

# 叫我程某某

昵称：叫我程某某  
园龄：2年11个月  
粉丝：8  
关注：2  
+加关注

博客园 首页 新随笔 联系 管理 订阅 XML

随笔- 133 文章- 0 评论- 2

### [Android Memory] Android内存管理、监测剖析

转载自：<http://blog.csdn.net/anlegor/article/details/23398785>

#### Android内存管理机制：

Android内存管理主要有：LowMemory Killer机制，Ashmem，PMEM/ION及Native内存和Dalvik内存管理管理和JV M垃圾回收机制。

#### LowMemory Killer机制：

源码位置drivers/staging/Android/lowmemorykiller.c

Android是一个多任务系统，也就是说可以同时运行多个程序，这个大家应该很熟悉。一般来说，启动运行一个程序是有一定的时间开销的，因此为了加快运行速度，当你退出一个程序时，Android并不会立即杀掉它，这样下次再运行该程序时，可以很快的启动。随着系统中保留的程序越来越多，内存肯定会出现不足，low memory killer就是在系统内存低于某值时，清除相关的程序，保障系统保持拥有一定数量的空闲内存。

Low memorykiller根据两个原则，进程的重要性和释放这个进程可获取的空闲内存数量，来决定释放的进程。

进程的重要性，由task\_struct->signal\_struct->oom\_adj决定，

Android将程序的重要性分成以下几类，按照重要性依次降低的顺序，每个程序都会有一个oom\_adj值，这个值越小，程序越重要，被杀的可能性越低：

名 称	oom_adj	解释
FOREGROUD_APP	0	前 台程序，可以理解为你正在使用的程序
VISIBLE_APP	1	用户可见的程序
SECONDARY_SERVER	2	后 台服务，比如说QQ会在后台运行服务
HOME_APP	4	HOME，就是主界面
HIDDEN_APP	7	被 隐藏的程序
CONTENT_PROVIDER	14	内容提供者，
EMPTY_APP	15	空程序，既不提供服务，也不提供内容

除了上述程序重要性分类之外，Android系统还维护着另外一张表minfree用于维护内存警戒值，这两张表构成一个对应关系，两张表在"/sys/module/lowmemorykiller/parameters/"下保存。

## 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  
更多链接

## 随笔分类

- Android Memory(3)
- Android Studio(8)
- Android 之Activity(15)
- Android 之开发者(7)
- Android之Adapter(1)
- Android之ContentProvider(1)
- Android之html5(3)
- Android之Intent(4)
- Android之JSON(2)
- Android之Service(4)
- Android之SharedPreferences(2)
- Android之SQLite(4)
- Android之SurfaceView(5)
- Android之Widget(1)
- Android之XML(3)
- Android之安全通信(1)
- Android之菜单(1)
- Android之传感器
- Android之单元测试(3)

Android之多点触控(2)  
Android之多线程 ( Handler ) (6)  
Android之环境配置  
Android之基础知识(3)  
Android之框架(7)  
Android之数据结构(6)  
Android之图形图像(8)  
Android之网络编程(5)  
Android之文件操作(1)  
Android之优化技术(13)  
Android之游戏开发  
Android自定义控件(8)  
Ant  
Eclipse(1)  
Eclipse插件(3)  
Git  
Gradle  
IOS(1)  
Java基础(4)  
Java之安全通信  
Java之集合类(1)  
Java之数据结构  
Java之线程池和对象池  
Linux(1)  
Mac OS  
python(2)  
shell编程  
SVN(ubuntu)  
Ubuntu Setup(2)  
UML  
web(1)  
编码解码  
待解决的问题  
工具  
设计模式

## 随笔档案

2016年5月 (2)  
2015年7月 (13)  
2015年6月 (118)

