

原

Android Input流程分析（三）：InputReader

2017年09月15日 01:30:18

Invoker123

阅读数：765

回到InputReader的loopOnce函数。
现在getEvents捞上来的RawEvent均保存在mEventBuffer中。

/native/services/inputflinger/InputReader.cpp

```
1 void InputReader::loopOnce() {
2     int32_t oldGeneration;
3     int32_t timeoutMillis;
4     bool inputDevicesChanged = false;
5     Vector<InputDeviceInfo> inputDevices;
6     { // acquire lock
7         AutoMutex _l(mLock);
8
9         oldGeneration = mGeneration;
10        timeoutMillis = -1;
11
12        uint32_t changes = mConfigurationChangesToRefresh;
13        if (changes) {
14            mConfigurationChangesToRefresh = 0;
15            timeoutMillis = 0;
16            refreshConfigurationLocked(changes);
17        } else if (mNextTimeout != LLONG_MAX) {
18            nsecs_t now = systemTime(SYSTEM_TIME_MONOTONIC);
19            timeoutMillis = toMillisecondTimeoutDelay(now, mNextTimeout);
20        }
21    } // release lock
22
23    size_t count = mEventHub->getEvents(timeoutMillis, mEventBuffer, EVENT_BUFFER_SIZE);
24
25    { // acquire lock
26        AutoMutex _l(mLock);
27        mReaderIsAliveCondition.broadcast();
28
29        if (count) {
30            processEventsLocked(mEventBuffer, count);
31        }
32
33        if (mNextTimeout != LLONG_MAX) {
34            nsecs_t now = systemTime(SYSTEM_TIME_MONOTONIC);
35            if (now >= mNextTimeout) {
36                #if DEBUG_RAW_EVENTS
37                    ALOGD("Timeout expired, latency=%0.3fms", (now - mNextTimeout) * 0.000001f);
38                #endif
39                mNextTimeout = LLONG_MAX;
40                timeoutExpiredLocked(now);
41            }
42        }
43    }
44
45    if (oldGeneration != mGeneration) {
46        inputDevicesChanged = true;
47        getInputDevicesLocked(inputDevices);
48    }
49    } // release lock
50
51    // Send out a message that the describes the changed input devices.
52    if (inputDevicesChanged) {
53        mPolicy->notifyInputDevicesChanged(inputDevices);
54    }
55
56    // Flush queued events out to the listener.
57    // This must happen outside of the lock because the listener could potentially call
58    // back into the InputReader's methods, such as getScanCodeState, or become blocked
59    // on another thread similarly waiting to acquire the InputReader lock thereby
60    // resulting in a deadlock. This situation is actually quite plausible because the
61    // listener is actually the input dispatcher, which calls into the window manager,
62    // which occasionally calls into the input reader.
63    mQueuedListener->flush();
```



0

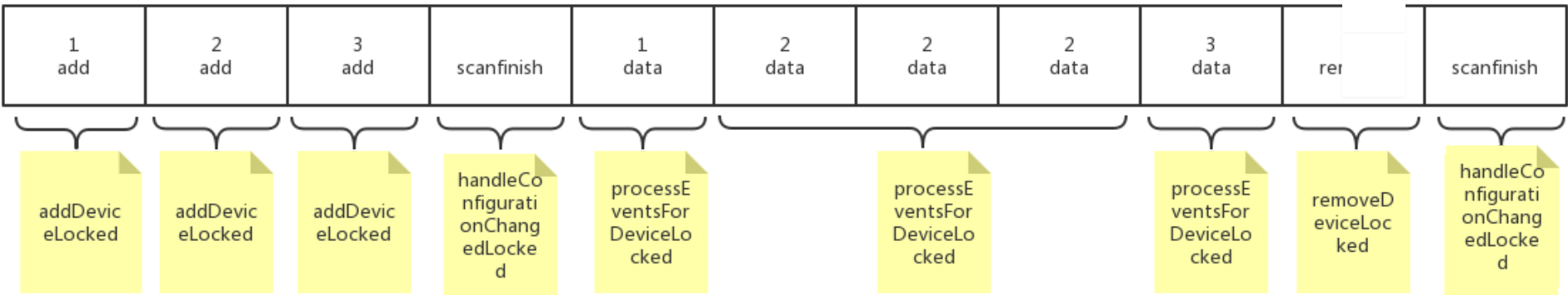


timeoutMillis决定了getEvent时poll的等待时间，默认为-1（poll会一直阻塞到有事件发生），若设备配置改变，则为0（立刻返回），键盘事件处理过程中mNextTimeout不是-1就是0，不会出现timeoutMillis为now和mNextTimeout时间差的情况。

遍历mEventBuffer中所有的RawEvent。

若发生的事件为设备增删事件（rawEvent[batchSize].type >= EventHubInterface::FIRST_SYNTHETIC_EVENT），则逐一地处理事件，处理完一件，count减1，rawEvent指针向前移动一格。若发生的是设备传送数据事件，则分批收集属于同一设备的数据传送事件去处理。

下图是mEventBuffer数组中一段的可能分布。对于设备增加，卸载和扫描完成事件，都是单独使用独立的函数进行处理的。对于设备数据事件，以设备id相同的连续RawEvent构成一个数组作为processEventsForDeviceLocked的rawEvent参数传入，数组的元素数量作为参数batchSize传入。



<http://blog.csdn.net/Invoker123>

```
1 void InputReader::processEventsLocked(const RawEvent* rawEvents, size_t count) {
2     for (const RawEvent* rawEvent = rawEvents; count;) {
3         int32_t type = rawEvent->type;
4         size_t batchSize = 1;
5         if (type < EventHubInterface::FIRST_SYNTHETIC_EVENT) {
6             int32_t deviceId = rawEvent->deviceId;
7             while (batchSize < count) {
8                 if (rawEvent[batchSize].type >= EventHubInterface::FIRST_SYNTHETIC_EVENT
9                     || rawEvent[batchSize].deviceId != deviceId) {
10                     break;
11                 }
12                 batchSize += 1;
13             }
14             #if DEBUG_RAW_EVENTS
15                 ALOGD("BatchSize: %d Count: %d", batchSize, count);
16             #endif
17             processEventsForDeviceLocked(deviceId, rawEvent, batchSize);
18         } else {
19             switch (rawEvent->type) {
20                 case EventHubInterface::DEVICE_ADDED:
21                     addDeviceLocked(rawEvent->when, rawEvent->deviceId);
22                     break;
23                 case EventHubInterface::DEVICE_REMOVED:
24                     removeDeviceLocked(rawEvent->when, rawEvent->deviceId);
25                     break;
26                 case EventHubInterface::FINISHED_DEVICE_SCAN:
27                     handleConfigurationChangedLocked(rawEvent->when);
28                     break;
29                 default:
30                     ALOG_ASSERT(false); // can't happen
31                     break;
32             }
33         }
34         count -= batchSize;
35         rawEvent += batchSize;
36     }
37 }
```

