

## 如何在安卓系统中侦测和调试内存泄露和越界



2017年03月28日 17:12:43

833

0

### 1.1 基本原理



ionc的libc初始时，会检测属性"libc.debug.malloc"，



/android/bionic/libc/bionic/malloc\_debug\_common.c



/static void malloc\_init\_impl(void)

```
if (!debug_level && __system_property_get("libc.debug.malloc", env)) {  
    debug_level = atoi(env);  
}
```

```
/* Debug level 0 means that we should use dlxxx allocation  
 * routines (default). */  
if (!debug_level) {  
    return;  
}
```

// Lets see which .so must be loaded for the requested debug level

switch (debug\_level) {

case 1:

case 5:

case 10:

so\_name = "/system/lib/libc\_malloc\_debug\_leak.so";

break;

case 20:

// Quick check: debug level 20 can only be handled in emulator.

if (!qemu\_running) {



子夜蓝风

+ 关注

原创

粉丝

喜欢

码云

49

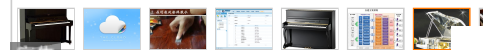
56

1

未开通



### 斯坦威钢琴价格



### 他的最新文章

[更多文章](#)[临窗小记](#)[linux 动态库 显式调用 与 隐式调用](#)[常用开源许可协议介绍](#)[ofono Architecture Introduction](#)[ubuntu下最全的软件安装、卸载、查看命令](#)

1	perform leak detection
5	fill allocated memory to detect overruns
10	fill memory and add sentinels to detect overruns
20	use special instrumented malloc/free routines for the emulator

如android/frameworks/base/media/libmediaplayerservice/MediaPlayerService.cpp在binder dump接口中调用get\_malloc\_leak\_info/free\_malloc\_leak\_info来获取当前全局表中的示例。

## 如何通过adb shell am播放音视频文件

```
{
    const size_t SIZE = 256;
    char buffer[SIZE];
    MyString8 result;

    typedef struct {
        size_t size;
        size_t dups;
        intptr_t * backtrace;
    } AllocEntry;

    uint8_t *info = NULL;
    size_t overallSize = 0;
    size_t infoSize = 0;
    size_t totalMemory = 0;
    size_t backtraceSize = 0;

    get_malloc_leak_info(&info, &overallSize, &infoSize, &totalMemory, &backtraceSize);

    . . . . .
    free_malloc_leak_info(info);
}

status_t MediaPlayerService::dump(int fd, const Vector<String16> &args)
{
    . . . . .
    #if defined(__arm__)
        bool dumpMem = false;
        for (size_t i = 0; i < args.size(); i++) {
            if (args[i] == String16("-m")) {
                dumpMem = true;
            }
        }
    }
    if (dumpMem) {
```



0



## 联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

🗣 网站客服

[关于](#) [招聘](#) [广告服务](#) [阿里云](#)

©2018 CSDN 京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



```
#endif
. . . . .
}
```

先使能memory debug功能为leak检测：在init.rc中加入以下代码后，重启板子：

```
#setprop libc.debug.malloc 1
```



0

使用以下命令来查看meida player service的内存使用情况



```
dumpsys media.player -m
```

#### 1.4 内存越界



内存越界，其原理是在每次malloc前，在返回给用户的内存前后，各建立一个16字节的越界监测区，并填充此区域为特殊字符。

free时，检测此区域是否被改写过。并打印出相应调用栈。



内存泄漏如何查看

先使能memory debug功能为leak检测：在init.rc中加入以下代码后，重启板子：

```
#setprop libc.debug.malloc 10
```

然后查看logcat中的信息，如果发生越界，便会assert,并且打印出调用栈。

```
I/DEBUG ( 648): *** **
I/DEBUG ( 648): Build fingerprint: 'CSR/sirfsocv7_android/sirfsocv7:2.3.7/GINGERBREAD/eng.xy05.20120806.111522:en
I/DEBUG ( 648): pid: 880, tid: 880 >>> /system/bin/mediaserver <<<
I/DEBUG ( 648): signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr deadbaad
I/DEBUG ( 648): r0 00000027 r1 deadbaad r2 a0000000 r3 00000000
I/DEBUG ( 648): r4 00000001 r5 00000000 r6 a92260e8 r7 be92e9fc
I/DEBUG ( 648): r8 00000000 r9 00000000 10 00000000 fp 00000000
I/DEBUG ( 648): ip afd46668 sp be92e800 lr afd193f9 pc afd15ec4 cpsr 60000030
I/DEBUG ( 648): d0 00000000bd6bc8e3 d1 0000000000000000
I/DEBUG ( 648): d2 0000000000000000 d3 0000000000000000
I/DEBUG ( 648): d4 0000000000000000 d5 0000000000000000
I/DEBUG ( 648): d6 0000000000000000 d7 3f80000000000000
I/DEBUG ( 648): d8 0000000000000000 d9 0000000000000000
I/DEBUG ( 648): d10 0000000000000000 d11 0000000000000000
```

