

wuyuwei45的专栏

目录视图

摘要视图

RSS 订阅

个人资料



wuyuwei45

访问：23638次

积分：888分

排名：第19459名

原创：64篇

转载：21篇

译文：3篇

评论：13条

文章搜索

文章分类

Misc (8)

Android系统 (16)

Android硬件抽象层 (2)

Linux应用 (8)

Linux系统 (14)

Linux驱动 (35)

ARM架构 (4)

C/C++ (4)

U-boot (6)

闲记 (1)

文章存档

2013年07月 (10)

2013年06月 (8)

2013年05月 (23)

2013年04月 (21)

2013年03月 (26)

阅读排行

Linux Gadget的一点研究 (1955)

Linux Gadget的一点研究 (1059)

Linux I2C驱动: i2c_dev (926)

Android中LCD背光驱动 (867)

博客专家福利

2015年1月微软MVP申请开始啦

专访荣浩：流程的永恒之道

推荐有礼--找出您心中的技术大牛

Android中LCD背光驱动

分类：Linux驱动 Android系统

2013-06-17 16:28 867人阅读 评论(0) 收藏 举报

目录(?)

1. Android的Setting

2. Android的背光JNI层

3. Android的背光HAL层

4. Linux的背光内核层

5. Linux的背光驱动层

6. 总结

其实Android的底层就是Linux，所以其驱动本质就是Linux驱动，但是这些Linux驱动是服务上层Android的，所以需要遵循上Android的一些接口规范。所以涉及到的Android驱动都应密切关注上传递的接口。本文介绍的LCD背光驱动就是从上层一直往下层展现，但是笔者毕竟不是专注于Android上层，碍于知识不充裕，所以对上层的东西介绍得相对简单。

1.Android的Setting

Android的设置里面管理了Andoird系统的所有设置，其中当然包括了屏幕亮度设置。

Setting的源码目录在：

mydroid/packages/apps/Settings/src/com/android/settings

亮度设置的java源文件在：

mydroid/packages/apps/Settings/src/com/android/settings/BrightnessPreference.java

打开这个文件看到：

[java]

01. public class BrightnessPreference extends SeekBarDialogPreference implements

02. SeekBar.OnSeekBarChangeListener, CheckBox.OnCheckedChangeListener {

03.

04. private SeekBar mSeekBar;

05. private CheckBox mCheckBox;

06.

07. private int mOldBrightness;

08. private int mOldAutomatic;

09.

10. private boolean mAutomaticAvailable;

11.

12. private boolean mRestoredOldState;

13.

14. // Backlight range is from 0 - 255. Need to make sure that user

15. // doesn't set the backlight to 0 and get stuck

16. private int mScreenBrightnessDim =

17. ...

Android的最上层已经将背光亮度量化为了[0,255]个等级，并且提示注意不要设置为0，所以在进行最低层的背光驱动编写时，可以合理按这个范围部署背光的亮度。

2.Android的背光JNI层

背光的JNI层源码在：

mydroid/frameworks/base/services/jni/com_android_server_LightsService.cpp

这一层就是调用HAL层的方法，为上一层实现一个设置亮度接口。

3.Android的背光HAL层

Java App和JNI一般是google维护的，所以源码位置相对固定，HAL有产品商开发维护的，所以位置是不固定的，

http://blog.csdn.net/wuyuwei45/article/details/9113389#3

1/6

Android/Linux USB Gadg	(674)
No rule to make target `c	(630)
移植Android时关于Linux	(426)
Linux Gadget的一点研究	(388)
MMC/SD卡驱动实例开发	(358)
Linux spi设备驱动	(351)

评论排行	
Linux Gadget的一点研究	(10)
Linux Gadget的一点研究	(2)
c++日志工具之——log4c	(1)
V4L2官方例程	(0)
Linux Gadget的一点研究	(0)
网页转载	(0)
守护进程：代码的分析	(0)
嵌入式常用笔试题	(0)
GStreamer：初识	(0)
Linux驱动调试手段：打E	(0)

推荐文章	
* ArcGIS 构建3D动画方法	
* Unity3D游戏开发之截屏保存精彩瞬间	
* 大数据让生活更加糟糕	
* 用Swift开发一个TODO应用	
* ActionBar 样式详解 -- 样式主题 简介 ActionBar 的 icon logo 标题 菜单样式修改	
* 使用Unity Render Textures实	