## 最新评论

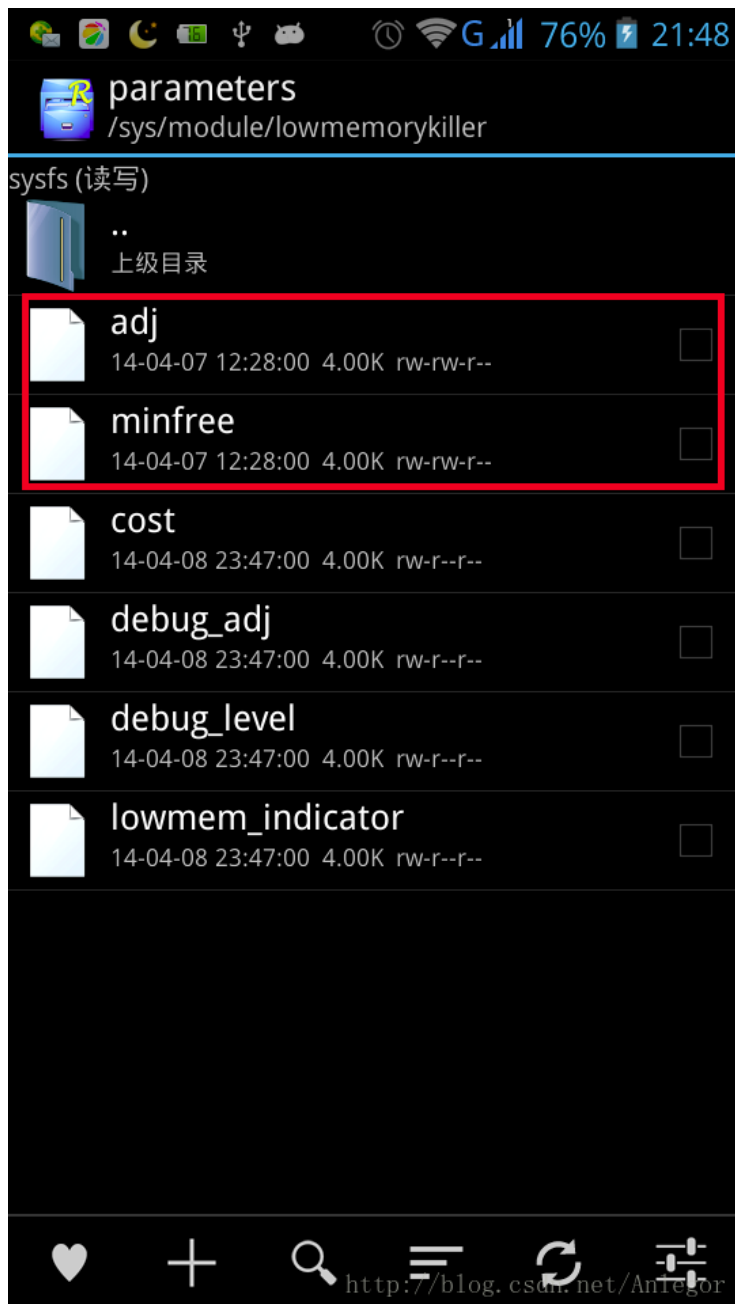
1. Re:Android SQLite学习指南  
给你顶下吧，哎，15个的浏览量  
--指尖下的幽灵  
2. Re:Android: 利用SurfaceView绘制股票  
票滑动直线解决延迟问题  
简单明了。。  
--WeiShao

## 阅读排行榜

1. socket、tcp、udp、http 的认识及区别(2592)  
2. 利用HTML5开发Android笔记 ( 上篇 ) (909)  
3. [Android Memory] Android内存管理、监测剖析(903)  
4. android 应用程序Activity之间数据传递与共享的几种途径(884)  
5. Android中View转换为Bitmap及getDrawingCache=null的解决方法(699)

## 评论排行榜

1. Android: 利用SurfaceView绘制股票  
票滑动直线解决延迟问题(1)



adj文件内容如下形如"0,1,2,4,9,15", minfree文件内容形如"8192,10240,12288,14336,16384,20480", 这样形成的表结构如下:

oom_adj	内存警戒值(以4K为单位)
0	8192
1	10240
2	12288
4	14336
9	16384
15	20480

例如, 当系统可用内存小于12384\*4K大小时, 会开始杀掉oom\_adj>=9级别的进程。  
进程的内存, 通过get\_mm\_rss获取, 在相同的oom\_adj下, 内存大的, 优先被杀。

Ashmem(匿名内存共享):

源码位置: kernel/mm/ashmem.c

为进程间提供提供大块共享内存, 同时为内核提供回收和管理这个内存的机制。

2. Android SQLite学习指南(1)

## 推荐排行榜

1. socket、tcp、udp、http 的认识及区别(1)

2. Android SQLite学习指南(1)

相比于malloc和anonymous/namedmmap等传统的内存分配机制，其优势是通过内核驱动提供了辅助内核的内存回收算法机制(pin/unpin)。什么是pin和unpin呢？具体来讲，就是当你使用Ashmem分配了一块内存，但是其中某些部分却不会被使用时，那么就可以将这块内存unpin掉。

unpin后，内核可以将它对应的物理页面回收，以作他用。你也不用担心进程无法对unpin掉的内存进行再次访问，因为回收后的内存还可以再次被获得(通过缺页handler)，因为unpin操作并不会改变已经 mmap的地址空间。

**AndroidPMEM/ION内存管理器：**

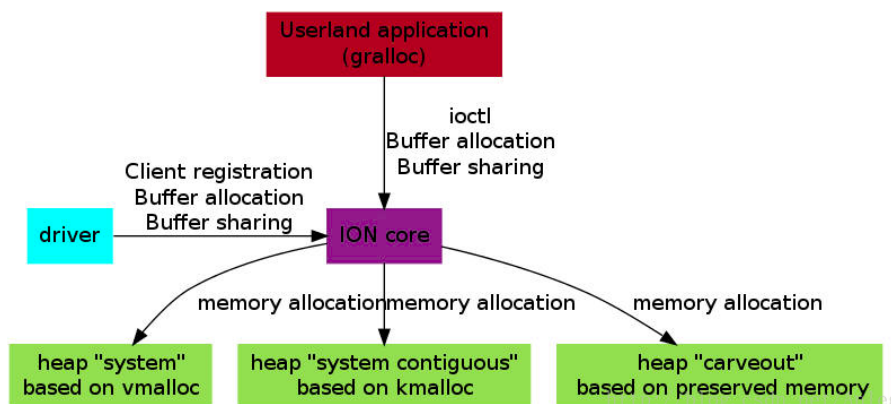
源码位置：drivers/misc/pmем.c

AndroidPmem是为了实现共享大尺寸连续物理内存而开发的一种机制，该机制对dsp，gpu等部件非常有用。Pmem相当于把系统内存划分出一部分单独管理，即不被linux mm管理，实际上linux mm根本看不到这段内存。

