# 调试工具种类

* attach debug to android process 
* Monitors 
* DDMS （Android Device Monitor）

# Monitors简介

1. Memory:

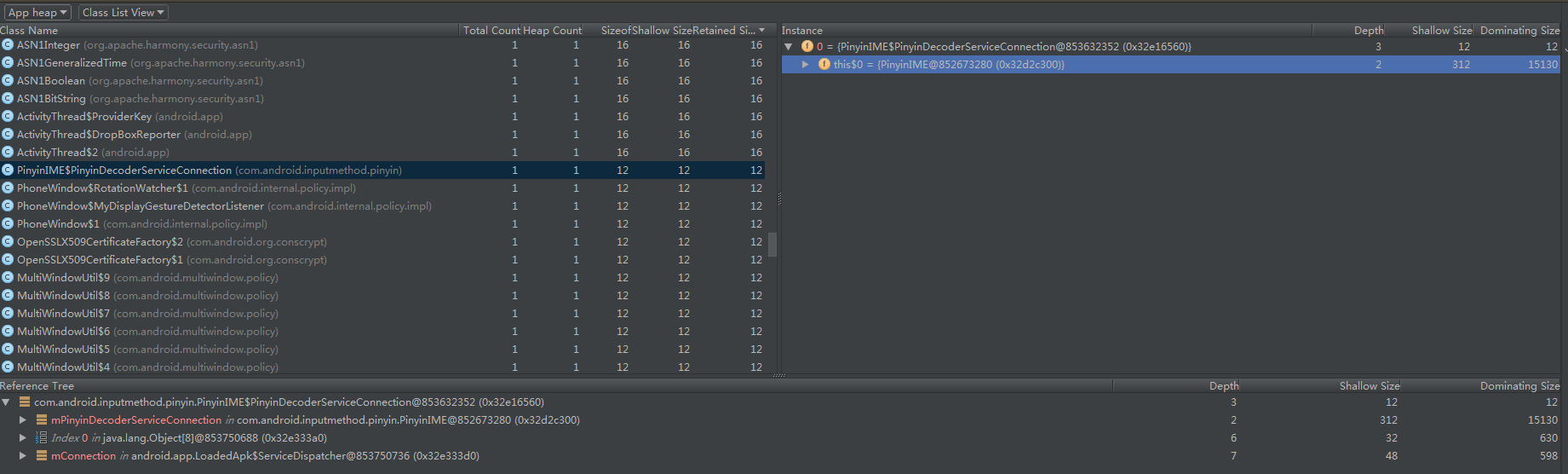


* （开启/关闭追踪）:开始追踪内存或不追踪。
* （启动GC）：强制进行GC。可以用于检查是否内存泄漏。
* （dump java heap）：

java heap是分配给实例类和数组对象运行数据区，所有java线程在运行期间共享heap中的数据。Java heap dump相当于java应用在运行的时候在某个时间点上打了个快照snapshot。

基本能够通过它观察出哪些对象占用了巨量内存，并且能够找出它们被什么对象把持住，导致无法释放。

点击Dump Java Heap后，APP会Freeze住。然后就是等待，这个时候最好别做其他事情，否则可能失败。大概几十秒后，就会进入读取hprof文件的界面了。



该面板里的信息可以有三种类型:app heap/image heap/zygote heap.

分别代表app 堆内存信息，图片堆内存信息，zygote进程的堆内存信息。

各属性中英文对照表

名称 Total Count Heap Count Sizeof Shallow size Retained Size

意义 内存中该类的对象个数 堆内存中该类的对象个数 物理大小 对象本身占有内存大小 释放该对象后，节省的内存大小

名称 depth Shallow Size Dominating Size

意义 深度 对象本身内存大小 管辖的内存大小

* （start allocation tracking）：

主要用于调试在复杂系统里面短时间内存暴增造成GC频繁的Case。使用方法：先点击一次，然后会看到Memory Recorder开始转动，然后自己开始在APP上面做相应的操作。在合适的时间再点一次，结束记录。

然后APP会Freeze，过一会儿就会进入到alloc文件的打开界面了。通过分析alloc数据，我们可以知道某一个thread里面的所有的调用过的方法所分配的堆大小，通过这些数据可以让我们针对方法级别堆程序进行优化。

1. CPU:

用于检测CPU占用情况，获取时间占用图以及调用次数。

<http://blog.csdn.net/xiaoyida11/article/details/52816688>

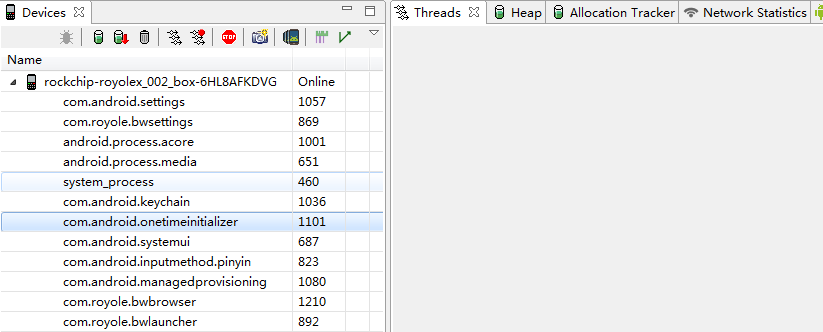
1. network:

