



聚酰亚胺薄膜



联系我们



请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心



空气源热泵



文章搜索

博客专栏



Android——4.2 Vold

文章：8篇

阅读：33768



Android——4.2_3G移植之路_RIL

文章：6篇

阅读：38783



Linux/Android - Input 系统

文章：8篇

阅读：48627

文章分类

【Android — 应用】 (25)

【Android — 框架】 (13)

【Android — 机制】 (24)

【Android — 驱动】 (8)

【Android — 编译】 (5)

【Linux — Driver】 (7)

【Embedded】 (25)

【Ubuntu】 (10)

【Workplace】 (3)

【Network】 (2)

【C/C++】 (6)

Source Code

 目录视图

 摘要视图



Linux/Android——输入子系统input_event传递 (二)

标签：input dev event input_dev touchscreen-ABS

2014年12月23日 20:13:38 11931人阅读 评论(0) 收藏

分类：【Android — 驱动】 (7)

版权声明：免责声明：本人在此发文（包括但不限于汉字、拼音、拉丁字母）均为随意敲击键盘所出，用于检验本人电脑键盘录入、屏幕机械、光电性能，并不代表本人局部或全部同意、支持或者反对观点。如需要详查请直接与键盘生产厂商法人代表联系。挖井排水无水表，不欠账。
来源：<https://blog.csdn.net/jscese/article/details/42099381>

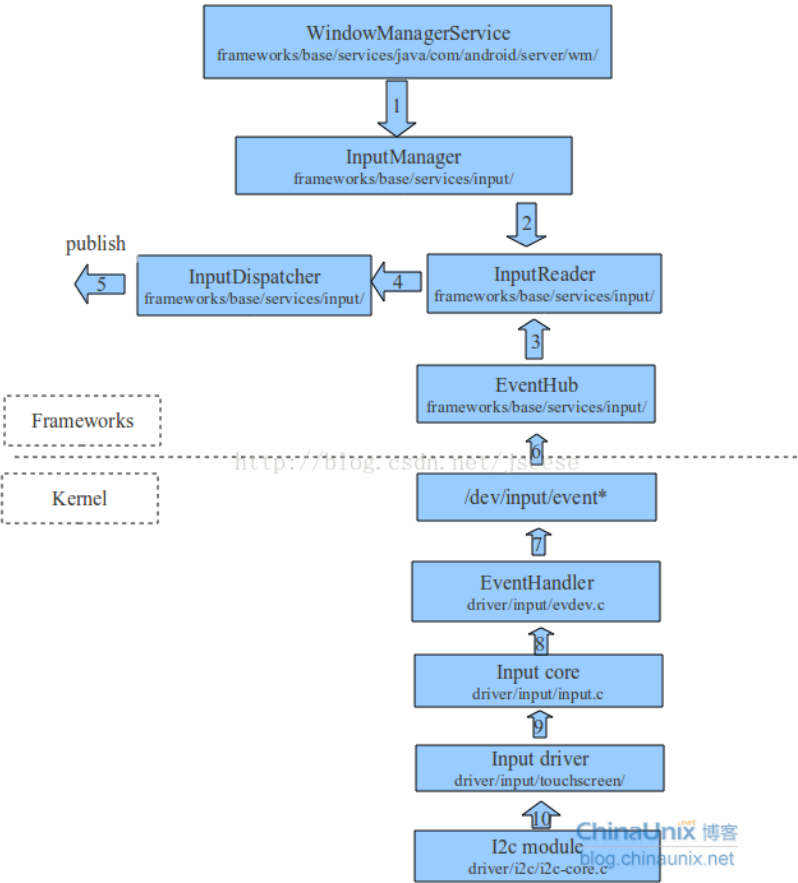
在前文 [Linux/Android——usb触摸屏驱动 - usbtouchscreen \(一\)](#) 中记录了如何在kernel中添加input类型为touchscreen的驱动，