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

[登录](#)
[注册](#)


I/DEBUG ( 648): scr 00000000  
I/DEBUG ( 648):  
I/DEBUG ( 648): #00 pc 00015ec4 /system/lib/libc.so  
I/DEBUG ( 648): #01 pc 0007858c /system/lib/gst/libglib-2.0.so.0.2400.1  
I/DEBUG ( 648): #02 pc 00077ebc /system/lib/gst/libglib-2.0.so.0.2400.1  
I/DEBUG ( 648): #03 pc 00078090 /system/lib/gst/libglib-2.0.so.0.2400.1  
I/DEBUG ( 648): #04 pc 00076e10 /system/lib/gst/libglib-2.0.so.0.2400.1  
I/DEBUG ( 648): #05 pc 000772a4 /system/lib/gst/libglib-2.0.so.0.2400.1  
I/DEBUG ( 648): #06 pc 000775cc /system/lib/gst/libglib-2.0.so.0.2400.1  
I/DEBUG ( 648): #07 pc 000776e0 /system/lib/gst/libglib-2.0.so.0.2400.1  
I/DEBUG ( 648): #08 pc 00041264 /system/lib/gst/libglib-2.0.so.0.2400.1  
I/DEBUG ( 648): #09 pc 00009bcc /system/lib/gst/libGSTPlayerImpFactory.so  
I/DEBUG ( 648): #10 pc 0000a908 /system/lib/gst/libGSTPlayerImpFactory.so  
I/DEBUG ( 648): #11 pc 0001c292 /system/lib/libmediaplayerservice.so  
I/DEBUG ( 648): #12 pc 00017b32 /system/lib/libmediaplayerservice.so  
I/DEBUG ( 648): #13 pc 00017bfc /system/lib/libmediaplayerservice.so  
I/DEBUG ( 648): #14 pc 000089ec /system/bin/mediaserver  
I/DEBUG ( 648): #15 pc 00014d72 /system/lib/libc.so  
I/DEBUG ( 648):  
I/DEBUG ( 648): code around pc:  
I/DEBUG ( 648): afd15ea4 2c006824 e028d1fb b13368db c064f8df  
I/DEBUG ( 648): afd15eb4 44fc2401 4000f8cc 49124798 25002027  
I/DEBUG ( 648): afd15ec4 f7f57008 2106eb6c ecc8f7f6 460aa901  
I/DEBUG ( 648): afd15ed4 f04f2006 95015380 95029303 e82ef7f6  
I/DEBUG ( 648): afd15ee4 462aa905 f7f62002 f7f5e83a 2106eb58  
I/DEBUG ( 648):  
I/DEBUG ( 648): code around lr:  
I/DEBUG ( 648): afd193d8 4a0e4b0d e92d447b 589c41f0 26004680  
I/DEBUG ( 648): afd193e8 686768a5 f9b5e006 b113300c 47c04628  
I/DEBUG ( 648): afd193f8 35544306 37fff117 6824d5f5 d1ef2c00  
I/DEBUG ( 648): afd19408 e8bd4630 bf0081f0 00028124 fffffff8  
I/DEBUG ( 648): afd19418 b086b570 f602fb01 9004460c a804a901  
I/DEBUG ( 648):  
I/DEBUG ( 648): stack:

I/DEBUG ( 648): be92e7c8 818f39cc /system/lib/gst/libglib-2.0.so.0.2400.1  
I/DEBUG ( 648): be92e7cc afd191c5 /system/lib/libc.so  
I/DEBUG ( 648): be92e7d0 afd4270c  
I/DEBUG ( 648): be92e7d4 afd426b8  
I/DEBUG ( 648): be92e7d8 00000000  
I/DEBUG ( 648): be92e7dc afd193f9 /system/lib/libc.so  
I/DEBUG ( 648): be92e7e0 00000001  
I/DEBUG ( 648): be92e7e4 be92e814  
I/DEBUG ( 648): be92e7e8 a92260e8  
I/DEBUG ( 648): be92e7ec be92e9fc  
I/DEBUG ( 648): be92e7f0 00000000  
I/DEBUG ( 648): be92e7f4 afd1871b /system/lib/libc.so  
I/DEBUG ( 648): be92e7f8 df002777  
I/DEBUG ( 648): be92e7fc e3a070ad  
I/DEBUG ( 648): #00 be92e800 0000000a  
I/DEBUG ( 648): be92e804 afd1ca1f /system/lib/libc.so  
I/DEBUG ( 648): be92e808 0000000a  
I/DEBUG ( 648): be92e80c afd426b8  
I/DEBUG ( 648): be92e810 a92260e8  
I/DEBUG ( 648): be92e814 fffffbdf  
I/DEBUG ( 648): be92e818 81933494  
I/DEBUG ( 648): be92e81c 81933494  
I/DEBUG ( 648): be92e820 818f39e4 /system/lib/gst/libglib-2.0.so.0.2400.1  
I/DEBUG ( 648): be92e824 81878590 /system/lib/gst/libglib-2.0.so.0.2400.1  
I/DEBUG ( 648): #01 be92e828 be92e844  
I/DEBUG ( 648): be92e82c 00000000  
I/DEBUG ( 648): be92e830 81933494  
I/DEBUG ( 648): be92e834 81933494  
I/DEBUG ( 648): be92e838 a9220d80 /system/lib/libmediaplayerservice.so  
I/DEBUG ( 648): be92e83c 81877ec0 /system/lib/gst/libglib-2.0.so.0.2400.1  
I/DEBUG ( 648): be92e840 818f38d0 /system/lib/gst/libglib-2.0.so.0.2400.1  
I/DEBUG ( 648): be92e844 818f3990 /system/lib/gst/libglib-2.0.so.0.2400.1  
I/DEBUG ( 648): be92e848 0002b800  
I/DEBUG ( 648): be92e84c 818f3990 /system/lib/gst/libglib-2.0.so.0.2400.1

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



但此时，仍然不能定位到底是哪段程序造成了此段地址越界(即不能定位到发生越界时的第一现场)，我们需要更为强大的内存检测工具- valgrind来做到代码级定位。

## 2. 使用valgrind工具检测进程内存情况

valgrind已经在Android 4.0和4.1中默认支持。我已经backport此包到CSR prima2 2.3中，

使用方法：

To run Valgrind on the system server:

```
0 #stop
#start
```

To run Valgrind on an application:

```
0 #stop
#start
```

0 #stop
#start

The system property server has a maximum length limit on property names.

Of course you can pass other arguments to valgrind or run other tools instead. You MUST have root for this to work.

需要注意的是：由于valgrind的介入，从而使systemserver/app启动过程非常缓慢。并且打印出类似以下的valgrind调试信息：

```
I//data/local/valgrind/bin/valgrind( 1042): ==1043== Thread 10:
I//data/local/valgrind/bin/valgrind( 1042): ==1043== Conditional jump or move depends on uninitialised value(s)
I//data/local/valgrind/bin/valgrind( 1042): ==1043== at 0x81135530: ??? (in /system/lib/egl/libGLESv1_CM_POWERVR_S
I//data/local/valgrind/bin/valgrind( 1042): ==1043==
I//data/local/valgrind/bin/valgrind( 1042): ==1043== Conditional jump or move depends on uninitialised value(s)
I//data/local/valgrind/bin/valgrind( 1042): ==1043== at 0x81135548: ??? (in /system/lib/egl/libGLESv1_CM_POWERVR_S
I//data/local/valgrind/bin/valgrind( 1042): ==1043==
I//data/local/valgrind/bin/valgrind( 1042): ==1043== Conditional jump or move depends on uninitialised value(s)
I//data/local/valgrind/bin/valgrind( 1042): ==1043== at 0x81135574: ??? (in /system/lib/egl/libGLESv1_CM_POWERVR_S
I//data/local/valgrind/bin/valgrind( 1042): ==1043==
I//data/local/valgrind/bin/valgrind( 1042): ==1043== Conditional jump or move depends on uninitialised value(s)
I//data/local/valgrind/bin/valgrind( 1042): ==1043== at 0x8113557C: ??? (in /system/lib/egl/libGLESv1_CM_POWERVR_S
I//data/local/valgrind/bin/valgrind( 1042): ==1043==
I//data/local/valgrind/bin/valgrind( 1042): ==1043== Use of uninitialised value of size 4
I//data/local/valgrind/bin/valgrind( 1042): ==1043== at 0x81135590: ??? (in /system/lib/egl/libGLESv1_CM_POWERVR_S
```

另外可使用带symbol的动态链接库文件进行调试。

Android Memory Usage :

