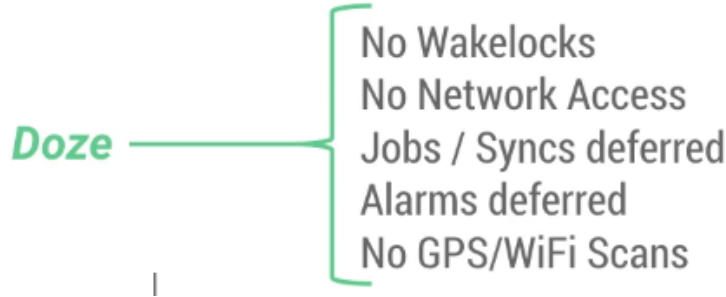


Android Doze模式源码分析

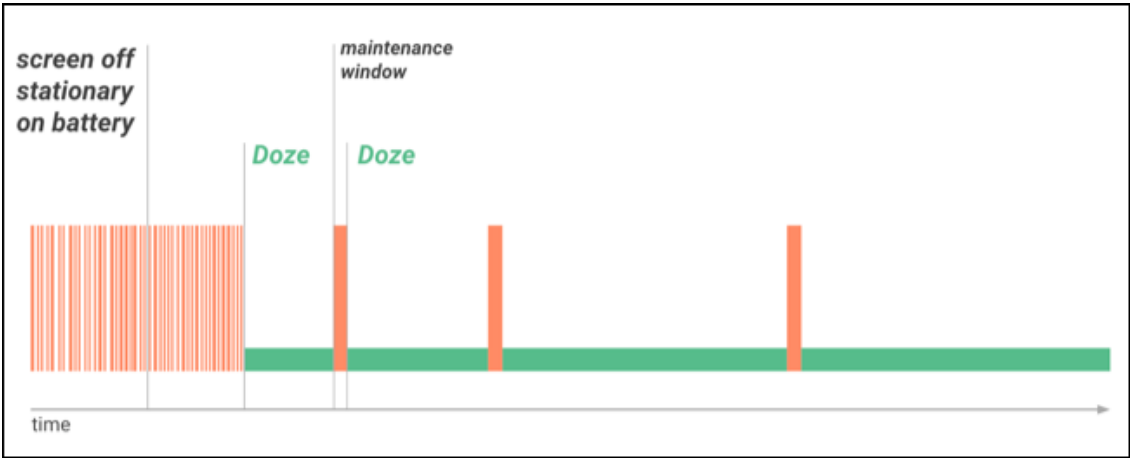
科技的仿生学无处不在，给予我们启发。为了延长电池是使用寿命，google从蛇的冬眠中得到体会，那就是在某种情况下也让手机进入类冬眠的情况，从而引入了今天的主题，Doze模式，Doze中文是打盹儿，打盹当然比活动节约能量了。

手机打盹儿的时候会怎样呢？



按照google的官方说法，Walklocks，网络访问，jobshedule，闹钟，GPS/WiFi扫描都会停止。这些停止后，将会节省30%的电量。

手机什么时候才会开始打盹呢？



上图是谷歌的Doze时序示意图，可以看出让手机打盹要满足三个条件

- 1.屏幕熄灭
- 2 .不插电
- 3.静止不动

这个是不是很仿生学呢？屏幕熄灭->闭上双眼，不插电->不吃东西，静止不动->安静地做个睡美人。生物不也是满足这些条件才能打盹吗？妙，是在妙！

打盹总得呼吸吧？上图中的maintenance window就是给你呼吸的！！呼吸的时候Walklocks，网络访问，jobshedule，闹钟，GPS/WiFi扫描这些都会恢复，来吧重重的吸一口新鲜空气吧！随着时间的推移，呼吸的间隔会越变越大，而每次呼吸的时间也会变长，当然，伙计，不会无限长！！最后都会归于一个定值。下面分析源码就知道了，biu！