第一次进入processEventsLocked函数走的必然是设备增加事件处理流程。首先获得EventHub::Device对应的InputDeviceIdentifier，classes和controllerNumber等成员信息来构造一个InputDevice。

/native/services/inputflinger/InputReader.cpp

```
1 void InputReader::addDeviceLocked(nsecs_t when, int32_t deviceId) {
2     ssize_t deviceIndex = mDevices.indexOfKey(deviceId);
3     if (deviceIndex >= 0) {
4         ALOGW("Ignoring spurious device added event for deviceId %d.", deviceId);
5         return;
6     }
7
8     InputDeviceIdentifier identifier = mEventHub->getDeviceIdentifier(deviceId);
```

```
11
12     InputDevice* device = createDeviceLocked(deviceId, controllerNumber, identifier, classes);
13     device->configure(when, &mConfig, 0);//配置Device的策略
14     device->reset(when);//重置Device
15
16     if (device->isIgnored()) {
17         ALOGI("Device added: id=%d, name='%s' (ignored non-input device)", deviceId,
18             identifier.name.string());
19     } else {
20         ALOGI("Device added: id=%d, name='%s', sources=0x%08x", deviceId,
21             identifier.name.string(), device->getSources());
22     }
23
24     mDevices.add(deviceId, device);//添加id-InputDevice*键值对到mDevices中
25     bumpGenerationLocked();
26
27     if (device->getClasses() & INPUT_DEVICE_CLASS_EXTERNAL_STYLUS) {
28         notifyExternalStylusPresenceChanged();
29     }
30 }
```

0

/native/services/inputflinger/InputReader.cpp

```
1 InputDevice* InputReader::createDeviceLocked(int32_t deviceId, int32_t controllerNumber,
2     const InputDeviceIdentifier& identifier, uint32_t classes) {
3     InputDevice* device = new InputDevice(&mContext, deviceId, bumpGenerationLocked(),
4         controllerNumber, identifier, classes);
5     ...
6     // Keyboard-like devices.
7     uint32_t keyboardSource = 0;
8     int32_t keyboardType = AINPUT_KEYBOARD_TYPE_NON_ALPHABETIC;
9     if (classes & INPUT_DEVICE_CLASS_KEYBOARD) {
10         keyboardSource |= AINPUT_SOURCE_KEYBOARD;
11     }
12     ...
13     if (keyboardSource != 0) {
14         device->addMapper(new KeyboardInputMapper(device, keyboardSource, keyboardType));
15     }
16     ...
17     return device;
18 }
```

InputDevice中有一个类型为Vector

```
1 void InputDevice::addMapper(InputMapper* mapper) {
2     mMappers.add(mapper);
3 }
```

这样便完成了设备的增加事件处理流程。接下来讲下事件传送事件的处理（先跳过扫描完成事件）。

/native/services/inputflinger/InputReader.cpp

```
1 void InputReader::processEventsForDeviceLocked(int32_t deviceId,
2     const RawEvent* rawEvents, size_t count) {
3     ssize_t deviceIndex = mDevices.indexOfKey(deviceId);
4     if (deviceIndex < 0) {
5         ALOGW("Discarding event for unknown deviceId %d.", deviceId);
6         return;
7     }
8
9     InputDevice* device = mDevices.valueAt(deviceIndex);
10    if (device->isIgnored()) {
11        //ALOGD("Discarding event for ignored deviceId %d.", deviceId);
12        return;
13    }
14
15    device->process(rawEvents, count);
16 }
```

processEventsForDeviceLocked中调用了InputDevice::process函数。这个函数中对每一个RawEvent,都会调用mMappers中的每一个InputMapper的process函数去处理它。

/native/services/inputflinger/InputReader.cpp

```

3      // We cannot simply ask each mapper to process them in bulk because mappers may
4      // have side-effects that must be interleaved. For example, joystick movement events and
5      // gamepad button presses are handled by different mappers but they should be dispatched
6      // in the order received.
7      size_t numMappers = mMappers.size();
8      for (const RawEvent* rawEvent = rawEvents; count--; rawEvent++) {
9  #if DEBUG_RAW_EVENTS
10         ALOGD("Input event: device=%d type=0x%04x code=0x%04x value=0x%08x when=%lld",
11             rawEvent->deviceId, rawEvent->type, rawEvent->code, rawEvent->value,
12             rawEvent->when);
13     #endif
14
15         if (mDropUntilNextSync) {
16             if (rawEvent->type == EV_SYN && rawEvent->code == SYN_REPORT) {
17                 mDropUntilNextSync = false;
18     #if DEBUG_RAW_EVENTS
19                 ALOGD("Recovered from input event buffer overrun.");
20     #endif
21             } else {
22     #if DEBUG_RAW_EVENTS
23                 ALOGD("Dropped input event while waiting for next input sync.");
24     #endif
25             }
26         } else if (rawEvent->type == EV_SYN && rawEvent->code == SYN_DROPPED) {
27             ALOGI("Detected input event buffer overrun for device %s.", getName().string());
28             mDropUntilNextSync = true;
29             reset(rawEvent->when);
30         } else {
31             for (size_t i = 0; i < numMappers; i++) {
32                 InputMapper* mapper = mMappers[i];
33                 mapper->process(rawEvent);
34             }
35         }
36     }
37 }

```

0

以KeyboardInputMapper为例进行介绍。在前面可以看到，要拥有KeyboardInputMapper，设备必须要属于这四个类型：INPUT_DEVICE_CLASS_KEYBOARD，INPUT_DEVICE_CLASS_ALPHAKEY，INPUT_DEVICE_CLASS_DPAD和INPUT_DEVICE_CLASS_GAMEPAD。如果原始输入事件类型为EV_KEY，先检查是否是键盘或游戏手柄的扫描码，如果是，则使用mapkey获得对应的键值码和flag。这些值将作为processKey的入参做最后的处理。