这在整个输入体系中最下层的设备驱动部分，往上一层就是linux内核的管理驱动input系统，kernel中位置：/kernel/drivers/input/input.c

撰写不易，转载需注明！

<http://blog.csdn.net/jscese/article/details/42099381>

到目前已经完全调通，可以正常使用了，现在记录一下这段时间接触到的Android 输入input 系统，先补上的层次图，蛮不错的：





聚酰亚胺薄膜



联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

💬 QQ客服 🗨 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

最新评论

Android——RIL 机制源码...

zisuchen : 你好问下, 如果我是在mtk平台上移植要怎么移植, mtk有改动rild改成mtk的代码

Android——build.pr...

LosingCarryJie : 虽然是篇老博客, 但是帮助我弄懂了源码! 感谢!

extern "C" 在C/C++...

诶阿星 : 看了几篇博客还是比较晕, 看了你的明白了, 谢谢楼主分享, 向你学习

OpenCV&OpenCL...

xavier19841016 : 看到你这个免责声明, 我笑了

Android——4.2 Vol...

Winston_jory : 楼主好®, 就连版权声明都写得这么形象。

OpenGL ES 入门 (一)

imwoohan : 期待下一期

OpenGL ES 入门 (一)

imwoohan : 赞一个。

4月跳槽路

UPON--知道个P : 唉! 我也有过类似的遭遇! 没法, 打铁还需自身硬! 练技术中。。。练完准备出山。

Unity3D——MonoBeha...

南、烟 : [reply]javi9111[/reply] 我也没在TV大厂待过, 回答不了你的问题, 在天朝有政府...

Unity3D——MonoBeha...

南、烟 : [reply]javi9111[/reply] 看你前面写了一大堆。。你想说啥 - - 。 uni...

上一篇博客里面的 `usbtouchscreen` 就是对应上图的I2c module的位置, 而在kernel中input的核心就是

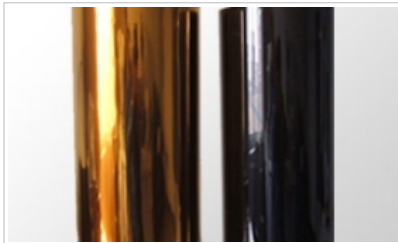
input_dev :

这个结构体表述的是一个输入设备的相关信息, 在usbtouchscreen 驱动中的 `usbtouch_probe` 会 `input_dev` 作为usbtouch设备的一部分。