没源码，说个球儿

下面以一台手机静静地放在桌面上，随着时间的推移，进入doze模式的过程来分析源码。

源码路径：[/frameworks/base/services/core/java/com/android/server/DeviceIdleController.java](#)

系统中用一个全局整形变量来表示当前doze的状态

```
1 private int mState;
```

状态值的可能取值有以下,一开始的状态是STATE\_ACTIVE。会依次经过1,2,3,4,状态后进入5状态，即STATE\_IDLE

```
1 private static final int STATE_ACTIVE = 0;
2 private static final int STATE_INACTIVE = 1;
3 private static final int STATE_IDLE_PENDING = 2;
4 private static final int STATE_SENSING = 3;
5 private static final int STATE_LOCATING = 4;
6 private static final int STATE_IDLE = 5;
7 private static final int STATE_IDLE_MAINTENANCE = 6;
```

首先屏幕熄灭，回调熄屏处理函数

```
1 private final DisplayManager.DisplayListener mDisplayListener
```

导航

- 博客园
- 首页
- 新随笔
- 联系
- 订阅 XML
- 管理

公告

昵称：屌丝迷途  
园龄：7年7个月  
粉丝：3  
关注：0  
[+加关注](#)

<	2017年8月						>
日	一	二	三	四	五	六	
30	31	1	2	3	4	5	
6	7	8	9	10	11	12	
13	14	15	16	17	18	19	
20	21	22	23	24	25	26	
27	28	29	30	31	1	2	
3	4	5	6	7	8	9	

统计

随笔 - 137  
文章 - 1  
评论 - 3  
引用 - 0

搜索

找找看

谷歌搜索

常用链接

- [我的随笔](#)
- [我的评论](#)
- [我的参与](#)
- [最新评论](#)
- [我的标签](#)

我的标签

- [Android\(11\)](#)
- [Android Studio\(5\)](#)
- [Android.mk\(4\)](#)
- [Gradle\(4\)](#)
- [Android编译\(3\)](#)
- [权限\(3\)](#)
- [应用架构\(3\)](#)
- [gps\(2\)](#)
- [SEAndroid\(2\)](#)
- [Service\(2\)](#)
- [更多](#)

随笔分类

- [Android Framework\(9\)](#)
- [Android定制移植\(26\)](#)
- [Android基础知识\(9\)](#)
- [Android开发\(17\)](#)
- [Android开源项目\(2\)](#)
- [编程开发\(1\)](#)

随笔档案

- [2017年3月 \(5\)](#)
- [2017年2月 \(4\)](#)
- [2016年12月 \(3\)](#)
- [2016年11月 \(10\)](#)
- [2016年10月 \(10\)](#)
- [2016年9月 \(4\)](#)
- [2016年8月 \(3\)](#)
- [2016年7月 \(3\)](#)
- [2016年6月 \(3\)](#)
- [2016年5月 \(2\)](#)
- [2016年4月 \(1\)](#)
- [2016年3月 \(3\)](#)
- [2016年2月 \(2\)](#)
- [2016年1月 \(3\)](#)

```
2         = new DisplayManager.DisplayListener() {
3             @Override public void onDisplayAdded(int displayId) {
4             }
5
6             @Override public void onDisplayRemoved(int displayId) {
7             }
8
9             @Override public void onDisplayChanged(int displayId) {
10                if (displayId == Display.DEFAULT_DISPLAY) {
11                    synchronized (DeviceIdleController.this) {
12                        updateDisplayLocked(); //屏幕状态改变
13                    }
14                }
15            }
16        };
```

进入updateDisplayLocked

```
1 void updateDisplayLocked() {
2     ...
3     becomeInactiveIfAppropriateLocked(); //看是否可以进入Inactive状态
4     ....
5 }
6 }
```

