东月之神

在单纯的观念里面，生命就容易变得比较深刻！

初探linux子系统集之led子系统(二)

分类： 初探linux子系统集 2014-07-09 20:39 1153人阅读 评论(1) 收藏 举报

巴西世界杯，德国7比1东道主，那个惨不忍睹啊，早上起来看新闻，第一眼看到7:1还以为点球也能踢成这样，后来想想，点球对多嘛6比1啊，接着就是各种新闻铺天盖地的来了。其实失败并没有什么，人生若是能够成功一次，那么再多的失败也是值得的，脚踏实地，失败了再次爬起来。很多时候，在一切都那么顺利的情况下，反而觉得没有啥意思，有一些挑战，一些失败，一些打击，接着，就会很努力很努力地去做，慢慢地就会被自己征服，很喜欢这种感受，废话说多了，还是接着研究led子系统吧。

对于led子系统中，有那么多得trigger，下面就来简单了解下。

1、default-on

```
<pre name="code" class="html">static voiddefon_trig_activate(struct led_classdev *led_cdev)
{
        led_set_brightness(led_cdev,led_cdev->max_brightness);
}

static structled_trigger defon_led_trigger = {
        .name     = "default-on",
        .activate = defon_trig_activate,
};

static voiddefon_trig_activate(struct led_classdev *led_cdev)
{
        led_set_brightness(led_cdev,led_cdev->max_brightness);
}

static structled_trigger defon_led_trigger = {
        .name     = "default-on",
        .activate = defon_trig_activate,
};
```

Default-on主要是设置led为最大亮度。

2、backlight

```
struct bl_trig_notifier {
        structled_classdev *led;        //led子系统设备
        intbrightness;                 //亮度
        intold_status;
        structnotifier_block notifier;     //内核通知链
```

```
        unsignedinvert;
};

static struct led_trigger bl_led_trigger ={
        .name                = "backlight",
        .activate  = bl_trig_activate,
        .deactivate       = bl_trig_deactivate
};
static void bl_trig_activate(structled_classdev *led)
{
        intret;

        structbl_trig_notifier *n;

        n= kzalloc(sizeof(struct bl_trig_notifier), GFP_KERNEL);
        led->trigger_data= n;
        if(!n) {
                dev_err(led->dev,"unable to allocate backlight trigger\n");
                return;
        }

        ret= device_create_file(led->dev, &dev_attr_inverted);
        if(ret)
                gotoerr_invert;

        n->led= led;
        n->brightness= led->brightness;
        n->old_status= UNBLANK;
        n->notifier.notifier_call= fb_notifier_callback;

        ret= fb_register_client(&n->notifier);
        if(ret)
                dev_err(led->dev,"unable to register backlight trigger\n");

        return;

err_invert:
        led->trigger_data= NULL;
        kfree(n);
}
```

其中fb_register_client注册到了framebuffer中的fb_notifier_list中，一旦framebuffer驱动中有事件，就会调用内核通知链中注册好的函数fb_notifier_callback。

关于内核通知链，这里就插播一曲来自网络的摘抄了：

大多数内核子系统都是相互独立的，因此某个子系统可能对其它子系统产生的事件感兴趣。为了满足这个需求，也即是让某个子系统在发生某个事件时通知其它的子系统，Linux内核提供了通知链的机制。通知链表只能够在内核的子系统之间使用，而不能够在内核与用户空间之间进行事件的通知。

通知链表是一个函数链表，链表上的每一个节点都注册了一个函数。当某个事情发生时，链表上所有节点对应的函数就会被执行。所以对于通知链表来说有一个通知方与一个接收方。在通知这个事件时所运行的函数由被通知方决定，实际上也即是被通知方注册了某个函数，在发生某个事件时这些函数就得到执行。其实和系统调用signal的思想差不多。

通知链技术可以概括为：事件的被通知者将事件发生时应该执行的操作通过函数指针方式保存在链表（通知链）中，然后当事件发生时通知者依次执行链表中每一个元素的回调函数完成通知。

```
static int fb_notifier_callback(struct notifier_block *p,
                                    unsignedlong event, void *data)
{
         structbl_trig_notifier *n = container_of(p,
                                                  structbl_trig_notifier, notifier);
         struct led_classdev*led = n->led;
         struct fb_event*fb_event = data;
         int *blank =fb_event->data;
         int new_status =*blank ? BLANK : UNBLANK;

         switch (event) {
         case FB_EVENT_BLANK :
                   if(new_status == n->old_status)
                            break;

                   if((n->old_status == UNBLANK) ^ n->invert) {
                            n->brightness= led->brightness;
                            led_set_brightness(led,LED_OFF);
                   } else {
                            led_set_brightness(led,n->brightness);
                   }

                   n->old_status= new_status;

                   break;
         }

         return 0;
}
```

