* [Android - how do I investigate an ANR?](http://stackoverflow.com/questions/704311/android-how-do-i-investigate-an-anr)

An ANR happens when some long operation takes place in the "main" thread. This is the event loop thread, and if it is busy, Android cannot process any further GUI events in the application, and thus throws up an ANR dialog.  
Now, in the trace you posted, the main thread seems to be doing fine, there is no problem. It is idling in the MessageQueue, waiting for another message to come in. In your case the ANR was likely a longer operation, rather than something that blocked the thread permanently, so the event thread recovered after the operation finished, and your trace went through after the ANR.  
Detecting where ANRs happen is easy if it is a permanent block (deadlock acquiring some locks for instance), but harder if it's just a temporary delay. First, go over your code and look for vunerable spots and long running operations. Examples may include using sockets, locks, thread sleeps, and other blocking operations from within the event thread. You should make sure these all happen in separate threads. If nothing seems the problem, use DDMS and enable the thread view. This shows all the threads in your application similar to the trace you have. Reproduce the ANR, and refresh the main thread at the same time. That should show you precisely whats going on at the time of the ANR

* [Designing for Responsiveness](http://developer.android.com/guide/practices/design/responsiveness.html)
* [Designing for Performance](http://developer.android.com/guide/practices/design/performance.html)
* [[android-discuss] ANR Tutorial](http://www.mailinglistarchive.com/html/android-discuss@googlegroups.com/2009-10/msg00573.html)
* [Android上的bug定位（troubleshooting）](http://swverification.blog.sohu.com/141315680.html)

对于android上的bug定位的文档很少，因为应用程序千差万别的，出现的问题也不尽相同，不过也是有规律可循，大的方向定位是可以做到的，我们对应用程上的问题可以得到相关的信息。

首 先，要对Java的Throwable比较熟悉，因为Android上的应用和服务都是Java的代码，它的Error和 Exception都是沿用Java的，比如Error有 AssertionError，VirtualMachineError，OutOfMemoryError和其他的Error类。Exception有 RuntimeException和IOException，请参考相应的文档查询，Adb logcat里面会把出现错误的Error或Exception打印出来。

分类1， **应用程序错误**， 什么样子的？大家如果用过android手机会碰到过xxxx process意外停止，Force close的对话框弹出来。这一般都是应用程序错误。这个过程一般有 uncaughtException,crash（TAG,e），handleApplicationError，sendSingal(SIGQUIT),logThreadStacks 然后会在/data/anr/traces.txt追加process crash信息。

举例：

11-04 08:55:37.114 W/AudioFlinger( 1032): write blocked for 55 msecs  
11-04 08:55:37.334 W/dalvikvm( 1103): threadid=35: thread exiting with uncaught exception (group=0x2aadda08)  
11-04 08:55:37.354 E/AndroidRuntime( 1103): Uncaught handler: thread WindowManagerPolicy exiting due to uncaught exception  
11-04 08:55:37.374 E/AndroidRuntime( 1103): \*\*\* EXCEPTION IN SYSTEM PROCESS.  System will crash.  
11-04 08:55:37.394 I/global  ( 1566): Default buffer size used in BufferedReader constructor. It would be better to be explicit if an 8k-char buffer is required.  
11-04 08:55:37.464 E/AndroidRuntime( 1103): java.lang.NullPointerException  
11-04 08:55:37.464 E/AndroidRuntime( 1103):  at android.graphics.Canvas.throwIfRecycled(Canvas.java:954)  
11-04 08:55:37.464 E/AndroidRuntime( 1103):  at android.graphics.Canvas.drawBitmap(Canvas.java:980)  
11-04 08:55:37.464 E/AndroidRuntime( 1103):  at com.android.internal.policy.impl.UnlockSliderView$AbstractView  
State.drawCanalBmp(UnlockSliderView.java:851)  
11-04 08:55:37.464 E/AndroidRuntime( 1103):  at com.android.internal.policy.impl.UnlockSliderView$AbstractView  
State.drawSlideCanalBmp(UnlockSliderView.java:822)  
11-04 08:55:37.464 E/AndroidRuntime( 1103):  at com.android.internal.policy.impl.UnlockSliderView$ViewStateSli  
ding.drawSlideImage(UnlockSliderView.java:1798)  
11-04 08:55:37.464 E/AndroidRuntime( 1103):  at com.android.internal.policy.impl.UnlockSliderView$ViewStateSli  
ding.onDraw(UnlockSliderView.java:1766)  
11-04 08:55:37.464 E/AndroidRuntime( 1103):  at com.android.internal.policy.impl.UnlockSliderView.onDraw(Unloc  
kSliderView.java:507)

分类2，**Java application Hang**， 当应用程序停止响应事件比如按键/Touch的时候，会诸如如下流 程，broadcastTimeout，appNotRespondingLocked(frameworks/base/services/java /com/android/server/am/ActivityManagerService.java)，sendSingal(SIGQUIT) (frameworks/base/core/java/android/os/Process.java),SingalCatcherThreadStart(dalvik/vm/SignalCatcher.c),logThreadStacks() (dalvik/vm/SignalCatcher.c)然后会在/data/anr/traces.txt追加process crash信息。