然后我们拔出usb，不充电，会回调充电处理函数

```
1 private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
2     @Override public void onReceive(Context context, Intent intent) {
3         if (Intent.ACTION_BATTERY_CHANGED.equals(intent.getAction())) {
4             int plugged = intent.getIntExtra("plugged", 0);
5             updateChargingLocked(plugged != 0); //充电状态改变
6         } else if (ACTION_STEP_IDLE_STATE.equals(intent.getAction())) {
7             synchronized (DeviceIdleController.this) {
8                 stepIdleStateLocked();
9             }
10        }
11    }
12 };
```

进入updateChargingLocked

```
1 void updateChargingLocked(boolean charging) {
2     ....
3     becomeInactiveIfAppropriateLocked(); //看是否可以进入Inactive状态
4     .....
5 }
```

最后不插电和熄灭屏幕后都会进入becomeInactiveIfAppropriateLocked，状态mState变成STATE\_INACTIVE，并且开启了一个定时器

```
1 void becomeInactiveIfAppropriateLocked() {
2     if (DEBUG) Slog.d(TAG, "becomeInactiveIfAppropriateLocked()");
3     //不插电和屏幕熄灭的条件都满足了
4     if (((!mScreenOn && !mCharging) || mForceIdle) && mEnabled && mState ==
STATE_ACTIVE) {
5         .....
6         mState = STATE_INACTIVE;
7         scheduleAlarmLocked(mInactiveTimeout, false);
8         .....
9     }
10 }
11
12 定时时长为常量30分钟
13 INACTIVE_TIMEOUT = mParser.getLong(KEY_INACTIVE_TIMEOUT,
14     !COMPRESS_TIME ? 30 * 60 * 1000L : 3 * 60 * 1000L);
```

手机静静地躺在桌面上30分钟后，定时器时间到达后，pendingintent会被发出,广播接收器进行处理

- 2015年12月 (7)
- 2015年11月 (13)
- 2015年10月 (4)
- 2015年9月 (1)
- 2015年8月 (6)
- 2015年7月 (3)
- 2015年6月 (1)
- 2015年5月 (1)
- 2015年4月 (6)
- 2015年3月 (11)
- 2015年2月 (2)
- 2015年1月 (3)
- 2014年12月 (1)
- 2014年11月 (9)
- 2014年10月 (8)
- 2014年9月 (3)
- 2013年4月 (2)

最新评论

- 1. Re:Android界面架构 (Activity,PhoneWiondow,DecorView)简介写的简单易懂，点赞但是有个问题是不是搞错了？DecorView 只是 FrameLayout 的子类，并不是PhoneWindow的子类好像在之前的源码里，DecorView 只是 PhoneWin..... --li-xyz
- 2. Re:Android 6.0权限全面详细分析和解决方案好文章 --广外帅哥
- 3. Re:Android Launcher 3 简单分析不错，支持 --lswzq

阅读排行榜


- 1. Android终止线程的方法(6111)
- 2. BMP图像数据格式详解(4655)
- 3. Android 6.0权限全面详细分析和解决方案(4068)
- 4. Android保持屏幕常亮的方法(2995)
- 5. [Android L]SEAndroid开放设备文件结点权限(读或写)方法(涵盖常用操作：sys/xxx、proc/xxx、SystemProperties)(2905)

评论排行榜


- 1. Android界面架构 (Activity,PhoneWiondow,DecorView)简介(1)
- 2. Android 6.0权限全面详细分析和解决方案(1)
- 3. Android Launcher 3 简单分析(1)

推荐排行榜

- 1. Android界面架构 (Activity,PhoneWiondow,DecorView)简介(3)
- 2. 两个Service之间相互监视的实现(1)
- 3. Android 6.0权限全面详细分析和解决方案(1)
- 4. BMP图像数据格式详解(1)
- 5. Android的开机流程及对应源码位置分析(1)



```
1 Intent intent = new Intent(ACTION_STEP_IDLE_STATE)
2     .setPackage("android")
3     .setFlags(Intent.FLAG_RECEIVER_REGISTERED_ONLY);
4 mAlarmIntent = PendingIntent.getBroadcast(getContext(), 0, intent, 0);
5
6 private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
7     @Override public void onReceive(Context context, Intent intent) {
8         if (Intent.ACTION_BATTERY_CHANGED.equals(intent.getAction())) {
9             int plugged = intent.getIntExtra("plugged", 0);
10            updateChargingLocked(plugged != 0);
11        } else if (ACTION_STEP_IDLE_STATE.equals(intent.getAction())) {
12            synchronized (DeviceIdleController.this) {
13                stepIdleStateLocked();    //接收到广播
14            }
15        }
16    }
17 };
```



