

# Android系统电池管理（PowerManagerService）框架分析及其在实体设备和虚拟设备上的差异（一）

📅 2020-04-24

🔖 android man pow roi service

## 0 前言

1. 移动设备的电池一直是影响移动设备用户体验的关键问题之一，电池容量、充电速度、以及系统待机时长都在用户考虑范围内。
2. 在众多嵌入式设备系统中，Android系统的设计的主要目的就是为通过触摸屏/显示屏和使用者交互，接收用户的触控指令并显示出用户所要的结果，其具有其他嵌入式系统没有的优秀的UI系统，但Android设备在使用中最耗电的也就基本是屏幕了（除了功率放大器之类的器件：LTE、WIFI、Bluetooth 等之外）。
3. 所以，现实中厂商一方面尽量增大电池容量、提升充电速度，另一方面尽可能降低Android系统的耗电量，或者减少不必要的电量浪费。
4. 这也是本文讨论的重点：**Android电源管理（用户空间+内核空间）**，代码来源Android AOSP 7.1.1版本。

## 1 电源管理概述

### 1.1 Linux 系统和Android系统的电源状态

ACPI State	Linux State	Description
S0	On(on)	正常工作状态
S1	Standby(standby)	CPU与RAM供电正常，但CPU不执行指令
S2	—	比S1更深的睡眠层次，这种模式通常不采用
S3	Suspend to RAM (mem)	挂起到内存
S4	Suspend to Disk(disk)	挂起到硬盘
S5	Shutdown	Soft Off, CPU、外设等断电，但电源依旧会为部分极低耗电设备供电
S6	Mechanical Off	全部断电

1.  
On：正常工作状态。
2.  
Standby也属于睡眠的一种方式，属于浅睡眠。  
该模式下CPU并未断电，依旧可以接收处理某些特定事件，视具体设备而定，恢复至正常工作状态的速度也比STR更快，但也更为耗电。举个例子来说，以该方式进入睡眠时，后续通过点击键盘也能将系统唤醒。而以mem进入的睡眠为深度睡眠，只能通过中断唤醒设备唤醒系统，如电源键(此时按电源键，不会经过正常的开机流程的BIOS、BOOTLOAD等)，此时按键盘是无法唤醒系统的。

码农家园

挂起到内存，俗称待机、睡眠（Sleep），进入该状态，系统的主要工作如下：

- 1) 将系统当前的运行状态等数据保存在内存中，此时仍需要向RAM供电，以保证后续快速恢复至工作状态
- 2) 冻结用户态的进程和内核态的任务（进入内核态的进程或内核自己的task）
- 3) 关闭外围设备，如显示屏、鼠标等,中断唤醒外设不会关闭，如电源键
- 4) CPU停止工作

4.

STD(Suspend to Disk):

挂起到硬盘，俗称休眠（Hibernation）将系统当前的运行状态等数据保存到硬盘上，并自动关机。下次开机时便从硬盘上读取之前保存的数据，恢复到休眠关机之前的状态。

譬如：在休眠关机时，桌面打开了一个应用，那么下一次开机启动时，该应用也处于打开状态。而正常的关机-开机流程，该应用是不会打开的。

??睡眠与休眠是2个不同的概念，睡眠属于STR，而休眠属于STD

??在我的Ubuntu18.04虚拟机上执行如下命令，可发现系统支持四种状态：

```
1 root@ubuntu:/home/zzb# cat /sys/power/state
2 freeze standby mem disk
```

??在我的一台Nexus 6P手机和一个emulator系统上同时执行同样命令，可发现实体手机和emulator系统都只支持mem这种睡眠方式。也就是说Android系统没有Linux意义上的休眠，只有睡眠（待机）状态来节电：

```
1 angler:/ # cat /sys/power/state
2 freeze mem
```

1.2 Android系统的Idle State

Android上的Idle状态分为二类：Cpu Idle和Device Idle

1.2.1 Cpu Idle

??Linux系统运行的基础是基于进程调度，实际上内核调度的线程（task），内核并不会区分线程与进程，都将他们当做一个线程（task）来处理；当所有的进程都没事儿干的时候，系统就会启用idle进程，使系统进入低功耗状态（如关闭一些服务、模块功能，降低CPU工作频率等），即idle状态，以达到省电的目的。