最新评论	
c++日志工具之——log4cplus canliaaa: 谢谢分享!	
Linux Gadget的一点研究之U盘和flexman09: 看完留影	
Linux Gadget的一点研究之U盘和flexman09: 看完留影	
Linux Gadget的一点研究之HID诶红尘六欲: @wuyuwei45:差不多, 不过我是在hidg_setup返回后, 具体看这里, 我已经写出来了。ht...	
Linux Gadget的一点研究之HID诶wuyuwei45: @hclydao:太好了, 你是在hidg_setup里增加对应的响应吗? 还是怎样? 望多交流赐教! 嘿嘿	
Linux Gadget的一点研究之HID诶红尘六欲: @wuyuwei45:已经解决了.感觉你的回复.	
Linux Gadget的一点研究之HID诶wuyuwei45: @hclydao:Hi,OxA应该是一个类请求, 你判断收到这个请求后其实是不用回复任何数据的, 直接将...	
Linux Gadget的一点研究之HID诶红尘六欲: @hclydao:应该是由composite_setup到hidg_setup的还是没有研究清楚 ...	
Linux Gadget的一点研究之HID诶红尘六欲: @wuyuwei45:非常感谢你的回复 我跟踪代码后发现前面host发过来的请求例如:0x09 0x...	
Linux Gadget的一点研究之HID诶wuyuwei45: @hclydao:你的问题已经涉及到的gadget驱动的UDC层部分了, 你具体的问题我倒是	

看产品上喜好, 笔者使用的TI OMAP4平台, 背光的HAL层代码就在:

mydroid/device/ti/xxx_product/liblights/light.c

先浏览欣赏一下light.c先

```
[cpp]
01. /*
02.  * Copyright (C) 2008 The Android Open Source Project
03.  *
04.  * Licensed under the Apache License, Version 2.0 (the "License");
05.  * you may not use this file except in compliance with the License.
06.  * You may obtain a copy of the License at
07.  *
08.  *     http://www.apache.org/licenses/LICENSE-2.0
09.  *
10.  * Unless required by applicable law or agreed to in writing, software
11.  * distributed under the License is distributed on an "AS IS" BASIS,
12.  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13.  * See the License for the specific language governing permissions and
14.  * limitations under the License.
15.  */
16.
17.
18. #define LOG_TAG "lights"
19.
20. #include <utils/log.h>
21.
22. #include <stdint.h>
23. #include <string.h>
24. #include <unistd.h>
25. #include <errno.h>
26. #include <fcntl.h>
27. #include <pthread.h>
28.
29. #include <sys/ioctl.h>
30. #include <sys/types.h>
31.
32. #include <hardware/lights.h>
33.
34. /*****
35.
36. static pthread_once_t g_init = PTHREAD_ONCE_INIT;
37. static pthread_mutex_t g_lock = PTHREAD_MUTEX_INITIALIZER;
38.
39. char const*const LCD_FILE
40.     = "/sys/class/leds/lcd-backlight/brightness";
41. char const*const KEYBOARD_FILE
42.     = "/sys/class/leds/keyboard-backlight/brightness";
43.
44. char const*const CHARGING_LED_FILE
45.     = "/sys/class/leds/battery-led/brightness";
46.
47. /*RGB file descriptors */
48. char const*const RED_LED_FILE
49.     = "/sys/class/leds/red/brightness";
50. char const*const RED_DELAY_ON_FILE
51.     = "/sys/class/leds/red/delay_on";
52. char const*const RED_DELAY_OFF_FILE
53.     = "/sys/class/leds/red/delay_off";
54. char const*const GREEN_LED_FILE
55.     = "/sys/class/leds/green/brightness";
56. char const*const GREEN_DELAY_ON_FILE
57.     = "/sys/class/leds/green/delay_on";
58. char const*const GREEN_DELAY_OFF_FILE
59.     = "/sys/class/leds/green/delay_off";
60. char const*const BLUE_LED_FILE
61.     = "/sys/class/leds/blue/brightness";
62. char const*const BLUE_DELAY_ON_FILE
63.     = "/sys/class/leds/blue/delay_on";
64. char const*const BLUE_DELAY_OFF_FILE
65.     = "/sys/class/leds/blue/delay_off";
66. ...

[cpp]
01. static int
02. set_light_backlight(struct light_device_t* dev,
03.                     struct light_state_t const* state)
```

没碰到过。...

```
04. {
05.     int err = 0;
06.     int brightness = rgb_to_brightness(state);
07.
08.     pthread_mutex_lock(&g_lock);
09.     err = write_int(LCD_FILE, brightness);
10.     pthread_mutex_unlock(&g_lock);
11.
12.     return err;
13. }
```

这里关注一下LCD背光的sys文件节点: "/sys/class/leds/lcd-backlight/brightness"

疑问1: 这个sys接口是谁约定的呢?

疑问2: 可不可以改这个接口呢?

