内容目录

CarPower休眠唤醒后，偶现关机/重启选择的界面，这个问题需要调查一下

# 调查一、了解input的大概数据流

kernel/include/dt-bindings/input/linux-event-codes.h

#define KEY\_POWER 116 /\* SC System Power Down \*/

device/chehejia/M01\_AE/qpnp\_pon.kl

key 116 POWER

frameworks/base/services/java/com/android/server/SystemServer.java

import com.android.server.input.InputManagerService;

import com.android.server.wm.WindowManagerService;

import com.android.server.policy.PhoneWindowManager;

InputManagerService inputManager = null;

//阶段一、初始化输入服务

inputManager = new InputManagerService(context);

wm = WindowManagerService.main(context, inputManager,

mFactoryTestMode != FactoryTest.FACTORY\_TEST\_LOW\_LEVEL, //工厂模式否

!mFirstBoot, mOnlyCore, new PhoneWindowManager()); //是否是首次开机，是否只是解析CoreApp

ServiceManager.addService(Context.INPUT\_SERVICE, inputManager);//“input”

//阶段二、启动输入服务

inputManager.start();

**阶段一、InputManager的初始化**

frameworks/base/services/core/java/com/android/server/input/InputManagerService.java

// Pointer to native input manager service object.

private final long mPtr;

private static native long nativeInit(InputManagerService service, Context context, MessageQueue messageQueue);

mPtr = nativeInit(this, mContext, mHandler.getLooper().getQueue());//通过JNI调用来启动native层的input系统，然后把返回值存放在mPtr中

frameworks/base/services/core/jni/com\_android\_server\_input\_InputManagerService.cpp

//这里实例化了NativeInputManagerService的一个对象，使用的Java层的MessageQueue的Looper，意味着Java层消息和Native消息是在同一个MessageQueue中的

static jlong nativeInit(JNIEnv\* env, jclass /\* clazz \*/,

jobject serviceObj, jobject contextObj, jobject messageQueueObj) {

sp<MessageQueue> messageQueue = android\_os\_MessageQueue\_getMessageQueue(env, messageQueueObj);

NativeInputManager\* im = new NativeInputManager(contextObj, serviceObj, messageQueue->getLooper());

im->incStrong(0);//增加对象引用计数

return reinterpret\_cast<jlong>(im);

}

NativeInputManager::NativeInputManager(jobject contextObj, jobject serviceObj, const sp<Looper>& looper) : mLooper(looper), mInteractive(true) {

sp<EventHub> eventHub = new EventHub();

mInputManager = new InputManager(eventHub, this, this);

}

frameworks/native/services/inputflinger/InputManager.cpp

InputManager::InputManager(

const sp<EventHubInterface>& eventHub,

const sp<InputReaderPolicyInterface>& readerPolicy,

const sp<InputDispatcherPolicyInterface>& dispatcherPolicy) {

mDispatcher = new InputDispatcher(dispatcherPolicy);

mReader = new InputReader(eventHub, readerPolicy, mDispatcher);

initialize();

}

void InputManager::initialize() {

mReaderThread = new InputReaderThread(mReader);//InputReaderThread线程，负责input事件的获取

mDispatcherThread = new InputDispatcherThread(mDispatcher);//InputDispatcherThread线程，负责input消息的发送

}

**阶段二、InputManager的启动**

切回InputManagerServer.java的start方法

public void start() {

nativeStart(mPtr);

}

com\_android\_server\_input\_InputManagerService.cpp

static void nativeStart(JNIEnv\* env, jclass /\* clazz \*/, jlong ptr) {

NativeInputManager\* im = reinterpret\_cast<NativeInputManager\*>(ptr);

status\_t result = im->getInputManager()->start();

if (result) {

jniThrowRuntimeException(env, "Input manager could not be started.");

}

}

InputManager.cpp

status\_t InputManager::start() {//同时给了PRIORITY\_URGENT\_DISPLAY这个级别跟SurfaceFlinger的priority是一样，可见是及时响应的

status\_t result = mDispatcherThread->run("InputDispatcher", PRIORITY\_URGENT\_DISPLAY);

if (result) {

ALOGE("Could not start InputDispatcher thread due to error %d.", result);

return result;

}

result = mReaderThread->run("InputReader", PRIORITY\_URGENT\_DISPLAY);

return OK;

}

**InputDispatcherThread**

InputDispatcherThread继承的是Thread类，由于它是Thread子类，于是继承它的run方法，进入run方法后会调用threadLoop()，在Thread类中它是虚函数，得由子类来复写，如下所示

bool InputDispatcherThread::threadLoop() {

mDispatcher->dispatchOnce();

return true;

}

void InputDispatcher::dispatchOnce() {

nsecs\_t nextWakeupTime = LONG\_LONG\_MAX;

{ // acquire lock

AutoMutex \_l(mLock);

mDispatcherIsAliveCondition.broadcast();

// Run a dispatch loop if there are no pending commands.