??idle状态又可以划分为不同的层级，以MTK的芯片为例，通常划分为以下几个状态:

State s	Description
soidle (screen idle)	亮屏 Idle 模式，该模式下与正常工作状态差别不大，唯一的区别就cpu处于空闲状态
rgidle	浅度 Idle 模式，cpu处于 WFI (wait for interrupt)，屏幕熄灭，同时关闭一些不需要的服务及模块，注意此状态cpu的时钟源与RTC模块是工作正常的，此时是可以通过TimerTask的定时触发激活系统的，TimerTask依赖于CPU的RTC模块，而Alarm则依赖于PMIC的RTC模块
dpidle (deep idle)	深度idle模式，该模式下cpu的时钟源和hrtimer（高精度定时器模块（RTC））被关闭，所有进程（包括系统进程）被冻结，即进入上文所述的睡眠状态

## 码农家园

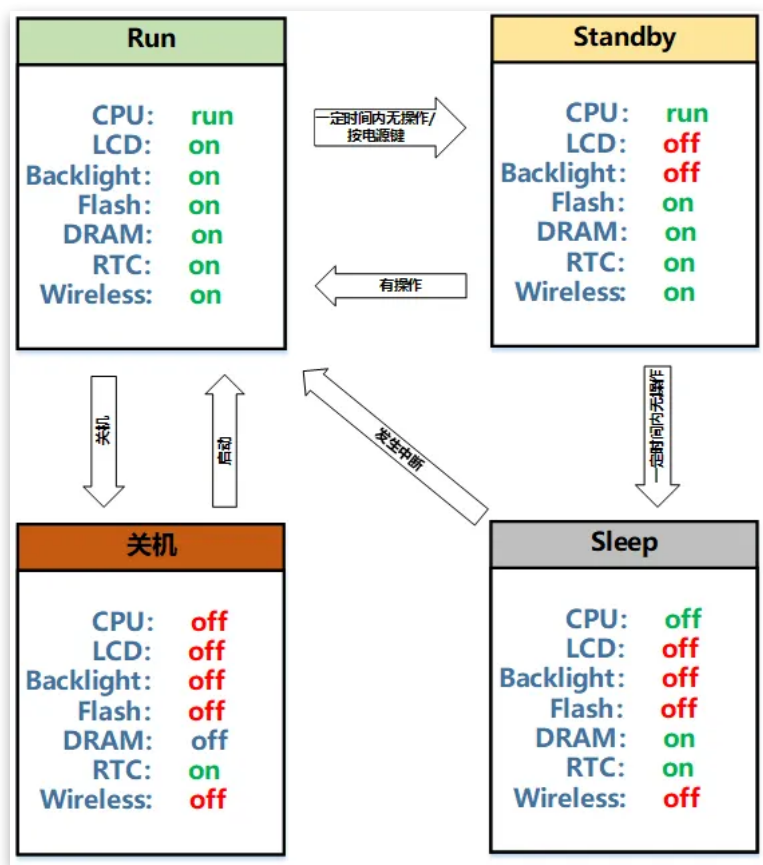
进入Doze模式。Doze模式的限制：

- 1. 网络接入被暂停
- 2. 系统忽略wake locks
- 3. 系统不会扫描Wi-Fi
- 4. 系统不允许sync adapters运行
- 5. 系统不允许JobScheduler运行

??结合以上分析的cpu idle不难发现Doze模式中的idle状态在概念属于浅idle状态，只是关闭了一些特定服务和模块，并非立即进入睡眠。

### 1.3 移动设备中主要模块工作的迁移状态

- Run 状态：系统正在运行状态，此时耗电最高。
- Standby 状态：系统未激活，Backlight为off，CPU时钟频率也降低。
- Sleep 状态：系统保持Standby状态时，关闭大部分周边设备的电源，CPU也进入Sleep状态，随机存储器或寄存器等挥发性内存可以选择性供电。RTC也为on状态。在系统时钟为off状态时，必须为RTC供电以计算时间。
- Shutdown 状态：系统虽然是Power off状态，但下一个Power on时，为实现时间同步及Power off提示等功能，RTC电源应一直保持on状态。



