# Debug

## 调试方法

### 串口调试

咖啡色接gnd口就好

打开CRT，选择波特率为115200，

串口只能链接一个！不支持多个会话

后台服务的话，加上&符号

logcat –v time &

dmesg &

logcat -v time&

######################################USB-USB-IN

logcat -b main -b system -b radio -b events -v time

### ADB

搜索关键字：\* FOR %%i IN (\*.apk)

git，googel，

目标文件：apk

日志加入

#扫描当前apk文件夹

@REM 循环安装本目录下的APK文件

FOR %%i IN (\*.apk) DO (

ECHO 正在安装：%%i

adb install -r %%i

)

#### APK安装

ADB APK安装

adb install

安装 APK 命令可以用adb install [-lrtsd] <file>或者adb install-multiple [-lrtsdp] <file...>，adb install-multiple表示批量安装。  
参数介绍：

* -l：锁定该程序
* -r：可以覆盖已有的应用，保留数据和缓存文件
* -t：允许测试该应用
* -s：安装在SD卡中
* -d：允许降低版本安装
* -p：部分应用程序安装

1. 无需数据线

且可解决部分机器在方法1时出现的“unable to connect to 192.168.1.199:5555”错误

在android设备上安装 “终端模拟器”等类似shell命令工具，使用下面命令（**需root权限**）：

TCP/IP方式：

setprop service.adb.tcp.port 5555

stop adbd

start adbd

usb方式：

setprop service.adb.tcp.port -1

stop adbd

start adbd

<https://twiceyuan.com/2014/11/21/Android%E7%BD%91%E7%BB%9C%E8%B0%83%E8%AF%95%E5%BC%80%E5%90%AF%E6%96%B9%E6%B3%95/>

#### 常见问题

##### 三星手机5.1调试

1. ~~左上角插入手机卡2才有用（待验证）~~

　　1、点开设置—更多—关于设备-》软件信息-》版本号。点内部版本号7到8次左右。

　　2、点过之后返回，就会看到开发者选项，这个勾选USB调试就行啦！

　　3、如果不能安装驱动，在三星官网下载PC套件Samsung kies(包含了链接所需要的驱动)。点击打开。【[PC套件下载地址](http://www.samsung.com/cn/support/usefulsoftware/KIES/)】http://www.samsung.com/cn/support/usefulsoftware/KIES/

4、那个KIES软件安装到电脑上，通过数据线连接电脑（链接模式为媒体设备MTP）即可。

实在连不上可以尝试重启一下

提示：若要解除阻止并通过USB数据线连接至+其他设备，请插入USIM&oq=若要解除阻止并通过USB数据线连接至+其他设备，请插入USIM

##### adb无法启动

现象1：端口被占用，Unable to create Debug Bridge: Unable to start adb server: error: could not install \*smartsocket\* listener: cannot bind to 127.0.0.1:5037

现象2：unknown host service

解决办法：kill 端口占用的进程，常见的是手机助手占用了。

Netstat –ano | findstr 5037



[Android Debugger port : Connection refused:](http://www.jianshu.com/p/96aaed625ad0) adb kill-server; adb start-server

Adb启动ok，无法发现设备

Adb是32位的，设备是64位的，请用android stdio的最新tool的adb来使用就好了。

##### MIX开启 USB安装/USB调试（安全设置）

需要登录小米账号，[more](http://www.miui.com/thread-5711795-1-1.html)

然后才可以执行adb shell。

参考：

[ANDROID开发常用的ADB命令整理kg](http://www.keyguan.com/blog/2017/02/11/android%E5%BC%80%E5%8F%91%E5%B8%B8%E7%94%A8%E7%9A%84adb%E5%91%BD%E4%BB%A4%E6%95%B4%E7%90%86/)

串口adb

Wifi-adb调试

Win-cmd:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| cls | Clear screen |  |  |  |
| Findstr | 正则匹配 |  |  |  |
|  |  |  |  |  |

#### 源码分析

ADB是Android debug bridge的缩写，它使用PC机可以通过USB或网络与android设备通讯。

源码：system/core/adb

先来看下编译脚本

##### Android.mk

可以看到，最终会有3个执行文件被生成，adbd和两个adb程序。adbd是手机终的守护进程；adb一个是windows、linux、darwin或freebsd运行的程序，另一个是目标机上运行的程序。

宏ADB\_HOST用于区分是PC端程序还是目标机端的程序。宏ADB\_HOST\_ON\_TARGET用于区分adb程序是否是在目标机上运行。这3个程序使用的是同一份源码，在内部，使用这些宏来区别不同的程序。

程序的入口在adb.c的main()函数：

##### 入口adb. main()函数

1. **int** main(**int** argc, **char** \*\*argv)
2. {
3. #if ADB\_HOST
4. adb\_sysdeps\_init();
5. adb\_trace\_init();
6. D("Handling commandline()\n");
7. **return** adb\_commandline(argc - 1, argv + 1);
8. #else
9. /\* If adbd runs inside the emulator this will enable adb tracing via
10. \* adb-debug qemud service in the emulator. \*/
11. adb\_qemu\_trace\_init();
12. **if**((argc > 1) && (!strcmp(argv[1],"recovery"))) {
13. adb\_device\_banner = "recovery";
14. recovery\_mode = 1;
15. }
17. start\_device\_log();
18. D("Handling main()\n");
19. **return** **adb\_main(0, DEFAULT\_ADB\_PORT)**;
20. #endif
21. }

先来看adbd程序，此时宏的设置是ADB\_HOST=0。上面代码中start\_device\_log()是log的初始化操作，可以重定向输出的log信息，接着进入adb\_main()函数。先来看下它的参数DEFAULT\_ADB\_PORT：

1. #if ADB\_HOST\_ON\_TARGET
2. /\* adb and adbd are coexisting on the target, so use 5038 for adb
3. \* to avoid conflicting with adbd's usage of 5037
4. \*/
5. #  define DEFAULT\_ADB\_PORT 5038
6. #else
7. #  define DEFAULT\_ADB\_PORT 5037
8. #endif

如果是目标机程序，它的值是5038，否则它的值是5037。这里没有定义ADB\_HOST\_ON\_TARGET， 所以它是5037。

##### adb\_main()的源代码如下：

1. **int** adb\_main(**int** is\_daemon, **int** server\_port)
2. {
3. #if !ADB\_HOST
4. **int** port;
5. **char** value[PROPERTY\_VALUE\_MAX];
7. umask(000);
8. #endif
10. atexit(adb\_cleanup);
11. #ifdef HAVE\_WIN32\_PROC
12. SetConsoleCtrlHandler( ctrlc\_handler, TRUE );
13. #elif defined(HAVE\_FORKEXEC)
14. // No SIGCHLD. Let the service subproc handle its children.
15. signal(SIGPIPE, SIG\_IGN);
16. #endif
18. init\_transport\_registration();
20. #if ADB\_HOST
21. HOST = 1;
22. usb\_vendors\_init();
23. usb\_init();
24. local\_init(DEFAULT\_ADB\_LOCAL\_TRANSPORT\_PORT);
25. adb\_auth\_init();
27. **char** local\_name[30];
28. build\_local\_name(local\_name, **sizeof**(local\_name), server\_port);
29. **if**(install\_listener(local\_name, "\*smartsocket\*", NULL)) {
30. exit(1);
31. }
32. #else
33. property\_get("ro.adb.secure", value, "0");
34. auth\_enabled = !strcmp(value, "1");
35. **if** (auth\_enabled)
36. adb\_auth\_init();
38. // Our external storage path may be different than apps, since
39. // we aren't able to bind mount after dropping root.
40. **const** **char**\* adb\_external\_storage = getenv("ADB\_EXTERNAL\_STORAGE");
41. **if** (NULL != adb\_external\_storage) {
42. setenv("EXTERNAL\_STORAGE", adb\_external\_storage, 1);
43. } **else** {
44. D("Warning: ADB\_EXTERNAL\_STORAGE is not set.  Leaving EXTERNAL\_STORAGE"
45. " unchanged.\n");
46. }
48. /\* don't listen on a port (default 5037) if running in secure mode \*/
49. /\* don't run as root if we are running in secure mode \*/
50. **if** (should\_drop\_privileges()) {
51. **struct** \_\_user\_cap\_header\_struct header;
52. **struct** \_\_user\_cap\_data\_struct cap;
54. **if** (prctl(PR\_SET\_KEEPCAPS, 1, 0, 0, 0) != 0) {
55. exit(1);
56. }
58. /\* add extra groups:
59. \*\* AID\_ADB to access the USB driver
60. \*\* AID\_LOG to read system logs (adb logcat)
61. \*\* AID\_INPUT to diagnose input issues (getevent)
62. \*\* AID\_INET to diagnose network issues (netcfg, ping)
63. \*\* AID\_GRAPHICS to access the frame buffer
64. \*\* AID\_NET\_BT and AID\_NET\_BT\_ADMIN to diagnose bluetooth (hcidump)
65. \*\* AID\_SDCARD\_R to allow reading from the SD card
66. \*\* AID\_SDCARD\_RW to allow writing to the SD card
67. \*\* AID\_MOUNT to allow unmounting the SD card before rebooting
68. \*\* AID\_NET\_BW\_STATS to read out qtaguid statistics
69. \*/
70. gid\_t groups[] = { AID\_ADB, AID\_LOG, AID\_INPUT, AID\_INET, AID\_GRAPHICS,
71. AID\_NET\_BT, AID\_NET\_BT\_ADMIN, AID\_SDCARD\_R, AID\_SDCARD\_RW,
72. AID\_MOUNT, AID\_NET\_BW\_STATS };
73. **if** (setgroups(**sizeof**(groups)/**sizeof**(groups[0]), groups) != 0) {
74. exit(1);
75. }
77. /\* then switch user and group to "shell" \*/
78. **if** (setgid(AID\_SHELL) != 0) {
79. exit(1);
80. }
81. **if** (setuid(AID\_SHELL) != 0) {
82. exit(1);
83. }
85. /\* set CAP\_SYS\_BOOT capability, so "adb reboot" will succeed \*/
86. header.version = \_LINUX\_CAPABILITY\_VERSION;
87. header.pid = 0;
88. cap.effective = cap.permitted = (1 << CAP\_SYS\_BOOT);
89. cap.inheritable = 0;
90. capset(&header, &cap);
92. D("Local port disabled\n");
93. } **else** {
94. **char** local\_name[30];
95. build\_local\_name(local\_name, **sizeof**(local\_name), server\_port);
96. **if**(install\_listener(local\_name, "\*smartsocket\*", NULL)) {
97. exit(1);
98. }
99. }
101. **int** usb = 0;
102. **if** (access(USB\_ADB\_PATH, F\_OK) == 0 || access(USB\_FFS\_ADB\_EP0, F\_OK) == 0) {
103. // listen on USB
104. usb\_init();
105. usb = 1;
106. }
108. // If one of these properties is set, also listen on that port
109. // If one of the properties isn't set and we couldn't listen on usb,
110. // listen on the default port.
111. property\_get("service.adb.tcp.port", value, "");
112. **if** (!value[0]) {
113. property\_get("persist.adb.tcp.port", value, "");
114. }
115. **if** (sscanf(value, "%d", &port) == 1 && port > 0) {
116. printf("using port=%d\n", port);
117. // listen on TCP port specified by service.adb.tcp.port property
118. local\_init(port);
119. } **else** **if** (!usb) {
120. // listen on default port
121. local\_init(DEFAULT\_ADB\_LOCAL\_TRANSPORT\_PORT);
122. }
124. D("adb\_main(): pre init\_jdwp()\n");
125. init\_jdwp();
126. D("adb\_main(): post init\_jdwp()\n");
127. #endif
129. **if** (is\_daemon)
130. {
131. // inform our parent that we are up and running.
132. #ifdef HAVE\_WIN32\_PROC
133. **DWORD**  count;
134. WriteFile( GetStdHandle( STD\_OUTPUT\_HANDLE ), "OK\n", 3, &count, NULL );
135. #elif defined(HAVE\_FORKEXEC)
136. fprintf(stderr, "OK\n");
137. #endif
138. start\_logging();
139. }
140. D("Event loop starting\n");
142. fdevent\_loop();
144. usb\_cleanup();
146. **return** 0;
147. }

(1) init\_transport\_registration()初始化fevent transport\_registration\_fde；

(2) 判断系统属性ro.adb.secure，目标板设置这个宏；

sagit:/ $ getprop | grep ro.adb.secure

[ro.adb.secure]: [1]

(3) 没有定义环境变量adb\_external\_storage；只有$EXTERNAL\_STORAGE

sagit:/ $ echo $EXTERNAL\_STORAGE

/sdcard

(4) should\_drop\_privileges()根据android编译环境should\_drop\_privileges返回不同的值，如果它的值是userdebug或eng，宏ALLOW\_ADBD\_ROOT的值被定义为1，执行install\_listener()，否则不会定义，这种情况下，由于adbd运行在root下，为保证它的安全性，它需要降级运行；

ifneq (,$(filter userdebug eng,$(TARGET\_BUILD\_VARIANT)))

LOCAL\_CFLAGS += -DALLOW\_ADBD\_ROOT=1

Endif

http://blog.csdn.net/xgbing/article/details/52058390

## Native奔溃调试

## [android native 内存泄露检查（libc.debug.malloc ）](http://blog.csdn.net/haima1998/article/details/51508947)

\*c c++代码，由于其特殊性质，没有虚拟机概念，内存则直接是由用户管理，比如申请，释放，都是需要用户主动去触发，如果用户出现使用了申请，但是用完之后，没有调用释放，则会引起内存泄露。这种叫真正意义的内存泄露，只有重启机子，才能恢复。

相对而已java端的内存泄露，指的是一个应用长期运行，导致相互引用，无法释放，GC没法回收，引起的有效内存越来越小，我们将此现象叫做，内存泄露，通过关闭此应用，重新打开即可恢复内存。因此看来，java内存泄露和c c ++ 的 还是有本质区别的。

java本身的虚拟机里面会关注对象的申请，释放，这些不需要用户直接注，java虚拟机通过管理机制，将调用c c++里面真正的malloc free 方法，封装起来，将java对象的生命周期和malloc free 进行关联，则可以保证在对象不使用的时候，内存紧张时，释放掉不再被引用的对象，GC回收就是在做这件事请。回到我们这节的主要内容，如何定位我们的c c++的内存泄露。

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。Android中kernel和应用程序都可能会有内存泄漏和越界。对于Java代码，在越界的时候虚拟机会加以检查并抛出异常。而对于C/C++代码，越界的时候就悄无声息地让程序出错或crash

### 调试手段

在adb shell之下:确保/system/lib/ libc\_malloc\_debug\_leak.so存在

root@zs600b:/ # stop

root@zs600b:/ # setprop libc.debug.malloc 1

root@zs600b:/ # start

执行成功之后，浏览器和cs-app都无法打开了。。。

指针溢出，野指针，堆内存指针释放多遍等问题如何调试？

Android这边打开bionic/libc/bionic/malloc\_debug\_common.c里的配置

如adb shell setprop persist.libc.debug.malloc 10 再重启

之后可以复现问题，抓bugreport来分析问题，当然要匹配symbols(out/target/product/project/symbols)

此时overhead会比较重，可能会有一些不预期的anr(出现ANR时请点击等待)，但不影响测试。

start & stop are tools under /system/bin (system.img)

### 常见问题

/system/bin/sh: Missing module libc\_malloc\_debug\_leak.so required for malloc debug level 1: dlopen failed: library "libc\_malloc\_debug\_leak.so" not found

userdebug才是自带so，否则自己手动push进去吧。。，。

可以看看eng或这user-debug版中的/system/lib/下是否有这两个文件，其中libc\_malloc\_debug\_qemu.so是模拟器用的。

### 复现场景

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 (double free)

2. 释放后又去使用 (used after free)

3. 使用越界 (比如申请了50节内存，结果在使用时多用了8字节的内存，这样就把后面的内存的内容踩坏，引起堆结构异常)

4. 释放时传给free()的地址不是malloc()申请的地址，比如：p = malloc(10); free(p + 5);

5. 内存泄露：申请内存后，忘记释放或某些代码路径没有释放

### 源码分析

此malloc的调试原理是：当系统发现我们有libc.debug.malloc的一些列配置成立时，此时系统会将malloc free 等方法，重新指向到 lib\_malloc\_debug.so里面的对应实现方法，lib\_malloc\_debug.so里面的方法，像比较而言，多了一些记录信息，将每次的申请时的地址，堆栈，so等信息记录下来，然后我们需要的时候，则通过工具ddms dump出来，进行分析每个申请的内存，是否正常的释放了，是否出现了内存泄露。

|  |
| --- |
| // The value of libc.debug.malloc.  #if !defined(LIBC\_STATIC)  static int g\_malloc\_debug\_level = 0;  #endif  // Initializes memory allocation framework once per process.  static void malloc\_init\_impl() {  const char\* so\_name = NULL;  // If debug level has not been set by memcheck option in the emulator,  // lets grab it from libc.debug.malloc system property.  if (g\_malloc\_debug\_level == 0 && \_\_system\_property\_get("libc.debug.malloc", env)) {  g\_malloc\_debug\_level = atoi(env);  }  // Choose the appropriate .so for the requested debug level.  switch (g\_malloc\_debug\_level) {  case 1:  case 5:  case 10:  so\_name = "libc\_malloc\_debug\_leak.so";  break;  }  // Load .so that implements the required malloc debugging functionality.  void\* malloc\_impl\_handle = dlopen(so\_name, RTLD\_LAZY);  // No need to init the dispatch table because we can only get  // here if debug level is 1, 5, 10, or 20.  static MallocDebug malloc\_dispatch\_table \_\_attribute\_\_((aligned(32)));  switch (g\_malloc\_debug\_level) {  case 1:  InitMalloc(malloc\_impl\_handle, &malloc\_dispatch\_table, "leak");  break;  }  } |

android系统启动流程

## framework debug 技巧

### 安装

拷贝文件到system/framework/;  
stop;  
start重启system\_server进程

### Java调试

其实整个调试过程非常简单：

* 打断点
* 跟踪代码（Step in/out/over等等）

### 打断点

在正确的进程的合适位置打断点:区别于普通app的进程，framwork的所在进程需要进行源码分析，比如ams是运行在system\_server中的，并且这些进程只能在root的机子调试（模拟器或者root真机）