如果触发了FB_EVENT_BLANK，那么就执行相应的操作。

3、timer

```
static structled_trigger timer_led_trigger = {
                           .name     = "timer",
                           .activate =timer_trig_activate,
                           .deactivate =timer_trig_deactivate,
};

static voidtimer_trig_activate(struct led_classdev *led_cdev)
{
                           int rc;

                           led_cdev->trigger_data= NULL;

                           rc =device_create_file(led_cdev->dev, &dev_attr_delay_on);
                           if (rc)
                            return;
                           rc =device_create_file(led_cdev->dev, &dev_attr_delay_off);
                           if (rc)
                            gotoerr_out_delayon;

                           led_blink_set(led_cdev,&led_cdev->blink_delay_on,
                                  &led_cdev->blink_delay_off);

                           led_cdev->trigger_data= (void *)1;

                           return;

err_out_delayon:
                           device_remove_file(led_cdev->dev,&dev_attr_delay_on);
}
```

当某个led_classdev与之连接后，这个触发器会在/sys/class/leds/<device>/下创建两个文件delay_on和delay_off。用户空间往这两个文件中写入数据后，相应的led会按照设置的高低电平的时间（ms）来闪烁。如果led_classdev注册了硬件闪烁的接口led_cdev->blink_set就是用硬件控制闪烁，否则用软件定时器来控制闪烁。

4、heatbeat

```
static structled_trigger heartbeat_led_trigger = {
                          .name     = "heartbeat",
                          .activate =heartbeat_trig_activate,
                          .deactivate = heartbeat_trig_deactivate,
};

structheartbeat_trig_data {
                          unsigned int phase;
                          unsigned int period;
                          struct timer_listtimer;
};

static voidheartbeat_trig_activate(struct led_classdev *led_cdev)
{
                          structheartbeat_trig_data *heartbeat_data;

                          heartbeat_data =kzalloc(sizeof(*heartbeat_data), GFP_KERNEL);
                          if (!heartbeat_data)
                           return;

                          led_cdev->trigger_data= heartbeat_data;
                          setup_timer(&heartbeat_data->timer,
                               led_heartbeat_function, (unsigned long)led_cdev);
                          heartbeat_data->phase= 0;
                          led_heartbeat_function(heartbeat_data->timer.data);
}
```

设置了heartbeat_data->phase，然后调用led_heartbeat_function。

```
static voidled_heartbeat_function(unsigned long data)
{
                          struct led_classdev*led_cdev = (struct led_classdev *) data;
                          structheartbeat_trig_data *heartbeat_data = led_cdev->trigger_data;
                          unsigned longbrightness = LED_OFF;
                          unsigned long delay =0;

                          /* acts like anactual heart beat -- ie thump-thump-pause... */
                          switch(heartbeat_data->phase) {
                          case 0:
                           /*
                            * The hyperbolic function below modifies the
                            * heartbeat period length in dependency of the
                            * current (1min) load. It goes through thepoints
                            * f(0)=1260, f(1)=860, f(5)=510,f(inf)->300.
                            */
                           heartbeat_data->period= 300 +
                                    (6720<< FSHIFT) / (5 * avenrun[0] + (7 << FSHIFT));
                           heartbeat_data->period=
                                    msecs_to_jiffies(heartbeat_data->period);
                           delay =msecs_to_jiffies(70);
                           heartbeat_data->phase++;
                           brightness =led_cdev->max_brightness;
                           break;
                          case 1:
                           delay =heartbeat_data->period / 4 - msecs_to_jiffies(70);
                           heartbeat_data->phase++;
                           break;
                          case 2:
                           delay =msecs_to_jiffies(70);
                           heartbeat_data->phase++;
                           brightness =led_cdev->max_brightness;
```

```
                                break;
                               default:
                                delay =heartbeat_data->period - heartbeat_data->period / 4 -
                                         msecs_to_jiffies(70);
                                heartbeat_data->phase= 0;
                                break;
                               }

                               led_set_brightness(led_cdev,brightness);
                               mod_timer(&heartbeat_data->timer,jiffies + delay);
}
```

通过定时来实现类似于心跳的led灯。

5、ide-disk

```
static voidledtrig_ide_timerfunc(unsigned long data)
{
                               if (ide_lastactivity!= ide_activity) {
                                ide_lastactivity =ide_activity;
                                /* INT_MAX will seteach LED to its maximum brightness */
                                led_trigger_event(ledtrig_ide,INT_MAX);
                                mod_timer(&ledtrig_ide_timer,jiffies + msecs_to_jiffies(10));
                               } else {
                                led_trigger_event(ledtrig_ide,LED_OFF);
                               }
}

static int __initledtrig_ide_init(void)
{
                               led_trigger_register_simple("ide-disk",&ledtrig_ide);
                               return 0;
}
```

通过定时器实现类似于硬盘灯的指示。

以上便是led子系统中的trigger的一些简单介绍。

- 上一篇初探linux子系统集之led子系统(一)
- 下一篇初探linux子系统集之led子系统(三)

eastmoon502136

- 访问：326849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

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)