Pmem和Ashmem都通过mmap来实现共享内存，其区别在于Pmem的共享区域是一段连续的物理内存，而Ashmem的共享区域在虚拟空间是连续的，物理内存却不一定连续。dsp和某些设备只能工作在连续的物理内存上，这样cpu与dsp之间的通信就需要通过Pmem来实现。

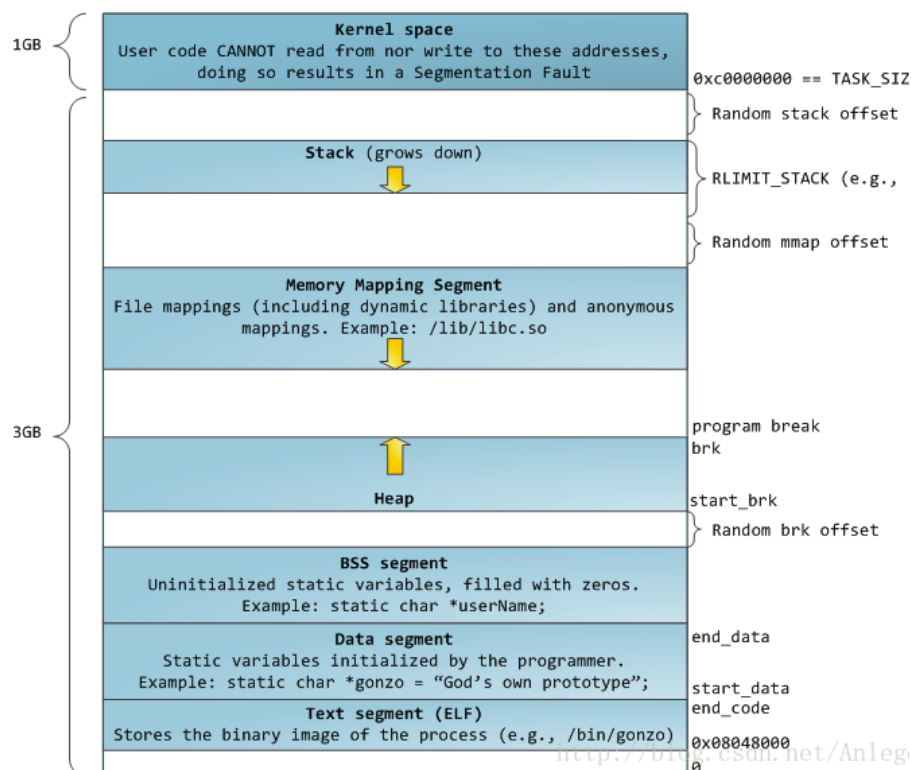
ION是google在Android4.0ICS为了解决内存碎片管理而引入的通用内存管理器，它会更加融合kernel。目前QCOM MSM, NVIDIA Tegra, TI OMAP, MRVL PXA都用ION替换PMEM。

ION 定义了四种不同的heap，实现不同的内存分配策略。

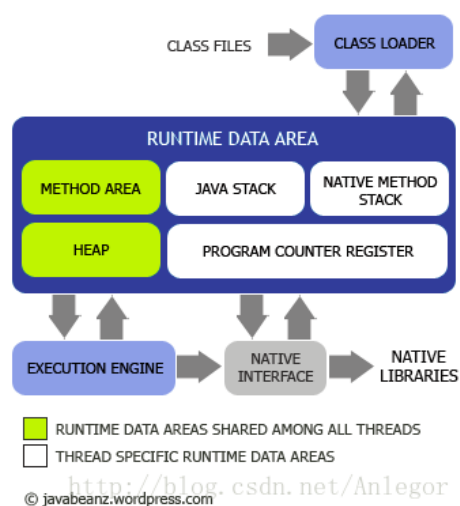


**AndroidNative内存和Dalvik内存管理：**

Native进程是采用C/C++实现，不包含dalvik实例的进程，/system/bin/目录下面的程序文件运行后都是以native进程形式存在的，如/system/bin/surfaceflinger、/system/bin/rild、procrank等就是native进程，地址空间结构如下：



java进程是Android中运行于dalvik虚拟机之上的进程。dalvik虚拟机的宿主进程由fork()系统调用创建，所以每一个java进程都是存在于一个native进程中，因此，java进程的内存分配比native进程复杂，因为进程中存在一个虚拟机实例。Android系统中的应用程序基本都是java进程，如桌面、电话、联系人、状态栏等等，进程结构如下：