### 跟踪代码

### 打印日志

-系统一般都会有debug标志，我们修改源码中的标志，打印就好，比如

修改后

-- Log.e("tag",logStr);

### REF

[android native 代码内存泄露 定位方案](http://www.jianshu.com/p/2b43abdd6647)

[[MTK软件原创] 如何调试malloc(堆越界)问题](http://bbs.16rd.com/thread-54815-1-1.html)

[more detail about adb shell start, stop and reboot](http://blog.csdn.net/johnnylq/article/details/6401531)

[android native 内存泄露检查（libc.debug.malloc ）](http://blog.csdn.net/haima1998/article/details/51508947)

# 日志分析logcat

Logcat是存到缓存到内存中的，英文

打印的都是 /dev/log

[Android6.0 Log的工作机制，源码分析](https://zhuanlan.zhihu.com/p/24372024)

adb logcat -c  
清除log缓存

adb bugreport  
查看bug报告

查看event节点，调试activity生命周期

logcat -b events -v time

logcat -b events -v time | grep am\_

查看slog

adb logcat -b system -v time | grep WindowManagerService

logcat -b system -v time | grep WindowManagerService > /sdcard/dlg.txt

cat /proc/kmsg 实时打印内核日志

内核的 dmesg 静态显示内核日志

## 日志抓取方案

Android日志主要分为kernel、radio、event、main这四种log。

dmesg

kernel log属于Linux内核的log ，可以通过读取/proc/kmsg或者通过串口来抓取。

adb 抓取kernel log的命令如下（需要有root权限）：

kernel.log抓取

adb shell rm -rf /sdcard/logs

adb shell mkdir /sdcard/logs

adb reboot

adb root

adb remount

adb shell

#dmesg > /sdcard/logs/kernel.log

cat /proc/kmsg > /sdcard/logs/kernel0324.log

adb shell

logcat -b main -b system -b radio -b events -v time > /sdcard/logs/logcat0324.log

adb shell

. hardware\_monitor.sh > /sdcard/logs/hardware0324.log

cp -rf /data/tombstones /sdcard/logs/

cp -rf /data/anr /sdcard/logs/

exit

adb pull /sdcard/logs

## 抓到pc端

adb shell logcat -b main -b system -b radio -b events -v time > logcat0606.log

# Android cmd

## Start/stop

## Pm

pm clear packagename 在root设备才生效！

pm list packages –f

查看所有已安装的应用的包名

pm disable <package name>

禁用启用app

* pm list packages [-f] [-d] [-e] [-s] [-3] [-i] [-u] [--user USER\_ID] [FILTER]  
  查看应用列表,**pm即PackageManager**

| **参数** | **解释** |
| --- | --- |
| -f | 显示应用关联的 apk 文件 |
| -d | 只显示 disabled 的应用 |
| -e | 只显示 enabled 的应用 |
| -s | 只显示系统应用 |
| -3 | 只显示第三方应用 |
| -i | 显示应用的 installer |
| -u | 包含已卸载应用 |
| <FILTER> | 包名包含 <FILTER> 字符串 |

## am

adb shell am broadcast -a android.intent.action.test.XX --ei ei3 1：发送传整数ei3 = 1d的XX广播

adb shell am start -n packageName/ActivityFullName：发送启动Activity的广播

am force-stop packageName

adb shell am start -a android.intent.action.ACTION\_REQUEST\_SHUTDOWN -d <http://gityuan.com>

adb shell am broadcast -a android.intent.action.ACTION\_REQUEST\_SHUTDOWN --ez android.intent.extra.KEY\_CONFIRM false

adb shell am start -n com.android.settings/com.android.settings.FallbackHome

查询版本

rm500:/ $ ps | grep dji

root 144 2 0 0 0 0000000000 S dji\_bat\_charge\_

root 325 1 45520 10472 0 0000000000 S /system/bin/djilink

system 1291 326 1574668 77052 0 0000000000 S com.dji

u0\_a38 1498 327 1819772 156924 0 0000000000 R dji.go.v4

u0\_a38 1523 1498 1799968 57196 0 0000000000 S dji.go.v4

rm500:/ $ dumpsys package dji.go.v4 | grep version

versionCode=31245 minSdk=19 targetSdk=21

versionName=4.2.12

## input

模拟上下左右 确定键值  
adb shell input keyevent  "value"  
value:是键值对应的数值  
 KEYCODE\_DPAD\_UP     19  
 KEYCODE\_DPAD\_DOWN  20  
 KEYCODE\_DPAD\_LEFT 21  
 KEYCODE\_DPAD\_RIGHT 22  
 KEYCODE\_DPAD\_CENTER 23

其他模拟手段：

<https://itimetraveler.github.io/2016/06/02/%E3%80%90Android%E3%80%91%E6%8A%80%E6%9C%AF%E8%B0%83%E7%A0%94%EF%BC%9A%E7%94%A8%E4%BB%A3%E7%A0%81%E6%A8%A1%E6%8B%9F%E5%B1%8F%E5%B9%95%E7%82%B9%E5%87%BB%E3%80%81%E8%A7%A6%E6%91%B8%E4%BA%8B%E4%BB%B6/>

## 实用功能

* 1、 adb shell input keyevent <keycode>  
  adb 命令代替键盘操作，不同的 keycode 能实现不同的功能  
  keycode 移步：<https://developer.android.com/reference/android/view/KeyEvent.html>
* 2、adb shell input text 123  
  输入文本在焦点处于某文本框时，可以通过 input 命令来输入文本。现在 123 出现在文本框中。
* 3、adb shell screencap -p /sdcard/sc.png  
  屏幕截图
* 4、adb shell screenrecord /sdcard/filename.mp4  
  录制 mp4 格式的视频保存到 /sdcard

# Linux cmd

## 网络

netstat | grep 8088：查看网络端口，通常用来调试。

### tcpdump

普通情况下，直接启动tcpdump将监视第一个网络接口上所有流过的数据包。

监视指定网络接口的数据包

tcpdump -i eth1

如果不指定网卡，默认tcpdump只会监视第一个网络接口，一般是eth0，下面的例子都没有指定网络接口。

#### 监视指定主机和端口的数据包

如果想要获取主机210.27.48.1接收或发出的telnet包，使用如下命令

tcpdump tcp port 23 host 210.27.48.1

对本机的udp 123 端口进行监视 123 为ntp的服务端口

tcpdump udp port 123

#### tcpdump 与wireshark联合使用

Wireshark(以前是ethereal)是Windows下非常简单易用的抓包工具。但在Linux下很难找到一个好用的图形化抓包工具。  
还好有Tcpdump。我们可以用Tcpdump + Wireshark 的完美组合实现：在 Linux 里抓包，然后在Windows 里分析包。

tcpdump tcp -i eth1 -t -s 0 -c 100 and dst port ! 22 and src net 192.168.1.0/24 -w ./target.cap

(1)tcp: ip icmp arp rarp 和 tcp、udp、icmp这些选项等都要放到第一个参数的位置，用来过滤数据报的类型  
(2)-i eth1 : 只抓经过接口eth1的包  
(3)-t : 不显示时间戳  
(4)-s 0 : 抓取数据包时默认抓取长度为68字节。加上-S 0 后可以抓到完整的数据包  
(5)-c 100 : 只抓取100个数据包  
(6)dst port ! 22 : 不抓取目标端口是22的数据包  
(7)src net 192.168.1.0/24 : 数据包的源网络地址为192.168.1.0/24  
(8)-w ./target.cap : 保存成cap文件，方便用ethereal(即wireshark)分析

### WIFI

#### Softap wpa\_cli的使用

在平台开发初期阶段，对于WIFI相关问题，上层应用如果还无法使用，或者需要定位问题的层次，这时可以借助wpa\_cli，下面介绍一些wpa\_cli 调试WIFI的基础用法。

借助下面命令可以排除：

Wifi扫描、Wifi连接、Wifi配置等问题。

在串口中执行下面命令：

root@k200:/ # wpa\_cli -i wlan0 -p wlan0

wpa\_cli v2.1-devel-4.3

Copyright (c) 2004-2013, Jouni Malinen <j@w1.fi> and contributors

This software may be distributed under the terms of the BSD license.

See README for more details.

Interactive mode

>

此时进入与wpa\_supplicant 的交换模式。输入h能查看所有wpa\_cli的支持的命令，由于命令太多，此处不贴出来。下面以表格形式介绍wpa\_cli基本命令以及用法：

|  |  |  |  |
| --- | --- | --- | --- |
| **命令** | **命令缩写** | **描述** | **测试** |
| status | stat | 显示当前连接的状态信息 | > stat  bssid=c8:3a:35:16:89:88  ssid=Tenda\_168988  id=1  mode=station  pairwise\_cipher=CCMP  group\_cipher=CCMP  key\_mgmt=WPA-PSK  wpa\_state=COMPLETED  ip\_address=192.168.0.142  address=00:50:43:02:ff:01  > |
| ping |  | 测试指令，看service端是否有响应 | > ping  PONG  > |
| interface |  | 显示当前可用的interface，wpa\_supplicant 支持多interface | > interface  Available interfaces:  wlan0  > |
| set |  | 设置环境变量，如果不带参数，显示当前环境变量 | > set  set variables:  EAPOL::heldPeriod (EAPOL state machine held period, in seconds)  EAPOL::authPeriod (EAPOL state machine authentication period, in seconds)  EAPOL::startPeriod (EAPOL state machine start period, in seconds)  EAPOL::maxStart (EAPOL state machine maximum start attempts)  dot11RSNAConfigPMKLifetime (WPA/WPA2 PMK lifetime in seconds)  dot11RSNAConfigPMKReauthThreshold (WPA/WPA2 reauthentication threshold  percentage)  dot11RSNAConfigSATimeout (WPA/WPA2 timeout for completing security  association in seconds)  > |
| reassociate | reas | 设置连接标签，开启扫描，查找可用的AP进行连接，如果当前是连接状态，只会扫描一次。 | <3>CTRL-EVENT-SCAN-RESULTS  <3>WPS-AP-AVAILABLE  <3>Trying to associate with 94:ba:56:00:32:14 (SSID='yl\_B19W4' freq=2462 MHz)  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=5 BSSID=94:ba:56:00:32:14  <3>CTRL-EVENT-STATE-CHANGE id=0 state=6 BSSID=94:ba:56:00:32:14  <3>Associated with 94:ba:56:00:32:14  <3>CTRL-EVENT-CONNECTED - Connection to 94:ba:56:00:32:14 completed (reauth) [id=0 id\_str=]  <3>CTRL-EVENT-STATE-CHANGE id=0 state=9 BSSID=00:00:00:00:00:00  <3>CTRL-EVENT-STATE-CHANGE id=0 state=0 BSSID=00:00:00:00:00:00  <3>CTRL-EVENT-DISCONNECTED bssid=00:00:00:00:00:00 reason=0  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=3 BSSID=00:00:00:00:00:00 |
| preauthenticate | pr | 针对一个AP进行预认证 | > pr c8:3a:35:16:89:88  OK  <3>RSN: failed to get master session key from pre-auth EAPOL state machines  <3>RSN: pre-authentication with c8:3a:35:16:89:88 failed |
| identity | id | 设置identity | > id 0 xxxx  OK  > save\_config  OK  >  cat wpa\_supplicant.conf  network={  ssid="yl\_B19W4"  key\_mgmt=NONE  identity="xxxx"  password="123455678"  disabled=1  } |
| password |  | 设置密码 | password 0<network id> 11111111  network={  ssid="yl\_B19W4"  key\_mgmt=NONE  identity="xxxx"  password="11111111111"  disabled=1  } |
| disconnect | disc | 断开连接，并且会告知wpa\_supplicant 不在连接其他AP | > disc  OK  <3>CTRL-EVENT-STATE-CHANGE id=1 state=0 BSSID=00:00:00:00:00:00  <3>CTRL-EVENT-DISCONNECTED bssid=00:00:00:00:00:00 reason=0 |
| reconnect | reconn | 重新连接，在运行上面的过程后，运行此命令 | > reconn  OK  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=3 BSSID=00:00:00:00:00:00  <3>CTRL-EVENT-SCAN-RESULTS  <3>WPS-AP-AVAILABLE  <3>Trying to associate with a4:a8:0f:fc:18:3a (SSID='XXXXXXX' freq=2462 MHz)  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=5 BSSID=a4:a8:0f:fc:18:3a  <3>CTRL-EVENT-STATE-CHANGE id=1 state=6 BSSID=a4:a8:0f:fc:18:3a  <3>Associated with a4:a8:0f:fc:18:3a  <3>CTRL-EVENT-CONNECTED - Connection to a4:a8:0f:fc:18:3a completed (reauth) [id=1 id\_str=]  <3>CTRL-EVENT-STATE-CHANGE id=1 state=9 BSSID=00:00:00:00:00:00 |
| quit | q | 退出wpa\_cli |  |
| terminate | term | 杀掉后台wpa\_supplicant进程 |  |
| reconfigure | recon | 重新导入配置文件，可以在调试时，修改配置文件后运行此命令，使配置文件生效 |  |
| scan | scan | 扫描 | scan  > OK  <3>CTRL-EVENT-SCAN-RESULTS  <3>WPS-AP-AVAILABLE |
| scan\_result | scan\_r | 显示扫描结果 | scan\_r  > bssid / frequency / signal level / flags / ssid  a4:a8:0f:fc:18:3a 2462 -59 [ESS] XXXXXXX  d4:ee:07:03:ee:0c 2427 -68 [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS] HiWiFi\_03EE0C  d8:5d:4c:25:d0:f2 2412 -72 [WPA-PSK-CCMP][WPA2-PSK-CCMP-preauth][ESS] OTG  c8:3a:35:16:89:88 2462 -83 [WPA-PSK-CCMP][ESS] Tenda\_168988 |
| bss |  | 获取扫描结果 某一个热点的具体信息 | bss <<idx> | <bssid>>  bss 1  > id=371  bssid=d4:ee:07:03:ee:0c  freq=2427  beacon\_int=100  capabilities=0x0431  qual=0  noise=0  level=-72  tsf=0000016060803119  ie=000d4869576946695f303345453043010882848b960c1218240301042a010032043048606c2d1a6e101bff000000000000000000000000000000000000000000003d16040d000000000000000000000000000000000000000030140100000fac040100000fac040100000fac020000dd160050f20101000050f20401000050f20401000050f202dd180050f2020101000003a4000027a4000042435e0062322f00dd06d4ee07010100  flags=[WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]  ssid=HiWiFi\_03EE0C  > |
| list\_networks | list\_n | 已经保存配置文件中的AP | > list  network id / ssid / bssid / flags  0 yl\_B19W4 c8:3a:35:16:89:88  1 XXXXXXX any  > |
| bssid |  | 配置ssid的bssid | bssid 0 c8:3a:35:16:89:88  结果下面可以看到 |
| add\_network | add\_n | 添加一个AP配置，在framework还没调通之前，可以用此命令进行配置 | > add\_n  7  > save\_config  OK  > list  network id / ssid / bssid / flags  0 yl\_B19W4 c8:3a:35:16:89:88 [DISABLED]  2 客厅里的飞看\_B186M any [DISABLED]  3 客厅里的飞看\_B1PKO any [DISABLED]  4 XXXXXXX any [CURRENT]  5 Tenda\_168988 any [DISABLED]  6 any [DISABLED]  7 any [DISABLED] --🡪添加的  >save\_config --🡪如果想保存到配置文件中 |
| set\_network | set\_n | 设置ap的属性，用set\_network 不接参数，可以看到所有可用信息。 | > set\_n  set\_network variables:  ssid (network name, SSID)  psk (WPA passphrase or pre-shared key)  key\_mgmt (key management protocol)  identity (EAP identity)  password (EAP password)  ...  ------------------------------------------------------  > set\_n 7 ssid "B2C2.5"  OK  > list  network id / ssid / bssid / flags  0 yl\_B19W4 c8:3a:35:16:89:88 [DISABLED]  2 客厅里的飞看\_B186M any [DISABLED]  3 客厅里的飞看\_B1PKO any [DISABLED]  4 XXXXXXX any [CURRENT]  5 Tenda\_168988 any [DISABLED]  6 any [DISABLED]  7 B2C2.5 any [DISABLED]  > set\_n 7 psk 22225555  FAIL  > set\_n 7 psk "22225555"  OK  > list  network id / ssid / bssid / flags  0 yl\_B19W4 c8:3a:35:16:89:88 [DISABLED]  2 客厅里的飞看\_B186M any [DISABLED]  3 客厅里的飞看\_B1PKO any [DISABLED]  4 XXXXXXX any [CURRENT]  5 Tenda\_168988 any [DISABLED]  6 any [DISABLED]  7 B2C2.5 any [DISABLED]  > save  OK  配置成功 |
| select\_network | select\_n | 选择一个已经配置的AP进行连接，这个命令用的比较多。上面set\_n 命令进行配置后，可以用这命令进行连接 | > sel 7  OK 这样supplicant会自动连接B2C2.5 这个热点  --------------  > stat  bssid=c8:3a:35:16:89:88  ssid=Tenda\_168988  id=5  mode=station  pairwise\_cipher=CCMP  group\_cipher=CCMP  key\_mgmt=WPA-PSK  wpa\_state=COMPLETED  ip\_address=192.168.0.142  address=00:50:43:02:ff:01  ---------------------------------------  查看当前状态  ---------------------------------------  > list  network id / ssid / bssid / flags  0 yl\_B19W4 c8:3a:35:16:89:88 [DISABLED]  2 客厅里的飞看\_B186M any [DISABLED]  3 客厅里的飞看\_B1PKO any [DISABLED]  4 XXXXXXX any [DISABLED]  5 Tenda\_168988 any [CURRENT]  ---------- 查看当前可选的AP  ----------  > sel 4  OK  <3>CTRL-EVENT-STATE-CHANGE id=5 state=0 BSSID=00:00:00:00:00:00  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=3 BSSID=00:00:00:00:00:00  <3>CTRL-EVENT-DISCONNECTED bssid=00:00:00:00:00:00 reason=0  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=0 BSSID=00:00:00:00:00:00  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=3 BSSID=00:00:00:00:00:00  <4>Failed to initiate AP scan  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=0 BSSID=00:00:00:00:00:00  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=3 BSSID=00:00:00:00:00:00  <4>Failed to initiate AP scan  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=0 BSSID=00:00:00:00:00:00  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=3 BSSID=00:00:00:00:00:00  <4>Failed to initiate AP scan  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=0 BSSID=00:00:00:00:00:00  <3>CTRL-EVENT-SCAN-RESULTS  <3>WPS-AP-AVAILABLE  <3>Trying to associate with a4:a8:0f:fc:18:3a (SSID='XXXXXXX' freq=2462 MHz)  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=5 BSSID=a4:a8:0f:fc:18:3a  <3>CTRL-EVENT-STATE-CHANGE id=4 state=6 BSSID=a4:a8:0f:fc:18:3a  <3>Associated with a4:a8:0f:fc:18:3a  <3>CTRL-EVENT-CONNECTED - Connection to a4:a8:0f:fc:18:3a completed (reauth) [id=4 id\_str=]  <3>CTRL-EVENT-STATE-CHANGE id=4 state=9 BSSID=00:00:00:00:00:00  ------- 选择进行连接 -------  stat  > bssid=a4:a8:0f:fc:18:3a  ssid=XXXXXXX  id=4  mode=station  pairwise\_cipher=NONE  group\_cipher=NONE  key\_mgmt=NONE  wpa\_state=COMPLETED  ip\_address=192.168.88.101  address=00:50:43:02:ff:01  >  查看当前状态，连接ok |
| enable\_network | enable\_n | 将network 的标志位  disable 置0，该位为1时，wifi在断开连接状态，不会去主动连接该热点。如果wifi不主动连接任何热点，可以查看配置文件。 | network={  ssid="客厅里的飞看\_B186M"  key\_mgmt=NONE  disabled=1  }  network={  ssid="客厅里的飞看\_B1PKO"  key\_mgmt=NONE  disabled=1  }  network={  ssid="XXXXXXX"  key\_mgmt=NONE  }  network={  ssid="Tenda\_168988"  psk="12345678"  key\_mgmt=WPA-PSK  disabled=1  } |
| disable\_network | disable\_n | 对应上面  disable\_network network\_id |  |
| remove\_network | remove\_n | 忘记网络  remove\_network id  会删除对应network配置 |  |
| get\_network | get\_n | 获取network的属性  对应之前的set\_network | set\_network variables:  ssid (network name, SSID)  psk (WPA passphrase or pre-shared key)  key\_mgmt (key management protocol)  identity (EAP identity)  password (EAP password)  ...  > get\_network 0 ssid  "yl\_B19W4"  > list  network id / ssid / bssid / flags  0 yl\_B19W4 c8:3a:35:16:89:88 [DISABLED]  2 客厅里的飞看\_B186M any [DISABLED]  3 客厅里的飞看\_B1PKO any [DISABLED]  4 XXXXXXX any [DISABLED]  5 Tenda\_168988 any [DISABLED]  6 any [DISABLED]  7 B2C2.5 any [DISABLED]  > |
| wps\_pbc |  | wps push button功能 | 前提:路由开启wps push button功能  下面我已经打开Tenda\_168988 wps push button功能，然后在盒子上运行wps\_pbc  wps\_pbc  > OK  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=3 BSSID=00:00:00:00:00:00  <3>CTRL-EVENT-SCAN-RESULTS  <3>WPS-AP-AVAILABLE  <3>CTRL-EVENT-SCAN-RESULTS  <3>WPS-AP-AVAILABLE  <3>CTRL-EVENT-SCAN-RESULTS  <3>WPS-AP-AVAILABLE  <3>CTRL-EVENT-SCAN-RESULTS  <3>WPS-AP-AVAILABLE-PBC  <3>Trying to associate with c8:3a:35:16:89:88 (SSID='Tenda\_168988' freq=2412 MHz)  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=5 BSSID=c8:3a:35:16:89:88  <3>CTRL-EVENT-STATE-CHANGE id=8 state=6 BSSID=c8:3a:35:16:89:88  <3>Associated with c8:3a:35:16:89:88  <3>CTRL-EVENT-EAP-STARTED EAP authentication started |
| wps\_pin |  | Wps pin 功能 | 前提:打开路由热点wps功能，并设置pin值 如12345678，用scan\_r查看你想要连接的ssid的bssid，然后用下面命令进行连接  > wps\_pin c8:3a:35:16:89:88 12345678  12345678  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=3 BSSID=00:00:00:00:00:00  <3>CTRL-EVENT-SCAN-RESULTS  <3>WPS-AP-AVAILABLE-AUTH  <3>Trying to associate with c8:3a:35:16:89:88 (SSID='Tenda\_168988' freq=2412 MHz)  <3>CTRL-EVENT-STATE-CHANGE id=-1 state=5 BSSID=c8:3a:35:16:89:88  <3>CTRL-EVENT-STATE-CHANGE id=8 state=6 BSSID=c8:3a:35:16:89:88  <3>Associated with c8:3a:35:16:89:88  <3>CTRL-EVENT-EAP-STARTED EAP authentication started |
| save\_config | save\_c | 保存信息到配置文件中，每次修改配置后记得用 |  |
|  |  |  |  |
|  |  |  |  |

