Android debug

系统级别的调试方法有哪几种？  
1. 跟踪

2. 断点

3．反寄存器

4. 调用栈分析 .....

app级别的调试方法有哪几种？

 -- JDB eclipse dump strictmode log.wtf ....

Ref link:

<http://www.cnblogs.com/mobile-veterans/archive/2012/07/20/2600722.html>

<http://www.cnblogs.com/mobile-veterans/archive/2012/07/20/2600715.html>

如何调试分析Android中发生的tombstone

Android中较容易出现以下三类问题:Force close / ANR / Tombstone

前两者主要是查看当前的进程或者系统框架层的状态和堆栈就基本可以分析出来，本文主要讨论一下tombstone的情况。

tombstone一般是由Dalvik错误、状态监视调试器、C层代码以及libc的一些问题导致的。

当系统发生tombstone的时候，kernel首先会上报一个严重的警告信号（signal）,上层接收到之后，进程的调试工具会把进程中当时的调用栈现场保存起来，并在系统创建了data/tombstones目录后把异常时的进程信息写在此目录里面，开发者需要通过调用栈来分析整个调用流程来找出出问题的点。

基本工具：

prebuilt/linux-x86/toolchain/arm-eabi-4.4.0/bin

在分析的时候仔细读取汇编会获得更多有用的异常发生时的信息。

1.arm-eabi-addr2line 将类似libxxx.so 0x00012345的调用栈16进制值翻译成文件名和函数名

arm-eabi-addr2line -e libxxx.so 0x00012345

2.arm-eabi-nm 列出文件的符号信息

arm-eabi-nm -l -C -n -S libdvm.so > dvm.data

3.arm-eabi-objdump 列出文件的详细信息

arm-eabi-objdump -C -d libc.so > libc.s

通过以上工具的分析 ，我们可以得到较完整的调用栈以及调用逻辑的汇编码。

然后需要结合ARM架构及ARM汇编的知识（有些情况下可能需要使用gdb）

来分析出现tombstone的原因，以下是本人遇到过的一些tombstone的情况：

1.无效的函数指针：指针为NULL或者已经被重新赋值

2.strlen崩溃：导致不完全的栈信息，栈被破坏

3.FILE操作：因为stdio并非线程安全的，多线程操作时，容易出现异常。

本文涉及到的tombstone处理的主要逻辑所在文件如下：

BootReceiver.java -- frameworks\base\services\java\com\android\server

Debuggerd.c -- system\core\debuggerd

ThreadLocal.java -- libcore\luni\src\main\java\java\lang