/native/services/inputflinger/InputReader.cpp

```

1 void KeyboardInputMapper::process(const RawEvent* rawEvent) {
2     switch (rawEvent->type) {
3     case EV_KEY: {
4         int32_t scanCode = rawEvent->code;//扫描码
5         int32_t usageCode = mCurrentHidUsage;
6         mCurrentHidUsage = 0;
7
8         if (isKeyboardOrGamepadKey(scanCode)) {
9             int32_t keyCode;
10            uint32_t flags;
11            if (getEventHub()->mapKey(getDeviceId(), scanCode, usageCode, &keyCode, &flags)) {
12                keyCode = AKEYCODE_UNKNOWN;
13                flags = 0;
14            }
15            processKey(rawEvent->when, rawEvent->value != 0, keyCode, scanCode, flags);
16        }
17        break;
18    }
19    case EV_MSC: {
20        if (rawEvent->code == MSC_SCAN) {
21            mCurrentHidUsage = rawEvent->value;
22        }
23        break;
24    }
25    case EV_SYN: {
26        if (rawEvent->code == SYN_REPORT) {
27            mCurrentHidUsage = 0;
28        }
29    }
30    }
31 }
32 }

```

KeyLayoutMap::mapKey函数根据传入的扫描码 (scanCode) ,找到对应的键值码 , 结果存放在outKeycode中 , outFlags表示的标志位会被设为KeyLayoutMap中保存的flag。

/frameworks/native/services/inputflinger/EventHub.cpp

```
1  status_t EventHub::mapKey(int32_t deviceId, int32_t scanCode, int32_t usageCode,
2      int32_t* outKeycode, uint32_t* outFlags) const {
3      AutoMutex _l(mLock);
4      Device* device = getDeviceLocked(deviceId);
5
6      if (device) {
7          // Check the key character map first.
8          sp<KeyCharacterMap> kcm = device->getKeyCharacterMap();//获得保存的KeyCharacterMap
9          if (kcm != NULL) {
10             if (!kcm->mapKey(scanCode, usageCode, outKeycode)) {
11                 *outFlags = 0;
12                 return NO_ERROR;
13             }
14         }
15
16         // Check the key layout next.
17         if (device->keyMap.haveKeyLayout()) //获得保存的keyLayoutMap
18             if (!device->keyMap.keyLayoutMap->mapKey(
19                 scanCode, usageCode, outKeycode, outFlags)) {
20                 return NO_ERROR;
21             }
22         }
23     }
24
25     *outKeycode = 0;
26     *outFlags = 0;
27     return NAME_NOT_FOUND;
28 }
29 }
```

/frameworks/native/libs/input/KeyCharacterMap.cpp

```
1  status_t KeyCharacterMap::mapKey(int32_t scanCode, int32_t usageCode, int32_t* outKeyCode) const {
2      if (usageCode) {
3          ssize_t index = mKeysByUsageCode.indexOfKey(usageCode);
4          if (index >= 0) {
5              #if DEBUG_MAPPING
6              ALOGD("mapKey: scanCode=%d, usageCode=0x%08x ~ Result keyCode=%d.",
7                  scanCode, usageCode, *outKeyCode);
8              #endif
9              *outKeyCode = mKeysByUsageCode.valueAt(index);
10             return OK;
11         }
12     }
13     if (scanCode) {
14         ssize_t index = mKeysByScanCode.indexOfKey(scanCode);
15         if (index >= 0) {
16             #if DEBUG_MAPPING
17             ALOGD("mapKey: scanCode=%d, usageCode=0x%08x ~ Result keyCode=%d.",
18                 scanCode, usageCode, *outKeyCode);
19             #endif
20             *outKeyCode = mKeysByScanCode.valueAt(index);
21             return OK;
22         }
23     }
24
25     #if DEBUG_MAPPING
26     ALOGD("mapKey: scanCode=%d, usageCode=0x%08x ~ Failed.", scanCode, usageCode);
27     #endif
28     *outKeyCode = AKEYCODE_UNKNOWN;
29     return NAME_NOT_FOUND;
30 }
31 }
```

/frameworks/native/libs/input/KeyLayoutMap.cpp

```
1  status_t KeyLayoutMap::mapKey(int32_t scanCode, int32_t usageCode,
2      int32_t* outKeyCode, uint32_t* outFlags) const {
3      const Key* key = getKey(scanCode, usageCode);
4      if (!key) {
5          #if DEBUG_MAPPING
6          ALOGD("mapKey: scanCode=%d, usageCode=0x%08x ~ Failed.", scanCode, usageCode);
```

0

```
9      *outFlags = 0;
10     return NAME_NOT_FOUND;
11 }
12
13 *outKeyCode = key->keyCode;
14 *outFlags = key->flags;
15
16 #if DEBUG_MAPPING
17     ALOGD("mapKey: scanCode=%d, usageCode=0x%08x ~ Result keyCode=%d, outFlags=0x%08x.",
18           scanCode, usageCode, *outKeyCode, *outFlags);
19 #endif
20     return NO_ERROR;
21 }
22
23 const KeyLayoutMap::Key* KeyLayoutMap::getKey(int32_t scanCode, int32_t usageCode) const {
24     if (usageCode) {
25         ssize_t index = mKeysByUsageCode.indexOfKey(usageCode);
26         if (index >= 0) {
27             return &mKeysByUsageCode.valueAt(index);
28         }
29     }
30     if (scanCode) {
31         ssize_t index = mKeysByScanCode.indexOfKey(scanCode);
32         if (index >= 0) {
33             return &mKeysByScanCode.valueAt(index);
34         }
35     }
36     return NULL;
37 }
38 }
```

