG\_LEDS\_APU) +**= leds-apu.o

**obj-$(CONFIG\_LEDS\_AS3645A) +**= leds-as3645a.o

**obj-$(CONFIG\_LEDS\_AN30259A) +**= leds-an30259a.o

**obj-$(CONFIG\_LEDS\_BCM6328) +**= leds-bcm6328.o

**obj-$(CONFIG\_LEDS\_BCM6358) +**= leds-bcm6358.o

**obj-$(CONFIG\_LEDS\_BD2802) +**= leds-bd2802.o

**obj-$(CONFIG\_LEDS\_CPCAP) +**= leds-cpcap.o

**obj-$(CONFIG\_LEDS\_LOCOMO) +**= leds-locomo.o

**obj-$(CONFIG\_LEDS\_LM3530) +**= leds-lm3530.o

**obj-$(CONFIG\_LEDS\_LM3533) +**= leds-lm3533.o

**obj-$(CONFIG\_LEDS\_LM3642) +**= leds-lm3642.o

**obj-$(CONFIG\_LEDS\_MIKROTIK\_RB532) +**= leds-rb532.o

**obj-$(CONFIG\_LEDS\_S3C24XX) +**= leds-s3c24xx.o

**obj-$(CONFIG\_LEDS\_NET48XX) +**= leds-net48xx.o

**obj-$(CONFIG\_LEDS\_WRAP) +**= leds-wrap.o

**obj-$(CONFIG\_LEDS\_COBALT\_QUBE) +**= leds-cobalt-qube.o

**obj-$(CONFIG\_LEDS\_COBALT\_RAQ) +**= leds-cobalt-raq.o

**obj-$(CONFIG\_LEDS\_SUNFIRE) +**= leds-sunfire.o

**obj-$(CONFIG\_LEDS\_PCA9532) +**= leds-pca9532.o

**obj-$(CONFIG\_LEDS\_GPIO\_REGISTER) +**= leds-gpio-register.o

**obj-$(CONFIG\_LEDS\_GPIO) +**= leds-gpio.o

**obj-$(CONFIG\_LEDS\_LP3944) +**= leds-lp3944.o

**obj-$(CONFIG\_LEDS\_LP3952) +**= leds-lp3952.o

**obj-$(CONFIG\_LEDS\_LP55XX\_COMMON) +**= leds-lp55xx-common.o

**obj-$(CONFIG\_LEDS\_LP5521) +**= leds-lp5521.o

**obj-$(CONFIG\_LEDS\_LP5523) +**= leds-lp5523.o

**obj-$(CONFIG\_LEDS\_LP5562) +**= leds-lp5562.o

**obj-$(CONFIG\_LEDS\_LP8501) +**= leds-lp8501.o

**obj-$(CONFIG\_LEDS\_LP8788) +**= leds-lp8788.o

**obj-$(CONFIG\_LEDS\_LP8860) +**= leds-lp8860.o

**obj-$(CONFIG\_LEDS\_TCA6507) +**= leds-tca6507.o

**obj-$(CONFIG\_LEDS\_TLC591XX) +**= leds-tlc591xx.o

**obj-$(CONFIG\_LEDS\_CLEVO\_MAIL) +**= leds-clevo-mail.o

**obj-$(CONFIG\_LEDS\_IPAQ\_MICRO) +**= leds-ipaq-micro.o

**obj-$(CONFIG\_LEDS\_HP6XX) +**= leds-hp6xx.o

**obj-$(CONFIG\_LEDS\_OT200) +**= leds-ot200.o

**obj-$(CONFIG\_LEDS\_FSG) +**= leds-fsg.o

**obj-$(CONFIG\_LEDS\_PCA955X) +**= leds-pca955x.o

**obj-$(CONFIG\_LEDS\_PCA963X) +**= leds-pca963x.o

**obj-$(CONFIG\_LEDS\_DA903X) +**= leds-da903x.o

**obj-$(CONFIG\_LEDS\_DA9052) +**= leds-da9052.o

**obj-$(CONFIG\_LEDS\_WM831X\_STATUS) +**= leds-wm831x-status.o

**obj-$(CONFIG\_LEDS\_WM8350) +**= leds-wm8350.o

**obj-$(CONFIG\_LEDS\_PWM) +**= leds-pwm.o

**obj-$(CONFIG\_LEDS\_REGULATOR) +**= leds-regulator.o

**obj-$(CONFIG\_LEDS\_INTEL\_SS4200) +**= leds-ss4200.o

**obj-$(CONFIG\_LEDS\_LT3593) +**= leds-lt3593.o

**obj-$(CONFIG\_LEDS\_ADP5520) +**= leds-adp5520.o

**obj-$(CONFIG\_LEDS\_MC13783) +**= leds-mc13783.o

**obj-$(CONFIG\_LEDS\_NS2) +**= leds-ns2.o

**obj-$(CONFIG\_LEDS\_NETXBIG) +**= leds-netxbig.o

**obj-$(CONFIG\_LEDS\_ASIC3) +**= leds-asic3.o

**obj-$(CONFIG\_LEDS\_MAX77693) +**= leds-max77693.o

**obj-$(CONFIG\_LEDS\_MAX8997) +**= leds-max8997.o

**obj-$(CONFIG\_LEDS\_LM355x) +**= leds-lm355x.o

**obj-$(CONFIG\_LEDS\_BLINKM) +**= leds-blinkm.o

**obj-$(CONFIG\_LEDS\_SYSCON) +**= leds-syscon.o

**obj-$(CONFIG\_LEDS\_MENF21BMC) +**= leds-menf21bmc.o