#### Softap hostapd\_cli的使用

hostapd\_cli相对wpa\_cli来说比较简单，能使用的命令也非常少。

借助下面命令可以解决：

wifi热点配置、查看设备连接时长等问题。

hostapd\_cli –h

Options:

-h help (show this usage text)

-v shown version information

-p<path> path to find control sockets (default: /var/run/hostapd)

-a<file> run in daemon mode executing the action file based on events

from hostapd

-B run a daemon in the background

-i<ifname> Interface to listen on (default: first interface found in the

socket path)

hostapd\_cli 运行:

hostapd\_cli -i uap0 -p /data/misc/wifi/hostapd

下面列表列出了hostapd\_cli能使用的指令，以及使用方法：

|  |  |  |  |
| --- | --- | --- | --- |
| 命令 | 命令缩写 | 描述 | 测试 |
| all\_sta | al | 显示当前连入的STA,并记录连上的时间长度 | > all\_sta  40:f3:08:39:b6:74  connected\_time=579  00:90:4c:78:5d:5c  connected\_time=3359  > all  00:90:4c:78:5d:5c  connected\_time=3458 |
| deauthenticate <addr> | dea | 解除验证 | > all  40:f3:08:39:b6:74 ---🡪我的设备  connected\_time=59  00:90:4c:78:5d:5c  connected\_time=4008  > dea 40:f3:08:39:b6:74 -🡪可以看到设备断开，又启动连接  OK |
| disassociate <addr> |  | 断开连接 | 过程与上面差不多 |
| wps\_pin |  | AP端的wps功能 |  |
| wps\_pbc |  | Ap端的wps功能 |  |
| wps\_config <SSID> <auth> <encr> <key> |  | 配置Ap的属性 |  |

-s snaplen ] [ -T type ] [ -w file ]  
 [ -W filecount ]  
 [ -E spi@ipaddr algo:secret,... ]  
 [ -y datalinktype ] [ -Z user ]  
 [ expression ]

#### 打开wifi-log

|  |
| --- |
| 1. <4>[ 3303.925164] wifi\_power\_write: poweren = 1 2. <6>[ 3303.925191] [WLAN\_RFKILL]: rockchip\_wifi\_power: 1 3. <6>[ 3303.925201] [BT\_RFKILL]: rfkill\_get\_bt\_power\_state: rfkill-bt driver has not Successful initialized 4. <6>[ 3303.925210] [WLAN\_RFKILL]: rockchip\_wifi\_ref\_voltage: 1 5. <6>[ 3303.925218] [WLAN\_RFKILL]: rockchip\_wifi\_ref\_voltage: wifi io reference voltage control is disabled. 6. <6>[ 3304.027411] extfa98xx\_control(): status: ON. 7. <6>[ 3304.027443] extfa98xx\_control(): profile: 0, vstep: 0. 8. <6>[ 3304.027604] rk\_rt5650:Enter::rk\_rt5650\_hw\_params----85 9. <6>[ 3304.027623] rk\_rt5650:Enter:rk\_rt5650\_hw\_params, 129, rate=48000 10. <6>[ 3304.027637] rk\_rt5650:Enter:rk\_rt5650\_hw\_params, 143, pll\_out/4/params\_rate(params) = 64 11. <6>[ 3305.805955] EHCI: rk\_ehci\_hcd\_enable, enable host controller 12. <6>[ 3306.027223] usb 1-1: new high-speed USB device number 10 using rockchip\_ehci\_host 13. <6>[ 3306.177152] usb 1-1: New USB device found, idVendor=0cf3, idProduct=1022 14. <6>[ 3306.177339] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3 15. <6>[ 3306.177435] usb 1-1: Product: USBWLAN 16. <6>[ 3306.177511] usb 1-1: Manufacturer: Qualcomm Atheros. 17. <6>[ 3306.177590] usb 1-1: SerialNumber: 12345678 18. <6>[ 3306.883487] [WLAN\_RFKILL]: wifi turn on power. 171 19. <6>[ 3306.908010] Calling CRDA to update world regulatory domain 20. <6>[ 3306.908644] World regulatory domain updated: 21. <6>[ 3306.908722] (start\_freq - end\_freq @ bandwidth), (max\_antenna\_gain, max\_eirp) 22. <6>[ 3306.908812] (2302000 KHz - 2382000 KHz @ 20000 KHz), (300 mBi, 500 mBm) 23. <6>[ 3306.908893] (2402000 KHz - 2482000 KHz @ 20000 KHz), (300 mBi, 500 mBm) 24. <6>[ 3306.908972] (2474000 KHz - 2494000 KHz @ 20000 KHz), (300 mBi, 500 mBm) 25. <6>[ 3306.909048] (5110000 KHz - 5250000 KHz @ 20000 KHz), (300 mBi, 500 mBm) 26. <6>[ 3306.909125] (5250000 KHz - 5710000 KHz @ 20000 KHz), (300 mBi, 500 mBm) 27. <6>[ 3306.909201] (5735000 KHz - 5835000 KHz @ 20000 KHz), (300 mBi, 500 mBm) 28. <6>[ 3306.917316] 6ath6kl: 3VAP/1, P2P enable, concurrent on , with dedicate p2p-device, multi-channel-concurrent enable, p2p-compat disable, sta-p2p-ie r 29. d, p2p\_wise\_scan enable 30. <6>[ 3306.918561] 6ath6kl: target bootstrap: 0x00000800 31. <6>[ 3306.919563] 6ath6kl: target's subtype is 0x7, HT20/40 2SS Dual-band 32. <6>[ 3306.924038] 6ath6kl: Create HTC cookie, type 1 num 440 33. <6>[ 3306.924091] 6ath6kl: Create HTC cookie, type 2 num 66 34. <6>[ 3306.924112] 6ath6kl: P2P flowctrl enabled 35. <6>[ 3306.924126] 6ath6kl: RX aggregation drop disabled 36. <6>[ 3306.924142] 6ath6kl: Using driver's regdb. 37. <6>[ 3307.139933] 6ath6kl: ar6004 hw 3.0 usb fw 3.5.0.607 38. <6>[ 3307.140166] Calling CRDA for country: CN 39. <6>[ 3307.140567] 6ath6kl: Ignore reg notifier call back, since initiator is driver 40. <6>[ 3307.140620] Regulatory domain changed to country: CN 41. <6>[ 3307.140660] (start\_freq - end\_freq @ bandwidth), (max\_antenna\_gain, max\_eirp) 42. <6>[ 3307.140710] (2402000 KHz - 2482000 KHz @ 20000 KHz), (400 mBi, 2000 mBm) 43. <6>[ 3307.140758] (5170000 KHz - 5250000 KHz @ 20000 KHz), (500 mBi, 2000 mBm) 44. <6>[ 3307.140804] (5735000 KHz - 5835000 KHz @ 20000 KHz), (500 mBi, 3200 mBm) 45. <6>[ 3307.146820] 6ath6kl: Wait defer tasks done... 46. <6>[ 3307.147262] 6ath6kl: Create dedicated p2p interface 47. <6>[ 3307.168883] 6ath6kl: Enable Firmware crash notiry. 48. <6>[ 3307.169621] usbcore: registered new interface driver ath6kl\_usb 49. <4>[ 3307.172931] Current WiFi chip is AR1021X. 50. <6>[ 3307.209728] extfa98xx\_control(): status: OFF. 51. <6>[ 3307.278231] IPv6: ADDRCONF(NETDEV\_UP): wlan0: link is not ready 52. <4>[ 3307.634653] Current WiFi chip is AR1021X. 53. <6>[ 3307.642981] IPv6: ADDRCONF(NETDEV\_UP): p2p0: link is not ready |

adb shell cat /data/misc/wifi/\*.conf  
查看wifi密码：

## lsusb

lsusb命令用于显示本机的USB设备列表，以及USB设备的详细信息。 lsusb命令是一个学习USB驱动开发，认识USB设备的助手，推荐大家使用，如果您的开发板中或者产品中没有lsusb命令可以自己移植一个，放到文件系统里面。来自: <http://man.linuxde.net/lsusb>

## grep

grep -E 'aaaaa|bbbbbb'

grep -E ‘net|dns’

# 性能测试工具之dumpsys

http://gityuan.com/2016/05/14/dumpsys-command/

在Android开发过程中，经常会遇到很多随机的问题，比如CPU过高、内存泄露等。Dumpsys是一个可以查看系统信息的命令，便于我们通过系统日志快速定位问题。

## dumpsys实现原理

dumpsys工具比较简单，通过调用ServiceManager服务的listServices来查询系统注册的所有服务，并通过checkService接口来获取服务的Binder远程代理对象，使用每个服务的dump函数打印该服务的相关信息。

相关源码路径：./frameworks/native/cmds/dumpsys/dumpsys.cpp

./frameworks/native/libs/binder/IServiceManager.cpp



图 1 dumpsys实现原理图

//dumpsys.cpp的函数入口

int main(int argc, char\* const argv[])

{

signal(SIGPIPE, SIG\_IGN);

//取得ServiceManager服务的远程Binder代理对象

sp<IServiceManager> sm = defaultServiceManager();

fflush(stdout);

if (sm == NULL) {

ALOGE("Unable to get default service manager!");

aerr << "dumpsys: Unable to get default service manager!" << endl;

return 20;

}

Vector<String16> services;

Vector<String16> args;

if (argc == 1) {

//使用ServiceManager服务的Binder远程代理对象查询所有的服务

services = sm->listServices();

services.sort(sort\_func);

args.add(String16("-a"));

} else {

//添加命令行指定的服务及参数

services.add(String16(argv[1]));

for (int i=2; i<argc; i++) {

args.add(String16(argv[i]));

}

}

//根据服务名称查找服务对应的Binder代理对象，如果存在，打印该服务

const size\_t N = services.size();

if (N > 1) {

// first print a list of the current services

aout << "Currently running services:" << endl;

//调用服务的dump函数打印服务相关信息

for (size\_t i=0; i<N; i++) {

sp<IBinder> service = sm->checkService(services[i]);

if (service != NULL) {

aout << " " << services[i] << endl;

}

}

}

for (size\_t i=0; i<N; i++) {

sp<IBinder> service = sm->checkService(services[i]);

if (service != NULL) {

if (N > 1) {

aout << "------------------------------------------------------------"

"-------------------" << endl;

aout << "DUMP OF SERVICE " << services[i] << ":" << endl;

}

//调用每个service中的dump方法，输出log

int err = service->dump(STDOUT\_FILENO, args);

if (err != 0) {

aerr << "Error dumping service info: (" << strerror(err)

<< ") " << services[i] << endl;

}

} else {

aerr << "Can't find service: " << services[i] << endl;

}

}

return 0;

}

## dumpsys使用方法

通过service list命令查看可以用于dumpsys的所有服务

130|root@android:/ # service list

Found 77 services:

0 sip: [android.net.sip.ISipService]

…

图 3 dumpsys Service示意图

大部分Android services都是在SystemServer中的main函数里注册的，它调用了一个叫init2的函数,然后ServerThead的run方法中进行服务注册。

.\framework\base\services\java\com\android\server\SystemServer.java)



其他 native services是在SystemServer中的main函数里注册的,它调用本地函数init1()，



然后调用的是JNI的里面的android\_server\_SystemServer\_init1函数。

.\framework\base\services\jni\com\_android\_server\_SystemServer.cpp



然后调用的system\_init函数。.\frameworks\base\cmds\system\_server

\library\System\_init.cpp

### CM系统上面可支持的dumpsys的服务如下表：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| HiDisplay | **SurfaceFlinger** | accessibility | account | **activity** | alarm | appwidget | hardware |
| audio | backup | battery | batteryinfo | bluetooth\_manager | clipboard | commontime\_management | connectivity |
| content | country\_detector | cpuinfo | dbinfo | device\_policy | diskstats | devicestoragemonitor | display |
| dreams | drm.drmManager | dropbox | entropy | ethernet | fm\_receiver | fm\_transmitter | gfxinfo |
| input | input\_method | isms | **meminfo** | iphonesubinfo | **mount** | assetredirection | phone |
| netpolicy | lock\_settings | notification | **package** | media.audio\_flinger | uimode | servicediscovery | power |
| netstats | pieservice | statusbar | updatelock | media.audio\_policy | usagestats | network\_management | profile |
| sip | textservices | throttle | vibrator | media.camera | search | permission |  |
| usb | telephony.registry | user | wallpaper | media.player | serial | samplingprofiler |  |
| wifip2p | simphonebook | wifi | window | sensorservice | location | scheduling\_policy |  |

黄色表示：无dump打印信息的服务，大家可以到相应服务（.\framework\native\services\）的源码下，查看dump接口，即可明白为什么黄色背景的服务没有dump打印信息。

HiDisplay ，Hisi添加的服务，用于查看支持的分辨率。参数字段为1表示支持。

红色表示：在以后调试中常用到的相关服务。

## dumpsys dropbox

Usage: dumpsys dropbox [--print|--file] [YYYY-mm-dd] [HH:MM:SS] [tag]

DropBoxManager是Android引入的用来持续化存储系统数据的机制，主要用于记录Android运行过程中，内核、系统进程、用户进程、等出现严重问题时的log.

