**lowmemorykiller阀值设置详解，后台进程管理教程**

Android将进程分为6个等级,foreground（前台进程）、visible（可见进程）、secondary server（次要服务）、hidden（后台进程）、content provider（内容供应节点）、empty（空进程）它们按优先级顺序由高到低依次是:

**1. FOREGROUND\_APP**

This is the process running the current foreground app. We'd really rather not kill it!  
用户正在使用的程序. 这个设的太高,用户看到得就会是一个正在使用的程序莫名其妙的消失了,然后自动回到桌面(因为它被系统kill了)所以最好别动它.

**2. VISIBLE\_APP**This is a process only hosting activities that are visible to the user, so we'd prefer they don't disappear.  
跟FOREGROUND\_APP类似,用户正在使用/看得到. 它们的区别就是VISIBLE\_APP可能不是用户focus的程序,但是用户看得到,或者没有覆盖到整个屏幕,只有屏幕的一部分. 所以可以适当的比FOREGROUND\_APP高一点.

**3. SECONDARY\_SERVER**This is a process holding a secondary server killing it will not have much of an impact as far as the user is concerned.  
所有应用的service. 系统级的service比如PhoneService不属于这类,它们是绝不会被Android结束掉的. 所以这个可以适当的设高一点点~ 注意, HOME(SenseUI)也包括在这里 因此还是别设的太高. 要不每次返回桌面都得等它重新load,特别是widget多的.

**4. HIDDEN\_APP**This is a process only hosting activities that are not visible, so it can be killed without any disruption.  
本来属于1或者2的程序, 在用户按了"back"或者"home"后,程序本身看不到了,但是其实还在运行的程序,它们就属于HIDDEN\_APP了. 干掉没什么影响.. 不过要了解并不是所有属于这一类的就应该马上结束掉,像pushmail, locale, 闹钟,等都属于这一类. 因此还是别设的过高. 真正"应该"一点返回键就退出的程序(真正没用的程序)在下面.

**5. CONTENT\_PROVIDER**This is a process with a content provider that does not have any clients attached to it. If it did have any clients, its adjustment would be the one for the highest-priority of those processes.  
5和6的区别具体不太了解..这个也是用处不大,但是还是比EMPTY\_APP稍微有点用, 所以高点没关系.

**6. EMPTY\_APP**

This is a process without anything currently running in it. Definitely the first to go! This value is initalized in the constructor, careful when refering to this static variable externally.

完全没用,杀了它只有好处没坏处,第一个干它!  
查看系统的OOM设置可以:

cat /sys/module/lowmemorykiller/parameters/minfree

显示出的应该是6个数字,以逗号隔开,例如:  
1536,2048,4096,5120,5632,6144

注意这些数字的单位是page. 1 page = 4 kb.上面的六个数字对应的就是(MB): 6,8,16,20,22,24  
这些数字也就是对应的内存阀值,一旦低于该值,Android便开始按顺序关闭进程. 因此Android开始结束优先级最低的EMPTY\_APP当可用内存小于24MB(6144\*4/1024).  
有一点没搞明白,它的可用内存不知道是从哪得到.明显不是free显示的可用内存,而且貌似compcache跟swap也不影响..  
  
要想重新设置该值:

echo "1536,2048,4096,5120,15360,23040" > /sys/module/lowmemorykiller/parameters/minfree

这样当可用内存低于90MB的时候便开始结束EMPTY\_APP. 而当可用内存低于60MB的时候才开始结束CONTENT\_PROVIDER组. 其余四个没动.  
**注意**:  
通过以上方法改变的数值并非永久.在下次重启后就又恢复到之前的设置. 若想让设置在每次开机执行,将

echo "1536,2048,4096,5120,15360,23040" > /sys/module/lowmemorykiller/parameters/minfree

加入到任意一个开机启动的配置文件. 一般在/system/init.d下的文件都是开机执行的(有的ROM也可能不在这里..) 只需用记事本打开任意一个文件,再把这行加入其中就好.

**存在一些问题：**

1. 如果各类进程的管理策略的阀值相当接近：6，8，16，20，22，24，最大的相差也不到8MB，在实际程序运行中，很容易导致多种类型的进程同时被关闭。如可用内存在25时，突然启动照相程序，系统可用内存急速，可能会导致空进程、内容供应节点、后台进程、次要服务等同时被关闭
2. 阀值上限较低：一般手机启动后，可用内存在50-100左右，但随着手机的使用，可用内存会逐步减少，最后降低到24MB左右，则系统开始启动进程管理机制，开始结束进程，但这个阀限制设在了24MB，相对来说偏低。其结果会导致系统使用一段时间后，整体速度变慢。很明显的就是，当手机长时间使用后，开启电话拨号，相册，照相机等应用时，系统的反应速度极慢。
3. 设置这个minfree阀值最难的地方是“后台进程”，设低了，ANDORID不会自动关闭你已经不用的“后台进程”。设太高的话，会自动关闭你还想用的“后台进程”，很多服务软件都是以后台进程模式提供服务的。

**基于以上几个问题，不难看出，目标也非常明确，主要解决两个矛盾：**

1. 拉开各进程的阀值层次，使得进程管理机制能更有效得工作
2. 提升阀值上限，空出更多的空余内存，以提升系统整体的运行速度

**进程管理策略设置原则：**

1. 前台进程、可见进程和次要服务是与用户体验息息相关的内容，这部分的进程管理策略要相对保守，给这些进程留下足够的运行空间
2. 压榨无用进程，腾出内存空间给主要程序使用

**下面总结了几种设置方式，适应不同的使用需要：**

* **游戏玩家/重度浏览器使用者配置：**
  + 用户特点：长时间专注于某一特定的，高内存需求的程序，对多任务的需求不高
  + 配置参数：
    - Foreground：6
    - Visible：8
    - Secondary Server：16
    - Hiden App：80
    - Content Provider：90
    - Empty:100
  + 配置理念：压榨后台进程，内容供应节点和空进程，将内存尽可能多得留给前台进程和系统，提升游戏速度和浏览器体验
  + 优点：程序启动和运行的速度最快
  + 缺点：多任务处理不理想，开启程序较多时，后台进程会被终止
* **多任务配置：**
  + 用户特点：同时运行多个应用程序，需要经常在多个程序间切换
  + 配置参数：
    - Foreground：6
    - Visible：8
    - Secondary Server：16
    - Hiden App：20
    - Content Provider：60
    - Empty:100
  + 配置理念：压榨空进程，给内容供应节点留有一定空间，最大限度提升后台程序的使用空间，提升多任务的处理能力
  + 优点：运行多个程序时，由于可支配内存较多，后台程序不容易被终止
  + 缺点：程序启动的速度和整体系统的运行速度可能会比游戏玩家配置略慢一些，由于经常运行多任务，平时系统的响应速度会受到一定影响
* **轻度用户/女生专用配置**
  + 用户特点：手机的主要功能是短信和电话，偶尔用用相机自拍
  + 配置参数：
    - Foreground：6
    - Visible：8
    - Secondary Server：16
    - Hiden App：24
    - Content Provider：32
    - Empty:48
  + 配置理念：压榨空进程，给内容供应节点留有一定空间，最大限度提升后台程序的使用空间，提升多任务的处理能力
  + 优点：比较均衡的配置，提升了系统的可用内存，使得系统的整体速度得到了提高，拉开了各级进程的管理策略层次，使得管理机制更有效率
  + 缺点：比较均衡的配置，无明显缺点