0

来看看processKey。down的值为RawEvent的code值，若不为0，表示该事件为按键按下的事件。如果是按下事件，首先根据屏幕的方向对键码值进行旋转变换。findKeyDown从 InputReader的Vector< KeyDown>类型成员mKeyDowns查找与scanCode对应的KeyDown。KeyDown是一个结构体，持有扫描码scanCode和对应的键值码keyCode两个成员。如果mKeyDowns里面找到了对应的KeyDown，说明发生了重复按键事件，使用这个KeyDown中的键码值。如果mKeyDowns里面没有找到对应的KeyDown，则在mKeyDowns中插入一个新的KeyDown。进入processKey函数时，mKeyDowns不一定为空，因为此时有其他的按键已经被按下还没释放。

当按键抬起时，则在mKeyDowns中移除对应的KeyDown。

/native/services/inputflinger/InputReader.cpp

```
1 void KeyboardInputMapper::processKey(nsecs_t when, bool down, int32_t keyCode,
2   int32_t scanCode, uint32_t policyFlags) {
3
4     if (down) {
5         // Rotate key codes according to orientation if needed.
6         if (mParameters.orientationAware && mParameters.hasAssociatedDisplay) {
7             keyCode = rotateKeyCode(keyCode, mOrientation);//根据屏幕的方向对键码值进行旋转变换
8         }
9
10        // Add key down.
11        ssize_t keyDownIndex = findKeyDown(scanCode);
12        if (keyDownIndex >= 0) {
13            // key repeat, be sure to use same keycode as before in case of rotation
14            keyCode = mKeyDowns.itemAt(keyDownIndex).keyCode;
15        } else {
16            // key down
17            if ((policyFlags & POLICY_FLAG_VIRTUAL)
18                && mContext->shouldDropVirtualKey(when,
19                                                    getDevice(), keyCode, scanCode)) {
20                return;
21            }
22            if (policyFlags & POLICY_FLAG_GESTURE) {
23                mDevice->cancelTouch(when);
24            }
25
26            mKeyDowns.push();
27            KeyDown& keyDown = mKeyDowns.editTop();
28            keyDown.keyCode = keyCode;
29            keyDown.scanCode = scanCode;
30        }
31
32        mDownTime = when;
33    } else {
34        // Remove key down.
35        ssize_t keyDownIndex = findKeyDown(scanCode);
```



```
40         keyCode = mKeyDowns.itemAt(keyDownIndex).keyCode;
41         mKeyDowns.removeAt(size_t(keyDownIndex));
42     } else {
43         // key was not actually down
44         ALOGI("Dropping key up from device %s because the key was not down. "
45             "keyCode=%d, scanCode=%d",
46             getDeviceName().string(), keyCode, scanCode);
47         return;
48     }
49 }
50
51 int32_t oldMetaState = mMetaState;
52 int32_t newMetaState = updateMetaState(keyCode, down, oldMetaState);
53 bool metaStateChanged = oldMetaState != newMetaState;
54 if (metaStateChanged) {
55     mMetaState = newMetaState;
56     updateLedState(false);
57 }
58
59 nsecs_t downTime = mDownTime;
60
61 // Key down on external an keyboard should wake the device.
62 // We don't do this for internal keyboards to prevent them from waking up in your pocket.
63 // For internal keyboards, the key layout file should specify the policy flags for
64 // each wake key individually.
65 // TODO: Use the input device configuration to control this behavior more finely.
66 if (down && getDevice()->isExternal()) {
67     policyFlags |= POLICY_FLAG_WAKE;
68 }
69
70
71 if (mParameters.handlesKeyRepeat) {
72     policyFlags |= POLICY_FLAG_DISABLE_KEY_REPEAT;
73 }
74
75 if (metaStateChanged) {
76     getContext()->updateGlobalMetaState();
77 }
78
79 if (down && !isMetaKey(keyCode)) {
80     getContext()->fadePointer();
81 }
82
83 NotifyKeyArgs args(when, getDeviceId(), mSource, policyFlags,
84     down ? AKEY_EVENT_ACTION_DOWN : AKEY_EVENT_ACTION_UP,
85     AKEY_EVENT_FLAG_FROM_SYSTEM, keyCode, scanCode, newMetaState, downTime);
86 getListener()->notifyKey(&args);
87 }
```

0

/frameworks/native/services/inputflinger/InputReader.h

```
1 struct KeyDown {
2     int32_t keyCode;
3     int32_t scanCode;
4 };
```

之后根据旧的metaState获得新的metaState。若某个键码值对应的键被按下，newMetaState对应的的标志位置1。若该键被释放，则取消AMETA_ALT_ON，AMETA_SHIFT_ON，AMETA_CTRL_ON，AMETA_META_ON和原来按下时设定的标志位。

/frameworks/native/libs/input/Keyboard.cpp

```
1 int32_t updateMetaState(int32_t keyCode, bool down, int32_t oldMetaState) {
2     int32_t mask;
3     switch (keyCode) {
4     case AKEYCODE_ALT_LEFT:
5         return setEphemeralMetaState(AMETA_ALT_LEFT_ON, down, oldMetaState);
6     case AKEYCODE_ALT_RIGHT:
7         return setEphemeralMetaState(AMETA_ALT_RIGHT_ON, down, oldMetaState);
8     case AKEYCODE_SHIFT_LEFT:
9         return setEphemeralMetaState(AMETA_SHIFT_LEFT_ON, down, oldMetaState);
10    case AKEYCODE_SHIFT_RIGHT:
11        return setEphemeralMetaState(AMETA_SHIFT_RIGHT_ON, down, oldMetaState);
12    case AKEYCODE_SYM:
13        return setEphemeralMetaState(AMETA_SYM_ON, down, oldMetaState);
14    case AKEYCODE_FUNCTION:
15
```

```
18     case AKEYCODE_CTRL_RIGHT:
19         return setEphemeralMetaState(AMETA_CTRL_RIGHT_ON, down, oldMetaState);
20     case AKEYCODE_META_LEFT:
21         return setEphemeralMetaState(AMETA_META_LEFT_ON, down, oldMetaState);
22     case AKEYCODE_META_RIGHT:
23         return setEphemeralMetaState(AMETA_META_RIGHT_ON, down, oldMetaState);
24     case AKEYCODE_CAPS_LOCK:
25         return toggleLockedMetaState(AMETA_CAPS_LOCK_ON, down, oldMetaState);
26     case AKEYCODE_NUM_LOCK:
27         return toggleLockedMetaState(AMETA_NUM_LOCK_ON, down, oldMetaState);
28     case AKEYCODE_SCROLL_LOCK:
29         return toggleLockedMetaState(AMETA_SCROLL_LOCK_ON, down, oldMetaState);
30     default:
31         return oldMetaState;
32     }
33 }
```