简单的显示网络占用情况。

1. GPU:

暂时未接触。

# DDMS



Threads:



ID: 虚拟机分配的唯一的线程ID,在Dalvik里，它们是从3开始的奇数。

Tid：linux的线程ID号

Stauts：线程状态，比较多，有下面的一些

running: 正在执行程序代码

sleeping：执行了Thread.sleep()

monitor：等待接受一个监听锁。

wait:：Object.wait()，等待被其他线程唤醒

native：正在执行native代码，

vmwait：等待虚拟机，（这个不是很懂，高手指教，这个状态在什么情况下发生）

zombie：线程在垂死的进程

init：线程在初始化（我们不可能看到）

starting：线程正在启动（我们不可能看到）

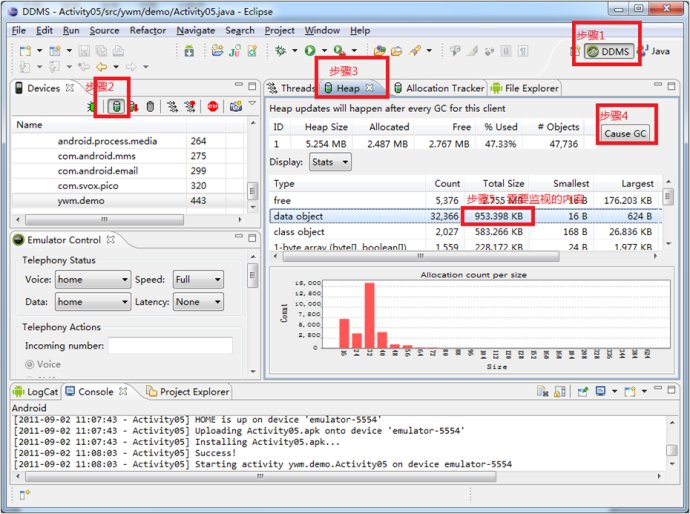
utime：执行用户代码的累计时间

stime：执行系统代码的累计时间

name：线程的名字

参考：<http://www.cnblogs.com/flyme/archive/2011/08/26/2154270.html>

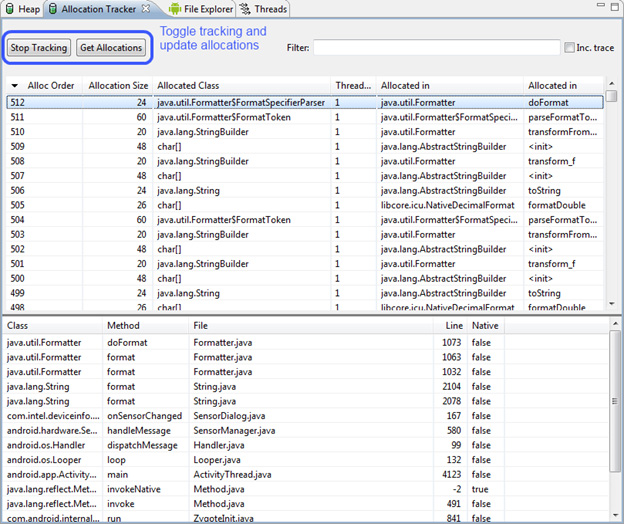
heap:



参考：<http://blog.sina.com.cn/s/blog_7f311ef50101a43i.html>

allocation tracker:

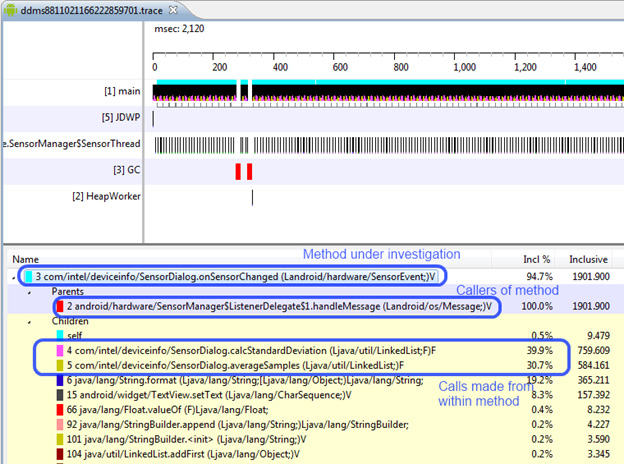
Allocation Tracker（分配跟踪器）视图中显示了有关分配的更深层细节。点击“Start Tracking（开始跟踪）”，在应用中执行某个操作，然后点击“Get Allocations（获得分配）”。



参考：<http://www.xuebuyuan.com/1291595.html>

method profiling:

Method Profiling（方法分析）是DDMS的一款工具，对于快速概览应用中时间的消耗分布非常有用，也可用于时间关键型函数的详细查看。



# attach debug to process

选择一个进程进行调试，其他步骤和debug一致。