// The dispatch loop might enqueue commands to run afterwards.

if (!haveCommandsLocked()) {

dispatchOnceInnerLocked(&nextWakeupTime);//事件分发

}

// Run all pending commands if there are any.

// If any commands were run then force the next poll to wake up immediately.

if (runCommandsLockedInterruptible()) {//事件执行

nextWakeupTime = LONG\_LONG\_MIN;

}

} // release lock

// Wait for callback or timeout or wake. (make sure we round up, not down)

nsecs\_t currentTime = now();

int timeoutMillis = toMillisecondTimeoutDelay(currentTime, nextWakeupTime);

mLooper->pollOnce(timeoutMillis);//获取键盘事件，如果没有消息的话就是阻塞epoll，底层的实现是基于pipe机制的

}

InputDispatch会调用dispatchOnceInnerLocked去分发消息；在消息分完完成后，又使用mLooper继续获取消息。当mLooper.pollOnce有读取到消息的时候就会往管道里头写新的内容，唤醒正在等待键盘事件的线程；当没有消息的时候，就无限的阻塞着，线程进入空闲等待状态

**InputReaderThread**

InputReader.cpp

bool InputReaderThread::threadLoop() {

mReader->loopOnce();

return true;

}

void InputReader::loopOnce() {

int32\_t oldGeneration;

int32\_t timeoutMillis;

bool inputDevicesChanged = false;

Vector<InputDeviceInfo> inputDevices;

{ // acquire lock

AutoMutex \_l(mLock);

oldGeneration = mGeneration;

timeoutMillis = -1;

uint32\_t changes = mConfigurationChangesToRefresh;

if (changes) {

mConfigurationChangesToRefresh = 0;

timeoutMillis = 0;

refreshConfigurationLocked(changes);

} else if (mNextTimeout != LLONG\_MAX) {

nsecs\_t now = systemTime(SYSTEM\_TIME\_MONOTONIC);

timeoutMillis = toMillisecondTimeoutDelay(now, mNextTimeout);

}

} // release lock

//阶段一，通过EventHub的getEvents方法来获取input事件，/dev/input/event\*

size\_t count = mEventHub->getEvents(timeoutMillis, mEventBuffer, EVENT\_BUFFER\_SIZE);

{ // acquire lock

AutoMutex \_l(mLock);

mReaderIsAliveCondition.broadcast();

if (count) {

processEventsLocked(mEventBuffer, count);//阶段二，开始处理读取出来的元事件

}

if (mNextTimeout != LLONG\_MAX) {

nsecs\_t now = systemTime(SYSTEM\_TIME\_MONOTONIC);

if (now >= mNextTimeout) {

#if DEBUG\_RAW\_EVENTS

ALOGD("Timeout expired, latency=%0.3fms", (now - mNextTimeout) \* 0.000001f);

#endif

mNextTimeout = LLONG\_MAX;

timeoutExpiredLocked(now);

}

}

if (oldGeneration != mGeneration) {

inputDevicesChanged = true;

getInputDevicesLocked(inputDevices);

}

} // release lock

// Send out a message that the describes the changed input devices.

if (inputDevicesChanged) {

mPolicy->notifyInputDevicesChanged(inputDevices);

}

mQueuedListener->flush();//阶段三，把QueuedInputListener中的消息全部都开始处理

}

InputListener.cpp

void QueuedInputListener::flush() {

size\_t count = mArgsQueue.size();

for (size\_t i = 0; i < count; i++) {

NotifyArgs\* args = mArgsQueue[i];

args->notify(mInnerListener);//mlnnerListener是InputDispatcher对象

delete args;

}

mArgsQueue.clear();