### 1.4 对Android来说电源管理的主要目的是什么？

??其实有两种重要场景，第一个是在不同环境为了节电或非节电目的需要调整屏幕亮度；第二个是在锁屏后要不要让手机进入Sleep状态应该由用户决定而不是系统。

- 屏幕亮度控制（LCD、Backlight）
  - 1. 一定时间内未操作设备，屏幕亮度会降低，然后将屏幕置为off
  - 2. 手动按电源键关闭屏幕，屏幕的on/off操作出发内核空间的LateResume及Early Suspend操作（这两个操作是Linux内核中为Android平台新添加的概念）。

## 码农家园

ep状态, 而是让这些应用一直运行 (比如执行下载任务、听歌等应用), 于是Android引入了WakeLock (唤醒锁) 的概念, 意思是: **在唤醒状态挂起锁设备。**


### 2 Android电源管理的系统架构

??Android平台的电源管理是基于Linux内核电源管理的, 传统Linux内核通常运行在有源状态下的设备中没有考虑移动设备的用电特点。所以, Android平台特地添加了针对移动设备用电特点的电源管理服务-Power Manager Service; 并在内核空间中引入了由用户空间Power Manager Service控制的LateResume、Early Suspen以及WakeLock概念。

#### 2.1 Android 电源管理的层级结构

??熟悉Android框架的知道, 整个Android电源管理框架也遵循Android的整体架构, 如下图所示:


1. 应用层: 包括PowerManagerService的Client应用程序, 及PowerManager。
2. App Framework层: 包括运行在system\_server中的线程之一: Java的PowerManagerService服务。
3. JNI层: 由PowerManagerService服务中Java Native代码的本地方法调用的实现函数。
4. HAL层: 抽象画化了Android用户空间到内核空间硬件设备的访问, 存在于Android电源管理的Legacy HAL中的库libpower.so将WakeLock的信息传递到内核空间。

Android电源管理结构: 层级结构+C/S

#### 2.2 PowerManager

??和众多系统服务一样, 为了使应用程序客户端访问远程服务, Android通过提供包装类 (Wrapper) 的管理器类提供利用远程服务的间接性标准化方法。

??PowerManager类由系统Fetcher类生成对象, 此时, RPC Binder的asInterface()方法返回PowerManagerService的代理对象引用。PowerManager可以通过获取返回的代理对象引用访问PowerManagerService, 如下图所示:

获取PowerManagerService代理对象引用

??客户端通过Android Java空间提供的getSystemService()方法可获得PowerManager类的对象引用, 进而可以由PowerManager随时访问PowerManagerService服务。