会对 `input_dev` 做一系列的初始化, 设置参数之类的, 具体可参考之前博客

`input_dev` 结构原型如下, `/kernel/include/linux/input.h`中定义 :

```
[objc]
1.  /**
2.   * struct input_dev - represents an input device
3.   * @name: name of the device
4.   * @phys: physical path to the device in the system hierarchy
5.   * @uniq: unique identification code for the device (if device has it)
6.   * @id: id of the device (struct input_id)
7.   * @propbit: bitmap of device properties and quirks
8.   * @evbit: bitmap of types of events supported by the device (EV_KEY,
9.   *   EV_REL, etc.)
10.  * @keybit: bitmap of keys/buttons this device has
11.  * @relbit: bitmap of relative axes for the device
12.  * @absbit: bitmap of absolute axes for the device
13.  * @mscbit: bitmap of miscellaneous events supported by the device
14.  * @ledbit: bitmap of leds present on the device
15.  * @sndbit: bitmap of sound effects supported by the device
16.  * @ffbit: bitmap of force feedback effects supported by the device
17.  * @swbit: bitmap of switches present on the device
18.  * @hint_events_per_packet: average number of events generated by the
19.  *   device in a packet (between EV_SYN/SYN_REPORT events). Used by
20.  *   event handlers to estimate size of the buffer needed to hold
21.  *   events.
22.  * @keycodemax: size of keycode table
23.  * @keycodesize: size of elements in keycode table
24.  * @keycode: map of scan codes to keycodes for this device
25.  * @getkeycode: optional legacy method to retrieve current keymap.
26.  * @setkeycode: optional method to alter current keymap, used to implement
27.  *   sparse keymaps. If not supplied default mechanism will be used.
28.  *   The method is being called while holding event_lock and thus must
29.  *   not sleep
30.  * @ff: force feedback structure associated with the device if device
31.  *   supports force feedback effects
32.  * @repeat_key: stores key code of the last key pressed; used to implement
33.  *   software autorepeat
34.  * @timer: timer for software autorepeat
35.  * @rep: current values for autorepeat parameters (delay, rate)
36.  * @mt: pointer to array of struct input_mt_slot holding current values
37.  *   of tracked contacts
38.  * @mtsize: number of MT slots the device uses
39.  * @slot: MT slot currently being transmitted
40.  * @trkid: stores MT tracking ID for the current contact
41.  * @absinfo: array of &struct input_absinfo elements holding information
42.  *   about absolute axes (current value, min, max, flat, fuzz,
43.  *   resolution)
44.  * @key: reflects current state of device's keys/buttons
45.  * @led: reflects current state of device's LEDs
46.  * @snd: reflects current state of sound effects
47.  * @sw: reflects current state of device's switches
48.  * @open: this method is called when the very first user calls
49.  *   input_open_device(). The driver must prepare the device
50.  *   to start generating events (start polling thread,
51.  *   request an IRQ, submit URB, etc.)
52.  * @close: this method is called when the very last user calls
53.  *   input_close_device().
54.  * @flush: purges the device. Most commonly used to get rid of force
55.  *   feedback effects loaded into the device when disconnecting
56.  *   from it
57.  * @event: event handler for events sent _to_ the device, like EV_LED
58.  *   or EV_SND. The device is expected to carry out the requested
59.  *   action (turn on a LED, play sound, etc.) The call is protected
60.  *   by @event_lock and must not sleep
61.  * @grab: input handle that currently has the device grabbed (via
62.  *   EVIOCGRAB ioctl). When a handle grabs a device it becomes sole
63.  *   recipient for all input events coming from the device
64.  * @event_lock: this spinlock is taken when input core receives
65.  *   and processes a new event for the device (in input_event()).
66.  *   Code that accesses and/or modifies parameters of a device
67.  *   (such as keymap or absmin, absmax, absfuzz, etc.) after device
68.  *   has been registered with input core must take this lock.
69.  * @mutex: serializes calls to open(), close() and flush() methods
```



