

东月之神

在单纯观念里面，生命就容易变得比较深刻！

-  目录视图
-  摘要视图
-  订阅

[专家坐阵，Javascript实战分享](#) [微信开发学习路线高级篇上线](#) [免费公开课平台正式上线啦](#) [有奖征文：云服务器使用初体验](#)

初探linux子系统集之led子系统(一)

分类：[初探linux子系统集](#) 2014-07-08 19:54 1446人阅读 [评论\(0\)](#) [收藏](#) [举报](#)

就像学编程第一个范例helloworld一样，学嵌入式，单片机、fpga之类的第一个范例就是点亮一盏灯。对于庞大的linux系统，当然可以编写一个字符设备驱动来实现我们需要的led灯，也可以直接利用gpio口，应用程序来拉高拉低管脚控制。不过，既然linux系统自己本来就带有led子系统，那么就可以好好利用之。好处不用多说了，主要对于应用层来说，不同平台都用linux的led子系统，那么应用程序不用做任何的改变，就可以在新的平台上运行，可移植性好。

linux的led子系统的源码路径：

```
Include/Linux/leds.h  
/drivers/leds
```

首先看一下led系统中的主要文件：

```
# LED Core  
obj-$(CONFIG_NEW_LEDS)                +=led-core.o  
obj-$(CONFIG_LEDS_CLASS)              += led-class.o  
obj-$(CONFIG_LEDS_TRIGGERS)           +=led-triggers.o  
  
# LED PlatformDrivers  
obj-$(CONFIG_LEDS_GPIO)                += leds-gpio.o  
  
# LED Triggers  
obj-$(CONFIG_LEDS_TRIGGER_TIMER)      +=ledtrig-timer.o  
obj-$(CONFIG_LEDS_TRIGGER_IDE_DISK)   +=ledtrig-ide-disk.o  
obj-$(CONFIG_LEDS_TRIGGER_HEARTBEAT) +=ledtrig-heartbeat.o  
obj-$(CONFIG_LEDS_TRIGGER_BACKLIGHT) +=ledtrig-backlight.o  
obj-$(CONFIG_LEDS_TRIGGER_GPIO)       +=ledtrig-gpio.o  
obj-$(CONFIG_LEDS_TRIGGER_DEFAULT_ON) += ledtrig-default-on.o
```

主要由leds.h、led-core.c、led-class.c、led-triggers.c，其中led-triggers又分为了timer、ide-disk、heartbeat、backlight、gpio、default-on等算法。

例子程序是leds-gpio，接下去会主要分析这个驱动实现。

首先简单看一下主要的文件

Leds.h

```
1、enum led_brightness{  
    LED_OFF        = 0,  
    LED_HALF       = 127,  
    ...  
}
```

```

        LED_FULL      = 255,
};