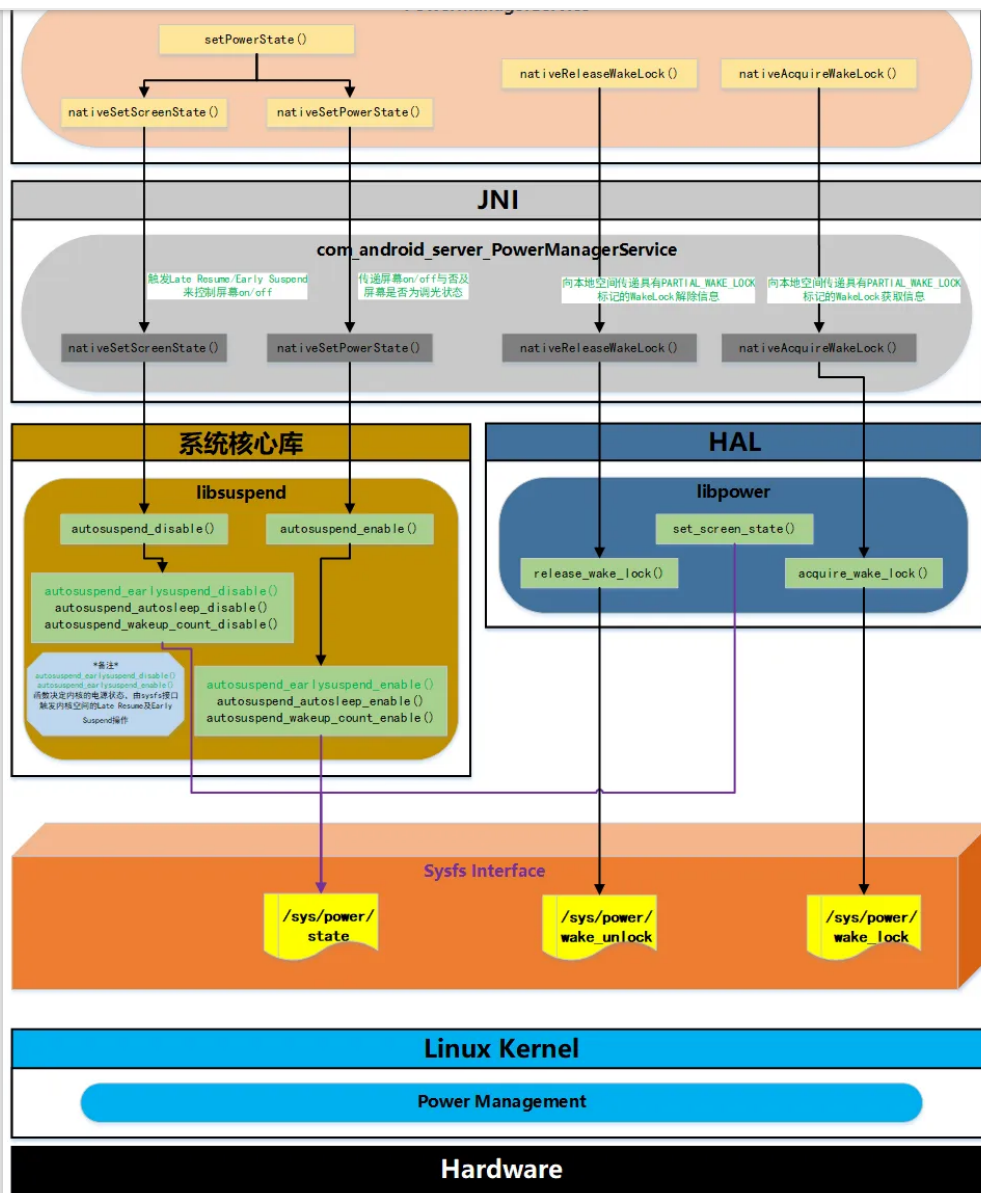
#### 2.3 PowerManagerService、本地空间中电源管理涉及到的主要方法调用流程

??PowerManagerService类在Android Java空间继承了IPowerManager.Stub服务存根类。实现PowerManagerService的实质性功能。PowerManagerService的主要操作是控制屏幕亮度和防止Sleep, 并将相应信息传递至本地及内核空间。

??下图中setPowerState()为决定PowerManagerService电源状态并根据电源状态控制灯光对象亮度的核心方法, 其能将屏幕亮度信息传递给本地空间, 其在系统框架中的调用流程已在图中标注。



## 码农家园



??其中这些函数的具体作用，我后面在分析代码时会涉及到一些。

## 2.4 内核空间

??对于内核而言，其以用户空间传递的信息为基础进行设备系统电源管理。Android平台在Linux原内核管理的基础上引入了**WakeLock**概念，即新增：Early Suspend状态和Late Resume状态。

??下图为Android内核电源管理流程图，系统由Running状态首先进入Early Suspend状态，直到屏幕熄灭时保持Running状态。在CPU Sleep状态下发生指定中断时，即使系统进入Resume状态也不会像Linux内核那样再次激活全部中断，而是只激活几个指定的中断，将耗电量降至最低。