疑问3: 为什么这里还有其他led的很多sys文件节点呢:

先解释疑问3, 因为一个Android设备, 不只是有LCD背光的LED, 还有可以有其他很多LED (当然也可以没有), 所以这里一并实现了这些LED的HAL, 产品商可以沿用这些HAL。

先来试验一把:

启动Android设备, 在Setting里更改亮度, 然后在串口命令行中或"adb shell"中运行命令 `cat /sys/class/leds/lcd-backlight/brightness`, 会发现Setting的更改后可以通过cat 显示Setting的更改。所以更加确认了这个"/sys/class/leds/lcd-backlight/brightness"接口是正确的。

因为HAL层是和Linux Kernel交互的, 所以这里如果能cat实现读取亮度等级, 那么在Linux kernel层就一定实现了sysfs接口。

4.Linux的背光内核层

背光的内核层源码在: `driver/leds/led-class.c`

这一层由内核开发者去维护, 不用我们操心。看看它的init函数

```
[cpp]
01. static int __init leds_init(void)
02. {
03.     leds_class = class_create(THIS_MODULE, "leds");
04.     if (IS_ERR(leds_class))
05.         return PTR_ERR(leds_class);
06.     leds_class->suspend = led_suspend;
07.     leds_class->resume = led_resume;
08.     leds_class->dev_attrs = led_class_attrs;
09.     return 0;
10. }
```

好明显, 正是由于它创建了sys class, 名字为"leds", 所以上面的背光sys文件节点"/sys/class/leds/"就有了来由, 那么剩下的"./lcd-backlight/brightness"又是怎么来的呢? 看看一个注册led设备类的函数。

```
[cpp]
01. /**
02.  * led_classdev_register - register a new object of led_classdev class.
03.  * @parent: The device to register.
04.  * @led_cdev: the led_classdev structure for this device.
05.  */
06. int led_classdev_register(struct device *parent, struct led_classdev *led_cdev)
07. {
08.     led_cdev->dev = device_create(leds_class, parent, 0, led_cdev,
09.                                  "%s", led_cdev->name);
10.     if (IS_ERR(led_cdev->dev))
11.         return PTR_ERR(led_cdev->dev);
12.
13.     #ifdef CONFIG_LEDS_TRIGGERS
14.         init_rwsem(&led_cdev->trigger_lock);
15.     ...
```

这个注册函数的接口最终会被我们要开发的背光驱动调用, 这个接口在/sys/class/leds/下又创建了一个设备接口, 名字是led_cdev->name。好明显这里的led_cdev->name应该就是"lcd-backlight", 究竟是不是真的这样呢? 继续看。

这个led-class.c在实现两个设备属性, 看代码:

```
[cpp]
01. static struct device_attribute led_class_attrs[] = {
02.     __ATTR(brightness, 0644, led_brightness_show, led_brightness_store),
```

```

03.     __ATTR(max_brightness, 0444, led_max_brightness_show, NULL),
04. #ifdef CONFIG_LEDS_TRIGGERS
05.     __ATTR(trigger, 0644, led_trigger_show, led_trigger_store),
06. #endif
07.     __ATTR_NULL,
08. };

```

看到了属性名字为“brightness”，这似乎越来越接近解释“/sys/class/leds/lcd-backlight/brightness”的由来了。

5.Linux的背光驱动层

Linux的背光驱动层就是完全由开发者去实现了，其实这层很简单，无非就是通过pwm实现设置背光。在这一层要注意将背光亮度量化为0~255，这是Android上层约定的。

这一层的采用Linux的platform device/driver 模型。它最终会调用Linux内核层leds-class.c的接口。

看看它的probe函数片段：

```

[cpp]
01. struct display_led_data {
02.     struct led_classdev pri_display_class_dev;
03.     struct led_classdev sec_display_class_dev;
04.     struct omap4_disp_led_platform_data *led_pdata;
05.     struct mutex pri_disp_lock;
06.     struct mutex sec_disp_lock;
07. };
08. static int omap4_XXX_display_probe(struct platform_device *pdev)
09. {
10.     int ret;
11.     struct display_led_data *info;
12.
13.     pr_info("%s:Enter\n", __func__);
14.
15.     if (pdev->dev.platform_data == NULL) {
16.         pr_err("%s: platform data required\n", __func__);
17.         return -ENODEV;
18.     }
19.
20.     info = kzalloc(sizeof(struct display_led_data), GFP_KERNEL);
21.     if (info == NULL) {
22.         ret = -ENOMEM;
23.         return ret;
24.     }
25.
26.     info->led_pdata = pdev->dev.platform_data;
27.     platform_set_drvdata(pdev, info);
28.
29.     info->pri_display_class_dev.name = "lcd-backlight";
30.
31.     info->pri_display_class_dev.brightness_set = omap4_XXX_primary_disp_store;
32.     ...
33.
34.
35.     ret = led_classdev_register(&pdev->dev,
36.                                &info->pri_display_class_dev);
37.     if (ret < 0) {
38.         pr_err("%s: Register led class failed\n", __func__);
39.         kfree(info);
40.         return ret;
41.     }
42.
43.     if (info->led_pdata->flags & LEDS_CTRL_AS_TWO_DISPLAYS) {
44.         pr_info("%s: Configuring the secondary LED\n", __func__);
45.         info->sec_display_class_dev.name = "lcd-backlight2";
46.         info->sec_display_class_dev.brightness_set =
47.             omap4_mirage_secondary_disp_store;
48.         info->sec_display_class_dev.max_brightness = LED_OFF;
49.         mutex_init(&info->sec_disp_lock);
50.
51.         ret = led_classdev_register(&pdev->dev,
52.                                    &info->sec_display_class_dev);
53.         ...