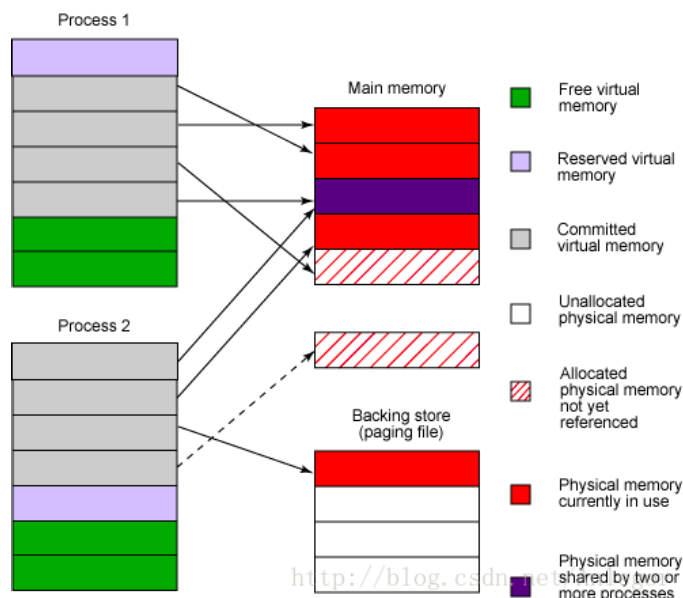
Stack空间（进栈和出栈）由操作系统控制，其中主要存储函数地址、函数参数、局部变量等等，所以Stack空间不需要很大，一般为几MB大小。

Heap空间的使用由程序员控制，程序员可以使用malloc、new、free、delete等函数调用来操作这片地址空间。Heap为程序完成各种复杂任务提供内存空间，所以空间比较大，一般为几百MB到几GB。正是因为Heap空间由程序员管理，所以容易出现使用不当导致严重问题。

进程空间中的heap空间是我们需要重点关注的。heap空间完全由程序员控制，我们使用的malloc、C++ new和java new所申请的空间都是heap空间，C/C++申请的内存空间在native heap中，而java申请的内存空间则在dalvik heap中。

进程的内存空间只是虚拟内存（或者叫作逻辑内存），而程序的运行需要的是实实在在的内存，即物理内存（RAM）。在必要时，操作系统会将程序运行中申请的内存（虚拟内存）映射到RAM，让进程能够使用物理内存。

RAM作为进程运行不可或缺的资源，对系统性能和稳定性有着决定性影响。另外，RAM的一部分被操作系统留作他用，比如显存等等，内存映射和显存等都是由操作系统控制，我们也不必过多地关注它，进程所操作的空间都是虚拟地址空间，无法直接操作RAM。示意图如下：



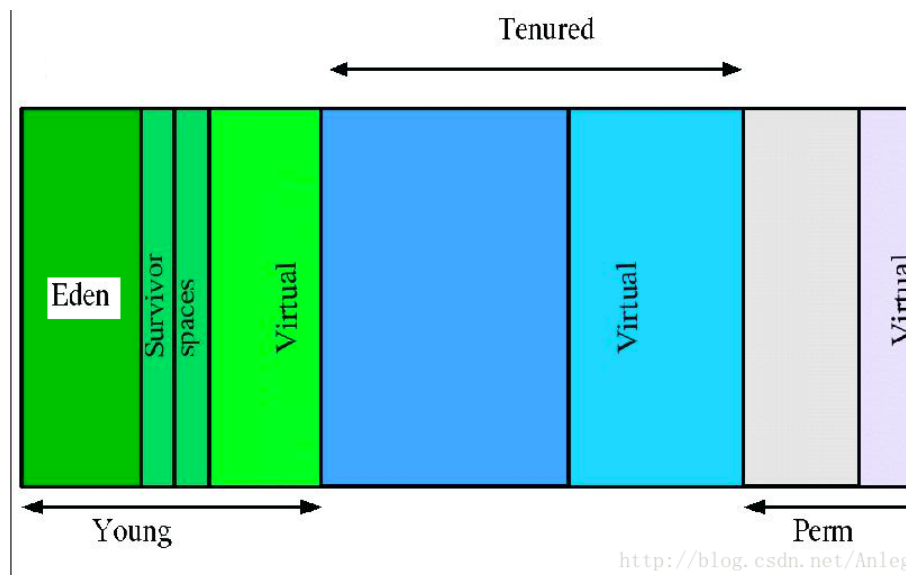
Android系统对dalvik的vm heapsize作了硬性限制，当java进程申请的java空间超过阈值时，就会抛出OOM异常，在/system/build.prop中有三项可以配置相关信息。

**dalvik.vm.heapstartsize**：堆分配的初始大小，调整这个值会影响到应用的流畅性和整体ram消耗。这个值越小，系统ram消耗越慢，但是由于初始值较小，一些较大的应用需要扩张这个堆，从而引发gc和堆调整的策略，会应用反应更慢。相反，这个值越大系统ram消耗越快，但是程序更流畅。

**dalvik.vm.heapgrowthlimit**：受控情况下的极限堆（仅仅针对dalvik堆，不包括native堆）大小，dvm heap是可增长的，但是正常情况下dvm heap的大小是不会超过dalvik.vm.heapgrowthlimit的值（非正常情况下面会详细说明）。这个值控制那些受控应用的极限堆大小，如果受控的应用dvm heap size超过该值，则将引发oom（out of memory）。