聚酰亚胺薄膜



联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

👤 QQ客服 🗨 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

```

70.  * @users: stores number of users (input handlers) that opened this
71.  * device. It is used by input_open_device() and input_close_device()
72.  * to make sure that dev->open() is only called when the first
73.  * user opens device and dev->close() is called when the very
74.  * last user closes the device
75.  * @going_away: marks devices that are in a middle of unregistering and
76.  * causes input_open_device*() fail with -ENODEV.
77.  * @sync: set to %true when there were no new events since last EV_SYN
78.  * @dev: driver model's view of this device
79.  * @h_list: list of input handles associated with the device. When
80.  * accessing the list dev->mutex must be held
81.  * @node: used to place the device onto input_dev_list
82.  */
83. struct input_dev {
84.     const char* name;
85.     const char* phys;
86.     const char* uniq;
87.     struct input_id id;
88.
89.     unsigned long propbit[BITS_TO_LONGS(INPUT_PROP_CNT)];
90.
91.     unsigned long evbit[BITS_TO_LONGS(EV_CNT)];
92.     unsigned long keybit[BITS_TO_LONGS(KEY_CNT)];
93.     unsigned long relbit[BITS_TO_LONGS(REL_CNT)];
94.     unsigned long absbit[BITS_TO_LONGS(ABS_CNT)];
95.     unsigned long mscbit[BITS_TO_LONGS(MSC_CNT)];
96.     unsigned long ledbit[BITS_TO_LONGS(LED_CNT)];
97.     unsigned long sndbit[BITS_TO_LONGS(SND_CNT)];
98.     unsigned long ffbbit[BITS_TO_LONGS(FF_CNT)];
99.     unsigned long swbit[BITS_TO_LONGS(SW_CNT)];
100.
101.     unsigned int hint_events_per_packet;
102.
103.     unsigned int keycodemax;
104.     unsigned int keycodesize;
105.     void* keycode;
106.
107.     int (*setkeycode)(struct input_dev *dev,
108.         const struct input_keymap_entry *ke,
109.         unsigned int old_keycode);
110.     int (*getkeycode)(struct input_dev *dev,
111.         struct input_keymap_entry *ke);
112.
113.     struct ff_device *ff;
114.
115.     unsigned int repeat_key;
116.     struct timer_list timer;
117.
118.     int rep[REP_CNT];
119.
120.     struct input_mt_slot *mt;
121.     int mtsize;
122.     int slot;
123.     int trkid;
124.
125.     struct input_absinfo *absinfo;
126.
127.     unsigned long key[BITS_TO_LONGS(KEY_CNT)];
128.     unsigned long led[BITS_TO_LONGS(LED_CNT)];
129.     unsigned long snd[BITS_TO_LONGS(SND_CNT)];
130.     unsigned long sw[BITS_TO_LONGS(SW_CNT)];
131.
132.     int (*open)(struct input_dev *dev);
133.     void (*close)(struct input_dev *dev);
134.     int (*flush)(struct input_dev *dev, struct file *file);
135.     int (*event)(struct input_dev *dev, unsigned int type, unsigned int code, int value);
136.
137.     struct input_handle __rcu *grab;
138.
139.     spinlock_t event_lock;
140.     struct mutex mutex;
141.
142.     unsigned int users;
143.     bool going_away;
144.
145.     bool sync;
146.
147.     struct device dev;
148.
149.     struct list_head h_list;
150.     struct list_head node;
151. };

```

我解释可能还会误导，源码上面的注释是最好的解释，都是描述一个input设备的相关信息。



聚酰亚胺薄膜



联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

💬 QQ客服 🗣 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

每一个input设备，都需要初始化一个这样的input_dev结构来描述记录此设备的一些特性，然input_register_device 注册到设备总线上以供后续使用

可以到系统运行目录的/proc/bus/input下 cat devices 查看总线上的已经注册上的input device

input_event :

设备驱动部分往上传递的就是触发的event事件了，还以usbtouchscreen的为例，回调函数为：

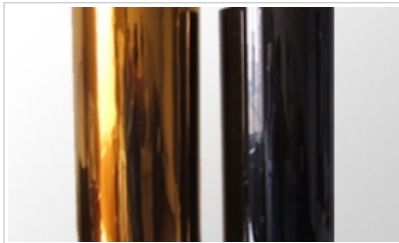
```
[objc]
1.  /*****
2.  * Generic Part
3.  */
4.  static void usbtouch_process_pkt(struct usbtouch_usb *usbtouch,
5.                                  unsigned char *pkt, int len)
6.  {
7.      struct usbtouch_device_info *type = usbtouch->type;
8.
9.      if (!type->read_data(usbtouch, pkt))
10.         return;
11.
12.      input_report_key(usbtouch->input, BTN_TOUCH, usbtouch->touch); // 上报触摸类型 。touch为按下
13.
14.      if (swap_xy) {
15.          input_report_abs(usbtouch->input, ABS_X, usbtouch->y);
16.          input_report_abs(usbtouch->input, ABS_Y, usbtouch->x);
17.      } else {
18.          input_report_abs(usbtouch->input, ABS_X, usbtouch->x);
19.          input_report_abs(usbtouch->input, ABS_Y, usbtouch->y); // 上报绝对坐标值
20.      }
21.      if (type->max_press)
22.          input_report_abs(usbtouch->input, ABS_PRESSURE, usbtouch->press);
23.      input_sync(usbtouch->input); // 同步操作
24.  }
```