Dropbox存储的log，位于/data/system/dropbox目录下，它可以记录的系统错误有：

* crash(应用程序强制关闭， Force Cloce)
* anr (应用程序没响应, Application Not Responding, ANR)
* wtf (What a Terrible Failure)
* strict\_mode (StrictMode Violation): 通常用来监测不应当在主线程执行的网络, 文件等操作
* lowmem (低内存): 在内存不足的时候, Android 会终止后台应用程序来释放内存, 但如果没有后台应用程序可被释放时, ActivityManagerService 就会在 DropBoxManager 中记录一次。
* watchdog：如果 WatchDog 监测到系统进程(system\_server)出现问题, 会增加一条 watchdog 记录到 DropBoxManager 中, 并终止系统进程的执行。
* netstats\_error：NetworkStatsService 负责收集并持久化存储网络状态的统计数据, 当遇到明显的网络状态错误时, 它会增加一条 netstats\_error 记录到 DropBoxManager。

## dumpsys HiDisplay

Hisi添加的服务，用于查看支持的分辨率。参数字段为1表示支持。

--- Format ---Brightness--- Contrast ---Saturation--- Hue --- DispMode ---EyePriority---

--- 6--- 50--- 50--- 50--- 50--- 0--- 0---

Screen Out Range : [0,0,0,0]

| MacroVersion : -1 | HDCP 0 |

| Is support 3d = 0 |

| HI\_UNF\_ENC\_FMT\_1080P\_60 = 1 |

| HI\_UNF\_ENC\_FMT\_1080P\_50 = 1 |

| HI\_UNF\_ENC\_FMT\_1080P\_30 = 0 |

## dumpsys activity

列出目标设备上的activity栈(back stack)和任务(task)的信息, 还有其他组件的一些信息和一些关于进程的信息。

获取所用应用: dumpsys activity

获取自己的应用: dumpsys activity | grep com.xxx.xxx.xx

获取处于栈顶的activity: dumpsys activity | grep mFocusedActivity

### dumpsys activity –h

通过帮助可以了解到dumpsys activity的全部用法。

### dumpsys activity a

Main stack:表示当前运行activity的应用。

Running activities：表示正在运行的activity。

Recent tasks:表示最近运行的activiy顺序。

Main stack:

\* TaskRecord{40f26e18 #3 A com.pplive.androidpad U 0}

numActivities=1 rootWasReset=true userId=0

affinity=com.pplive.androidpad

…………………………………………….

…………………………………………….

Running activities (most recent first):

TaskRecord{40f26e18 #3 A com.pplive.androidpad U 0}

Run #1: ActivityRecord{4123fb58 u0 com.pplive.androidpad/.ui.RecommendActivity}

TaskRecord{40f49860 #2 A com.cyanogenmod.trebuchet U 0}

Run #0: ActivityRecord{40f47930 u0 com.cyanogenmod.trebuchet/.Launcher}

Recent tasks:

\* Recent #0: TaskRecord{40f26e18 #3 A com.pplive.androidpad U 0}

numActivities=1 rootWasReset=true userId=0

\* Recent #1: TaskRecord{40f648f0 #4 I com.android.deskclock/.AlarmAlertFullScreen U 0}

intent={flg=0x800000 cmp=com.android.deskclock/.AlarmAlertFullScreen}

--------------------------------------------------------------------------------------------------------------------------------

### **场景1：查询某个App所有的Service状态**

dumpsys activity s com.sina.weibo



解读：

* Service类名为com.morgoo.droidplugin.PluginManagerService；
* 运行在进程pid=7220，进程名为com.sina.weibo，uid=10094；
* 通过bindService连接该服务的进程pid=7306，进程名为com.sina.weibo:PluginP03。

### **场景2：查询某个App所有的广播状态**

dumpsys activity s com.sina.weibo



* android.intent.action.SCREEN\_ON代表手机亮屏广播；
* 接收该广播的receiver有很多个，其中一个所在进程为pid=7220，进程名为com.sina.weibo

### **场景3：查询某个App所有的Activity状态**

* 输出结果较多，尤其是View Hierarchy，下面截取部分：

dumpsys activity a com.sina.weibo



解读：

* 格式：TaskRecord{Hashcode #TaskId Affinity UserId=0 Activity个数=1}；所以上图信息解析后就是TaskId=1802，Affinity=com.sina.weibo，当前Task中Activity个数为1。
* effectiveUid为当前task所属Uid，mCallingUid为调用者Uid=u0a94，mCallingPackage为调用者包名，这里是com.sina.weibo；
* realActivity:task中的已启动的Activity组件名com.sina.weibo/.SplashActivity。

### **场景4：查询某个App的进程状态**

dumpsys activity p com.sina.weibo



* 格式：ProcessRecord{Hashcode pid:进程名/uid}，进程pid=7306，进程名为com.sina.weibo:PluginP03，uid=10094.
* 该进程中还有Services，Connections, Providers, Receivers，可以看出该进程是没有Activity的进程。

### **其他**

* 还有很多场景，会用到不同的参数，这里就不再一一列举，建议大家多去尝试，慢慢地就更加熟练，再比如：
* dumpsys activity top //当前界面app状态
* dumpsys activity oom //查看进程状态

### 栈顶详细消息

dumpsys activity top查看栈顶Activity,可以用来获取包名

可以看到Activity的名字和状态以及这个Activity的视图结构，很厉害的样子



## dumpsys appwidget

User: 0

Providers:

[0] provider com.andrew.apollo/.appwidgets.AppWidgetSmall:

min=(64001x10241) minResize=(64001x10241) updatePeriodMillis=0 resizeMode=03 autoAdvanceViewId=-1 initialLayout=#7f030009 uid=10000 zombie=false

[1] provider com.andrew.apollo/.appwidgets.AppWidgetLarge:

min=(64001x32769) minResize=(64001x32769) updatePeriodMillis=0 resizeMode=03 autoAdvanceViewId=-1 initialLayout=#7f030005 uid=10000 zombie=false

[2] provider com.andrew.apollo/.appwidgets.AppWidgetLargeAlternate:

min=(64001x32769) minResize=(64001x32769) updatePeriodMillis=0 resizeMode=03 autoAdvanceViewId=-1 initialLayout=#7f030006 uid=10000 zombie=false

[3] provider com.andrew.apollo/.appwidgets.RecentWidgetProvider:

min=(64001x46081) minResize=(64001x28161) updatePeriodMillis=0 resizeMode=33 autoAdvanceViewId=-1 initialLayout=#7f030007 uid=10000 zombie=false

[4] provider com.android.browser/.widget.BookmarkThumbnailWidgetProvider:

min=(46081x46081) minResize=(28161x10241) updatePeriodMillis=0 resizeMode=31 autoAdvanceViewId=-1 initialLayout=#7f04000e uid=10004 zombie=false

[5] provider com.android.calculator2/.CalculatorWidget:

min=(64001x64001) minResize=(64001x64001) updatePeriodMillis=0 resizeMode=23 autoAdvanceViewId=-1 initialLayout=#7f040010 uid=10009 zombie=false

[6] provider com.android.calendar/.widget.CalendarAppWidgetProvider:

min=(28161x46081) minResize=(28161x28161) updatePeriodMillis=0 resizeMode=33 autoAdvanceViewId=-1 initialLayout=#7f04000d uid=10010 zombie=false

[7] provider com.android.contacts/.socialwidget.SocialWidgetProvider:

min=(66561x10241) minResize=(66561x10241) updatePeriodMillis=86400000 resizeMode=01 autoAdvanceViewId=-1 initialLayout=#7f04008e uid=10001 zombie=false

[8] provider com.android.deskclock/com.android.alarmclock.AnalogAppWidgetProvider:

min=(28161x28161) minResize=(28161x28161) updatePeriodMillis=0 resizeMode=01 autoAdvanceViewId=-1 initialLayout=#7f040006 uid=10016 zombie=false

[9] provider com.android.deskclock/com.android.alarmclock.DigitalAppWidgetProvider:

min=(115201x51201) minResize=(94721x30721) updatePeriodMillis=0 resizeMode=33 autoAdvanceViewId=-1 initialLayout=#7f040015 uid=10016 zombie=false

[10] provider com.android.email/.provider.WidgetProvider:

min=(46081x28161) minResize=(46081x28161) updatePeriodMillis=0 resizeMode=33 autoAdvanceViewId=-1 initialLayout=#7f040045 uid=10020 zombie=false

[11] provider com.android.gallery3d/.gadget.PhotoAppWidgetProvider:

min=(61441x61441) minResize=(61441x61441) updatePeriodMillis=86400000 resizeMode=01 autoAdvanceViewId=-1 initialLayout=#7f040004 uid=10024 zombie=false

[12] provider com.cyanogenmod.lockclock/.ClockWidgetProvider:

min=(76801x30721) minResize=(51201x30721) updatePeriodMillis=0 resizeMode=33 autoAdvanceViewId=-1 initialLayout=#7f030000 uid=10034 zombie=false

[13] provider com.android.quicksearchbox/.SearchWidgetProvider:

min=(66561x10241) minResize=(66561x10241) updatePeriodMillis=0 resizeMode=01 autoAdvanceViewId=-1 initialLayout=#7f040006 uid=10042 zombie=false

[14] provider com.android.settings/.widget.SettingsAppWidgetProvider:

min=(83201x12801) minResize=(83201x12801) updatePeriodMillis=0 resizeMode=01 autoAdvanceViewId=-1 initialLayout=#7f0400b7 uid=1000 zombie=false

[15] provider net.cactii.flash2/.TorchWidgetProvider:

min=(8961x10241) minResize=(8961x10241) updatePeriodMillis=3600000 resizeMode=03 autoAdvanceViewId=-1 initialLayout=#7f030004 uid=10048 zombie=false

AppWidgetIds:

[0] id=1

hostId=1024 com.cyanogenmod.trebuchet/10049

provider=com.andrew.apollo/.appwidgets.AppWidgetLarge

host.callbacks=com.android.internal.appwidget.IAppWidgetHost$Stub$Proxy@425a4070

views=android.widget.RemoteViews@425c53e8

[1] id=2

hostId=1024 com.cyanogenmod.trebuchet/10049

provider=com.cyanogenmod.lockclock/.ClockWidgetProvider

host.callbacks=com.android.internal.appwidget.IAppWidgetHost$Stub$Proxy@425a4070

views=android.widget.RemoteViews@42959f58

Hosts:

[0] hostId=1024 com.cyanogenmod.trebuchet/10049:

callbacks=com.android.internal.appwidget.IAppWidgetHost$Stub$Proxy@425a4070

instances.size=2 zombie=false

Deleted Providers:

Deleted Hosts:

## dumpsys connectivity

Connectivity Service提供有关网络当前状态信息。

dumpsys connectivity用于查看wifi和以太网网络连接状态、IP地址等信息。

NetworkStateTracker for WIFI:

NetworkInfo: type: wifi[], state: UNKNOWN/IDLE, reason: (unspecified), extra: (none), roaming: false, failover: false, isAvailable: false

{LinkAddresses: [] Routes: [] DnsAddresses: [] Domains: null}

android.net.wifi.WifiStateTracker@4276f068

NetworkStateTracker for BLUETOOTH:

NetworkInfo: type: BLUETOOTH\_TETHER[], state: UNKNOWN/IDLE, reason: (unspecified), extra: (none), roaming: false, failover: false, isAvailable: false

{LinkAddresses: [] Routes: [] DnsAddresses: [] Domains: null}

android.bluetooth.BluetoothTetheringDataTracker@4253c1b8

NetworkStateTracker for ETHERNET:

Active network: ethernet

NetworkInfo: type: ethernet[], state: CONNECTED/CONNECTED, reason: (unspecified), extra: (none), roaming: false, failover: false, isAvailable: true

{InterfaceName: eth0 LinkAddresses: [10.9.8.121/24,] Routes: [10.9.8.0/24 -> 0.0.0.0,0.0.0.0/0 -> 10.9.8.254,] DnsAddresses: [10.9.8.254,] Domains: }

android.net.ethernet.EthernetStateTracker@4243e6b0

NetworkStateTracker for WIFI\_P2P:

NetworkInfo: type: wifi\_p2p[], state: UNKNOWN/IDLE, reason: (unspecified), extra: (none), roaming: false, failover: false, isAvailable: false

{LinkAddresses: [] Routes: [] DnsAddresses: [] Domains: null}

android.net.wifi.WifiStateTracker@4253b7e0

Network Requester Pids:

9:

1:

13:

7:

FeatureUsers:

NetworkTranstionWakeLock is currently not held.

It was last requested for

mUpstreamIfaceTypes:

Tether state:

Inet condition reports:

## dumpsys content

data connected: true

auto sync: u0=true

memory low: false

accounts: 0

now: 229020101 (2014-03-02 14:54:40)

offset: 23:44:56 (HH:MM:SS)

uptime: 63:37:00 (HH:MM:SS)

time spent syncing: 0:00 (HH:MM:SS), sync not in progress

next alarm time: 230420022 (23:19 (HH:MM:SS) from now)

notification info: isActive false, startTime null

Active Syncs: 0

SyncQueue: 0 operation(s)

Sync Status

Sync adapters for UserInfo{0:Owner:13}:

ServiceInfo: SyncAdapterType {name=com.android.email.provider, type=com.android.exchange, userVisible=true, supportsUploading=false, isAlwaysSyncable=false, allowParallelSyncs=false, settingsActivity=null}, ComponentInfo{com.android.exchange/com.android.exchange.EmailSyncAdapterService}, uid 10021

ServiceInfo: SyncAdapterType {name=com.android.email.provider, type=com.android.email, userVisible=true, supportsUploading=false, isAlwaysSyncable=false, allowParallelSyncs=false, settingsActivity=null}, ComponentInfo{com.android.email/com.android.email.service.PopImapSyncAdapterService}, uid 10020

ServiceInfo: SyncAdapterType {name=com.android.calendar, type=com.android.exchange, userVisible=true, supportsUploading=true, isAlwaysSyncable=false, allowParallelSyncs=false, settingsActivity=null}, ComponentInfo{com.android.exchange/com.android.exchange.CalendarSyncAdapterService}, uid 10021

ServiceInfo: SyncAdapterType {name=com.android.contacts, type=com.android.exchange, userVisible=true, supportsUploading=true, isAlwaysSyncable=false, allowParallelSyncs=false, settingsActivity=null}, ComponentInfo{com.android.exchange/com.android.exchange.ContactsSyncAdapterService}, uid 10021

Observer tree:

settings/secure/long\_press\_timeout: pid=1974 uid=1000 user=0 target=42555fd8

settings/secure/incall\_power\_button\_behavior: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/secure/ring\_home\_button\_behavior: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/secure/default\_input\_method: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/secure/default\_input\_method: pid=1974 uid=1000 user=0 target=42711a88

settings/secure/enabled\_input\_methods: pid=1974 uid=1000 user=0 target=42711a88

settings/secure/selected\_input\_method\_subtype: pid=1974 uid=1000 user=0 target=42711a88

settings/secure/accessibility\_enabled: pid=1974 uid=1000 user=-1 target=42578ca0

settings/secure/touch\_exploration\_enabled: pid=1974 uid=1000 user=-1 target=42578ca0

settings/secure/accessibility\_display\_magnification\_enabled: pid=1974 uid=1000 user=-1 target=42578ca0

settings/secure/enabled\_accessibility\_services: pid=1974 uid=1000 user=-1 target=42578ca0

settings/secure/touch\_exploration\_granted\_accessibility\_services: pid=1974 uid=1000 user=-1 target=42578ca0

settings/secure/pubkey\_blacklist: pid=1974 uid=1000 user=0 target=426adff0

settings/secure/serial\_blacklist: pid=1974 uid=1000 user=0 target=426a5c30

settings/secure/adb\_port: pid=1974 uid=1000 user=0 target=425fb080

settings/secure/screensaver\_enabled: pid=1974 uid=1000 user=-1 target=426c20d8

settings/secure/screensaver\_activate\_on\_sleep: pid=1974 uid=1000 user=-1 target=426c20d8

settings/secure/screensaver\_activate\_on\_dock: pid=1974 uid=1000 user=-1 target=426c20d8

settings/secure/locationPackagePrefixBlacklist: pid=1974 uid=1000 user=-1 target=425e9f88

settings/secure/locationCoarseAccuracy: pid=1974 uid=1000 user=0 target=424c1e80

settings/secure/location\_providers\_allowed: pid=1974 uid=1000 user=-1 target=424b9f40

settings/secure/location\_providers\_allowed: pid=2092 uid=10046 user=0 target=42667c30

settings/secure/user\_setup\_complete: pid=2092 uid=10046 user=0 target=426d3800

settings/secure/kill\_app\_longpress\_back: pid=2092 uid=10046 user=-1 target=42592100

settings/system: pid=2300 uid=10049 user=0 target=42676dd0

settings/system: pid=2300 uid=10049 user=0 target=429f7770

settings/system: pid=2300 uid=10049 user=0 target=42a288c8

settings/system: pid=2300 uid=10049 user=0 target=425d61b0

settings/system: pid=2300 uid=10049 user=0 target=42a18cc0

settings/system: pid=2300 uid=10049 user=0 target=42461128

settings/system/battery\_light\_enabled: pid=1974 uid=1000 user=0 target=42371158

settings/system/battery\_light\_pulse: pid=1974 uid=1000 user=0 target=42371158

settings/system/battery\_light\_low\_color: pid=1974 uid=1000 user=0 target=42371158

settings/system/battery\_light\_medium\_color: pid=1974 uid=1000 user=0 target=42371158

settings/system/battery\_light\_full\_color: pid=1974 uid=1000 user=0 target=42371158

settings/system/quiet\_hours\_enabled: pid=1974 uid=1000 user=0 target=42371158

settings/system/quiet\_hours\_enabled: pid=1974 uid=1000 user=0 target=4270f348

settings/system/quiet\_hours\_start: pid=1974 uid=1000 user=0 target=42371158

settings/system/quiet\_hours\_start: pid=1974 uid=1000 user=0 target=4270f348

settings/system/quiet\_hours\_end: pid=1974 uid=1000 user=0 target=42371158

settings/system/quiet\_hours\_end: pid=1974 uid=1000 user=0 target=4270f348

settings/system/quiet\_hours\_dim: pid=1974 uid=1000 user=0 target=42371158

settings/system/quiet\_hours\_dim: pid=1974 uid=1000 user=0 target=4270f348

settings/system/end\_button\_behavior: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/home\_wake\_screen: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/volume\_wake\_screen: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/volbtn\_music\_controls: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/accelerometer\_rotation: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/user\_rotation: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/screen\_off\_timeout: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/screen\_off\_timeout: pid=1974 uid=1000 user=-1 target=426c20d8