进入stepIdleStateLocked,该函数是状态转换处理的主要函数



```
1 void stepIdleStateLocked() {
2     if (DEBUG) Slog.d(TAG, "stepIdleStateLocked: mState=" + mState);
3     EventLogTags.writeDeviceIdleStep();
4
5     final long now = SystemClock.elapsedRealtime();
6     if ((now+mConstants.MIN_TIME_TO_ALARM) >
mAlarmManager.getNextWakeFromIdleTime()) {
7         // Whoops, there is an upcoming alarm. We don't actually want to go idle.
8         if (mState != STATE_ACTIVE) {
9             becomeActiveLocked("alarm", Process.myUid());
10        }
11        return;
12    }
13
14    switch (mState) {
15        case STATE_INACTIVE:
16            // We have now been inactive long enough, it is time to start looking
17            // for significant motion and sleep some more while doing so.
18            startMonitoringSignificantMotion(); //观察是否有小动作
19            scheduleAlarmLocked(mConstants.IDLE_AFTER_INACTIVE_TIMEOUT, false); //设置观察小动作要观察多久
20            mState = STATE_IDLE_PENDING; //状态更新为STATE_IDLE_PENDING
21            break;
22        case STATE_IDLE_PENDING: //小动作观察结束，很厉害，一直都没有小动作，会进入这里
23            mState = STATE_SENSING; //状态更新为STATE_SENSING
24            scheduleSensingAlarmLocked(mConstants.SENSING_TIMEOUT); //设置传感器感应时长
25            mAnyMotionDetector.checkForAnyMotion(); //传感器感应手机有没有动
26            break;
27        case STATE_SENSING: //传感器也没发现手机动，就来最后一发，看GPS有没有动
28            mState = STATE_LOCATING; //状态更新为STATE_LOCATING
29            scheduleSensingAlarmLocked(mConstants.LOCATING_TIMEOUT); //设置GPS观察时长
30            mLocationManager.requestLocationUpdates(mLocationRequest,
mGenericLocationListener,
31                mHandler.getLooper()); //GPS开始感应
32            break;
33        case STATE_LOCATING: //GPS也发现没动
34            cancelSensingAlarmLocked();
35            cancelLocatingLocked();
36            mAnyMotionDetector.stop(); //这里没有break，直接进入下一个case
37        case STATE_IDLE_MAINTENANCE:
38            scheduleAlarmLocked(mNextIdleDelay, true); //设置打盹多久后进行呼吸
39            mNextIdleDelay = (long)(mNextIdleDelay * mConstants.IDLE_FACTOR); //更新下次打盹多久后进行呼吸
40            if (DEBUG) Slog.d(TAG, "Setting mNextIdleDelay = " + mNextIdleDelay);
41            mNextIdleDelay = Math.min(mNextIdleDelay, mConstants.MAX_IDLE_TIMEOUT);
42            mState = STATE_IDLE; //噢耶 终于进入了STATE_IDLE
43            mHandler.sendEmptyMessage(MSG_REPORT_IDLE_ON);
44            break;
45        case STATE_IDLE: //打盹完了，呼吸一下就是这里了
46            scheduleAlarmLocked(mNextIdlePendingDelay, false);
47            mState = STATE_IDLE_MAINTENANCE; //状态更新为STATE_IDLE_MAINTENANCE
48            mNextIdlePendingDelay = Math.min(mConstants.MAX_IDLE_PENDING_TIMEOUT,
```

```
49             (long)(mNextIdlePendingDelay * mConstants.IDLE_PENDING_FACTOR));
50             //更新下次呼吸的时间
51             mHandler.sendMessage(MSG_REPORT_IDLE_OFF);
52             break;
53         }
54     }
```