可以看到通过 input_report_* 上报事件到input.c中，这也就是上面层次图中的箭头 9 在/kernel/include/linux/input.h :

```
[objc]
1.  static inline void input_report_key(struct input_dev *dev, unsigned int code, int value)
2.  {
3.      input_event(dev, EV_KEY, code, !!value);
4.  }
5.
6.  static inline void input_report_rel(struct input_dev *dev, unsigned int code, int value)
7.  {
8.      input_event(dev, EV_REL, code, value);
9.  }
10.
11.  static inline void input_report_abs(struct input_dev *dev, unsigned int code, int value)
12.  {
13.      input_event(dev, EV_ABS, code, value);
14.  }
```

可以看到不同的report 都调用进了input_event，只是传参不同，接下来的事就全交由input.c 来做了！

```
[objc]
1.  /**
2.  * input_event() - report new input event
3.  * @dev: device that generated the event
4.  * @type: type of the event
5.  * @code: event code
6.  * @value: value of the event
7.  *
8.  * This function should be used by drivers implementing various input
9.  * devices to report input events. See also input_inject_event().
10.  *
11.  * NOTE: input_event() may be safely used right after input device was
12.  * allocated with input_allocate_device(), even before it is registered
13.  * with input_register_device(), but the event will not reach any of the
```



聚酰亚胺薄膜



联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

💬 QQ客服 🗨 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

Linux/Android——输入子系统input_event传递 (二) - CSDN博客

```

14.  * input handlers. Such early invocation of input_event() may be used
15.  * to 'seed' initial state of a switch or initial position of absolute
16.  * axis, etc.
17.  */
18. void input_event(struct input_dev *dev,
19.                 unsigned int type, unsigned int code, int value)
20. {
21.     unsigned long flags;
22.
23.     if (is_event_supported(type, dev->evbit, EV_MAX)) { //判断是否是注册时的event类型，驱动probe时注册input
        //时设置了能响应的event类型
24.
25.         spin_lock_irqsave(&dev->event_lock, flags); //自旋锁枷锁
26.
27.         add_input_randomness(type, code, value);
28.         input_handle_event(dev, type, code, value); //进一步处理传上来的这个 event
29.         spin_unlock_irqrestore(&dev->event_lock, flags); //解锁
30.     }
31. }

```

可以看到在这里首先就是过滤了事件类型，这个也是在usbtouchscreen中的probe中初始化过的！

类型有如下几种：

```

[objc]
1.  /*
2.  * Event types
3.  */
4.
5.  #define EV_SYN          0x00
6.  #define EV_KEY          0x01
7.  #define EV_REL          0x02
8.  #define EV_ABS          0x03
9.  #define EV_MSC          0x04
10. #define EV_SW           0x05
11. #define EV_LED           0x11
12. #define EV_SND           0x12
13. #define EV_REP           0x14
14. #define EV_FF            0x15
15. #define EV_PWR           0x16
16. #define EV_FF_STATUS     0x17
17. #define EV_MAX           0x1f
18. #define EV_CNT           (EV_MAX+1)

```

input_handle_event：

由上面的input_event 调入进这个handle处理。这里会根据type进行分类处理：

```

[objc]
1. static void input_handle_event(struct input_dev *dev,
2.                               unsigned int type, unsigned int code, int value)
3. {
4.     int disposition = INPUT_IGNORE_EVENT; //初始为不做处理
5.
6.     switch (type) {
7.
8.     case EV_SYN:
9.         switch (code) {
10.            case SYN_CONFIG:
11.                disposition = INPUT_PASS_TO_ALL;
12.                break;
13.
14.            case SYN_REPORT:
15.                if (!dev->sync) {
16.                    dev->sync = true;
17.                    disposition = INPUT_PASS_TO_HANDLERS;
18.                }
19.                break;
20.
21.            ...
22.
23.            case EV_KEY:
24.
25.                if (is_event_supported(code, dev->keybit, KEY_MAX) && //按键code是否被keybit支持