0

/frameworks/native/libs/input/Keyboard.cpp

```
1 static int32_t setEphemeralMetaState(int32_t mask, bool down, int32_t oldMetaState) {
2     int32_t newMetaState;
3     if (down) {
4         newMetaState = oldMetaState | mask;
5     } else {
6         newMetaState = oldMetaState &
7             ~(mask | AMETA_ALT_ON | AMETA_SHIFT_ON | AMETA_CTRL_ON | AMETA_META_ON);
8     }
9
10    if (newMetaState & (AMETA_ALT_LEFT_ON | AMETA_ALT_RIGHT_ON)) {
11        newMetaState |= AMETA_ALT_ON;
12    }
13
14    if (newMetaState & (AMETA_SHIFT_LEFT_ON | AMETA_SHIFT_RIGHT_ON)) {
15        newMetaState |= AMETA_SHIFT_ON;
16    }
17
18    if (newMetaState & (AMETA_CTRL_LEFT_ON | AMETA_CTRL_RIGHT_ON)) {
19        newMetaState |= AMETA_CTRL_ON;
20    }
21
22    if (newMetaState & (AMETA_META_LEFT_ON | AMETA_META_RIGHT_ON)) {
23        newMetaState |= AMETA_META_ON;
24    }
25    return newMetaState;
26 }
```

最后一步，将一些信息封装成一个NotifyKeyArgs对象。getListener函数返回的是一个InputDispatcher对象，调用其notifyKey函数。至此，流程走到了InputDispatcher这边。



想对作者说点什么

转载 **Android Input**子系统：**Input**事件的产生、读取和分发，**InputReader**、**InputDispatcher** 阅读数 364

EventHub:InputManagerService:在上一篇博文AndroidInput子系统：Input进程的创建，监听线程的启动中，我们... [博文](#) 来自：[依然怡然](#)

Android 输入系统（三）InputReader 阅读数 1万+

在之前，整理了输入系统服务InputManagerService（服务的生成、启动），EventHub（读取原始输入事件、设备... [博文](#) 来自：[奋斗的菜鸟ing](#)

Android 输入系统之InputReader篇 阅读数 3501

Androidinput系列文章 [博文](#) 来自：[愿景的博客](#)

Android输入子系统之InputReader读取键盘消息过程分析 阅读数 853

InputReader读取键盘消息过程分析在Android输入子系统之启动过程分析中，InputManagerService启动之后，会... [博文](#) 来自：[chenweiaiyanan...](#)

在 **Android 4.4.4** 上，**分析 input -- android framework** 部分 阅读数 1万+

安卓input子系统是通过事件管道通过系统的各个层的。在最低层，物理输入设备产生了描述的状态变化的信号，如... [博文](#) 来自：[快乐&&平凡](#)

Inputreader.cpp里virtualkey的流程 阅读数 676

首先是voidTouchInputMapper::process(constRawEvent*rawEvent){ if(rawEvent->type==EV_SYN&&rawEve... [博文](#) 来自：[u013308744的专栏](#)

Android input处理机制（一）InputReader

本文主要总结了android源码InputReader处理消息的机制，本简要说明了新老android源码在此方面的差异。...

博文 来自： Ron的专栏

阅读数 2272

0

Android Input流程分析(三):InputReader - Invoker123..._CSDN博客

11-9

Android 输入系统(三)InputReader - 奋斗的菜鸟ing - CSDN博客

3-22

LimitInput.cpp

限制输入一定长度的字符，可以实现控制台限制输入一定长度的字符。

03-04

下载

Android Framework Input 机制分析

阅读数 827

App进程的Java层的ViewRoot对象，请求与底层建立通信，通过Binder机制调用WindowManagerService|进而转...

博文 来自： Ron的专栏

Android Input Framework(三)---InputReader&InputDisp..._CSDN博客

3-13

Android Input流程分析(三):InputReader 09-15 阅读数 662 回到InputReader的...博文 来自: Invoker123的博客 Android input子系统之InputReader获取输入事件详细...

Android input子系统之InputReader获取输入事件详细分..._CSDN博客

11-9

Android Input流程分析(三):InputReader - invoker123的博客(新手的一点见解) 09-15 431 回到InputReader的loopOnce函数。

现在getEvents捞上来的RawEven...



小码哥_WS
135篇文章
排名:千里之外

关注



江湖人称小白哥
99篇文章
排名:5000+

关注



慢慢的燃烧
1354篇文章
排名:642

关注



frank_zyp
52篇文章
排名:千里之外

关注

Android Input流程分析（二）：EventHub

阅读数 563

AndroidInput流程分析（二）：EventHub

现在，InputReader线程已经开始运行

博文 来自： Invoker123的博客

Android input处理机制(一)InputReader - Ron的专栏 - CSDN博客

11-11

Android输入系统(三)InputReader的加工类型和InputDisp..._CSDN博客

3-22

Android Input流程分析(三):InputReader 09-15 阅读数 692 回到InputReader的...博文 来自: Invoker123的博客 Android按键超时的ANR原理小结 09-22 阅读数 ...

BufferReader与BufferInputStream 区别及用法

阅读数 3535

以reader结尾的都是以字符方式读入，而以stream结尾的都是字节形式importjava.io.BufferedReader;importjava.i...

博文 来自： hander的专栏

MTK TP input子系统笔记

阅读数 591

MTKTPinput子系统

博文 来自： sinat_30545941的...

Android input处理机制(三)InputDispatcher - Ron的专栏 - CSDN博客

11-15

Android Input流程分析(四):InputDispatcher - Invoker123的博客 12-20 413 之前提到,InputReader将Key的信息封装成一个NotifyKeyArgs对象,调用InputDispatcher的no...

android input子系统--InputReader EventHub::getevent..._CSDN博客

10-20

EventHub::getEvents中:分析mClosingDevices在android系统中的调用过程 // Report any devices that had last been added/removed. while (mClosingDevices) { ...

如何复制百度文库中的文章，方法你绝对想不到！

阅读数 477

很多人经常会上百度搜索资料，结果发现在百度文库那边可以找到，兴奋了半天却发现下载时要币的，或者登陆上去...