**dalvik.vm.heapsize**：不受控情况下的极限堆大小，这个就是堆的最大值。不管它是不是受控的。这个值会影响非受控应用的dalvikheap size。一旦dalvik heap size超过这个值，直接引发oom。

JVM垃圾回收机制：



JVM的垃圾原理是这样的，它把对象分为年轻代（Young）、年老代（Tenured）、持久代（Perm），对不同生命周期的对象使用不同的垃圾回收算法。

**年轻代(Young)**：年轻代分为三个区，一个eden区，两个Survivor区。程序中生成的大部分新的对象都在Eden区中，当Eden区满时，还存活的对象将被复制到其中一个Survivor区，当此Survivor区的对象占用空间满了时，此区存活的对象又被复制到另外一个Survivor区，当这个Survivor区也满了的时候，从第一个Survivor区复制过来的并且此时还存活的对象，将被复制到年老代。

**年老代（Tenured）**：年老代存放的是上面年轻代复制过来的对象，也就是在年轻代中还存活的对象，并且区满了复

制过来的。一般来说，年老代中的对象生命周期都比较长。

持久代（Perm）：用于存放静态的类和方法，持久代对垃圾回收没有显著的影响。

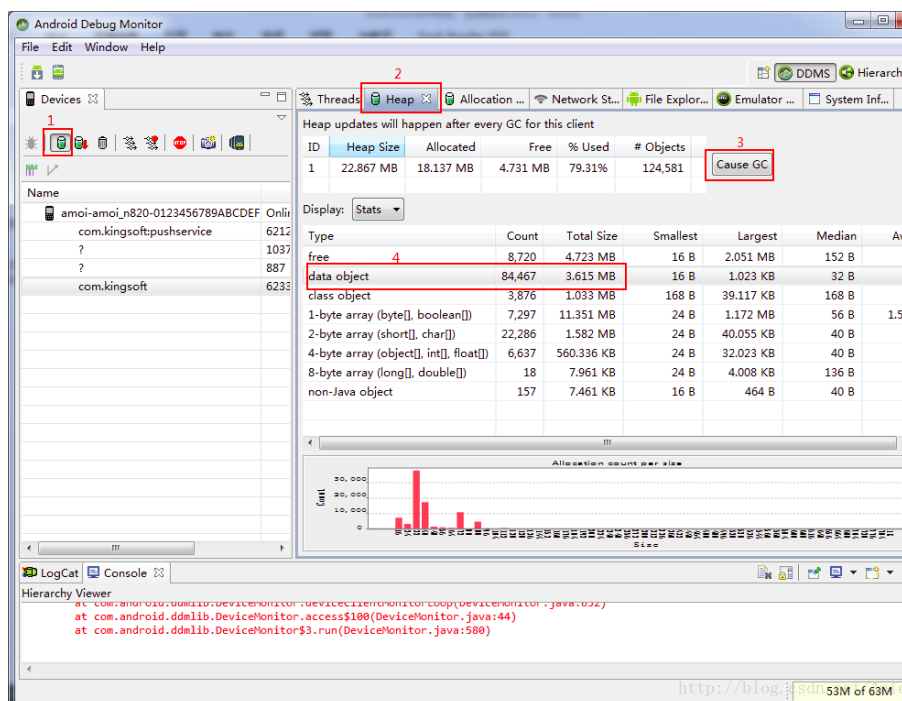
Android内存分析工具：

内存监控工具DDMS：

Ddms可以在eclipse中安装对应的插件，同时在androidsdk的tools文件夹也有同样的工具，并且功能比eclipse插件更完备。

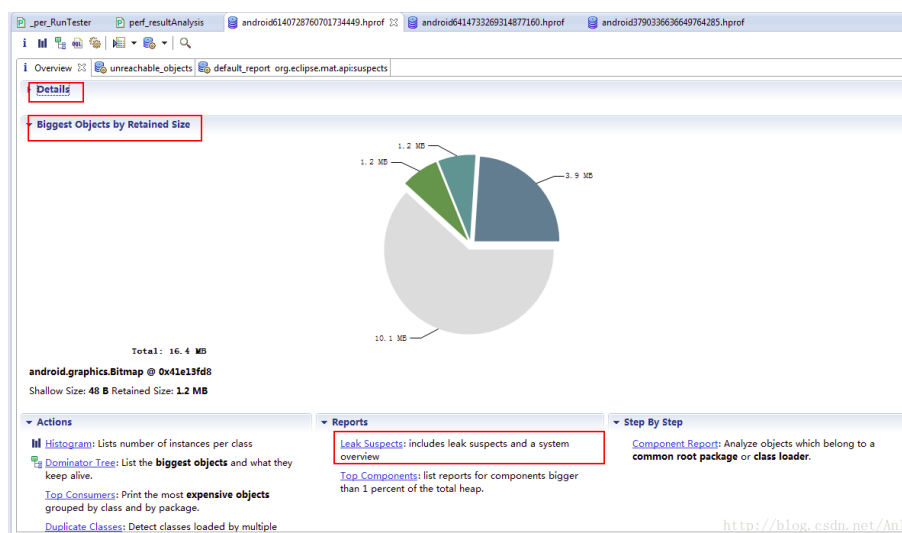
切换到eclipse的ddms视图后，从设备中选择要监控的进程（手机需要开启usb调试，被监控应用的manifest中android:debuggable应该为true）