}

小结一下InputReader 这三个阶段

1、通过EventHub的getEvents方法来获取input事件。

EventHub负责打开/dev/input/目录下的所有设备，然后为每一个设备创建一个Device，并把这个Device放入EventHub所定义的数组们Device中。之后，就是把这个设备纳入监视范围。然后就是开始等待事件的发生，一旦有事件发生，就从产生事件的设备中读取出这些设备，把这些事件转化为RawEvent类型放入InputReader提供的事件数组中，之后返回。到这里，从EventHub获取事件就结束了。

2、从EventHub中读取出若干事件，然会对这些事件进行预处理。

3、把QueuedInputListener中的事件分发出去。

# 调查二、long press key debug log

EventHub的数据流里未提供是谁触发的长按，所以现在先在framework层加一下long press key debug log，问题复现是有log可查

device/chehejia/M01\_AE/qpnp\_pon.kl

key 116 POWER

frameworks/native/include/input/InputEventLabels.h

DEFINE\_KEYCODE(POWER),

frameworks/native/include/android/keycodes.h

AKEYCODE\_POWER = 26,

framework/base/services/core/java/com/android/server/policy/PhoneWindowManager.java

case KeyEvent.KEYCODE\_POWER: {

Slog.w(TAG, "interceptKeyBeforeQueueing: KeyEvent.KEYCODE\_POWER debug for PowerStateMachineImpl test!!!");

/\*

// Any activity on the power button stops the accessibility shortcut

cancelPendingAccessibilityShortcutAction();

result &= ~ACTION\_PASS\_TO\_USER;

isWakeKey = false; // wake-up will be handled separately

if (down) {

interceptPowerKeyDown(event, interactive);

} else {

interceptPowerKeyUp(event, interactive, canceled);

}

\*/

break;

}

10-19 04:43:56.161 1286 1465 D PowerStateMachineImpl: recv MCU\_PWR\_QUERY(2)

10-19 04:44:41.900 1286 1465 D PowerStateMachineImpl: Recv NM mode 0

10-19 04:44:41.927 1286 1796 D PowerStateMachineImpl: Transition from PRESLEEP to SLEEP

10-19 04:45:04.268 562 860 W WindowManager: interceptKeyBeforeQueueing: KeyEvent.KEYCODE\_POWER mask for PowerStateMachineImpl!!!

10-19 04:45:04.315 562 860 W WindowManager: interceptKeyBeforeQueueing: KeyEvent.KEYCODE\_POWER mask for PowerStateMachineImpl!!!

10-19 04:45:04.410 1286 1796 D PowerStateMachineImpl: Transition from SLEEP to PARTLYWORK

10-19 04:45:04.937 1286 1465 D PowerStateMachineImpl: recv wakeup reason (51)

10-19 04:45:21.673 1286 1465 D PowerStateMachineImpl: Recv NM mode 3

# 调查三、关机界面触发

frameworks/base/services/core/java/com/android/server/policy/PhoneWindowManager.java

private void powerLongPress() {//长按处理的接口

final int behavior = getResolvedLongPressOnPowerBehavior();

Slog.w(TAG, "powerLongPress mask for PowerStateMachineImpl, behavior = " + behavior);

switch (behavior) {

case LONG\_PRESS\_POWER\_NOTHING:

break;

case LONG\_PRESS\_POWER\_GLOBAL\_ACTIONS:

Slog.w(TAG, "LONG\_PRESS\_POWER\_GLOBAL\_ACTIONS mask for PowerStateMachineImpl");

mPowerKeyHandled = true;

performHapticFeedbackLw(null, HapticFeedbackConstants.LONG\_PRESS, false);

showGlobalActionsInternal();//触发关机/重启选择界面的“总”入口1

break;

case LONG\_PRESS\_POWER\_SHUT\_OFF:

case LONG\_PRESS\_POWER\_SHUT\_OFF\_NO\_CONFIRM:

...

break;

}

}

GlobalActions mGlobalActions;//frameworks/base/services/core/java/com/android/server/policy/GlobalActions.java

WindowManagerFuncs mWindowManagerFuncs;//frameworks/base/services/core/java/com/android/server/wm/WindowManagerService.java，实现这些callback

void showGlobalActionsInternal() {

sendCloseSystemWindows(SYSTEM\_DIALOG\_REASON\_GLOBAL\_ACTIONS);//关闭系统dialogs

if (mGlobalActions == null) {

mGlobalActions = new GlobalActions(mContext, mWindowManagerFuncs);