举例：

11-04 15:02:00.795 I/dalvikvm( 1270): processname:com.android.phone  
11-04 15:02:00.795 I/dalvikvm( 1270): crashstring:Java Crash/Hang (SIGQUIT)  
11-04 15:02:00.795 I/dalvikvm( 1270): crashlog:

其他的android错误，通常开发人员会用Log.e来甄别这些信息。

**Dalvik/ Core Libraries的错误**

类似于01-06 17:27:24.526: INFO/DEBUG(963): crashlog:  
01-06 17:27:24.526: INFO/DEBUG(963): \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*  
01-06 17:27:24.526: INFO/DEBUG(963): Build fingerprint: 'xxxxbuild/test-  
keys'  
01-06 17:27:24.526: INFO/DEBUG(963): pid: 2539, tid: 2539  >>> /system/bin/vold <<<  
01-06 17:27:24.526: INFO/DEBUG(963): signal 11 (SIGSEGV), fault addr 00000000  
01-06 17:27:24.526: INFO/DEBUG(963):  r0 00000000  r1 00020000  r2 00000000  r3 80808080  
01-06 17:27:24.526: INFO/DEBUG(963):  r4 00000000  r5 00000000  r6 beff7490  r7 000111cc  
01-06 17:27:24.526: INFO/DEBUG(963):  r8 00000000  r9 00000000  10 00000000  fp 00000000  
01-06 17:27:24.526: INFO/DEBUG(963):  ip 00000000  sp beff7468  lr 0000a6c1  pc afe0e1b8  cpsr 40000010  
01-06 17:27:24.576: INFO/DEBUG(963):  #00  pc 0000e1b8  /system/lib/libc.so  
01-06 17:27:24.576: INFO/DEBUG(963):  #01  pc 0000a6be  /system/bin/vold  
01-06 17:27:24.576: INFO/DEBUG(963):  #02  pc 0000a858  /system/bin/vold  
01-06 17:27:24.576: INFO/DEBUG(963):  #03  pc 0000a918  /system/bin/vold  
01-06 17:27:24.586: INFO/DEBUG(963):  #04  pc 000097b2  /system/bin/vold  
01-06 17:27:24.586: INFO/DEBUG(963):  #05  pc 0001fd7a  /system/lib/libc.so  
01-06 17:27:24.586: INFO/DEBUG(963):  #06  pc 0000bcd2  /system/lib/libc.so  
01-06 17:27:24.586: INFO/DEBUG(963):  #07  pc b000157e  /system/bin/linker  
01-06 17:27:24.586: INFO/DEBUG(963): code:  
01-06 17:27:24.586: INFO/DEBUG(963):  afe0e1a8   
01-06 17:27:24.586: INFO/DEBUG(963):  e3120003  
01-06 17:27:24.586: INFO/DEBUG(963):  e28cc001

。。。

具 体调用过程\_\_linker\_init()  (bionic/linker/linker.c) -> debugger\_init() (bionic/linker/debugger.c)-> debugger\_signal\_handler()Called when a signal is received from Kernel, uses socket() to connect to The “android:debuggerd” socket, and write()s to the socket  -> main()(system/core/debuggerd/debuggerd.c)The debuggerd daemon creates a socket server android:debuggerd and loops forever,waiting for some client to write into the socket  -> handle\_crashing\_process()->engrave\_tombstone() (system/core/debuggerd/debuggerd.c)Dump stack trace and registers in /data/tombstone/ folder在/data/tombstone/会看到这些东西

**kernel错误**：比较难定位，如果系统发生crash现象，可以开机之后取得最后一次的kernel日志来定位，可以在/proc/last\_kmsg得到相关的东西来定位系统是不是 kernel问题。在日志里面通常最后会有Kernel panic 。。。。

**Modem的问题**，这个跟平台有关系，比如G1是高通平台，如果发生在Modem的问题crash了，你可以到/proc/last\_amsslog 找到，发给高通解决。比如从/proc/last\_kmsg得到

05-29 09:09:02.409 <0>[54026.002654] Kernel panic - not syncing: Modem has crashed... 这样可以定位到modem的问题，找到/proc/last\_amsslog 相关信息，不过这个是二进制文件的。

总 结：如果是用户程序 Exception或者Dalvik Error Invoking Runtime都会通过AcitivityMangerService发出SIGQUIT信号给process，再调用SignalCatcher.c再 把crashed process信息放在/data/anr/traces.txt里面。

如果是SYSTEM process????或者通过Log.e(TAG,str,trowable）会直接报告exception到logcat里面。

如果是Dalvik Error, Invoking debuggerd/C code/LibC Error, 会调用Tombstone，然后把信息打印到/data/tombstones目录里面。

如果是kernel错误，会直接放到 /proc/last\_kmsg文件（下次重启后会有）

如果是Modem错误，会直接有 /proc/last\_amass文件出现（高通平台）。