如下图操作，我们主要关注dataobject类型的total size数据在使用过程中是否出现异常的数据波动。

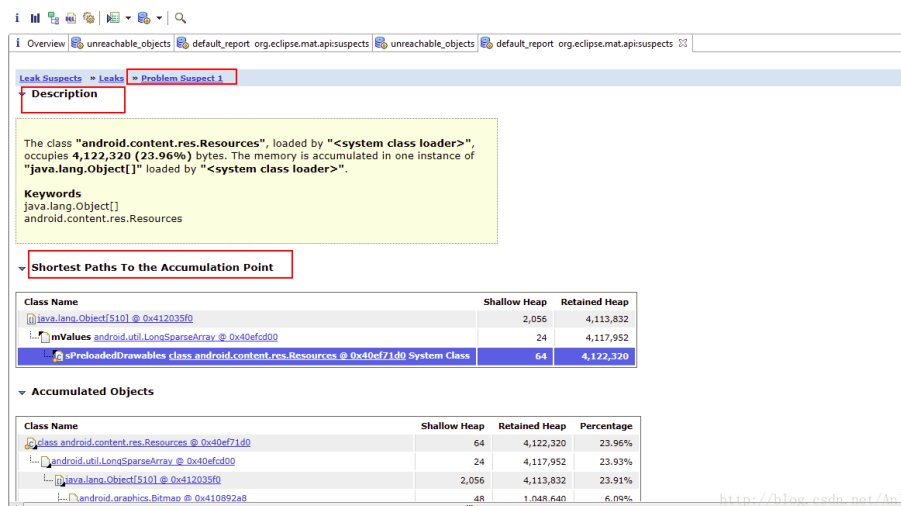


Eclipse MAT插件：

Mat是eclipse的一个插件，安装完成之后，在ddms视图下，同时在左侧设备视图上方选择“Update Heap”和“Dump HPROF file”按钮，在弹出的窗口中选择“Leak Suspects Report”并finish，即可出现下面的结果视图：



进入leaksuspects之后，就能列出所有可能有泄漏的地方以及代码片段



The class "android.content.res.Resources", loaded by "<system class loader>", occupies 4,122,320 (23.96%) bytes. The memory is accumulated in one instance of "java.lang.Object[]" loaded by "<system class loader>".

**Keywords**  
java.lang.Object[]  
android.content.res.Resources

**Shortest Paths To the Accumulation Point**

Class Name	Shallow Heap	Retained Heap
java.lang.Object[510] @ 0xd12035f0	2,056	4,113,832
mValues android.util.LongSparseArray @ 0xd40efcd00	24	4,117,952
sPreloadedDrawables class android.content.res.Resources @ 0xd40ef71d0 System Class	64	4,122,320

**Accumulated Objects**

Class Name	Shallow Heap	Retained Heap	Percentage
class android.content.res.Resources @ 0xd40ef71d0	64	4,122,320	23.96%
android.util.LongSparseArray @ 0xd40efcd00	24	4,117,952	23.93%
java.lang.Object[510] @ 0xd12035f0	2,056	4,113,832	23.91%
android.graphics.Bitmap @ 0xd10892a8	48	1,048,640	6.09%

Adb shell的dumpsys meminfo命令：

```
E:\Andriod\sdk\platform-tools>adb shell dumpsys meminfo com.kingsoft
Applications Memory Usage (kB):
Uptime: 117709487 Realtime: 299133867

** MEMINFO in pid 6233 [com.kingsoft] **

      Pss      Shared   Private  Heap     Heap     Heap
      Size    Dirty    Dirty    Size    Alloc    Free
-----
Native      0         0         0    35948     8584     663
Dalvik    16863    10436    16668    23415    19952    3463
Cursor      0         0         0
Ashmem      0         0         0
Other dev   3064     6160         0
.so mmap    3624     2816     680
.jar mmap   0         0         0
.apk mmap   113         0         0
.ttf mmap   243         0         0
.dex mmap   568         0         0
Other mmap  1026     420     348
Unknown    13388    1004    13372
TOTAL     38889    20836    31068    59363    28536    4126

Objects
  Views:      483      ViewRootImpl:      1
  AppContexts: 4      Activities:      3
  Assets:      5      AssetManagers:      5
  Local Binders: 30      Proxy Binders:      26
  Death Recipients: 1
  OpenSSL Sockets: 0

SQL
  MEMORY_USED:      85
  PAGECACHE_OVERFLOW: 12      MALLOC_SIZE:      62

DATABASES
  pgsz      dbsz      Lookaside(h)      cache      Dbname
  4         40         20      0/15/1      /data/data/com.kingsoft/datab
ases/webview.db

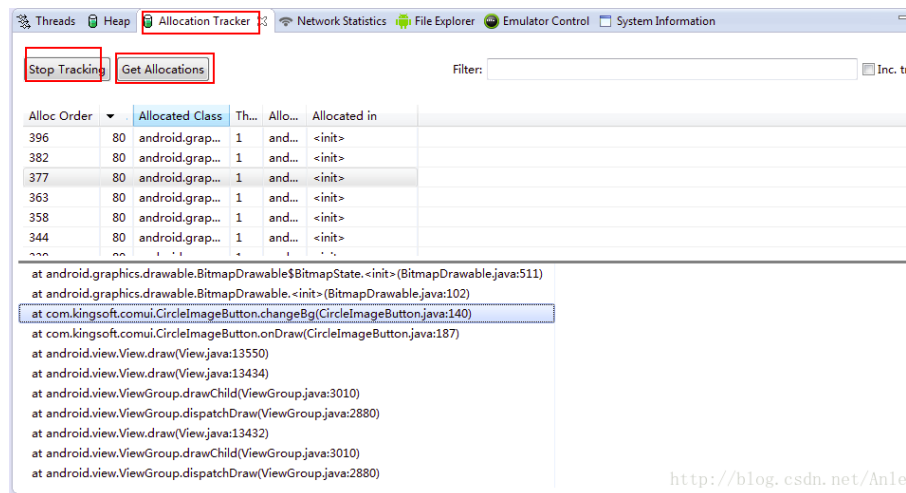
Asset Allocations
  zip:/mnt/asec/com.kingsoft-1/pkg.apk:/assets/fonts/segoeui.ttf: 505K
  zip:/mnt/asec/com.kingsoft-1/pkg.apk:/assets/fonts/segoeui.ttf: 505K
```