settings/system/pointer\_location: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/fancy\_rotation\_anim: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/expanded\_desktop\_state: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/expanded\_desktop\_state: pid=2092 uid=10046 user=-1 target=42592100

settings/system/expanded\_desktop\_style: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/expanded\_desktop\_style: pid=2092 uid=10046 user=-1 target=42592100

settings/system/accelerometer\_rotation\_angles: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/key\_home\_long\_press\_action: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/key\_menu\_action: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/key\_menu\_long\_press\_action: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/key\_assist\_action: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/key\_assist\_long\_press\_action: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/key\_app\_switch\_action: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/key\_app\_switch\_long\_press\_action: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/hardware\_key\_rebinding: pid=1974 uid=1000 user=-1 target=423e4cc0

settings/system/pointer\_speed: pid=1974 uid=1000 user=-1 target=423fed08

settings/system/show\_touches: pid=1974 uid=1000 user=-1 target=4273b4a0

settings/system/stylus\_icon\_enabled: pid=1974 uid=1000 user=0 target=424d7780

settings/system/swap\_volume\_keys\_on\_rotation: pid=1974 uid=1000 user=0 target=4263dc28

settings/system/status\_bar\_ime\_switcher: pid=1974 uid=1000 user=0 target=42642658

settings/system/notification\_light\_pulse: pid=1974 uid=1000 user=0 target=42728698

settings/system/notification\_light\_pulse\_default\_color: pid=1974 uid=1000 user=0 target=42728698

settings/system/notification\_light\_pulse\_default\_led\_on: pid=1974 uid=1000 user=0 target=42728698

settings/system/notification\_light\_pulse\_default\_led\_off: pid=1974 uid=1000 user=0 target=42728698

settings/system/notification\_light\_pulse\_custom\_enable: pid=1974 uid=1000 user=0 target=42728698

settings/system/notification\_light\_pulse\_custom\_values: pid=1974 uid=1000 user=0 target=42728698

settings/system/quiet\_hours\_mute: pid=1974 uid=1000 user=0 target=4270f348

settings/system/quiet\_hours\_still: pid=1974 uid=1000 user=0 target=4270f348

settings/system/mode\_ringer\_streams\_affected: pid=1974 uid=1000 user=0 target=42496408

settings/system/volume\_link\_notification: pid=1974 uid=1000 user=0 target=42496408

settings/system/safe\_headset\_volume: pid=1974 uid=1000 user=0 target=42496408

settings/system/volume\_keys\_control\_ring\_stream: pid=1974 uid=1000 user=0 target=42496408

settings/system/vibrate\_input\_devices: pid=1974 uid=1000 user=-1 target=425bde08

settings/system/auto\_brightness\_lux: pid=1974 uid=1000 user=-1 target=42593288

settings/system/auto\_brightness\_backlight: pid=1974 uid=1000 user=-1 target=42593288

settings/system/screen\_brightness: pid=1974 uid=1000 user=-1 target=426c20d8

settings/system/screen\_brightness: pid=2092 uid=10046 user=0 target=424942e0

settings/system/screen\_brightness\_mode: pid=1974 uid=1000 user=-1 target=426c20d8

settings/system/screen\_brightness\_mode: pid=2092 uid=10046 user=0 target=424942e0

settings/system/screen\_brightness\_mode: pid=2092 uid=10046 user=0 target=4269a730

settings/system/auto\_brightness\_responsiveness: pid=1974 uid=1000 user=-1 target=426c20d8

settings/system/status\_bar\_collapse\_on\_dismiss: pid=2092 uid=10046 user=0 target=4274a998

settings/system/status\_bar\_battery: pid=2092 uid=10046 user=0 target=42453af8

settings/system/status\_bar\_battery: pid=2092 uid=10046 user=0 target=4276e268

settings/system/status\_bar\_battery: pid=2092 uid=10046 user=0 target=42494958

settings/system/bugreport\_in\_power\_menu: pid=2092 uid=10046 user=0 target=424942e0

settings/system/next\_alarm\_formatted: pid=2092 uid=10046 user=0 target=424942e0

settings/system/quick\_settings\_tiles: pid=2092 uid=10046 user=-1 target=42454728

settings/system/qs\_dyanmic\_alarm: pid=2092 uid=10046 user=-1 target=42454728

settings/system/qs\_dyanmic\_bugreport: pid=2092 uid=10046 user=-1 target=42454728

settings/system/qs\_dyanmic\_dock\_battery: pid=2092 uid=10046 user=-1 target=42454728

settings/system/qs\_dyanmic\_ime: pid=2092 uid=10046 user=-1 target=42454728

settings/system/qs\_dyanmic\_usbtether: pid=2092 uid=10046 user=-1 target=42454728

settings/system/qs\_dyanmic\_wifi: pid=2092 uid=10046 user=-1 target=42454728

settings/system/expanded\_view\_widget: pid=2092 uid=10046 user=0 target=42667c30

settings/system/expanded\_hide\_scrollbar: pid=2092 uid=10046 user=0 target=42667c30

settings/system/expanded\_haptic\_feedback: pid=2092 uid=10046 user=0 target=42667c30

settings/system/haptic\_feedback\_enabled: pid=2092 uid=10046 user=0 target=42667c30

settings/system/expanded\_widget\_buttons: pid=2092 uid=10046 user=0 target=42667c30

settings/system/expanded\_ring\_mode: pid=2092 uid=10046 user=0 target=42667c30

settings/system/vibrate\_when\_ringing: pid=2092 uid=10046 user=0 target=42667c30

settings/system/nav\_buttons: pid=2092 uid=10046 user=-1 target=42592100

settings/system/pie\_controls: pid=2092 uid=10046 user=-1 target=42592100

settings/system/pie\_positions: pid=2092 uid=10046 user=-1 target=42592100

settings/system/pie\_sensitivity: pid=2092 uid=10046 user=-1 target=42592100

settings/system/status\_bar\_brightness\_control: pid=2092 uid=10046 user=0 target=4269a730

settings/system/status\_bar\_signal: pid=2092 uid=10046 user=0 target=42663730

settings/system/status\_bar\_signal: pid=2092 uid=10046 user=-1 target=42632938

settings/system/status\_bar\_am\_pm: pid=2092 uid=10046 user=0 target=4255f290

settings/system/status\_bar\_am\_pm: pid=2092 uid=10046 user=0 target=423da5b8

settings/system/status\_bar\_clock: pid=2092 uid=10046 user=0 target=4255f290

settings/system/status\_bar\_clock: pid=2092 uid=10046 user=0 target=423da5b8

settings/system/status\_bar\_notif\_count: pid=2092 uid=10046 user=0 target=423720a0

settings/system/enable\_stylus\_gestures: pid=2300 uid=10049 user=0 target=42727690

settings/bookmarks: pid=1974 uid=1000 user=0 target=426313c0

settings/global: pid=1974 uid=1000 user=0 target=425f5d78

settings/global/wifi\_suspend\_optimizations\_enabled: pid=1974 uid=1000 user=0 target=42717528

settings/global/wifi\_networks\_available\_notification\_on: pid=1974 uid=1000 user=0 target=426362d0

settings/global/http\_proxy: pid=1974 uid=1000 user=0 target=424739f8

settings/global/wifi\_watchdog\_poor\_network\_test\_enabled: pid=1974 uid=1000 user=0 target=4248f960

settings/global/wifi\_watchdog\_on: pid=1974 uid=1000 user=0 target=4248fd00

settings/global/nsd\_on: pid=1974 uid=1000 user=0 target=42496cb0

settings/global/dock\_audio\_media\_enabled: pid=1974 uid=1000 user=0 target=42496408

settings/global/device\_provisioned: pid=1974 uid=1000 user=0 target=426694b0

settings/global/device\_provisioned: pid=2092 uid=10046 user=0 target=42749818

settings/global/sampling\_profiler\_ms: pid=1974 uid=1000 user=0 target=4254ad48

settings/global/stay\_on\_while\_plugged\_in: pid=1974 uid=1000 user=-1 target=426c20d8

settings/global/overlay\_display\_devices: pid=1974 uid=1000 user=0 target=425b3d48

settings/global/throttle\_polling\_sec: pid=1974 uid=1000 user=0 target=4245b490

settings/global/throttle\_threshold\_bytes: pid=1974 uid=1000 user=0 target=4245b490

settings/global/throttle\_value\_kbitsps: pid=1974 uid=1000 user=0 target=4245b490

settings/global/throttle\_reset\_day: pid=1974 uid=1000 user=0 target=4245b490

settings/global/throttle\_notification\_type: pid=1974 uid=1000 user=0 target=4245b490

settings/global/throttle\_help\_uri: pid=1974 uid=1000 user=0 target=4245b490

settings/global/throttle\_max\_ntp\_cache\_age\_sec: pid=1974 uid=1000 user=0 target=4245b490

settings/global/auto\_time: pid=1974 uid=1000 user=0 target=425678c0

settings/global/auto\_time: pid=2240 uid=1001 user=0 target=425259d0

settings/global/sms\_short\_code\_confirmation: pid=2240 uid=1001 user=0 target=4277d788

settings/global/sms\_outgoing\_check\_max\_count: pid=2240 uid=1001 user=0 target=4277d788

settings/global/auto\_time\_zone: pid=2240 uid=1001 user=0 target=42518120

settings/global/sms\_short\_code\_rule: pid=2240 uid=1001 user=0 target=4276f348

settings/global/data\_roaming: pid=2240 uid=1001 user=0 target=4264a4d0

com.cyanogenmod.trebuchet.settings/favorites: pid=2300 uid=10049 user=0 target=4277d240

com.cyanogenmod.trebuchet.settings/appWidgetReset: pid=2300 uid=10049 user=0 target=4266beb0

user\_dictionary/words: pid=2207 uid=10031 user=0 target=42780b70

telephony/carriers: pid=2240 uid=1001 user=0 target=426d7b40

com.android.contacts/contacts: pid=2207 uid=10031 user=0 target=42735ab8

pid 1974: 99 observers

pid 2092: 41 observers

pid 2300: 9 observers

pid 2240: 7 observers

pid 2207: 2 observers

Total number of nodes: 140

Total number of observers: 158

## dumpsys cpuinfo

查看每个进程占用用户空间和内核空间的情况。

Load: 5.08 / 5.07 / 5.07

CPU usage from 105554ms to 45554ms ago:

12% 2893/com.pplive.androidpad: 11% user + 1.3% kernel / faults: 21594 minor

2.2% 3097/com.android.deskclock: 1.4% user + 0.8% kernel / faults: 12 minor

1.8% 1590/surfaceflinger: 1.1% user + 0.6% kernel

0.4% 1596/displaysetting: 0% user + 0.4% kernel

0.3% 18/HI\_VPSS\_Process: 0% user + 0.3% kernel

0.2% 16/log\_udisk\_task: 0% user + 0.2% kernel

0% 2233/system\_server: 0% user + 0% kernel

0% 20/HI\_HDMI\_kThread: 0% user + 0% kernel

0% 1598/android\_ir\_user: 0% user + 0% kernel

0% 2347/com.android.systemui: 0% user + 0% kernel

0% 4/kworker/0:0: 0% user + 0% kernel

0% 15/kworker/1:1: 0% user + 0% kernel

0% 17/x5hdqam\_sop: 0% user + 0% kernel

0% 21/HI\_HDMI\_kCEC: 0% user + 0% kernel

0% 448/cfinteractive: 0% user + 0% kernel

0% 2538/com.cyanogenmod.trebuchet: 0% user + 0% kernel / faults: 2 minor

0% 3133/com.pplive.androidpad:remote: 0% user + 0% kernel / faults: 1 minor

9.1% TOTAL: 7.1% user + 1.9% kernel + 0% softirq

## dumpsys dbinfo

Applications Database Info:

\*\* Database info for pid 2318 [com.android.location.fused] \*\*

\*\* Database info for pid 2092 [com.android.systemui] \*\*

\*\* Database info for pid 2207 [com.android.inputmethod.latin] \*\*

\*\* Database info for pid 1974 [system] \*\*

Connection pool for /data/system/users/0/accounts.db:

Open: true

Max connections: 1

Available primary connection:

Connection #0:

isPrimaryConnection: true

onlyAllowReadOnlyOperations: true

Most recently executed operations:

0: [2014-02-28 10:18:25.208] executeForCursorWindow took 2ms - succeeded, sql="SELECT uid FROM grants GROUP BY uid"

1: [2014-02-28 10:18:25.207] prepare took 0ms - succeeded, sql="SELECT uid FROM grants GROUP BY uid"

2: [1970-01-02 00:00:06.060] executeForCursorWindow took 0ms - succeeded, sql="SELECT \_id, type, name FROM accounts"

3: [1970-01-02 00:00:06.055] prepare took 1ms - succeeded, sql="SELECT \_id, type, name FROM accounts"

4: [1970-01-02 00:00:06.054] executeForCursorWindow took 0ms - succeeded, sql="SELECT \_id, type, name FROM accounts"

5: [1970-01-02 00:00:06.053] prepare took 0ms - succeeded, sql="SELECT \_id, type, name FROM accounts"

6: [1970-01-02 00:00:06.030] executeForCursorWindow took 16ms - succeeded, sql="SELECT uid FROM grants GROUP BY uid"

7: [1970-01-02 00:00:06.029] prepare took 0ms - succeeded, sql="SELECT uid FROM grants GROUP BY uid"

8: [1970-01-02 00:00:06.029] executeForLong took 0ms - succeeded, sql="PRAGMA user\_version;"

9: [1970-01-02 00:00:06.029] prepare took 0ms - succeeded, sql="PRAGMA user\_version;"

10: [1970-01-02 00:00:06.007] executeForString took 21ms - succeeded, sql="SELECT locale FROM android\_metadata UNION SELECT NULL ORDER BY locale DESC LIMIT 1"

11: [1970-01-02 00:00:06.002] execute took 5ms - succeeded, sql="CREATE TABLE IF NOT EXISTS android\_metadata (locale TEXT)"

12: [1970-01-02 00:00:05.996] executeForLong took 5ms - succeeded, sql="PRAGMA wal\_autocheckpoint=100"

13: [1970-01-02 00:00:05.994] executeForLong took 2ms - succeeded, sql="PRAGMA wal\_autocheckpoint"

14: [1970-01-02 00:00:05.992] executeForLong took 2ms - succeeded, sql="PRAGMA journal\_size\_limit=524288"

15: [1970-01-02 00:00:05.989] executeForLong took 2ms - succeeded, sql="PRAGMA journal\_size\_limit"

16: [1970-01-02 00:00:05.986] executeForString took 3ms - succeeded, sql="PRAGMA synchronous"

17: [1970-01-02 00:00:05.979] executeForString took 7ms - succeeded, sql="PRAGMA journal\_mode=PERSIST"

18: [1970-01-02 00:00:05.958] executeForString took 21ms - succeeded, sql="PRAGMA journal\_mode"

19: [1970-01-02 00:00:05.956] executeForLong took 2ms - succeeded, sql="PRAGMA foreign\_keys"

Available non-primary connections:

<none>

Acquired connections:

<none>

Connection waiters:

<none>

Connection pool for /data/system/locksettings.db:

Open: true

Max connections: 4

Available primary connection:

Connection #0:

isPrimaryConnection: true

onlyAllowReadOnlyOperations: false

Most recently executed operations:

0: [1970-01-02 00:00:04.209] prepare took 0ms - succeeded, sql="PRAGMA user\_version;"

1: [1970-01-02 00:00:04.208] executeForString took 0ms - succeeded, sql="SELECT locale FROM android\_metadata UNION SELECT NULL ORDER BY locale DESC LIMIT 1"

2: [1970-01-02 00:00:04.208] execute took 0ms - succeeded, sql="CREATE TABLE IF NOT EXISTS android\_metadata (locale TEXT)"

3: [1970-01-02 00:00:04.208] executeForLong took 0ms - succeeded, sql="PRAGMA wal\_autocheckpoint=100"

4: [1970-01-02 00:00:04.207] executeForLong took 1ms - succeeded, sql="PRAGMA wal\_autocheckpoint"

5: [1970-01-02 00:00:04.207] executeForLong took 0ms - succeeded, sql="PRAGMA journal\_size\_limit=524288"

6: [1970-01-02 00:00:04.207] executeForLong took 0ms - succeeded, sql="PRAGMA journal\_size\_limit"

7: [1970-01-02 00:00:04.207] executeForString took 0ms - succeeded, sql="PRAGMA synchronous"

8: [1970-01-02 00:00:04.201] executeForString took 6ms - succeeded, sql="PRAGMA journal\_mode"

9: [1970-01-02 00:00:04.201] executeForLong took 0ms - succeeded, sql="PRAGMA foreign\_keys"

10: [1970-01-02 00:00:04.201] executeForLong took 0ms - succeeded, sql="PRAGMA page\_size"

Available non-primary connections:

Connection #1:

isPrimaryConnection: false

onlyAllowReadOnlyOperations: true

Most recently executed operations:

0: [1970-01-02 00:00:05.843] executeForCursorWindow took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?", bindArgs=["0", "lock\_pattern\_autolock"]

1: [1970-01-02 00:00:05.838] prepare took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?"

2: [1970-01-02 00:00:05.838] executeForCursorWindow took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?", bindArgs=["0", "lockscreen.password\_type\_alternate"]

3: [1970-01-02 00:00:05.837] prepare took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?"

4: [1970-01-02 00:00:05.837] executeForCursorWindow took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?", bindArgs=["0", "lockscreen.password\_type"]

5: [1970-01-02 00:00:05.836] prepare took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?"

6: [1970-01-02 00:00:04.230] executeForCursorWindow took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?", bindArgs=["0", "lockscreen.password\_type"]

7: [1970-01-02 00:00:04.229] prepare took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?"