博文 来自： 教程大咖的CSDN...

Android之input系统流程 - CSDN博客

8-15

http://blog.csdn.net/leerobin83/article/details/6873238

转载Android Input子系统:Input事件的产生、读取和分发..._CSDN博客

11-9

在上一篇博文Android Input子系统:Input进程的创建,监听线程的启动中,我们学习了...Android Input流程分析(三):InputReader - Invoker123的博客 09-15 433 回到...

如何复制百度文库中的文章。。。这个必须留一份。。。

阅读数 884

很多人经常会上百度搜索资料，结果发现在百度文库那边可以找到，兴奋了半天却发现下载时要币的，或者登陆上去...

博文 来自： 可口可乐的博客

教你复制百度文库中的内容

阅读数 3275

百度文库中的很多内容都是需要金币，即使不需要金币，你下载里面的内容，还要注册一个账号。那么现在教大家一...

博文 来自： chen_panpan的博客

Android输入事件从读取到分发三：InputDispatcherThread线程分发事件的过程

阅读数 2709

分析完事件的读取后，东忙西忙，不知不觉已过去了快五个月了...也不是说没有事件把这部分源码分析完，而是实在...

博文 来自： 阳光玻璃杯

input子系统三 input系统启动和EventHub数据读取	阅读数 520
一、框架介绍由下图可以看出，在系统服务启动时会通过InputManager启动InputReader和InputDispatcher，创建... 博文 来自： frank_zyp的博客	0
如何抓取 framework input 事件相关 log	阅读数 20
[size=medium][DESCRIPTION]出现事件输入相关的问题时,建议先follow[url]http://429564140.iteye.com/blog/2... 博文 来自： mengxianguang3...	
android sensor 框架分析---sensor数据流分析	阅读数 1355
5,sensor数据流分析前面几章做了很多准备和铺垫,这章终于可以分析sensor数据的传输流程了。主要步骤如下,1,服务... 博文 来自： Jack的博客	
Android 4.1 Input设备流程分析	03-25
Android 4.1 Input设备流程分析包含触摸流程的详细分析图	下载

Android Input Framework(一)	阅读数 354
1InputFramework概述Android输入系统在整个图形系统框架中扮演了很重要的角色，主要负责用户消息的管理，具... 博文 来自： 编程菜鸟进阶之路	

Android input子系统之InputReader获取输入事件详细分析 - - EventHub->getevents	阅读数 318
此文章只分析EventHub获取输入事件的getevents函数的具体实现首先在EventHub的构造函数中,将以下变量进行初... 博文 来自： qq_30025621的博客	

AOSP源码分析：Android Input事件的产生、读取和分发	阅读数 861
大家好，今天为大家推荐来自MIUI的Cheeeelok同学的AOSP源码分析系列文章，本文依然从源码的角度带大家理解... 博文 来自： 技术视界	

Linux/Android——Input系统之InputReader (七)	阅读数 8206
在前文Linux/Android——Input系统之frameworks层InputManagerService(六)这里介绍了android层input服务的... 博文 来自： jscese	

Android 4.0 事件输入(Event Input)系统	阅读数 4万+
1.TouchScreen功能在Android4.0下不工作 原来在Android2.3.5下能正常工作的TouchScreen功能，移植到And... 博文 来自： MyArrow的专栏	

Android_Input分析	阅读数 461
http://wenku.baidu.com/view/1f6650906bec0975f465e22f.htmlInputSubsystem分析 Android2.3.7的Input子... 博文 来自： 傻傻的我	

Android Input System	阅读数 1412
1，概述 废话少说，直接上图，input从kernel到androidframeworks一锅端式结构框架图。该图kernel部分是以... 博文 来自： 有风吹过。。。	

Android Touch事件传递机制全面解析（从WMS到View树）	阅读数 1万+
转眼间近一年没更新博客了，工作一忙起来，很难有时间来写博客了，由于现在也在从事Android开发相关的工作，... 博文 来自： 兰亭风雨的专栏	

Input事件输入系统之应用MotionEvent触摸事件处理流程	阅读数 1115
Input事件输入系统之应用MotionEvent触摸事件处理流程 输入事件一般分为KeyEvent按键事件，和MotionEve... 博文 来自： xiaomingheni的博客	

Android触摸屏坐标转换	阅读数 1940
一、坐标转换机制概述特有名词：TP：触摸屏（TouchPanel）LCD：显示屏（LiquidCrystalDisplay）在Android... 博文 来自： ningyaodong的博客	

InputManagerService分析一：IMS的启动与事件传递	阅读数 1万+
从这一节里面，我们开始介绍WindowManagerService部分	博文 来自： lilian0118的专栏

Android 输入事件系统之 EventHub 和 Input Lib(事件解析库)	阅读数 1602
从 Android事件输入系统整体框架 一文可知InputLibs是一个事件解析库，完成事件解析、keycode转换，设备配置... 博文 来自： 学无止境，静下心...	

android5.0 物理键盘与软键盘同时使用修改	阅读数 2940
最近工作中遇到在android5.0系统插入硬件盘物理设备后，软键盘无法弹出的问题，在网上查找了相关资料：参考:ht... 博文 来自： u010823818的博客	

android6.0 otg连接设备 点亮屏幕（案例）	阅读数 2383
现在我们otg连接设备的时候有点亮屏幕的需求。解决方法我们可以在识别设备的时候，去调用PowerManager的wa... 博文 来自： kc58236582的博客	

input事件的处理	阅读数 1878
转自http://blog.csdn.net/coldsnow33/article/details/17054985接InputReader::loopOnce()的if(count){process... 博文 来自： qwaszx523的博客	

Android之Input子系统事件分发流程	阅读数 2万+
一、Android4.2系统服务侧1.服务端注册过程frameworks/base/core/java/android/view/ViewRootImpl.javapubl... 博文 来自： 程序改变生活	



真传奇不是靠充的！装备全靠捡，0付费复古传奇！

Android View绘制三大流程探索及常见问题

View绘制的三大流程，指的是measure（测量）、layout（布局）、draw（绘制）measure负责确定View的测量宽...

博文 来自： 梦工厂 阅读数 3570

input子系统整体流程全面分析（触摸屏驱动为例）

input输入子系统整体流程 input子系统在内核中的实现，包括输入子系统（InputCore），事件处理层（Ev...