Adbshell的cat命令：

```
E:\Andriod\sdk\platform-tools>adb shell cat /proc/6233/status
Name:   com.kingsoft
State:  S (sleeping)
Tgid:   6233
Pid:    6233
PPid:   105
TracerPid:      0
Uid:     10242   10242   10242   10242
Gid:     10242   10242   10242   10242
FDs size: 256
Groups: 1006 1013 1015 1028 3003
UmlPeak: 370976 kB
UmlSize: 338072 kB
UmlLck:  0 kB
UmlPin:  0 kB
UmlHWM:  92996 kB
UmlRSS:  68596 kB
UmlData: 51456 kB
UmlStk:   136 kB
UmlExe:   8 kB
UmlLib:  32036 kB
UmlPTE:   196 kB
UmlSwap:  0 kB
Threads: 23
SigQ:    0/7840
SigPnd:  0000000000000000
ShdPnd:  0000000000000000
SigBlk:  0000000000009204
SigIgn:  0000000000001000
SigCgt:  00000002000084a8
CapInh:  0000000000000000
CapPrm:  0000000000000000
CapEff:  0000000000000000
CapBnd:  ffffffff00000000
Cpus_allowed:    3
Cpus_allowed_list: 0-1
voluntary_ctxt_switches: 193709
nonvoluntary_ctxt_switches: 1334345
```

应用内存分配跟踪工具Allocation tracker：

同样该功能用到ddms，只是里面提供的一个子功能而已，能够知道所有对象的分配是在代码的哪个类，哪个文件的哪一行。



AndroidOOM的常见原因：

非静态内部类的静态实例容易造成内存泄漏

activity使用静态成员

使用handler时的内存问题

注册某个对象后未反注册

集合中对象没清理造成的内存泄露

资源对象没关闭造成的内存泄露

一些不良代码成内存压力：Bitmap使用不当；构造Adapter时，没有使用缓存的 convertView；在经常调用的方



法中创建对象（例如循环）

参考资料：

Androidmemory fundamentals

[http://www.slideshare.net/tarasleskiv/android-memory-fundamentals?qid=c8462372-7470-4459-a76c-552c678724c0&v=qf1&b=&from\\_search=2](http://www.slideshare.net/tarasleskiv/android-memory-fundamentals?qid=c8462372-7470-4459-a76c-552c678724c0&v=qf1&b=&from_search=2)

Forensic Memory Analysis of Android'sDalvik Virtual Machine

[http://www.slideshare.net/SOURCEConference/forensic-memory-analysis-of-androids-dalvik-virtual-machine?qid=c8462372-7470-4459-a76c-552c678724c0&v=qf1&b=&from\\_search=6](http://www.slideshare.net/SOURCEConference/forensic-memory-analysis-of-androids-dalvik-virtual-machine?qid=c8462372-7470-4459-a76c-552c678724c0&v=qf1&b=&from_search=6)

Memory management for \_android\_apps

[http://www.slideshare.net/shaobin0604/memory-management-forandroidapps?qid=c8462372-7470-4459-a76c-552c678724c0&v=qf1&b=&from\\_search=11](http://www.slideshare.net/shaobin0604/memory-management-forandroidapps?qid=c8462372-7470-4459-a76c-552c678724c0&v=qf1&b=&from_search=11)

android内存管理机制分析

<http://www.doc88.com/p-992213371842.html>

Android 操作系统的内存回收机制

<http://blog.jobbole.com/25169/>

Android系统匿名共享内存（AnonymousShared Memory）C++调用接口分析

<http://blog.csdn.net/luoshengyang/article/details/6939890>

Android系统匿名共享内存Ashmem（AnonymousShared Memory）简要介绍和学习计划 - 老罗的Android之旅 - 博客频道 - CSDN.NET