8: [1970-01-02 00:00:04.228] executeForCursorWindow took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?", bindArgs=["0", "lock\_pattern\_autolock"]

9: [1970-01-02 00:00:04.227] prepare took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?"

10: [1970-01-02 00:00:04.227] executeForCursorWindow took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?", bindArgs=["0", "lockscreen.password\_type\_alternate"]

11: [1970-01-02 00:00:04.227] prepare took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?"

12: [1970-01-02 00:00:04.226] executeForCursorWindow took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?", bindArgs=["0", "lockscreen.password\_type"]

13: [1970-01-02 00:00:04.226] prepare took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?"

14: [1970-01-02 00:00:04.223] executeForCursorWindow took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?", bindArgs=["0", "lock\_pattern\_autolock"]

15: [1970-01-02 00:00:04.222] prepare took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?"

16: [1970-01-02 00:00:04.222] executeForCursorWindow took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?", bindArgs=["0", "lockscreen.password\_type\_alternate"]

17: [1970-01-02 00:00:04.221] prepare took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?"

18: [1970-01-02 00:00:04.221] executeForCursorWindow took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?", bindArgs=["0", "lockscreen.password\_type"]

19: [1970-01-02 00:00:04.220] prepare took 0ms - succeeded, sql="SELECT value FROM locksettings WHERE user=? AND name=?"

Acquired connections:

<none>

Connection waiters:

<none>

Connection pool for /data/data/com.android.providers.settings/databases/settings.db:

Open: true

Max connections: 4

Available primary connection:

Connection #0:

isPrimaryConnection: true

onlyAllowReadOnlyOperations: false

Most recently executed operations:

0: [2014-03-03 01:03:03.255] executeForLastInsertedRowId took 161ms - succeeded, sql="INSERT INTO system(value,name) VALUES (?,?)", bindArgs=["com.andrew.apollo/com.andrew.apollo.MediaButtonIntentReceiver", "media\_button\_receiver"]

1: [2014-03-03 01:03:03.255] prepare took 0ms - succeeded, sql="INSERT INTO system(value,name) VALUES (?,?)"

2: [2014-03-01 07:21:20.182] executeForLastInsertedRowId took 11ms - succeeded, sql="INSERT INTO system(value,name) VALUES (?,?)", bindArgs=["1", "com.baidu.pushservice.PushSettings.connect\_state"]

3: [2014-03-01 07:21:20.182] prepare took 0ms - succeeded, sql="INSERT INTO system(value,name) VALUES (?,?)"

4: [2014-03-01 07:21:20.019] executeForLastInsertedRowId took 100ms - succeeded, sql="INSERT INTO system(value,name) VALUES (?,?)", bindArgs=["0", "com.baidu.pushservice.PushSettings.connect\_state"]

5: [2014-03-01 07:21:20.019] prepare took 0ms - succeeded, sql="INSERT INTO system(value,name) VALUES (?,?)"

6: [2014-02-28 10:18:01.156] executeForLastInsertedRowId took 12ms - succeeded, sql="INSERT INTO system(value,name) VALUES (?,?)", bindArgs=["1", "com.baidu.pushservice.PushSettings.connect\_state"]

7: [2014-02-28 10:18:01.156] prepare took 0ms - succeeded, sql="INSERT INTO system(value,name) VALUES (?,?)"

8: [1970-01-02 00:00:11.493] executeForLastInsertedRowId took 40ms - succeeded, sql="INSERT INTO system(value,name) VALUES (?,?)", bindArgs=["0", "com.baidu.pushservice.PushSettings.connect\_state"]

9: [1970-01-02 00:00:11.493] prepare took 0ms - succeeded, sql="INSERT INTO system(value,name) VALUES (?,?)"

10: [1970-01-02 00:00:03.007] prepare took 0ms - succeeded, sql="PRAGMA user\_version;"

11: [1970-01-02 00:00:03.006] executeForString took 0ms - succeeded, sql="SELECT locale FROM android\_metadata UNION SELECT NULL ORDER BY locale DESC LIMIT 1"

12: [1970-01-02 00:00:03.006] execute took 0ms - succeeded, sql="CREATE TABLE IF NOT EXISTS android\_metadata (locale TEXT)"

13: [1970-01-02 00:00:03.000] executeForLong took 0ms - succeeded, sql="PRAGMA wal\_autocheckpoint=100"

14: [1970-01-02 00:00:03.000] executeForLong took 0ms - succeeded, sql="PRAGMA wal\_autocheckpoint"

15: [1970-01-02 00:00:03.000] executeForLong took 0ms - succeeded, sql="PRAGMA journal\_size\_limit=524288"

16: [1970-01-02 00:00:03.000] executeForLong took 0ms - succeeded, sql="PRAGMA journal\_size\_limit"

17: [1970-01-02 00:00:03.000] executeForString took 0ms - succeeded, sql="PRAGMA synchronous"

18: [1970-01-02 00:00:02.980] executeForString took 19ms - succeeded, sql="PRAGMA journal\_mode"

19: [1970-01-02 00:00:02.979] executeForLong took 1ms - succeeded, sql="PRAGMA foreign\_keys"

Available non-primary connections:

Connection #1:

isPrimaryConnection: false

onlyAllowReadOnlyOperations: true

Most recently executed operations:

0: [1970-01-02 00:00:03.104] executeForCursorWindow took 1ms - succeeded, sql="SELECT shortcut, intent FROM bookmarks"

1: [1970-01-02 00:00:03.072] executeForCursorWindow took 32ms - succeeded, sql="SELECT shortcut, intent FROM bookmarks"

2: [1970-01-02 00:00:03.070] prepare took 0ms - succeeded, sql="SELECT shortcut, intent FROM bookmarks"

3: [1970-01-02 00:00:03.018] executeForCursorWindow took 1ms - succeeded, sql="SELECT name, value FROM system LIMIT 201"

4: [1970-01-02 00:00:03.018] prepare took 0ms - succeeded, sql="SELECT name, value FROM system LIMIT 201"

5: [1970-01-02 00:00:03.016] executeForCursorWindow took 1ms - succeeded, sql="SELECT name, value FROM secure LIMIT 201"

6: [1970-01-02 00:00:03.016] prepare took 0ms - succeeded, sql="SELECT name, value FROM secure LIMIT 201"

7: [1970-01-02 00:00:03.015] executeForCursorWindow took 0ms - succeeded, sql="SELECT value FROM secure WHERE name=?", bindArgs=["long\_press\_timeout"]

8: [1970-01-02 00:00:03.014] prepare took 0ms - succeeded, sql="SELECT value FROM secure WHERE name=?"

9: [1970-01-02 00:00:03.014] executeForCursorWindow took 0ms - succeeded, sql="SELECT name, value FROM global LIMIT 201"

10: [1970-01-02 00:00:03.013] prepare took 0ms - succeeded, sql="SELECT name, value FROM global LIMIT 201"

11: [1970-01-02 00:00:03.012] executeForCursorWindow took 0ms - succeeded, sql="SELECT value FROM secure WHERE (name=?)", bindArgs=["android\_id"]

12: [1970-01-02 00:00:03.011] prepare took 0ms - succeeded, sql="SELECT value FROM secure WHERE (name=?)"

13: [1970-01-02 00:00:03.010] executeForLong took 0ms - succeeded, sql="PRAGMA user\_version;"

14: [1970-01-02 00:00:03.010] executeForString took 0ms - succeeded, sql="SELECT locale FROM android\_metadata UNION SELECT NULL ORDER BY locale DESC LIMIT 1"

15: [1970-01-02 00:00:03.009] execute took 1ms - succeeded, sql="CREATE TABLE IF NOT EXISTS android\_metadata (locale TEXT)"

16: [1970-01-02 00:00:03.009] executeForLong took 0ms - succeeded, sql="PRAGMA wal\_autocheckpoint=100"

17: [1970-01-02 00:00:03.009] executeForLong took 0ms - succeeded, sql="PRAGMA wal\_autocheckpoint"

18: [1970-01-02 00:00:03.009] executeForLong took 0ms - succeeded, sql="PRAGMA journal\_size\_limit=524288"

19: [1970-01-02 00:00:03.009] executeForLong took 0ms - succeeded, sql="PRAGMA journal\_size\_limit"

Acquired connections:

<none>

Connection waiters:

<none>

\*\* Database info for pid 2584 [com.cyanogenmod.lockclock] \*\*

\*\* Database info for pid 2300 [com.cyanogenmod.trebuchet] \*\*

Connection pool for /data/user/0/com.cyanogenmod.trebuchet/databases/launcher.db:

Open: true

Max connections: 1

Available primary connection:

Connection #0:

isPrimaryConnection: true

onlyAllowReadOnlyOperations: true

Most recently executed operations:

0: [2014-03-03 01:41:33.859] executeForCursorWindow took 1ms - succeeded, sql="SELECT \* FROM favorites"

1: [2014-03-03 01:41:33.856] prepare took 0ms - succeeded, sql="SELECT \* FROM favorites"

2: [2014-02-28 10:22:38.190] executeForLastInsertedRowId took 313ms - succeeded, sql="INSERT INTO favorites(cellX,icon,cellY,screen,iconPackage,itemType,intent,title,iconResource,\_id,spanX,container,spanY,iconType) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?)", bindArgs=[5, <byte[]>, 0, 2, "com.youku.tv", 1, "#Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x10200000;component=com.youku.tv/.WelcomeActivity;end", "优酷", "com.youku.tv:drawable/icon", 17, 1, -100, 1, 0]

3: [2014-02-28 10:22:38.189] prepare took 1ms - succeeded, sql="INSERT INTO favorites(cellX,icon,cellY,screen,iconPackage,itemType,intent,title,iconResource,\_id,spanX,container,spanY,iconType) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?)"

4: [2014-02-28 10:22:38.124] executeForCursorWindow took 3ms - succeeded, sql="SELECT intent FROM favorites WHERE (intent=?)", bindArgs=["#Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;component=com.youku.tv/.WelcomeActivity;end"]

5: [2014-02-28 10:22:38.122] prepare took 0ms - succeeded, sql="SELECT intent FROM favorites WHERE (intent=?)"

6: [2014-02-28 10:22:38.109] executeForCursorWindow took 10ms - succeeded, sql="SELECT itemType, container, screen, cellX, cellY, spanX, spanY FROM favorites"

7: [2014-02-28 10:22:38.106] prepare took 1ms - succeeded, sql="SELECT itemType, container, screen, cellX, cellY, spanX, spanY FROM favorites"

8: [1970-01-02 00:00:06.600] executeForCursorWindow took 19ms - succeeded, sql="SELECT \* FROM favorites"

9: [1970-01-02 00:00:06.598] prepare took 0ms - succeeded, sql="SELECT \* FROM favorites"

10: [1970-01-02 00:00:05.846] executeForCursorWindow took 10ms - succeeded, sql="SELECT MAX(\_id) FROM favorites"

11: [1970-01-02 00:00:05.845] prepare took 1ms - succeeded, sql="SELECT MAX(\_id) FROM favorites"

12: [1970-01-02 00:00:05.817] executeForLong took 28ms - succeeded, sql="PRAGMA user\_version;"

13: [1970-01-02 00:00:05.816] prepare took 1ms - succeeded, sql="PRAGMA user\_version;"

14: [1970-01-02 00:00:05.816] executeForString took 0ms - succeeded, sql="SELECT locale FROM android\_metadata UNION SELECT NULL ORDER BY locale DESC LIMIT 1"

15: [1970-01-02 00:00:05.815] execute took 0ms - succeeded, sql="CREATE TABLE IF NOT EXISTS android\_metadata (locale TEXT)"

16: [1970-01-02 00:00:05.809] executeForLong took 0ms - succeeded, sql="PRAGMA wal\_autocheckpoint=100"

17: [1970-01-02 00:00:05.809] executeForLong took 0ms - succeeded, sql="PRAGMA wal\_autocheckpoint"

18: [1970-01-02 00:00:05.809] executeForLong took 0ms - succeeded, sql="PRAGMA journal\_size\_limit=524288"

19: [1970-01-02 00:00:05.808] executeForLong took 0ms - succeeded, sql="PRAGMA journal\_size\_limit"

Available non-primary connections:

<none>

Acquired connections:

<none>

Connection waiters:

<none>

\*\* Database info for pid 12945 [com.youku.tv:multiscreen] \*\*

\*\* Database info for pid 12912 [com.hiapk.markettv] \*\*

Connection pool for /data/data/com.hiapk.markettv/databases/hiapk\_apk.db:

Open: true

Max connections: 1

Available primary connection:

Connection #0:

isPrimaryConnection: true

onlyAllowReadOnlyOperations: true

Most recently executed operations:

0: [2014-03-03 01:03:18.340] executeForCursorWindow took 0ms - succeeded, sql="SELECT a.\*, b.source, b.recd\_id FROM app\_download a LEFT JOIN app\_download\_extend b ON (a.app\_id=b.app\_id)"

1: [2014-03-03 01:03:18.339] prepare took 0ms - succeeded, sql="SELECT a.\*, b.source, b.recd\_id FROM app\_download a LEFT JOIN app\_download\_extend b ON (a.app\_id=b.app\_id)"

2: [2014-03-03 01:03:18.327] executeForLong took 0ms - succeeded, sql="PRAGMA user\_version;"

3: [2014-03-03 01:03:18.327] prepare took 0ms - succeeded, sql="PRAGMA user\_version;"

4: [2014-03-03 01:03:18.326] executeForString took 1ms - succeeded, sql="SELECT locale FROM android\_metadata UNION SELECT NULL ORDER BY locale DESC LIMIT 1"

5: [2014-03-03 01:03:18.326] execute took 0ms - succeeded, sql="CREATE TABLE IF NOT EXISTS android\_metadata (locale TEXT)"

6: [2014-03-03 01:03:18.325] executeForLong took 1ms - succeeded, sql="PRAGMA wal\_autocheckpoint=100"

7: [2014-03-03 01:03:18.325] executeForLong took 0ms - succeeded, sql="PRAGMA wal\_autocheckpoint"

8: [2014-03-03 01:03:18.325] executeForLong took 0ms - succeeded, sql="PRAGMA journal\_size\_limit=524288"

9: [2014-03-03 01:03:18.325] executeForLong took 0ms - succeeded, sql="PRAGMA journal\_size\_limit"

10: [2014-03-03 01:03:18.325] executeForString took 0ms - succeeded, sql="PRAGMA synchronous"

11: [2014-03-03 01:03:18.325] executeForString took 0ms - succeeded, sql="PRAGMA journal\_mode=PERSIST"

12: [2014-03-03 01:03:18.324] executeForString took 1ms - succeeded, sql="PRAGMA journal\_mode"

13: [2014-03-03 01:03:18.324] executeForLong took 0ms - succeeded, sql="PRAGMA foreign\_keys"

14: [2014-03-03 01:03:18.324] executeForLong took 0ms - succeeded, sql="PRAGMA page\_size"

Available non-primary connections:

<none>

Acquired connections:

<none>

Connection waiters:

<none>

Connection pool for /data/data/com.hiapk.markettv/databases/hiapk\_market.db:

Open: true

Max connections: 1

Available primary connection:

Connection #0:

isPrimaryConnection: true

onlyAllowReadOnlyOperations: true

Most recently executed operations:

0: [2014-03-03 01:03:18.466] executeForCursorWindow took 1ms - succeeded, sql="SELECT \* FROM software\_installed"

1: [2014-03-03 01:03:18.466] prepare took 0ms - succeeded, sql="SELECT \* FROM software\_installed"

2: [2014-03-03 01:03:18.312] executeForLong took 1ms - succeeded, sql="PRAGMA user\_version;"

3: [2014-03-03 01:03:18.312] prepare took 0ms - succeeded, sql="PRAGMA user\_version;"

4: [2014-03-03 01:03:18.311] executeForString took 1ms - succeeded, sql="SELECT locale FROM android\_metadata UNION SELECT NULL ORDER BY locale DESC LIMIT 1"

5: [2014-03-03 01:03:18.310] execute took 1ms - succeeded, sql="CREATE TABLE IF NOT EXISTS android\_metadata (locale TEXT)"

6: [2014-03-03 01:03:18.309] executeForLong took 0ms - succeeded, sql="PRAGMA wal\_autocheckpoint=100"

7: [2014-03-03 01:03:18.309] executeForLong took 0ms - succeeded, sql="PRAGMA wal\_autocheckpoint"

8: [2014-03-03 01:03:18.309] executeForLong took 0ms - succeeded, sql="PRAGMA journal\_size\_limit=524288"

9: [2014-03-03 01:03:18.309] executeForLong took 0ms - succeeded, sql="PRAGMA journal\_size\_limit"

10: [2014-03-03 01:03:18.309] executeForString took 0ms - succeeded, sql="PRAGMA synchronous"

11: [2014-03-03 01:03:18.308] executeForString took 1ms - succeeded, sql="PRAGMA journal\_mode=PERSIST"

12: [2014-03-03 01:03:18.307] executeForString took 1ms - succeeded, sql="PRAGMA journal\_mode"

13: [2014-03-03 01:03:18.307] executeForLong took 0ms - succeeded, sql="PRAGMA foreign\_keys"

14: [2014-03-03 01:03:18.307] executeForLong took 0ms - succeeded, sql="PRAGMA page\_size"

Available non-primary connections:

<none>

Acquired connections:

<none>

Connection waiters:

<none>

\*\* Database info for pid 12853 [com.qiyi.video] \*\*

\*\* Database info for pid 2683 [com.qiyi.video:bdservice\_v1] \*\*

Connection pool for /data/data/com.qiyi.video/databases/moplus\_server\_config.db:

Open: true

Max connections: 1

Available primary connection:

Connection #0:

isPrimaryConnection: true

onlyAllowReadOnlyOperations: true

Most recently executed operations:

0: [2014-02-28 10:18:29.824] executeForCursorWindow took 0ms - succeeded, sql="select \* from server\_config\_table where type=0"

1: [2014-02-28 10:18:29.823] prepare took 1ms - succeeded, sql="select \* from server\_config\_table where type=0"

2: [1970-01-02 00:00:10.971] executeForCursorWindow took 1ms - succeeded, sql="select \* from server\_config\_table where type=1"

3: [1970-01-02 00:00:10.970] prepare took 1ms - succeeded, sql="select \* from server\_config\_table where type=1"