```

Led的亮度，分为三等级，关、中间、最亮。

```

2、struct led_classdev{
    constchar      *name;    // Led的名字
    int            brightness; //led亮度
    int            max_brightness; //led最大亮度
    int            flags;

    /*Lower 16 bits reflect status */
#define LED_SUSPENDED      (1 << 0)
    /*Upper 16 bits reflect control information */
#define LED_CORE_SUSPENDRESUME (1 << 16)

    /*Set LED brightness level */
    /*Must not sleep, use a workqueue if needed */
    void            (*brightness_set)(struct led_classdev*led_cdev,
                                     enum led_brightness brightness);    //亮度设置函数指针

    /*Get LED brightness level */
    enumled_brightness (*brightness_get)(struct led_classdev *led_cdev); //获取亮度函数指针

    int            (*blink_set)(struct led_classdev*led_cdev,
                                unsigned long *delay_on,
                                unsigned long *delay_off);    //闪烁时点亮和熄灭的时间设置

    structdevice    *dev;
    structlist_head node;    //leds-list的node
    constchar      *default_trigger;    //默认trigger的名字

    unsignedlong    blink_delay_on,blink_delay_off;    //闪烁的开关时间
    structtimer_list blink_timer;    //闪烁的定时器链表
    int            blink_brightness;    //闪烁的亮度

#ifdef CONFIG_LEDS_TRIGGERS
    /*Protects the trigger data below */
    structrw_semaphore trigger_lock;    //trigger的锁

    structled_trigger *trigger;    //led的trigger
    structlist_head  trig_list;    //trigger的链表
    void            *trigger_data;    //trigger的数据
#endif
};

3、struct led_trigger {
    /*Trigger Properties */
    constchar      *name;    //trigger的名字
    void            (*activate)(struct led_classdev*led_cdev);    //激活trigger
    void            (*deactivate)(struct led_classdev*led_cdev);

    /*LEDs under control by this trigger (for simple triggers) */
    rwlock_t      leddev_list_lock;
    structlist_head led_cdevs;    //led设备的链表

    /*Link to next registered trigger */
    structlist_head next_trig;
};

4、/* For the leds-gpiodriver */
struct gpio_led {
    constchar *name;    //led的名字
    constchar *default_trigger;    //默认的trigger
    unsigned gpio;    //gpio口
    unsigned active_low : 1;
    unsigned retain_state_suspended : 1;
    unsigned default_state : 2;
    /*default_state should be one of LEDS_GPIO_DEFSTATE_ (ON|OFF|KEEP) */
};

```

```

5、struct gpio_led_platform_data {
    int          num_leds;           led的个数
    const struct gpio_led *leds;    led结构体

#define GPIO_LED_NO_BLINK_LOW      0      /*No blink GPIO state low */
#define GPIO_LED_NO_BLINK_HIGH    1      /*No blink GPIO state high */
#define GPIO_LED_BLINK            2      /* Please, blink */
    int          (*gpio_blink_set)(unsigned gpio,int state,
                                   unsignedlong *delay_on,
                                   unsignedlong *delay_off);
};

```

led-core.c

```

DECLARE_RWSEM(leds_list_lock);
EXPORT_SYMBOL_GPL(leds_list_lock);

LIST_HEAD(leds_list);
EXPORT_SYMBOL_GPL(leds_list);

```

主要是声明了leds的链表和锁。

Led-class.c

1、 leds_init

主要是创建leds_class，赋值suspend和resume以及dev_attrs。

led_class_attrs

```

static struct device_attribute led_class_attrs[] = {
    __ATTR(brightness,0644, led_brightness_show, led_brightness_store),
    __ATTR(max_brightness,0444, led_max_brightness_show, NULL),
#ifdef CONFIG_LEDS_TRIGGERS
    __ATTR(trigger,0644, led_trigger_show, led_trigger_store),
#endif
    __ATTR_NULL,
};

```

2、 led_classdev_register

创建classdev设备，也即Leds_class类中实例化一个对象，类似于c++的new一个对象，leds有很多种，而这里是注册一个特定的led，内核中的面向对象思想也极其丰富。

加到leds_list链表中，初始化blinktimer，指定blink_timer的function和data，设置trigger，然后一个新的led设备就注册好了，就可以使用了。

led-triggers.c

1、 led_trigger_register

扫描trigger链表中是否有同名的trigger，接着把当前trigger加入到链表中，如果led_classdev中有默认的trigger，那么就设置这个默认的。

好了，简单看了下led子系统中比较重要的结构体和函数，那么接下去就可以通过leds-gpio这个驱动来进一步了解led子系统了。

版权声明：本文为博主东月之神原创文章，未经博主允许不得转载。

- 上一篇[初探linux子系统集之写在前言](#)
- 下一篇[初探linux子系统集之led子系统\(二\)](#)

顶

0

踩

0

猜你在找

查看评论

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

个人资料



[eastmoon502136](#)