[http://elinux.org/Android\\_Memory\\_Usage](http://elinux.org/Android_Memory_Usage)

valgrind in android:

[https://bugs.kde.org/show\\_bug.cgi?id=266035#c17](https://bugs.kde.org/show_bug.cgi?id=266035#c17)



连接介绍：

[blog.csdn.net/sduliulun/article/details/7732906](http://blog.csdn.net/sduliulun/article/details/7732906)

0



目前您尚未登录，请 [登录](#) 或 [注册](#) 后进行评论

## 定位多线程内存越界问题实践总结



killmice

2014年08月08日 18:53

📖 5854

定位多线程内存越界问题实践总结 2013/2/4 杨志丰 yangzhifeng83@gmail.com 关键字 多线程，内存越界，valgrind，electr  
ic-fence，mpro...

## android native 内存泄露检查（libc.debug.malloc）

转自：<http://www.csdn123.com/html/blogs/20131011/81402.htm>



haima1998

2016年05月26日 17:11

📖 4212

1. Introduction Android对内存的使用包括内存泄漏和...

## 区块链概念股大揭秘！这些股值得入手！

【网易官方股票交流群】添加微信好友，进群免费领牛股→

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册






## android 系统内存检查

 u011279649 2016年03月09日 07:35  810

1. Introduction Android对内存的使用包括内存泄漏和内存越界，内存泄漏会导致系统内存减少，最终分配不到内存，这样大的程序就不能运行，甚至系统没有内存而崩溃。Andro...


## Android内存泄露检测工具 and 实际开发中遇到的内存泄露问题解析

 内存泄露是平常开发中经常遇到的，有些时候稍不注意就会发生，而且还不易察觉，这就需要工具来帮助检测。本文主要介绍内存检测工具和我在开发中遇到的内存泄露问题和解决方案。内存泄露的原理具体的原理涉及到虚...

 ard361401376 2016年05月23日 17:41  2256

## Android Native进程内存泄露检测

 forestcell 2016年12月17日 16:16  821

 说明：本文为博主原创文章，未经博主允许不得转载。 目录(?)[+] Android Native进程内存泄露检测 简介 对于Android的 nati...

## 程序猿学炒股投资，拒绝死工资！

网易官方股票交流群！免费送您3支牛股



## Android中native进程内存泄露的调试技巧（一）-- libc debug

libc.debug.malloc // 1 - For memory leak detections. // 5 - For filling allocated / freed memory...

 agwtpcbox 2016年11月30日 14:50  1258

## android hwui内存越界分析

 dyfrank 2014年11月05日 21:43  2139

android内存检查工具：1. address sanitizer

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册



记录一下主要步骤： 1. 编译valgrind, 我的编译环境：Ubuntu 12.10 x86-64, android ndk r8, 目标android设备的os是andro  
id ...

(转)记一次内存优化的分享

wangbin\_jxust 2015年05月18日 09:43 621

原文:记一次内存优化的分享 公司游戏已经进行到最后一个阶段了，经过最后一次引擎升级之后使用的是3.3的版本。以前产品  
目标是只需要兼容1G或者以上内存就可以，我们也一直没关注低内存的运行...



Android native 内存泄露检查 ( libc.debug.malloc )

内存泄露和内存越界 u010481276 2018年01月03日 11:27 91



赠送《趣味背单词》，学英语就是这么简单

英语单词还在死记硬背？教你一招速记方法搞定英语单词



使用DDMS中的native heap检查Android native内存泄露

ddms native heap wuhengde 2017年05月04日 21:19 1969

Android：性能优化之利用LeakCanary检测内存泄漏及解决办法

什么是内存泄漏？有些对象只有有限的生命周期。当它们的任务完成之后，它们将被垃圾回收。如果在对象的生命周期本该  
结束的时候，这个对象还被一系列的引用，这就会导致内存泄漏。随着泄漏的累积，app将消耗完...

HMYANG314 2017年05月16日 16:31 366

Android中native进程内存泄露的调试技巧

broadview2006 2013年01月31日 09:47 3095

Android中native进程内存泄露的调试技巧 红狼博客 代码基于Android2.3.x版本 Android为Java程序提供了方便的内存泄露信  
息和工具（如MAT），便于查找。但是，对于...

## Android中native进程内存泄露的调试技巧（一）



l\_nan

2015年02月04日 18:03

8077

基于Android5.0版本 Android为Java程序提供了方便的内存泄露信息和工具

（如MAT），便于查找。但是，对于纯粹C/C++ 编写的native进程，却不那么容易查找内存泄露。传统的C/C...