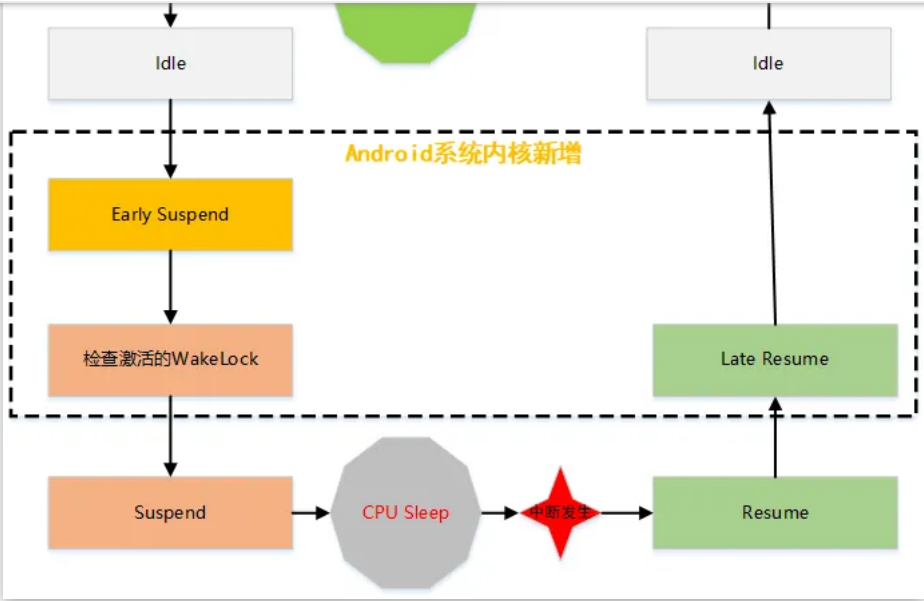
??**Early Suspend**：预挂起机制是Android特有的挂起机制，这个机制作用是关闭一些与显示相关的外设，比如LCD背光、重力感应器、触摸屏，但是其他外设如WIFI、蓝牙等模块等并未关闭。

此时，系统依旧可以处理事件，如音乐播放软件，息屏后依旧能播放音乐。

需要注意的是Early Suspend机制与**WakeLock**机制相互独立，就算有应用持有**wakelock**锁，系统依旧可以通过Early Suspend机制关闭与显示相关的外设。

??**Late Resume**：迟唤醒机制，用于唤醒预挂起的设备

??通常，息屏后，系统将先通过Early Suspend机制进入Idle状态，如果满足进入睡眠的条件（没有进程持有唤醒锁）则会通过Linux的Suspend机制进入Sleep(睡眠)状态。



2.4.1 WakeLock是什么

??唤醒锁，一种锁机制，用于阻止系统进入睡眠状态，只要有应用获取到改锁，那么系统就无法进入睡眠状态。

??该机制起初是早期Android为Linux内核打得一个补丁，并想合入到linux内核，但被Linux社区拒绝，后续Linux内核引入自己的Wakelock机制,Android系统也使用的是linux的Wakelock机制，所以该机制并非Android特有的机制。

??Android系统提供了两种类型的锁，每一个类型又可分为超时锁与普通锁，超时锁，超时会自动释放，而普通锁则必需要手动释放：

2.4.2 睡眠触发

在wakelock中，有3个地方可以让系统从early\_suspend进入suspend状态。

1. wake\_unlock，系统每释放一个锁，就会检查是否还存其他激活的wakelock，若不存在则执行Linux的标准suspend流程进入睡眠状态
2. 在超时锁的超时回调函数，判断是否存在其他激活的wakelock，若不存在，则进入睡眠状态
3. autosleep机制，android 4.1引入该机制，亮屏时会向autosleep节点写入off，熄屏则会写入mem。Android一灭屏，就会尝试进入睡眠，失败之后系统处于idle进程超过一定时间，则又尝试进入睡眠，判断标准同上，若存在wakelock则进入失败。

可以adb连接一台手机：

```
1 angler:/ # ls /sys/power/
2 autosleep pm_async pm_freeze_timeout state wake_lock wake_unlock wakeup_count
```

也可以adb连接一台emulator：

```
1 arm64:/ # ls /sys/power/
2 mem_sleep pm_async pm_debug_messages pm_freeze_timeout pm_print_times pm_test
```



参考文章

- 1. Android电源管理基础知识整理
- 2. Android Doze模式源码分析
- 3. Android AutoSleep休眠机制