}

final boolean keyguardShowing = isKeyguardShowingAndNotOccluded();//keyguard是否在显示

//keyguardShowing = false, isDeviceProvisioned() = true

mGlobalActions.showDialog(keyguardShowing, isDeviceProvisioned());//弹出关机/重启选择界面入口2

if (keyguardShowing) {

// since it took two seconds of long press to bring this up, poke the wake lock so they have some time to see the dialog.

mPowerManager.userActivity(SystemClock.uptimeMillis(), false);//通知power发生了一次用户时间, 让用户可以看到dialog

}

}

frameworks/base/services/core/java/com/android/server/policy/GlobalActions.java

public void showDialog(boolean keyguardShowing, boolean deviceProvisioned) {//false, true

Slog.d(TAG, "heshang showDialog1 " + keyguardShowing + " " + deviceProvisioned);

mKeyguardShowing = keyguardShowing;//keyguard正在显示

mDeviceProvisioned = deviceProvisioned;

mShowing = true;

if (mStatusBarConnected) {

Slog.d(TAG, "heshang showDialog2 " + keyguardShowing + " " + deviceProvisioned);

mStatusBarInternal.showGlobalActions(mDisplayId);

mHandler.postDelayed(mShowTimeout, 5000);

} else {

Slog.d(TAG, "heshang showDialog3 " + keyguardShowing + " " + deviceProvisioned);

// SysUI isn't alive, show legacy menu.

ensureLegacyCreated();

mLegacyGlobalActions.showDialog(mKeyguardShowing, mDeviceProvisioned);

}

}

frameworks/base/services/core/java/com/android/server/policy/LegacyGlobalActions.java

//如果已经存在mDialog就将原来的dismiss掉，之后发送消息重新show出来，否则就新创建出来一个显示.

public void showDialog(boolean keyguardShowing, boolean isDeviceProvisioned) {

if (DEBUG) Slog.d(TAG, "showDialog " + keyguardShowing + " " + deviceProvisioned);

**Slog.w(TAG, "who showDialog start " + keyguardShowing + " " + deviceProvisioned + " for PowerStateMachineImpl");**//加入谁触发了重启/关机diag的代码

**Thread.dumpStack();**

**Slog.w(TAG, "who showDialog end " + keyguardShowing + " " + deviceProvisioned + " for PowerStateMachineImpl");**

调试效果如下

12-19 15:11:40.159 565 713 W GlobalActions: heshang showDialog1.1 false true

12-19 15:11:40.150 855 973 D UDS : cannot connect server

12-19 15:11:40.159 565 713 W System.err: java.lang.Exception: Stack trace

12-19 15:11:40.159 565 713 W System.err: at java.lang.Thread.dumpStack(Thread.java:1348)

12-19 15:11:40.159 565 713 W System.err: at com.android.server.policy.GlobalActions.showDialog(GlobalActions.java:64)

12-19 15:11:40.159 565 713 W System.err: at com.android.server.policy.PhoneWindowManager.showGlobalActionsInternal(PhoneWindowManager.java:1813)

12-19 15:11:40.159 1418 1545 D vol.k : onReceive ACTION\_CLOSE\_SYSTEM\_DIALOGS

12-19 15:11:40.159 565 713 W System.err: at com.android.server.policy.PhoneWindowManager.**powerLongPress**(PhoneWindowManager.java:1653)

12-19 15:11:40.159 565 713 W System.err: at com.android.server.policy.PhoneWindowManager.-wrap20(Unknown Source:0)

12-19 15:11:40.159 565 713 W System.err: at com.android.server.policy.PhoneWindowManager$PolicyHandler.handleMessage(PhoneWindowManager.java:949)

12-19 15:11:40.159 565 713 W System.err: at android.os.Handler.dispatchMessage(Handler.java:106)

12-19 15:11:40.159 565 713 W System.err: at android.os.Looper.loop(Looper.java:164)

12-19 15:11:40.159 565 713 W System.err: at android.os.HandlerThread.run(HandlerThread.java:65)

12-19 15:11:40.160 565 713 W System.err: at com.android.server.ServiceThread.run(ServiceThread.java:46)

12-19 15:11:40.160 565 713 W System.err: at com.android.server.UiThread.run(UiThread.java:42)