## Android常见问题集锦



xiaoyaoyou1212

2016年04月02日 22:22

4531

在开发过程中，每个人或多或少会遇到各种各样的问题，有些问题依据代码思路调试就可以定位出来，而大部分的问题都是经验性问题，遇到过就很容易解决，但在第一次遇到时往往会花费大量时间来定位问题。针对此种情况，下文...



### 程序员不会英语怎么行？



教你一个数学公式秒懂天下英语



## Android memory corruption debugger



kongxinsun

2017年11月29日 14:10

74

memory corruption是最难搞的问题之一，这是因为：1. 破坏内存的地方和内存破坏的结果常常是分离的，所以很难定位。2. 有些破坏在特定case下才会出现，不好复现。 Memor...

## Android memory leak detect



kongxinsun

2017年11月29日 15:10

91

What is memory leak Native process memory leak detect Java process memory leak detect Kmemleak usage...

## Malloc内存泄露和内存越界问题的研究



u011761947

2013年10月22日 11:11

1165

Malloc内存泄露和内存越界问题的研究 2013-03-02 22:59:32 分类： LINUX 原文地址：Malloc内存泄露和内存越界问题的研究 作者：gongcb Ma...

## Android : dumpsys功能之一：原理介绍



wlqingwei

2015年02月27日 17:31

2397

描述Android中dumpsys的功能实现原理

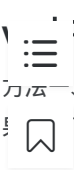
一般情况下，如果我们重启web应用是通过重启tomcat的话，则不存在内存泄漏问题。但如果不重启tomcat而对web应用进行重加载则可能会导致内存泄漏，因为重加载后有可能导致原来的某些内存无法让G...

### 带你一键升级英语单词记忆技能！

@先生，你有一本免费<英语单词速记>宝典待领取



0



### 内存泄露检测方法



whatday

2015年12月29日 15:41

3415

万云一、1、头文件：#define \_CRTDBG\_MAP\_ALLOC #include #include 注意 #include 语句必须采用上文所示顺序。如...了顺序，所使用的函数可...



### 说下越界访问的调试方法



glider

2016年11月06日 22:15

404

越界访问的一般提示是“access violation reading location 0xFFFFFFFF”，翻译过来就是“内存越界访问”。这个提示和一般提示的不同之处是，程序不会停在越界...

### Android Handler内存泄露分析



goodlixueyong

2016年02月19日 23:02

989

用lint对Android代码进行扫描时，有这样一个关于Handler的警告信息：This Handler class should be static or leaks might occur。产生...

### C语言中的指针和内存泄漏几种情况



qq\_32319583

2016年12月14日 16:30

712

C语言中的指针和内存泄漏几种情况

### java 内存泄露调试和解决



caomiao2006

2015年09月15日 00:35

1279

说起Java的内存泄露，其实定义不是那么明确。首先，如果JVM没有bug，那么理论上是不会出现“无法回收的堆空间”，也就是说C/C++中的那种内存泄露在Java中不存在的。其次，如果由于Java程序...

## 英语文档看不懂？教你一个公式秒懂英语！

跨界老码农教你学英语，带你有效提升阅读英文技术文档的能力→



### ！👍 调试malloc(堆越界)问题




sandform 2016年05月26日 16:24 1886

如何调试malloc(堆越界)问题 [DESCRIPTION] 有一类NE比较特殊，就是堆引起的异常(调用malloc申请的内存后使用不当引起的异常)：1. 申请后多次释放 (doub...



### 📖 1. 在类的源文件(.m)中，@interface部分的作用

.h里面的@interface，不消说，是典型的头文件，它是供其它Class调用  guhunv33 2015年12月12日 13:41 353  
在.m文件的@implementation部分，@property和functions，都能够被其它Class“看到”。 而.  
m里面的@...

### 面向对象编程的概念



po\_shi 2017年09月05日 22:40 121

假如你之前从未使用过面向对象类的编程语言，那么在写作代码之前，你需要了解一些基本的概念。这篇教程将向你介绍对象（objects），类(classes),继承（inheritance），接口（inter...

### dumpsys命令的用法



yuanhuihui208 2015年07月10日 16:28 5642

dumpsys命令是android手机自带的调试工具，下面详解讲解dumpsys用法 一. 初识dumpsysadb shell //进入手机shell...

### 如何在安卓系统上使用arm-linux-gdb调试内核

wlw0071986 2016年03月15日 15:04 1570

现在很多安卓平台都没有把gdb调试工具编译进去，因此需要我们自己安装交叉编译环境下的gdb工具。具体实现只需几步即可：1. 下载最新的arm-linux-gdb源码包 下载地址：http:...