4: [1970-01-02 00:00:10.912] execute took 58ms - succeeded, sql="COMMIT;"

5: [1970-01-02 00:00:10.912] executeForChangedRowCount took 0ms - succeeded, sql="PRAGMA user\_version = 1"

6: [1970-01-02 00:00:10.912] prepare took 0ms - succeeded, sql="PRAGMA user\_version = 1"

7: [1970-01-02 00:00:10.911] executeForChangedRowCount took 1ms - succeeded, sql="CREATE TABLE server\_config\_table (\_id INTEGER PRIMARY KEY,name TEXT,type INTEGER,value TEXT);"

8: [1970-01-02 00:00:10.911] prepare took 0ms - succeeded, sql="CREATE TABLE server\_config\_table (\_id INTEGER PRIMARY KEY,name TEXT,type INTEGER,value TEXT);"

9: [1970-01-02 00:00:10.910] execute took 0ms - succeeded, sql="BEGIN EXCLUSIVE;"

10: [1970-01-02 00:00:10.910] executeForLong took 0ms - succeeded, sql="PRAGMA user\_version;"

11: [1970-01-02 00:00:10.909] prepare took 1ms - succeeded, sql="PRAGMA user\_version;"

12: [1970-01-02 00:00:10.753] execute took 156ms - succeeded, sql="COMMIT"

13: [1970-01-02 00:00:10.753] execute took 0ms - succeeded, sql="REINDEX LOCALIZED"

14: [1970-01-02 00:00:10.752] execute took 1ms - succeeded, sql="INSERT INTO android\_metadata (locale) VALUES(?)", bindArgs=["zh\_CN"]

15: [1970-01-02 00:00:10.752] execute took 0ms - succeeded, sql="DELETE FROM android\_metadata"

16: [1970-01-02 00:00:10.751] execute took 1ms - succeeded, sql="BEGIN"

17: [1970-01-02 00:00:10.751] executeForString took 0ms - succeeded, sql="SELECT locale FROM android\_metadata UNION SELECT NULL ORDER BY locale DESC LIMIT 1"

18: [1970-01-02 00:00:10.599] execute took 152ms - succeeded, sql="CREATE TABLE IF NOT EXISTS android\_metadata (locale TEXT)"

19: [1970-01-02 00:00:10.598] executeForLong took 0ms - succeeded, sql="PRAGMA wal\_autocheckpoint=100"

Available non-primary connections:

<none>

Acquired connections:

<none>

Connection waiters:

<none>

\*\* Database info for pid 2357 [com.android.smspush] \*\*

\*\* Database info for pid 2240 [com.android.phone] \*\*

Connection pool for /data/data/com.android.providers.telephony/databases/telephony.db:

Open: true

Max connections: 1

Available primary connection:

Connection #0:

isPrimaryConnection: true

onlyAllowReadOnlyOperations: true

Most recently executed operations:

0: [1970-01-02 00:00:11.709] executeForCursorWindow took 0ms - succeeded, sql="SELECT apn FROM carriers WHERE (\_id = -1) ORDER BY name ASC"

1: [1970-01-02 00:00:11.707] prepare took 0ms - succeeded, sql="SELECT apn FROM carriers WHERE (\_id = -1) ORDER BY name ASC"

2: [1970-01-02 00:00:11.706] executeForLong took 1ms - succeeded, sql="PRAGMA user\_version;"

3: [1970-01-02 00:00:11.706] prepare took 0ms - succeeded, sql="PRAGMA user\_version;"

4: [1970-01-02 00:00:11.701] executeForString took 1ms - succeeded, sql="SELECT locale FROM android\_metadata UNION SELECT NULL ORDER BY locale DESC LIMIT 1"

5: [1970-01-02 00:00:11.701] execute took 0ms - succeeded, sql="CREATE TABLE IF NOT EXISTS android\_metadata (locale TEXT)"

6: [1970-01-02 00:00:11.700] executeForLong took 0ms - succeeded, sql="PRAGMA wal\_autocheckpoint=100"

7: [1970-01-02 00:00:11.700] executeForLong took 0ms - succeeded, sql="PRAGMA wal\_autocheckpoint"

8: [1970-01-02 00:00:11.688] executeForLong took 12ms - succeeded, sql="PRAGMA journal\_size\_limit=524288"

9: [1970-01-02 00:00:11.688] executeForLong took 0ms - succeeded, sql="PRAGMA journal\_size\_limit"

10: [1970-01-02 00:00:11.688] executeForString took 0ms - succeeded, sql="PRAGMA synchronous"

11: [1970-01-02 00:00:11.687] executeForString took 0ms - succeeded, sql="PRAGMA journal\_mode=PERSIST"

12: [1970-01-02 00:00:11.682] executeForString took 5ms - succeeded, sql="PRAGMA journal\_mode"

13: [1970-01-02 00:00:11.681] executeForLong took 0ms - succeeded, sql="PRAGMA foreign\_keys"

14: [1970-01-02 00:00:11.681] executeForLong took 0ms - succeeded, sql="PRAGMA page\_size"

Available non-primary connections:

<none>

Acquired connections:

<none>

Connection waiters:

<none>

\*\* Database info for pid 2280 [com.coship.krdservice] \*\*

\*\* Database info for pid 2222 [com.hisilicon.dlna.dms] \*\*

\*\* Database info for pid 2272 [com.bel.android.dspmanager] \*\*

## dumpsys diskstats

系统内部存储设备的使用情况。

diskstats服务注册位置：

(源码路径：.\framework\base\services\java\com\android\server\SystemServer.java)



dump实现位置:

(源码路径：.\framework\base\services\java\com\android\server\DiskStatsService.java)



dumpsys diskstats用法：

Latency: 0ms [512B Data Write]

Data-Free: 894012K / 1032088K total = 86% free

Cache-Free: 31512K / 100784K total = 31% free

System-Free: 234744K / 503944K total = 46% free

图 8 dumpsys diskstats示意图

## dumpsys devicestoragemonitor

Current DeviceStorageMonitor state:

mFreeMem=894 MB mTotalMemory=0.98 GB

mFreeMemAfterLastCacheClear=894 MB

mLastReportedFreeMem=895 MB mLastReportedFreeMemTime=-3h41m36s839ms

mLowMemFlag=false mMemFullFlag=false

mClearSucceeded=false mClearingCache=false

mMemLowThreshold=101 MB mMemFullThreshold=1.00 MB

mMemCacheStartTrimThreshold=75.84 MB mMemCacheTrimToThreshold=151 MB

## dumpsys package

查看一些系统信息和所有应用的信息。

不建议使用dumpsys package去打印所有信息，可以通过添加参数去获得需要的信息。

如果一定要dumpsys package,则可到盒子的可写分区/mnt，重定向到文本里。

dumpsys package > package.txt

后续的命令都可以重定向到文本，这样便于查看分析问题。

### dumpsys package –h(help)

Package manager dump options:

[-h] [-f] [cmd] ...

-f: print details of intent filters

-h: print this help

cmd may be one of:

l[ibraries]: list known shared libraries

f[ibraries]: list device features

r[esolvers]: dump intent resolvers

perm[issions]: dump permissions

pref[erred]: print preferred package settings

preferred-xml: print preferred package settings as xml

prov[iders]: dump content providers

p[ackages]: dump installed packages

s[hared-users]: dump shared user IDs

m[essages]: print collected runtime messages

v[erifiers]: print package verifier info

<package.name>: info about given package

### dumpsys package l

作用：列出所有共享库。

Libraries:

android.test.runner -> /system/framework/android.test.runner.jar

com.android.location.provider -> /system/framework/com.android.location.provider.jar

javax.obex -> /system/framework/javax.obex.jar

com.android.future.usb.accessory -> /system/framework/com.android.future.usb.accessory.jar

### dumpsys package f

作用：列出设备特征，这个主要判断设备是否支持某个应用。

Android Market会根据uses-feature过滤所有你设备不支持的应用。通过使用<uses-feature>元素，一个应用可以指定它所支持的硬件型号。

例如：

当你的游戏需要一个支持多点触控的屏幕的时候，我们可以使用 <uses-feature>元素来剔除所有不支持多点触控的设备，就像下面这样：  
<uses-feature android:name="android.hardware.touchscreen.multitouch" android:required="true"/>

通过dumpsys package f命令查看设备特征，即可确定设备是否支持某些应用。

Features:

android.hardware.wifi

android.hardware.location.network

android.hardware.location

android.hardware.sensor.gyroscope

android.hardware.screen.landscape

android.hardware.screen.portrait

com.cyanogenmod.android

android.hardware.wifi.direct

android.hardware.usb.accessory

android.hardware.camera.any

android.hardware.bluetooth

android.hardware.touchscreen.multitouch.distinct

### dumpsys package p

可以查看某个应用程序的版本号（versionName）、安装时间（firstInstallTime）、升级时间（lastUpdateTime）、安装路径（codePath）、共享用户（sharedUser）、权限（grantedPermissions）

Package [com.android.email] (41106fe0):

userId=10019 gids=[3003, 1015, 1028]

sharedUser=null

pkg=Package{411df440 com.android.email}

codePath=/system/app/Email2.apk

resourcePath=/system/app/Email2.apk

nativeLibraryPath=/data/app-lib/Email2

versionCode=410000

applicationInfo=ApplicationInfo{411df4e0 com.android.email}

flags=[ SYSTEM HAS\_CODE ALLOW\_CLEAR\_USER\_DATA ALLOW\_BACKUP ]

versionName=4.1

dataDir=/data/data/com.android.email

targetSdk=17

supportsScreens=[small, medium, large, xlarge, resizeable, anyDensity]

timeStamp=2014-02-25 13:55:03

firstInstallTime=2014-02-25 13:55:03

lastUpdateTime=2014-02-25 13:55:03

signatures=PackageSignatures{411070a8 [410e7c80]}

permissionsFixed=false haveGids=true installStatus=1

pkgFlags=[ SYSTEM HAS\_CODE ALLOW\_CLEAR\_USER\_DATA ALLOW\_BACKUP ]

User 0: installed=true stopped=false notLaunched=false enabled=0

disabledComponents:

com.android.email.activity.MessageCompose

com.android.email.activity.ShortcutPicker

com.android.email.service.AttachmentDownloadService

com.android.email.service.MailService

enabledComponents:

com.android.email.widget.WidgetConfiguration

grantedPermissions:

android.permission.READ\_EXTERNAL\_STORAGE

android.permission.ACCESS\_NETWORK\_STATE

### dumpsys package s

查看共享用户的应用程序。每个应用程序以SharedUser[包名]分开，可以互相访问数据。

参考：<http://blog.csdn.net/a345017062/article/details/6236263>

SharedUser [android.uid.bluetooth] (41011880):

userId=1002 gids=[3003, 3002, 3001, 1015, 3005, 1016, 3008, 1028]

grantedPermissions:

com.android.email.permission.ACCESS\_PROVIDER

android.permission.READ\_EXTERNAL\_STORAGE

android.permission.WRITE\_EXTERNAL\_STORAGE

SharedUser [android.uid.calendar] (40fc6210):

userId=10011 gids=[3003, 1015, 1028]

grantedPermissions:

android.permission.READ\_EXTERNAL\_STORAGE

android.permission.USE\_CREDENTIALS

android.permission.WRITE\_EXTERNAL\_STORAGE

android.permission.GET\_ACCOUNTS

### dumpsys package $package-name

查看某个应用程序的安装路径、权限、开机启动（Non-Data Actions的android.intent.action.BOOT\_COMPLETED表示开机启动）等信息。

Receiver Resolver Table:

Schemes:

file:

411d6f38 com.android.providers.downloads/.DownloadReceiver filter 411d73f8

Non-Data Actions:

android.intent.action.BOOT\_COMPLETED:

411d6f38 com.android.providers.downloads/.DownloadReceiver filter 411d71c8

android.net.conn.CONNECTIVITY\_CHANGE:

411d6f38 com.android.providers.downloads/.DownloadReceiver filter 411d71c8

## dumpsys meminfo

### dumpsys meminfo –h(help)

meminfo dump options: [-a] [--oom] [process]

-a: include all available information for each process.

--oom: only show processes organized by oom adj.

If [process] is specified it can be the name or

pid of a specific process to dump.

### dumpsys meminfo

显示各个进程所占内存信息。

* Total PSS by process ：各个进程占用的物理内存。
* Total PSS by OOM adjustment ：OOM 杀进程后，可以要回来的内存。
* Total PSS by category ：按memory使用类型统计内存使用情况。

Applications Memory Usage (kB):

Uptime: 34986361 Realtime: 34986361

Total PSS by process:

32849 kB: system (pid 2214)

26765 kB: com.android.settings (pid 3372)

23280 kB: com.android.browser (pid 3643)

Total PSS by OOM adjustment:

28935 kB: Persistent

14770 kB: com.android.systemui (pid 2326)

7231 kB: com.android.phone (pid 2469)

3611 kB: com.bel.android.dspmanager (pid 2483)

Total PSS by category:

91823 kB: Dalvik

8184 kB: .apk mmap

Total PSS: 199528 kB

KSM: 0 kB saved from shared 0 kB

### dumpsys meminfo $package\_name or $pi//使用程序的包名或者进程id

dumpsys meminfo pkname

一般来说内存占用大小有如下规律：VSS >= RSS >= PSS >= USS  
VSS - Virtual Set Size 虚拟耗用内存（包含共享库占用的内存）是单个进程全部可访问的地址空间  
RSS - Resident Set Size 实际使用物理内存（包含共享库占用的内存）是单个进程实际占用的内存大小，对于单个共享库， 尽管无论多少个进程使用，实际该共享库只会被装入内存一次。  
PSS - Proportional Set Size 实际使用的物理内存（比例分配共享库占用的内存）  
USS - Unique Set Size 进程独自占用的物理内存（不包含共享库占用的内存）USS 是一个非常非常有用的数字， 因为它揭示了运行一个特定进程的真实的内存增量大小。如果进程被终止， USS 就是实际被返还给系统的内存大小。  
**USS** 是针对某个进程开始有可疑内存泄露的情况，进行检测的最佳数字。怀疑某个程序有内存泄露可以查看这个值是否一直有增加

android程序内存被分为2部分：native和dalvik，dalvik就是我们平常说的java堆，我们创建的对象是在这里面分配的，而bitmap是直接在native上分配的

对于内存的限制是 native+dalvik 不能超过最大限制。

可以通过查看cat /system/ build.prop里面的dalvik.vm.heapsize=512m字段确定应用程序默认内存大小。

Applications Memory Usage (kB):

Uptime: 35562088 Realtime: 35562088

\*\* MEMINFO in pid 5749 [com.tencent.mobileqq] \*\*

Shared Private Heap Heap Heap

Pss Dirty Dirty Size Alloc Free

------ ------ ------ ------ ------ ------

Native 20 8 20 4676 3064 15

Dalvik 11955 5200 11628 19056 10892 8164

Cursor 0 0 0

Ashmem 0 0 0

Other dev 4 20 0

.so mmap 1739 2168 1276

.jar mmap 4 0 0

.apk mmap 539 0 0

.ttf mmap 121 0 0

.dex mmap 6031 0 96

Other mmap 194 16 144

Unknown 1368 376 1348

TOTAL 21975 7788 14512 23732 13956 8179

Objects

Views: 41 ViewRootImpl: 2

AppContexts: 6 Activities: 2

Assets: 2 AssetManagers: 2

Local Binders: 21 Proxy Binders: 26

Death Recipients: 1

OpenSSL Sockets: 0

SQL

MEMORY\_USED: 0

PAGECACHE\_OVERFLOW: 0 MALLOC\_SIZE: 62

当Heap Size:Native+Dalvik超过512m,则会出现OOM。

meminfo服务注册位置：

代码路径./frameworks/base/services/java/com/android/server/am/ActivityManagerService.java）



meminfo对应的dump函数实现封装在MemBinder(m)里面

## dumpsys mount

用于查看盒子开机启动后执行移动硬盘或U盘挂载和卸载流程和次数。

……………….

…………………

mConnection:

13:00:04 - SND -> {1 volume list}

13:00:04 - RCV <- {110 1 block /mnt/sdcard 1 UUID= DevType=SDCARD 66319 (null)}

13:00:04 - RCV <- {200 1 Volumes listed.}

13:00:04 - RMV <- {110 1 block /mnt/sdcard 1 UUID= DevType=SDCARD 66319 (null)}

13:00:04 - RMV <- {200 1 Volumes listed.}

13:00:05 - RCV <- {605 Volume block /mnt/sdcard state changed from 1 (Idle-Unmounted) to 3 (Checking)}

13:00:05 - Error handling '605 Volume block /mnt/sdcard state changed from 1 (Idle-Unmounted) to 3 (Checking)': java.lang.NullPointerException

13:00:05 - RCV <- {605 Volume block /mnt/sdcard state changed from 3 (Checking) to 4 (Mounted) UUID= DevType=SDCARD 66319 (null)}

图 4第一次开机启动dumpsys mount状态示意图

……………..