博文 来自： u012497906的专栏 阅读数 1507

android4.4电源管理——Input系统(Power键处理)

SystemServer.javapublicvoidinitAndLoop(){ wm=WindowManagerService.main(context,power,displ...

博文 来自： Vincent的专栏 阅读数 2740

Android OTA升级原理和流程分析（三）---Android系统的三种启动模式

AndroidOTA升级原理和流程分析（三）---Android系统的三种启动模式 转载自：http://blog.chinaunix.net/ui...

博文 来自： ylyuanlu的专栏 阅读数 7583

Android 事件驱动流程

Android input event Android 事件驱动流程

博文 来自： 04-01 阅读数 0 下载



传奇十年，卸载算我输！这游戏爆率是真高！

Android开发之input子系统一

一、了解Android系统主机默认携带input子系统，并且开机就会产生默认的mouse和keyboard事件，这样使得用户...

博文 来自： 王涛的博客 阅读数 425

Android中InputManagerService里的InputReader和InputDispatcher

最近工作中遇到InputManagerService相关的问题，所以将InputReader和InputDispatcher看了一遍，水平有限，...

博文 来自： 浩海天空 阅读数 3344

Android输入事件InputReader和InputDispatcher分析

.C++frameworks/native/services/inputflinger/InputDispatcher.cppvoidInputDispatcher::notifyKey(constN...

博文 来自： unbroken 阅读数 392

Android系统MotionEvent处理InputReader线程基本原理总结

基本原理触摸事件数据的传递基本流程大致应该分为如下几个阶段首先，当然是硬件感应-->固件软件--&a...

博文 来自： fisher_2005的专栏 阅读数 104

Android4.0 input分析 android部分

前言官方关于androidinput介绍http://source.android.com/devices/tech/input/overview.html 安卓input子系...

博文 来自： dkleikesa的专栏 阅读数 4762



一刀一怪，一怪一神装，战力全飙升，秒天秒地秒全服！

Linux/Android——Input系统之InputMapper 处理 (八)【转】

本文转载自：http://blog.csdn.net/jscese/article/details/43561773前文Linux/Android——Input系统之InputRe...

博文 来自： weixin_33958585... 阅读数 24

Android的Input流程分析(好文)

原址参考其实Android5.0中事件输入子系统的框架和流程没有本质变化。Service端的实现在/frameworks/native/s...

博文 来自： unbroken 阅读数 336

基于PyTorch的深度学习入门教程（六）——数据并行化

前言本文参考PyTorch官网的教程，分为五个基本模块来介绍PyTorch。为了避免文章过长，这五个模块分别在五篇...

博文 来自： 雁回晴空的博客专栏 阅读数 4720

Android实现QQ分享及注意事项

一、获取APPID和帮助文档可以参看新手引导和接入说明：http://wiki.open.qq.com/wiki/移动应用接入wiki索引分...

博文 来自： 水寒 阅读数 5547

CNN笔记：通俗理解卷积神经网络

通俗理解卷积神经网络（cs231n与5月dl班课程笔记） 1 前言 2012年我在北京组织过8期machine l...

博文 来自： 结构之法 算法之道 阅读数 21万+

Android开发本地及网络Mp3音乐播放器(十二)创建NetMusicListAdapter、SearchResult显示...

实现功能： 实现NetMusicListAdapter（网络音乐列表适配器） 实现SearchResult（搜索音乐对象） 使用Jsoup组...

博文 来自： iwanghang(一个播... 阅读数 9291

【小程序】微信小程序开发实践

帐号相关流程注册范围 企业 政府 媒体 其他组织换句话说讲就是不让个人开发者注册。:)填写企业信息不能使用和之前...

博文 来自： 小雨同学的技术博客 阅读数 25万+

HttpClient使用详解

Http协议的重要性相信不用我多说了，HttpClient相比传统JDK自带的URLConnection，增加了易用性和灵活性（具...

博文 来自： 鹏霄万里展雄飞 阅读数 80万+

android客户端与服务器端交互 如何保持session

最近在开发项目的过程中，遇到android与web服务器要在同一session下通信的问题。 在解决问题前先回顾下Sessi...

博文 来自： charming的专栏 阅读数 4万+

三菱FX系列PLC与PC通讯的实现之专有协议（计算机联接）的程序设计之一

阅读内容为：FX系列微型可编程控制器用户手册（通讯篇）中计算机链接功能章节。 采用本方法通信，pc端的实现...