```

需关注这里设置了info->pri_display_class_dev.name = "lcd-backlight";，然后调用led_classdev_register函数注册，所以完美解释了

led_cdev->name就是 "lcd-backlight", 完美解释了"/sys/class/leds/lcd-backlight/brightness"的由来。
另外info->pri_display_class_dev.brightness_set = omap4_xxx_primary_disp_store;中的
omap4_xxx_primary_disp_store()就是我们需要实现的设置背光亮度函数，函数里面就是pwm操作。
到这里又有一个疑问4：为什么只实现设置背光亮度的接口，而没有实现读取当前亮度量化值的接口？
其实Android上层自行处理了这个获取亮度量化值的事情，也就是说Android上层设置了亮度是多少，上层会执行保留设置结果。无需再通过下层读取。

6.总结

- 6.1针对之前的疑问1/2，其实LCD背光的sys接口路径是可以改的，但是需要Linux内核层和Android HAL层配合来改，单单改一方都是会导致Android Setting无法调节背光。
- 6.2需注意在实现Linux背光驱动时的亮度量化关系，也就是注意上层传递下来的亮度设置范围是0~255。
- 6.3Android底层的Linux驱动都是服务于上层Java的，在做Android底层的Linux驱动时需要明确和上层的接口依赖关系，否则无法重用google或芯片厂商实现的接口，从而导致功能无法用。
- 6.4.再一次证明了Android系统服务运行效率极低。设置背光，其实最终就是设置了一下pwm寄存器即可，但是从Android最顶层一层层调用下来，真是“费尽周折”，怪不得Android设备的硬件配置明显远优于ios设备，但是流畅体验性却不明显优于ios设备。

上一篇 [Android/Linux USB Gadget : 续](#)
下一篇 [GPS调试-1](#)

主题推荐

[android](#) [linux驱动](#) [linux内核](#) [ios设备](#) [linux kernel](#)

猜你在找

在Ubuntu上为Android系统编写Linux内核驱动程序
在Ubuntu上为Android系统编写Linux内核驱动程序
在Ubuntu上为Android系统编写Linux内核驱动程序
在Android上编译linux内核驱动程序
在Ubuntu上为Android增加硬件抽象层（HAL）模块访问

在Ubuntu上为Android系统内置C可执行程序测试Linux内
在Ubuntu上为Android系统编写Linux内核驱动程序
在Ubuntu上为Android增加硬件抽象层（HAL）模块访问
（一）在Ubuntu上为Android系统编写Linux内核驱动程
在Ubuntu上为Android增加硬件抽象层（HAL）模块访问

[查看评论](#)

暂无评论

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 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

核心技术类目

全部主题 [Hadoop](#) [AWS](#) [移动游戏](#) [Java](#) [Android](#) [iOS](#) [Swift](#) [智能硬件](#) [Docker](#) [OpenStack](#)
[VPN](#) [Spark](#) [ERP](#) [IE10](#) [Eclipse](#) [CRM](#) [JavaScript](#) [数据库](#) [Ubuntu](#) [NFC](#) [WAP](#) [jQuery](#)
[BI](#) [HTML5](#) [Spring](#) [Apache](#) [.NET](#) [API](#) [HTML](#) [SDK](#) [IIS](#) [Fedora](#) [XML](#) [LBS](#) [Unity](#)
[Splashtop](#) [UML](#) [components](#) [Windows Mobile](#) [Rails](#) [QEMU](#) [KDE](#) [Cassandra](#) [CloudStack](#) [FTC](#)
[coremail](#) [OPhone](#) [CouchBase](#) [云计算](#) [iOS6](#) [Rackspace](#) [Web App](#) [SpringSide](#) [Maemo](#)
[Compuware](#) [大数据](#) [aptech](#) [Perl](#) [Tornado](#) [Ruby](#) [Hibernate](#) [ThinkPHP](#) [HBase](#) [Pure](#) [Solr](#)

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [银行汇款帐号](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)
[网站客服](#) [杂志客服](#) [微博客服](#) webmaster@csdn.net [400-600-2320](tel:400-600-2320)

京 ICP 证 070598 号

北京创新乐知信息技术有限公司 版权所有

江苏乐知网络技术有限公司 提供商务支持

Copyright © 1999-2014, CSDN.NET, All Rights Reserved 