```



聚酰亚胺薄膜



联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

💬 QQ客服 💬 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心


```
26.         !!test_bit(code, dev->key) != value) { //key是键盘当前所有键状态，测试code对应键状态，value传
按键状态。此句表示按键状态应有变化
27.
28.         if (value != 2) {
29.             __change_bit(code, dev->key); //改变key的值以改变按键状态。
30.             if (value)
31.                 input_start_autorepeat(dev, code); //如果按键值为按下，则开始重复按键操作。具体会不会重
ut_start_autorepeat还会根据evbit中有没有置位重复事件等判断。
32.             else
33.                 input_stop_autorepeat(dev); //如果是松开按键则应停止重复按键相关操作。
34.         }
35.
36.         disposition = INPUT_PASS_TO_HANDLERS;
37.     }
38.     break;
39.
40. ...
41.
42.     case EV_ABS:
43.         if (is_event_supported(code, dev->absbit, ABS_MAX)) //同上面一样看是否支持
44.             disposition = input_handle_abs_event(dev, code, &value); //这个函数可以跟进去看，是做筛选的
是不会返回INPUT_IGNORE_EVENT，后面如果有跟上次相同的ABS坐标就会被过滤掉，返回IGNORE
45.         // err("jscese display disposition vlue ==0x%x,code==0x%x, value== 0x%x\n",disposition,code,valu
break;
46.
47.
48. ...
49.
50.     }
51.
52.     if (disposition != INPUT_IGNORE_EVENT && type != EV_SYN)
53.         dev->sync = false;
54.
55.     if ((disposition & INPUT_PASS_TO_DEVICE) && dev->event)
56.         dev->event(dev, type, code, value);
57.
58.     if (disposition & INPUT_PASS_TO_HANDLERS)
59.         input_pass_event(dev, type, code, value); //更深一步调用，最终都是 调用到 event(**)方法
60.
61. }
```

这里先记录整个输入系统从设备驱动到上层的关系，以及从kernel中的驱动调用到input系统中的传递过程看到调用了input.c中的一些函数传递，但是对input核心还是没多少概念，

下篇解析记录一下input这个核心模块~

- [上一篇](#) 同步 与 异步
- [下一篇](#) Linux/Android——input子系统核心 (三)





写下你的评论...



聚酰亚胺薄膜



联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

🗣 QQ客服 🗣 客服论坛

关于 招聘 广告服务  百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

Linux/Android——输入子系统input_event传递 (二) - CSDN博客

input子系统详解

 ylyuanlu 2011-08-20 18:30:36 

Input子系统详解 一 . Input子系统架构 Linux系统提供了input子系统，按键、触摸屏、键盘、鼠标等输入都可以利用input函数来实现设备驱动，下面是Input子系统架构： In...

input子系统 KeyPad-Touch上报数据格式与机制

----- 本文系本站原创,欢迎转载! 转载请注明出处:http...

 android_huber 2012-04-17 14:48:55  9790

Linux之解析鼠标input事件数据

 qq_21792169 2016-03-05 17:03:03 

Linux中USB鼠标驱动： http://blog.csdn.net/qq_21792169/article/details/48790745 或者直接Linux自带的USB鼠标驱动

input 子系统架构总结

 lbmygf 2012-03-21 16:57:05 

Linux输入子系统(Input Subsystem) Linux 的输入子系统不仅支持鼠标、键盘等常规输入设备,而且还支持蜂鸣器、设备。本章将对 Linux 输 ...