Math.min(mConstants.MAX\_IDLE\_PENDING\_TIMEOUT,  
(long)(mNextIdlePendingDelay \* mConstants.IDLE\_PENDING\_FACTOR));  
这一句看到了吗？取最小值，这里就是保证了idle和窗口的时间不会变成无限大。  
为了让各位有个感官的体验，上面的一些时间我直接列出来吧

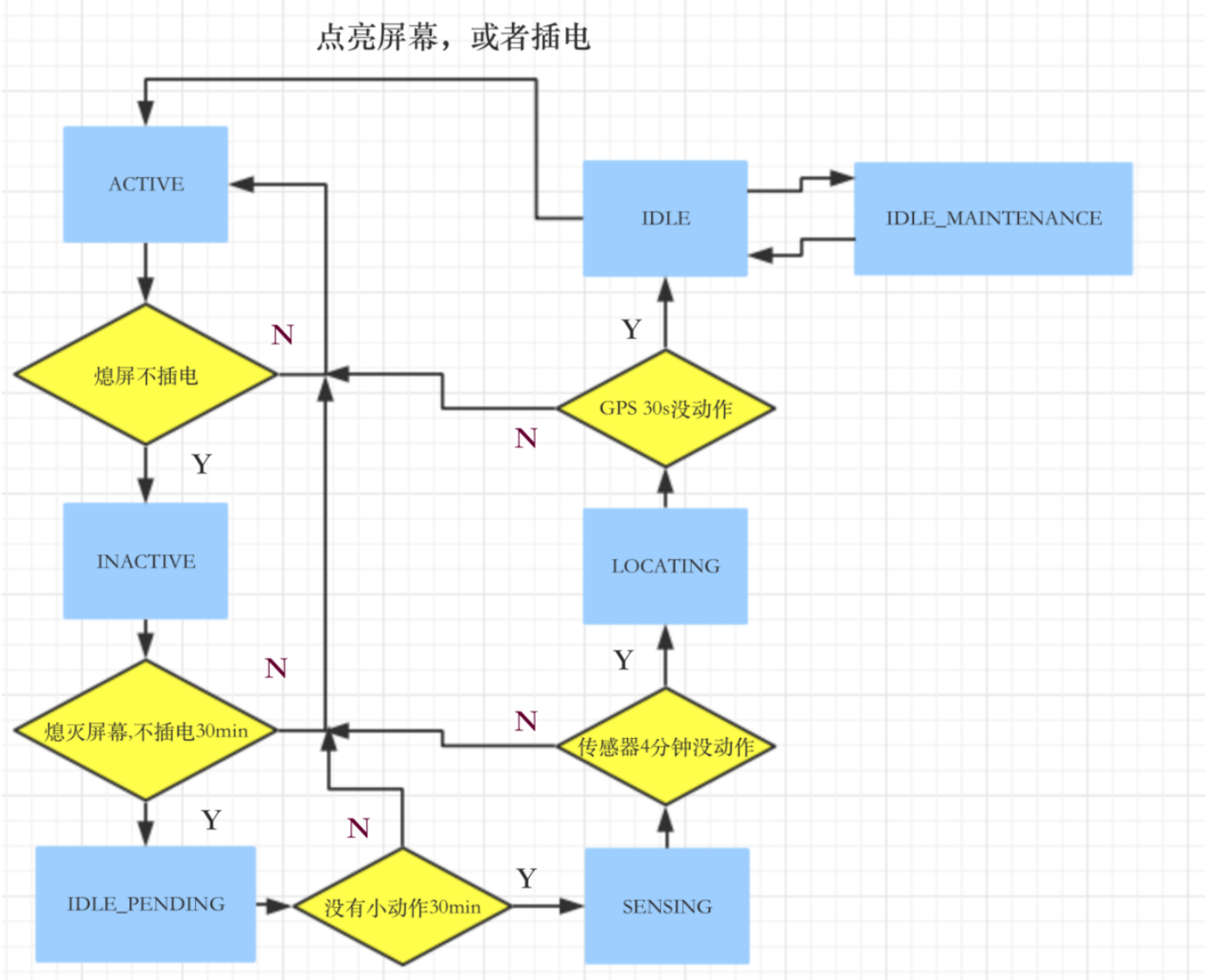
熄屏不插电进入INACTIVE时间上面说了30分钟

观察小动作的时间30分钟  
IDLE\_AFTER\_INACTIVE\_TIMEOUT = mParser.getLong(KEY\_IDLE\_AFTER\_INACTIVE\_TIMEOUT,  
!COMPRESS\_TIME ? 30 \* 60 \* 1000L : 3 \* 60 \* 1000L);

观察传感器的时间4分钟  
SENSING\_TIMEOUT = mParser.getLong(KEY\_SENSING\_TIMEOUT,  
!DEBUG ? 4 \* 60 \* 1000L : 60 \* 1000L);

观察GPS的时间30秒  
LOCATING\_TIMEOUT = mParser.getLong(KEY\_LOCATING\_TIMEOUT,  
!DEBUG ? 30 \* 1000L : 15 \* 1000L);

所以进入idle的总时间为30分钟+30分钟+4分钟+30s=1小时4分钟30秒，哈哈哈哈！！  
下面给张状态转换图看看，没到达idle状态前，基本上有什么风吹草动都会变回ACTIVE状态。而变成IDLE状态后，只能插电或者点亮屏幕才离开IDLE状态。就像人入睡前，很容易被吵醒，而深度入眠后，估计只有闹钟能闹醒你了！！



上面说了这么多，跟我应用开发有什么关系？  
其实，没多大关系，看下源码不行嚒。  
不过作为一种新的机制，最好测试下你的应用在这几种状态下是否能够正常运行，起码不能挂掉啊。  
google提供了adb的指令来强制变换状态，这样你就不用干等着它状态变化了。

```