<http://blog.csdn.net/luoshengyang/article/details/6651971>

程序源代码分析 - 老罗的Android之旅 - 博客频道 - CSDN.NET

<http://blog.csdn.net/luoshengyang/article/details/6664554>

Android系统匿名共享内存Ashmem（AnonymousShared Memory）在进程间共享的原理分析 - 老罗的Android之旅 - 博客频道 - CSDN.NET

<http://blog.csdn.net/luoshengyang/article/details/6666491>

Windows内存与进程管理器底层分析

<http://www.cnblogs.com/zudn/archive/2010/12/29/1920593.html>

什么是内存直接映像技术？它有何特点？

[http://zhidao.baidu.com/link?url=e8NFxuOaY9UsOxKGkIWmTR20-gpDGy48eE1SerGNCjAYbg3Xj-Fp0pqBSAKFy\\_AEiyKQOWmx3QHGN4ldWqec0M1wxBowDOT59Bua2O0DjZC](http://zhidao.baidu.com/link?url=e8NFxuOaY9UsOxKGkIWmTR20-gpDGy48eE1SerGNCjAYbg3Xj-Fp0pqBSAKFy_AEiyKQOWmx3QHGN4ldWqec0M1wxBowDOT59Bua2O0DjZC)

Mysteries of Windows Memory ManagementRevealed

<http://media.ch9.ms/teched/na/2011/ppt/WCL406.pptx>

全面介绍Windows内存管理机制及C++内存分配实例

<http://blog.csdn.net/yeming81/article/details/2046193>

技术内幕：Android对Linux内核的增强

[http://tech.it168.com/a2011/0805/1228/000001228471\\_all.shtml](http://tech.it168.com/a2011/0805/1228/000001228471_all.shtml)

android内存管理了解\_百度文库

[http://wenku.baidu.com/link?url=bGaPQbl7u1\\_JmKSJ5nVBv1dHtvGSXREyv\\_7PeSt9\\_FCx20I5oK1EaEgno9d6Ke\\_d4Kao92yoP\\_7X6gOOToylezMwEn-inKnBxb7SAoJ3iH6m](http://wenku.baidu.com/link?url=bGaPQbl7u1_JmKSJ5nVBv1dHtvGSXREyv_7PeSt9_FCx20I5oK1EaEgno9d6Ke_d4Kao92yoP_7X6gOOToylezMwEn-inKnBxb7SAoJ3iH6m)

Android low memory killer 详解-tuyer-ChinaUnix博客

<http://blog.chinaunix.net/uid-20321537-id-3228776.html>

Linux 设备文件简介

[http://www.jinbuguo.com/kernel/device\\_files.html](http://www.jinbuguo.com/kernel/device_files.html)

Explore Android Internals

[http://www.slideshare.net/jserv/android-internals-30176596?gid=a3be858d-e483-423c-be0e-a85e0a08fdd6&v=qf1&b=&from\\_search=4](http://www.slideshare.net/jserv/android-internals-30176596?gid=a3be858d-e483-423c-be0e-a85e0a08fdd6&v=qf1&b=&from_search=4)

Inside Android's Dalvik VM - NEJUG Nov 2011

[http://www.slideshare.net/dougqh/inside-androids-dalvik-vm-nejug-nov-2011?gid=a3be858d-e483-423c-be0e-a85e0a08fdd6&v=qf1&b=&from\\_search=3](http://www.slideshare.net/dougqh/inside-androids-dalvik-vm-nejug-nov-2011?gid=a3be858d-e483-423c-be0e-a85e0a08fdd6&v=qf1&b=&from_search=3)

android之ION内存管理器（1）-- 简介

<http://blog.csdn.net/crazyjiang/article/details/7927260>

Android进程的内存管理分析

<http://blog.csdn.net/gemmem/article/details/8920039>

Gc in android

<http://www.slideshare.net/VikasBalikai/gc-in-android>

Android内存泄漏分析及调试

<http://blog.csdn.net/gemmem/article/details/13017999>

Android OOM介绍及分析方法

<http://blog.csdn.net/gemmem/article/details/8964397>

分类: [Android Memory](#)



叫我程某某  
关注 - 2  
粉丝 - 8

+加关注

0

0

« 上一篇: [\[Android Memory\] Android性能测试小工具Emmagee](#)

» 下一篇: [\[Java基础\] Java多线程-工具篇-BlockingQueue](#)

posted @ 2015-06-13 18:42 叫我程某某 阅读(903) 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

【力荐】普惠云计算 0门槛体验 30+云产品免费半年

【推荐】可嵌入您系统的“在线Excel”! SpreadJS 纯前端表格控件

【推荐】阿里云“全民云计算”优惠升级



最新IT新闻:

- 太懒，还是太贪？大厂为何发行满屏Bug的游戏
  - 腾讯WeGame游戏平台全体验：速度很爽
  - 4G版Surface Pro、Surface Book 2代将于10月发布
  - 小米AI音箱虚拟形象小爱同学定妆：二次元萌妹
  - 作为马云的贵人，孙正义正在影响着未来的全球科技格局
- » 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...

Copyright ©2017 叫我程某某