**obj-$(CONFIG\_LEDS\_KTD2692) +**= leds-ktd2692.o

**obj-$(CONFIG\_LEDS\_POWERNV) +**= leds-powernv.o

**obj-$(CONFIG\_LEDS\_IS31FL319X) +**= leds-is31fl319x.o

**obj-$(CONFIG\_LEDS\_IS31FL32XX) +**= leds-is31fl32xx.o

**obj-$(CONFIG\_LEDS\_PM8058) +**= leds-pm8058.o

**obj-$(CONFIG\_LEDS\_MLXCPLD) +**= leds-mlxcpld.o

**obj-$(CONFIG\_LEDS\_MLXREG) +**= leds-mlxreg.o

**obj-$(CONFIG\_LEDS\_NIC78BX) +**= leds-nic78bx.o

**obj-$(CONFIG\_LEDS\_MT6323) +**= leds-mt6323.o

**obj-$(CONFIG\_LEDS\_LM3692X) +**= leds-lm3692x.o

**obj-$(CONFIG\_LEDS\_SC27XX\_BLTC) +**= leds-sc27xx-bltc.o

**obj-$(CONFIG\_LEDS\_LM3601X) +**= leds-lm3601x.o

# LED SPI Drivers

**obj-$(CONFIG\_LEDS\_CR0014114) +**= leds-cr0014114.o

**obj-$(CONFIG\_LEDS\_DAC124S085) +**= leds-dac124s085.o

# LED Userspace Drivers

**obj-$(CONFIG\_LEDS\_USER) +**= uleds.o

# LED Triggers

**obj-$(CONFIG\_LEDS\_TRIGGERS) +**= trigger/

**$(CONFIG\_NEW\_LEDS)是引用变量****CONFIG\_NEW\_LEDS，这个变量在内核根目录.config里面定义CONFIG\_NEW\_LEDS=y，.config是由XXX\_defconfig生成**

第一个编译的是led-core.c说明，有一个函数调用了他的函数，按照Makefile的调用关系找到驱动模型的入口函数类似module\_init(led\_init);

以led连接到gpio为例，led的驱动模型是bus-》platform bus-》led-gpio

文件：led-core.c ->led-class.c->

led-class.c ->led-gpio.c->

led-gpio.c

**函数：led\_init\_core()→of\_led\_classdev\_register()->**

**devm\_of\_led\_classdev\_register()->create\_gpio\_led()->**

**module\_platform\_driver(gpio\_led\_driver);**

**对 module\_platform\_driver(gpio\_led\_driver);展开**

**根据**

**#define** module\_platform\_driver(\_\_platform\_driver) \

module\_driver(\_\_platform\_driver, platform\_driver\_register, \

platform\_driver\_unregister)

得到

module\_driver(**gpio\_led\_driver, platform\_driver\_register, platform\_driver\_unregister**);

**根据**

**#define** module\_driver(\_\_driver, \_\_register, \_\_unregister, ...) \

**static** **int** \_\_init \_\_driver**##**\_init(**void**) \

{ \

**return** \_\_register(&(\_\_driver) , **##**\_\_VA\_ARGS\_\_); \

} \

module\_init(\_\_driver**##**\_init); \

**static** **void** \_\_exit \_\_driver**##**\_exit(**void**) \

{ \

\_\_unregister(&(\_\_driver) , **##**\_\_VA\_ARGS\_\_); \

} \

module\_exit(\_\_driver**##**\_exit);

得到

**static** **int** \_\_init **gpio\_led\_driver\_init**(**void**)

{

**return** **platform\_driver\_register**(&(**gpio\_led\_driver**) , **##**\_\_VA\_ARGS\_\_);

}

module\_init(**gpio\_led\_driver\_init**);

**static** **void** \_\_exit **gpio\_led\_driver\_exit**(**void**)

{

**platform\_driver\_register**(&(**gpio\_led\_driver**) , **##**\_\_VA\_ARGS\_\_);

}

module\_exit(**gpio\_led\_driver\_exit**);

\_\_VA\_ARGS\_\_可变参数个数为0，则编译时去掉##及前面的，，最终得到

**static** **int** \_\_init **gpio\_led\_driver\_init**(**void**)

{

**return** **platform\_driver\_register**(&(**gpio\_led\_driver**));

}

module\_init(**gpio\_led\_driver\_init**);

**static** **void** \_\_exit **gpio\_led\_driver\_exit**(**void**)

{

**platform\_driver\_register**(&(**gpio\_led\_driver**));

}

module\_exit(**gpio\_led\_driver\_exit**);

由此可见出现了module\_init(**gpio\_led\_driver\_init**);