分析Power key的处理流程

 bmj 2013-05-25 10:34:54 

<http://blog.csdn.net/pillarbuaa/article/details/76345910rks/base/policy/src/com/android/internal/policy>

Linux--内核Uevent事件机制 与 Input子系统

 wh8_2011 2016-06-15 07:36:50 

一、Uevent机制 1.前提摘要 （1）Sysfs文件系统 内核设备模型主要的模块和用户之间能看到的相关部分就是sysfs了。内核在启动的时候会注册sys...

input子系统 事件流程浅析

 leesagacious 2015-12-05 23:44:32 

事件(struct input_event)从设备驱动层 -> 核心层->事件处理层的经过 struct input_event { struct timeval time; //事件...

input子系统四 input事件处理

 coldsnow33 2013-10-18 17:24:34 

input事件处理流程 input driver -> input core ->event handler -> userspace 给应用程序。 一 事件分发跟踪 核心层的上报接口是i...

Input event 分析

 hello_yj 2015-04-15 13:30:20 

以gsensor器件为例：1、Gsensor上报ABS_DISTANCE后，接着上报sync事件，当event core 检测到sync事件时，认为nt数据完成【参考下文代码得出】。 ...

技术外文文献看不懂？教你一个公式秒懂英语

不背单词和语法，一个公式学好英语



Linux--内核Uevent事件机制 与 Input子系统

 lx1584685501 2015-06-05 17:05:13 

一、Uevent机制 1.前提摘要 （1）Sysfs文件系统 内核设备模型主要的模块和用户之间能看到的相关部分就是sysfs了。内核在启动的时候会注册sysfs文件系统，并...

<https://blog.csdn.net/jscese/article/details/42099381>

7/8



聚酰亚胺薄膜



联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

👤 QQ客服 💬 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

input子系统框架分析

duan_xiaosu 2017-03-30 15:47:50

1. input框架介绍：Linux input子系统主要分为三层：驱动、输入core、事件处理层。驱动根据core提供的接口，向上... 的动作 (input_report_**)。cor...

linux input 子系统分析 三

YAOZHENGUO2006 2011-09-14 19:48:59

linux input子系统分析--子系统核心.事件处理层.事件传递过程 一. 输入子系统核心分析。 1.输入子系统核心对应与/d... put/input.c文件,这个也...

struct input_event详解

bingqingsuimeng 2012-11-13 12:57:23

查看/dev/input/eventX是什么类型的事件，cat /proc/bus/input/devices 设备有着自己特殊的按键键码，我需要将一... 按键，比如0 - 9，X - Z等模拟...

开发一个app多少钱

开发一个手机app需要多少钱 费用

百度广告



input_event结构体详解

liwei405499 2014-12-19 10:47:15

关键结构体input_event信息：struct input_event { struct timeval time; __u16 type; __u16 code; __s32 val...

struct input_event结构体定义

g47020055 2015-09-15 17:54:40

查看/dev/input/eventX是什么类型的事件，cat /proc/bus/input/devices设备有着自己特殊的按键键码，我需要将一... 按键，比如0 - 9，X - Z等模拟成标准按键，...

Linux C中读取/dev/input/event设备来判断键盘按键是否按下

code：#include #include #include #include #include #include #include #define DEV_PATH "/dev/input/eve...

 zgrjkflmkyc 2014-09-23 15:49:37 14614

input设备使用方法和input_event说明

newtonnl 2013-04-24 00:15:25

1.定义的结构体继承input_devstruct bma150_data { struct i2c_client *bma150_client; struct bma150_platform_d...

技术外文文献看不懂？教你一个公式秒懂英语

不背单词和语法，一个公式学好英语



linux键盘input_event浅析

tdstds 2014-01-23 17:39:20

input_event(mxckbd_dev, EV_KEY, mxckpd_keycodes[scancode], 0); void input_event(struct input_dev ...

深入理解Android输入系统

u010312937 2016-11-22 14:09:21

《深入理解Android 卷III》即将发布，作者是张大伟。此书填补了深入理解Android Framework卷中的一个主要空白，F... d Framework中和UI相关的部分。在一个特别讲...

https://blog.csdn.net/jscese/article/details/42099381

8/8