- 访问：326851次
- 积分：4115
- 等级：
- 排名：第3840名
- 原创：119篇
- 转载：0篇
- 译文：0篇
- 评论：219条

个性签名

别驻足，梦想要不停追逐；别认输，熬过黑夜才有日出。要记住，成功就在下一步；路很苦，汗水是最美的书！

文章搜索

文章分类

- [android](#)(13)
- [linux](#)(21)
- [证券投资](#)(6)
- [linux总线驱动](#)(19)
- [OK6410\(arm11\)](#)(9)
- [单片机](#)(1)
- [c](#)(7)
- [c++](#)(1)
- [网络](#)(3)
- [数据结构](#)(2)
- [算法](#)(3)

- [人生顿悟](#)(6)
- [电子小玩意](#)(1)
- [深入学习国嵌实验](#)(10)
- [linux内核源码0.11学习摘录](#)(4)
- [产品](#)(6)
- [杂感](#)(2)
- [初探linux子系统集](#)(5)

文章存档

- [2015年07月](#)(1)
- [2015年05月](#)(2)
- [2014年07月](#)(6)
- [2014年03月](#)(1)
- [2013年12月](#)(7)
- [2013年11月](#)(1)
- [2013年08月](#)(2)
- [2013年07月](#)(2)
- [2013年06月](#)(2)
- [2013年05月](#)(1)
- [2013年04月](#)(3)
- [2013年03月](#)(11)
- [2013年02月](#)(4)
- [2013年01月](#)(11)
- [2012年12月](#)(2)
- [2012年11月](#)(5)
- [2012年10月](#)(12)
- [2012年09月](#)(7)
- [2012年08月](#)(16)
- [2012年07月](#)(16)
- [2012年06月](#)(7)

阅读排行

- [和菜鸟一起学linux之bluez学习记录I](#)(51429)
- [和菜鸟一起学linux之wifi学习记录](#)(14127)
- [和菜鸟一起学电子小玩意之四轴飞行器原理](#)(13001)
- [和菜鸟一起学linux之V4L2摄像头应用流程](#)(10208)
- [和菜鸟一起学android4.0.3源码之硬件gps简单移植](#)(8974)
- [和菜鸟一起学linux之DBUS基础学习记录](#)(7661)
- [和菜鸟一起学android4.0.3源码之红外遥控器适配](#)(7354)
- [和菜鸟一起学android4.0.3源码之USB wifi移植心得](#)(6734)
- [和菜鸟一起学android4.0.3源码之bluetooth移植心得](#)(6338)
- [和菜鸟一起学android4.0.3源码之wifi的简单分析](#)(6029)

评论排行

- [和菜鸟一起学android4.0.3源码之wifi的简单分析](#)(24)
- [和菜鸟一起学android4.0.3源码之bluetooth移植心得](#)(21)
- [和菜鸟一起学linux之wifi学习记录](#)(19)
- [和菜鸟一起学android4.0.3源码之硬件gps简单移植](#)(12)
- [和菜鸟一起学linux之V4L2摄像头应用流程](#)(11)
- [和菜鸟一起学android4.0.3源码之wifi direct的简单分析](#)(11)
- [和菜鸟一起学android4.0.3源码之鼠标光标绘制简略版](#)(9)
- [和菜鸟一起学android4.0.3源码之USB wifi移植心得](#)(8)
- [和菜鸟一起学OK6410之最简单字符驱动](#)(7)
- [和菜鸟一起学linux之dlna的学习记录](#)(7)

推荐文章

- [* 美团Android DEX自动拆包及动态加载简介](#)
- [* Android实战技巧之四十四: Hello,Native!](#)
- [* 如何面试Python后端工程师?](#)
- [* Android自定义控件之仿汽车之家下拉刷新](#)
- [* 【android】音乐播放器之service服务设计](#)
- [* 【FastDev4Android框架开发】Android MVP开发模式详解\(十九\)](#)