12-19 15:11:40.160 565 713 W GlobalActions: heshang showDialog1.2 false true

mKeyguardShowing = keyguardShowing;

mDeviceProvisioned = isDeviceProvisioned;

if (mDialog != null) {

mDialog.dismiss();//dismiss dialog

mDialog = null;

// Show delayed, so that the dismiss of the previous dialog completes

mHandler.sendEmptyMessage(MESSAGE\_SHOW);

} else {

handleShow();//如果不存在mDialog, 就调用handleShow处理

}

}

private void handleShow() {

awakenIfNecessary();

mDialog = createDialog();

prepareDialog();

// If we only have 1 item and it's a simple press action, just do this action.

if (mAdapter.getCount() == 1

&& mAdapter.getItem(0) instanceof SinglePressAction

&& !(mAdapter.getItem(0) instanceof LongPressAction)) {

((SinglePressAction) mAdapter.getItem(0)).onPress();

} else {

if (mDialog != null) {

WindowManager.LayoutParams attrs = mDialog.getWindow().getAttributes();

attrs.setTitle("LegacyGlobalActions");

mDialog.getWindow().setAttributes(attrs);

mDialog.show();

mDialog.getWindow().getDecorView().setSystemUiVisibility(View.STATUS\_BAR\_DISABLE\_EXPAND);

}

}

}

private GlobalActionsDialog createDialog() {

...

mItems = new ArrayList<Action>();

String[] defaultActions = mContext.getResources().getStringArray(

com.android.internal.R.array.config\_globalActionsList);

ArraySet<String> addedKeys = new ArraySet<String>();

for (int i = 0; i < defaultActions.length; i++) {

String actionKey = defaultActions[i];

Log.e(TAG, "heshang createDialog 2, i = " + i + ", actionKey = " + actionKey);

12-19 14:42:09.878 564 711 E LegacyGlobalActions: heshang createDialog 2, i = 0, actionKey = power

12-19 14:42:09.879 564 711 E LegacyGlobalActions: heshang createDialog 2, i = 1, actionKey = restart

12-19 14:42:09.879 564 711 E LegacyGlobalActions: heshang createDialog 2, i = 2, actionKey = bugreport

12-19 14:42:09.879 564 711 E LegacyGlobalActions: heshang createDialog 2, i = 3, actionKey = users

if (addedKeys.contains(actionKey)) {

// If we already have added this, don't add it again.

continue;

}

if (GLOBAL\_ACTION\_KEY\_POWER.equals(actionKey)) {//长按power键动作

mItems.add(new PowerAction());

} else if (GLOBAL\_ACTION\_KEY\_AIRPLANE.equals(actionKey)) {

mItems.add(mAirplaneModeOn);

} else if (GLOBAL\_ACTION\_KEY\_BUGREPORT.equals(actionKey)) {

if (Settings.Global.getInt(mContext.getContentResolver(),

Settings.Global.BUGREPORT\_IN\_POWER\_MENU, 0) != 0 && isCurrentUserOwner()) {

mItems.add(new BugReportAction());

}

} else if (GLOBAL\_ACTION\_KEY\_SILENT.equals(actionKey)) {

if (mShowSilentToggle) {

mItems.add(mSilentModeAction);

}

} else if (GLOBAL\_ACTION\_KEY\_USERS.equals(actionKey)) {

if (SystemProperties.getBoolean("fw.power\_user\_switcher", false)) {

addUsersToMenu(mItems);

}

} else if (GLOBAL\_ACTION\_KEY\_SETTINGS.equals(actionKey)) {

mItems.add(getSettingsAction());

} else if (GLOBAL\_ACTION\_KEY\_LOCKDOWN.equals(actionKey)) {

mItems.add(getLockdownAction());

} else if (GLOBAL\_ACTION\_KEY\_VOICEASSIST.equals(actionKey)) {

mItems.add(getVoiceAssistAction());

} else if (GLOBAL\_ACTION\_KEY\_ASSIST.equals(actionKey)) {

mItems.add(getAssistAction());

} else if (GLOBAL\_ACTION\_KEY\_RESTART.equals(actionKey)) {

mItems.add(new RestartAction());

} else {

Log.e(TAG, "Invalid global action key " + actionKey);

}

// Add here so we don't add more than one.

addedKeys.add(actionKey);

}

}