1 adb shell dumpsys battery unplug //相当于不插电
```

```
2 adb shell dumpsys device idle step //让状态转换
```

转自：http://www.jianshu.com/p/8fb25f53bed4?utm\_campaign=haruki&utm\_content=note&utm\_medium=reader\_share&utm\_source=qq#

分类: [Android Framework](#)

标签: [Doze](#)

[好文要顶](#)[关注我](#)[收藏该文](#)



屌丝迷途

关注 - 0

粉丝 - 3

+加关注

0

0

« 上一篇：[Setting 之dashboard 点击跳转流程](#)  
» 下一篇：[简介](#)

posted on 2017-02-07 14:02 [屌丝迷途](#) 阅读(1049) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#)网站首页。

- 【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库
- 【推荐】极光开发者服务平台，五大功能一站集齐
- 【推荐】腾讯云域名+云解析 限时折扣抓紧抢购
- 【推荐】阿里云“全民云计算”优惠升级
- 【推荐】一小时搭建人工智能应用，让技术更容易入门



最新IT新闻:

- [DevOps团队之殇](#)
  - [魅族官微一天三发声明：Flow耳机事件来龙去脉详解](#)
  - [雷军：我给技术类创业者的3个建议](#)
  - [联通联合lephone、芒果TV推视频芒卡 月租9元含40G流量](#)
  - [Android 8.0 Oreo需手动允许每一个“未知来源”app的安装](#)
- » [更多新闻...](#)



最新知识库文章:

- [做到这一点，你也可以成为优秀的程序员](#)
  - [写给立志做码农的大学生](#)
  - [架构腐化之谜](#)
  - [学会思考，而不只是编程](#)
  - [编写Shell脚本的最佳实践](#)
- » [更多知识库文章...](#)