博文 来自： benaic2001的博客 阅读数 1万+

如何在ArcGIS Online中构建自己的应用程序模板初级篇-显示地图					阅读数 4万 +
开发ArcGIS Online应用程序模板之前，需要了解怎么使用ArcGIS API for JavaScript。	在ArcGIS Online当中如...	博文	来自：	ArcGIS产品与技术...	
再谈iOS 7的手势滑动返回功能					阅读数 8万 +
之前随手写过一篇《使用UIScreenEdgePanGestureRecognizer实现swipe to pop效果》，挺粗糙的。现在使用默...		博文	来自：	JasonLee的专栏	
[ASP.NET]二维码的创建					阅读数 5439
又好一段时间没有写写东西了，继续回归原来的模式，多做记录，最近要实现个unity的二维码方面的功能，首先就要...		博文	来自：	学无止境的专栏	
jquery/js实现一个网页同时调用多个倒计时(最新的)					阅读数 44万 +
jquery/js实现一个网页同时调用多个倒计时(最新的) 最近需要网页添加多个倒计时. 查阅网络,基本上都是干遍一律的...		博文	来自：	Websites	
将Excel文件导入数据库（POI+Excel+MySQL+jsp页面导入）第一次优化					阅读数 3万 +
本篇文章是根据我的上篇博客，给出的改进版，由于时间有限，仅做了一个简单的优化。相关文章：将excel导入数据...		博文	来自：	Lynn_Blog	
webService学习（二）——调用自定义对象参数					阅读数 2万 +
webService学习（二）——调用自定义对象参数 本文主要内容：1、如何通过idea进行webService Client的简单...		博文	来自：	止水的专栏	
人脸检测工具face_recognition的安装与应用					阅读数 5万 +
人脸检测工具face_recognition的安装与应用		博文	来自：	roguesir的博客	
C#实现开发windows服务实现自动从FTP服务器下载文件（自行设置分/时执行）					阅读数 2万 +
最近在做一个每天定点从FTP自动下载节目.xml并更新到数据库的功能。首先想到用 FileSystemWatcher来监控下载...		博文	来自：	kongwei521的专栏	
eclipse复制粘贴卡死					阅读数 2488
找了很多资料，最后总结在一起的解决eclipse复制粘贴时卡死的一些方案		博文	来自：	寒尘的专栏	
微信支付V3微信公众号支付PHP教程(thinkPHP5公众号支付)/JSSDK的使用					阅读数 14万 +
扫二维码关注，获取更多技术分享 本文承接之前发布的博客《微信支付V3微信公众号支付PHP教程/thinkPHP5公众...		博文	来自：	Marswill	
linux上安装Docker(非常简单的安装方法)					阅读数 20万 +
最近比较有空，大四出来实习几个月了，作为实习狗的我，被叫去研究Docker了，汗汗！ Docker的三大核心概念：...		博文	来自：	我走小路的博客	
如何在ubuntu 16.04上安装 RealSense（相机型号：Intel SR300）					阅读数 3633
前人栽树，后人乘凉~ 小白参考网上数篇教程（其实最主要是自己的安装记录，方便之后查找错误） https://github...		博文	来自：	z17816876284的...	
openfire 3.8.2 源码部署 /开发配置 / 二次开发					阅读数 6643
最近新搞了openfire 从网上找了很多源码部署的相关文章但都是大同小异，拷贝加修改，我如是按照各个文章版本部...		博文	来自：	StillCity的专栏	
强连通分量及缩点tarjan算法解析					阅读数 57万 +
强连通分量：简言之 就是找环（每条边只走一次，两两可达）孤立的一个点也是一个连通分量 使用tarjan算法 在...		博文	来自：	九野的博客	
【HTTP】Fiddler（一）- Fiddler简介					阅读数 30万 +
1.为什么是Fiddler? 抓包工具有很多，小到最常用的web调试工具firebug，达到通用的强大的抓包工具wireshark.为...		博文	来自：	专注、专心	
centos 查看命令源码					阅读数 8万 +
# yum install yum-utils 设置源: [base-src] name=CentOS-5.4 - Base src - baseurl=http://vault.ce...		博文	来自：	linux/unix	
OpenCV+OpenGL 双目立体视觉三维重建					阅读数 4万 +
0.绪论这篇文章主要为了研究双目立体视觉的最终目标——三维重建，系统的介绍了三维重建的整体步骤。双目立体...		博文	来自：	shiter编写程序的艺...	
mybatis一级缓存(session cache)引发的问题					阅读数 2万 +
mybatis一级缓存(session cache)引发的问题		博文	来自：	flysharkym的专栏	
python图片处理类之~PIL.Image模块(ios android icon图标自动生成处理)					阅读数 5万 +
1.从pyCharm提示下载PIL包 http://www.pythonware.com/products/pil/ 2.解压后，进入到目录下 cd /Users/ji...		博文	来自：	专注于cocos+unit...	
Libusb库在Android下的使用例程					阅读数 6807
转载请注明：http://blog.csdn.net/hubbybob1/article/details/54863662 阅读本文前请先了解相关基础内容，操...		博文	来自：	hubbybob1专栏	
Hadoop+HBase完全分布式安装					阅读数 4313
记录下完全分布式HBase数据库安装步骤准备3台机器：10.202.7.191 / 10.202.7.139 / 10.202.9.89所需准备的Jar包...		博文	来自：	Dobbin	
DataTables 的 实例 《一》					阅读数 1万 +
1.加载需要的js/css文件 2. function del(id){ alert(id); } var table; \$(document).ready(function(...		博文	来自：	辛修灿的博客	

关于SpringBoot bean无法注入的问题（与文件包位置有关）	阅读数 17万+
问题场景描述整个项目通过Maven构建，大致结构如下： 核心Spring框架一个module spring-boot-base service... 博文 来自： 开发随笔	
Android平台Camera实时滤镜实现方法探讨(五)--GLSurfaceView实现Camera预览	阅读数 2万+
前面有一篇探讨了如何在片段着色器中将YUV数据转换为RGB数据并显示，但采用samplerExternalOES将SurfaceTe... 博文	
JavaWeb多文件上传及zip打包下载	阅读数 6536
项目中经常会使用到文件上传及下载的功能。本篇文章总结场景在JavaWeb环境下，多文件上传及批量打包下载功能... 博文 来自： kidQ的博客	
R语言逻辑回归、ROC曲线和十折交叉验证	阅读数 5万+
自己整理编写的逻辑回归模板，作为学习笔记记录分享。数据集用的是14个自变量Xi，一个因变量Y的australian数据... 博文 来自： Tiaaaaa的博客	
聚类算法流程 文本分类流程 算术编码流程 NAL Unit结构分析 H.264宏块残差分析	
ios获取idfa android title搜索 server的安全控制模型是什么 sql ios 动态修改约束 数据分析课程 数据分析课程	



Invoker123

关注

原创26

粉丝14

喜欢5

评论0

等级：

博客 已

访问：1万+

积分：410

排名：15万+

勋章：

恒

最新文章

剖析Android的Smart Lock

Android O HIDL框架

SystemService的Watchdog

standard、singleTop、singleTask和singleInstance原理分析

从一个关机时间过长的bug谈起

个人分类

Framework17篇

Hal1篇

Linux1篇

Android安全3篇

ART5篇

归档

2019年1月1篇

2018年12月1篇

2018年11月1篇

2018年10月2篇

2018年9月3篇

展开

热门文章

Android Sensor HAL层分析
阅读数 2218

SEAndroid流程分析
阅读数 1176

process_config解析fstab文件
阅读数 800

Android Input流程分析（三）：
InutReader



CSDN学院



CSDN企业招聘

 QQ客服

 kefu@csdn.net

 客服论坛

 400-660-0108


工作时间 8:30-22:00

关于我们

招聘

广告服务

网站地图

 百度提供站内搜索 京ICP备19004658号

©1999-2019 北京创新乐知网络技术有限公司

网络110报警服务

经营性网站备案信息

北京互联网违法和不良信息举报中心

中国互联网举报中心

家长监护

0