……………

mConnection:

13:00:04 - SND -> {1 volume list}

13:00:04 - RCV <- {110 1 block /mnt/sdcard 1 UUID= DevType=SDCARD 66319 (null)}

13:00:04 - RCV <- {200 1 Volumes listed.}

13:00:04 - RMV <- {110 1 block /mnt/sdcard 1 UUID= DevType=SDCARD 66319 (null)}

13:00:04 - RMV <- {200 1 Volumes listed.}

13:00:05 - RCV <- {605 Volume block /mnt/sdcard state changed from 1 (Idle-Unmounted) to 3 (Checking)}

13:00:05 - Error handling '605 Volume block /mnt/sdcard state changed from 1 (Idle-Unmounted) to 3 (Checking)': java.lang.NullPointerException

13:00:05 - RCV <- {605 Volume block /mnt/sdcard state changed from 3 (Checking) to 4 (Mounted) UUID= DevType=SDCARD 66319 (null)}

13:04:19 - RCV <- {605 Volume sda1 /mnt/sda/sda1 state changed from -1 (Initializing) to 0 (No-Media)}

13:04:19 - RCV <- {605 Volume sda1 /mnt/sda/sda1 state changed from 0 (No-Media) to 1 (Idle-Unmounted)}

13:04:21 - RCV <- {630 Volume sda1 /mnt/sda/sda1 disk inserted (8:1)}

13:04:21 - RCV <- {605 Volume sda1 /mnt/sda/sda1 state changed from 1 (Idle-Unmounted) to 3 (Checking)}

13:04:21 - Error handling '605 Volume sda1 /mnt/sda/sda1 state changed from 1 (Idle-Unmounted) to 3 (Checking)': java.lang.NullPointerException

13:04:21 - RCV <- {605 Volume sda1 /mnt/sda/sda1 state changed from 3 (Checking) to 4 (Mounted) UUID= DevType=USB 2049 (null)}

图 5插上U盘后dumpsys mount示意图

|  |  |
| --- | --- |
| SDcard常用状态 |  |
| Environment.MEDIA\_MOUNTED | sd卡在手机上正常使用状态 |
| Environment.MEDIA\_UNMOUNTED | 用户手工到手机设置中卸载sd卡之后的状态 |
| Environment.MEDIA\_REMOVED | 用户手动卸载，然后将sd卡从手机取出之后的状态 |
| Environment.MEDIA\_BAD\_REMOVAL | 用户未到手机设置中手动卸载sd卡，直接拨出之后的状态 |
| Environment.MEDIA\_SHARED | 手机直接连接到电脑作为u盘使用之后的状态 |
| Environment.MEDIA\_CHECKINGS | 手机正在扫描sd卡过程中的状态 |
| Environment.MEDIA\_UNMOUNTABLE | SDCard 不可被安装 如果 SDCard 是存在但不可以被安装 |
| Environment.MEDIA\_NOFS | 表明对象为空白或正在使用不受支持的文件系统 |

表格 1 SDcard常用状态

在图5中红色部分表示U盘挂载上去的状态变化过程。

………………

………………

mConnection:

13:00:04 - SND -> {1 volume list}

13:00:04 - RCV <- {110 1 block /mnt/sdcard 1 UUID= DevType=SDCARD 66319 (null)}

13:00:04 - RCV <- {200 1 Volumes listed.}

13:00:04 - RMV <- {110 1 block /mnt/sdcard 1 UUID= DevType=SDCARD 66319 (null)}

13:00:04 - RMV <- {200 1 Volumes listed.}

13:00:05 - RCV <- {605 Volume block /mnt/sdcard state changed from 1 (Idle-Unmounted) to 3 (Checking)}

13:00:05 - Error handling '605 Volume block /mnt/sdcard state changed from 1 (Idle-Unmounted) to 3 (Checking)': java.lang.NullPointerException

13:00:05 - RCV <- {605 Volume block /mnt/sdcard state changed from 3 (Checking) to 4 (Mounted) UUID= DevType=SDCARD 66319 (null)}

13:04:19 - RCV <- {605 Volume sda1 /mnt/sda/sda1 state changed from -1 (Initializing) to 0 (No-Media)}

13:04:19 - RCV <- {605 Volume sda1 /mnt/sda/sda1 state changed from 0 (No-Media) to 1 (Idle-Unmounted)}

13:04:21 - RCV <- {630 Volume sda1 /mnt/sda/sda1 disk inserted (8:1)}

13:04:21 - RCV <- {605 Volume sda1 /mnt/sda/sda1 state changed from 1 (Idle-Unmounted) to 3 (Checking)}

13:04:21 - Error handling '605 Volume sda1 /mnt/sda/sda1 state changed from 1 (Idle-Unmounted) to 3 (Checking)': java.lang.NullPointerException

13:04:21 - RCV <- {605 Volume sda1 /mnt/sda/sda1 state changed from 3 (Checking) to 4 (Mounted) UUID= DevType=USB 2049 (null)}

13:10:35 - RCV <- {632 Volume sda1 /mnt/sda/sda1 bad removal (8:1)}

13:10:35 - RCV <- {605 Volume sda1 /mnt/sda/sda1 state changed from 4 (Mounted) to 5 (Unmounting)}

13:10:36 - RCV <- {605 Volume sda1 /mnt/sda/sda1 state changed from 5 (Unmounting) to 1 (Idle-Unmounted)}

图 7拔掉U盘后dumpsys mount示意图

褐色部分表示U盘卸载过程。

通过dumpsys mount 可以查看盒子启动过后，U盘挂载和卸载的状态和次数，直到盒子重启后，记录才会被清除。

## dumpsys display

DISPLAY MANAGER (dumpsys display)

mHeadless=false

mOnlyCode=false

mSafeMode=false

mPendingTraversal=false

mAllDisplayBlankStateFromPowerManager=2

mNextNonDefaultDisplayId=1

mDefaultViewport=DisplayViewport{valid=true, displayId=0, orientation=0, logicalFrame=Rect(0, 0 - 1280, 720), physicalFrame=Rect(0, 0 - 1280, 720), deviceWidth=1280, deviceHeight=720}

mExternalTouchViewport=DisplayViewport{valid=false, displayId=0, orientation=0, logicalFrame=Rect(0, 0 - 0, 0), physicalFrame=Rect(0, 0 - 0, 0), deviceWidth=0, deviceHeight=0}

mSingleDisplayDemoMode=false

Display Adapters: size=2

LocalDisplayAdapter

OverlayDisplayAdapter

mCurrentOverlaySetting=

mOverlays: size=0

Display Devices: size=1

DisplayDeviceInfo{"内置屏幕": 1280 x 720, 60.000004 fps, density 160, 160.15764 x 160.42105 dpi, touch INTERNAL, FLAG\_DEFAULT\_DISPLAY, FLAG\_ROTATES\_WITH\_CONTENT, FLAG\_SECURE, FLAG\_SUPPORTS\_PROTECTED\_BUFFERS, rotation 0, type BUILT\_IN, address null}

mAdapter=LocalDisplayAdapter

mDisplayToken=android.os.BinderProxy@423fd038

mCurrentLayerStack=0

mCurrentOrientation=0

mCurrentLayerStackRect=Rect(0, 0 - 1280, 720)

mCurrentDisplayRect=Rect(0, 0 - 1280, 720)

mCurrentSurface=null

mBuiltInDisplayId=0

mPhys=PhysicalDisplayInfo{1280 x 720, 60.000004 fps, density 1.0, 160.15764 x 160.42105 dpi, secure true}

mBlanked=false

Logical Displays: size=1

Display 0:

mDisplayId=0

mLayerStack=0

mHasContent=true

mPrimaryDisplayDevice=内置屏幕

mBaseDisplayInfo=DisplayInfo{"内置屏幕", app 1280 x 720, real 1280 x 720, largest app 1280 x 720, smallest app 1280 x 720, 60.000004 fps, rotation 0, density 160, 160.15764 x 160.42105 dpi, layerStack 0, type BUILT\_IN, address null, FLAG\_SECURE, FLAG\_SUPPORTS\_PROTECTED\_BUFFERS}

mOverrideDisplayInfo=DisplayInfo{"内置屏幕", app 1280 x 720, real 1280 x 720, largest app 1280 x 1255, smallest app 720 x 695, 60.000004 fps, rotation 0, density 160, 160.15764 x 160.42105 dpi, layerStack 0, type BUILT\_IN, address null, FLAG\_SECURE, FLAG\_SUPPORTS\_PROTECTED\_BUFFERS}

## dumpsys window

Window Manager Service 位于Surface Flinger 之上，将要绘制到机器画面上的内容传递给Surface Flinger

### dumpsys window lastanr

用于查看从开机到现场是否发生ANR事件。

WINDOW MANAGER LAST ANR (dumpsys window lastanr)

<no ANR has occurred since boot>

### dumpsys window policy

WINDOW MANAGER POLICY STATE (dumpsys window policy)

mSafeMode=false mSystemReady=true mSystemBooted=true

mLidState=-1 mLidOpenRotation=-1 mHdmiPlugged=true

mUiMode=1 mDockMode=0 mCarDockRotation=-1 mDeskDockRotation=-1

mUserRotationMode=0 mUserRotation=0 mAllowAllRotations=-1

mCurrentAppOrientation=0

mCarDockEnablesAccelerometer=true mDeskDockEnablesAccelerometer=true

mLidKeyboardAccessibility=0 mLidNavigationAccessibility=0 mLidControlsSleep=true

mLongPressOnPowerBehavior=-1 mHasSoftInput=true

mScreenOnEarly=true mScreenOnFully=true mOrientationSensorEnabled=false

mUnrestrictedScreen=(0,0) 1280x720

mRestrictedScreen=(0,0) 1280x720

mStableFullscreen=(0,0)-(1280,720)

mStable=(0,25)-(1280,720)

mSystem=(0,0)-(1280,720)

mCur=(0,0)-(1280,720)

mContent=(0,0)-(1280,720)

mDock=(0,0)-(1280,720)

mDockLayer=268435456 mStatusBarLayer=151000

mShowingLockscreen=false mShowingDream=false mDreamingLockscreen=false

mStatusBar=Window{42600d28 u0 StatusBar}

mFocusedWindow=Window{42688fc8 u0 com.youku.tv/com.youku.tv.ui.activity.AllViedosActivity}

mFocusedApp=Token{427490e8 ActivityRecord{4277a408 u0 com.youku.tv/.ui.activity.AllViedosActivity}}

mWinDismissingKeyguard=Window{424a1310 u0 com.youku.tv/com.youku.player.activity.YoukuTVPlayerActivity}

mTopFullscreenOpaqueWindowState=Window{42688fc8 u0 com.youku.tv/com.youku.tv.ui.activity.AllViedosActivity}

mTopIsFullscreen=true mHideLockScreen=false

mForceStatusBar=false mForceStatusBarFromKeyguard=false

mDismissKeyguard=0 mWinDismissingKeyguard=Window{424a1310 u0 com.youku.tv/com.youku.player.activity.YoukuTVPlayerActivity} mHomePressed=false

mAllowLockscreenWhenOn=false mLockScreenTimeout=-1 mLockScreenTimerActive=false

mEndcallBehavior=2 mIncallPowerBehavior=1 mRingHomeBehavior=1 mLongPressOnHomeBehavior=2

mLandscapeRotation=0 mSeascapeRotation=2

mPortraitRotation=3 mUpsideDownRotation=1

mHdmiRotation=0 mHdmiRotationLock=true

### dumpsys window animator

WINDOW MANAGER ANIMATOR STATE (dumpsys window animator)

App Animators:

App Animator #3 com.android.server.wm.AppWindowAnimator@427cda60

App Animator #2 com.android.server.wm.AppWindowAnimator@4277fdc0

App Animator #1 com.android.server.wm.AppWindowAnimator@425f78c8

App Animator #0 com.android.server.wm.AppWindowAnimator@42482f28

Wallpaper tokens:

Wallpaper #0 WindowToken{426e4120 android.os.Binder@426f7178}

DisplayContentsAnimator #0:

Window #0: WindowStateAnimator{42745db0 com.android.systemui.ImageWallpaper}

Window #1: WindowStateAnimator{425ba800 com.youku.tv/com.youku.tv.ui.activity.AllViedosActivity}

Window #2: WindowStateAnimator{42704d88 StatusBar}

mBulkUpdateParams=0x8 ORIENTATION\_CHANGE\_COMPLETE

mWallpaperTarget=null

mWpAppAnimator=null

### dumpsys window sessions

WINDOW MANAGER SESSIONS (dumpsys window sessions)

Session Session{4272d218 1974:1000}:

mNumWindow=0 mClientDead=false mSurfaceSession=android.view.SurfaceSession@425eb868

Session Session{425e8270 2092:u0a10046}:

mNumWindow=2 mClientDead=false mSurfaceSession=android.view.SurfaceSession@424bd1a8

Session Session{4246e9f0 3563:u0a10059}:

mNumWindow=2 mClientDead=false mSurfaceSession=android.view.SurfaceSession@42740888

Session Session{423e6e50 2300:u0a10049}:

mNumWindow=1 mClientDead=false [mSurfaceSession=android.view.SurfaceSession@426be340](mailto:mSurfaceSession=android.view.SurfaceSession@426be340)

### dumpsys window tokens

WINDOW MANAGER TOKENS (dumpsys window tokens)

All tokens:

WindowToken{42771590 null}

AppWindowToken{425170f8 token=Token{42515000 ActivityRecord{42514bc8 u0 com.cyanogenmod.trebuchet/.Launcher}}}

WindowToken{426e4120 android.os.Binder@426f7178}

AppWindowToken{42595520 token=Token{423ca098 ActivityRecord{42549870 u0 com.youku.tv/.ui.activity.HomeActivityWithViewPager}}}

AppWindowToken{4277fca8 token=Token{427490e8 ActivityRecord{4277a408 u0 com.youku.tv/.ui.activity.AllViedosActivity}}}

WindowToken{4258ffd8 android.os.Binder@4258ff98}

Wallpaper tokens:

Wallpaper #0 WindowToken{426e4120 android.os.Binder@426f7178}

Application tokens in Z order:

App #2 AppWindowToken{4277fca8 token=Token{427490e8 ActivityRecord{4277a408 u0 com.youku.tv/.ui.activity.AllViedosActivity}}}:

windows=[Window{42688fc8 u0 com.youku.tv/com.youku.tv.ui.activity.AllViedosActivity}]

windowType=2 hidden=false hasVisible=true

app=true userId=0

allAppWindows=[Window{42688fc8 u0 com.youku.tv/com.youku.tv.ui.activity.AllViedosActivity}]

groupId=9 appFullscreen=true requestedOrientation=0

hiddenRequested=false clientHidden=false willBeHidden=false reportedDrawn=true reportedVisible=true

numInterestingWindows=1 numDrawnWindows=1 inPendingTransaction=false allDrawn=true (animator=true)

startingData=null removed=false firstWindowDrawn=true

App #1 AppWindowToken{42595520 token=Token{423ca098 ActivityRecord{42549870 u0 com.youku.tv/.ui.activity.HomeActivityWithViewPager}}}:

windows=[Window{4251d7f8 u0 com.youku.tv/com.youku.tv.ui.activity.HomeActivityWithViewPager}]

windowType=2 hidden=true hasVisible=true

app=true userId=0

allAppWindows=[Window{4251d7f8 u0 com.youku.tv/com.youku.tv.ui.activity.HomeActivityWithViewPager}]

groupId=9 appFullscreen=true requestedOrientation=0

hiddenRequested=true clientHidden=true willBeHidden=false reportedDrawn=false reportedVisible=false

numInterestingWindows=1 numDrawnWindows=1 inPendingTransaction=false allDrawn=true (animator=true)

startingData=null removed=false firstWindowDrawn=true

App #0 AppWindowToken{425170f8 token=Token{42515000 ActivityRecord{42514bc8 u0 com.cyanogenmod.trebuchet/.Launcher}}}:

windows=[Window{423f1628 u0 com.cyanogenmod.trebuchet/com.cyanogenmod.trebuchet.Launcher}]

windowType=2 hidden=true hasVisible=true

app=true userId=0

allAppWindows=[Window{423f1628 u0 com.cyanogenmod.trebuchet/com.cyanogenmod.trebuchet.Launcher}]

groupId=2 appFullscreen=true requestedOrientation=-1

hiddenRequested=true clientHidden=true willBeHidden=false reportedDrawn=false reportedVisible=false

numInterestingWindows=1 numDrawnWindows=1 inPendingTransaction=false allDrawn=true (animator=true)

startingData=null removed=false firstWindowDrawn=true

### dumpsys window windows

WINDOW MANAGER WINDOWS (dumpsys window windows)

Window #0 Window{42600d28 u0 StatusBar}:

mDisplayId=0 mSession=Session{425e8270 2092:u0a10046} mClient=android.os.BinderProxy@427311b0

mOwnerUid=10046 mShowToOwnerOnly=false

mAttrs=WM.LayoutParams{(0,0)(fillx25) gr=#37 sim=#20 ty=2000 fl=#1800048 fmt=-3}

Requested w=1280 h=25 mLayoutSeq=8017

mPolicyVisibility=false mPolicyVisibilityAfterAnim=false mAttachedHidden=false

mHasSurface=true mShownFrame=[0.0,0.0][1280.0,25.0] isReadyForDisplay()=false

WindowStateAnimator{42704d88 StatusBar}:

Surface: shown=false layer=151000 alpha=1.0 rect=(0.0,0.0) 1280.0 x 25.0

Window #1 Window{42688fc8 u0 com.youku.tv/com.youku.tv.ui.activity.AllViedosActivity}:

mDisplayId=0 mSession=Session{4246e9f0 3563:u0a10059} mClient=android.os.BinderProxy@4263d3c0

mOwnerUid=10059 mShowToOwnerOnly=true

mAttrs=WM.LayoutParams{(0,0)(fillxfill) sim=#120 ty=1 fl=#1810580 pfl=0x8 wanim=0x1030001}

Requested w=1280 h=720 mLayoutSeq=8017

mHasSurface=true mShownFrame=[0.0,0.0][1280.0,720.0] isReadyForDisplay()=true

WindowStateAnimator{425ba800 com.youku.tv/com.youku.tv.ui.activity.AllViedosActivity}:

Surface: shown=true layer=21015 alpha=1.0 rect=(0.0,0.0) 1280.0 x 720.0

Window #2 Window{4251d7f8 u0 com.youku.tv/com.youku.tv.ui.activity.HomeActivityWithViewPager}:

mDisplayId=0 mSession=Session{4246e9f0 3563:u0a10059} mClient=android.os.BinderProxy@42495cb8

mOwnerUid=10059 mShowToOwnerOnly=true

mAttrs=WM.LayoutParams{(0,0)(fillxfill) sim=#120 ty=1 fl=#1810580 pfl=0x8 wanim=0x1030001}

Requested w=1280 h=720 mLayoutSeq=207

mHasSurface=false mShownFrame=[0.0,0.0][1280.0,720.0] isReadyForDisplay()=false

WindowStateAnimator{426ba9d8 com.youku.tv/com.youku.tv.ui.activity.HomeActivityWithViewPager}:

Window #3 Window{423f1628 u0 com.cyanogenmod.trebuchet/com.cyanogenmod.trebuchet.Launcher}:

mDisplayId=0 mSession=Session{423e6e50 2300:u0a10049} mClient=android.os.BinderProxy@42781e98

mOwnerUid=10049 mShowToOwnerOnly=true

mAttrs=WM.LayoutParams{(0,0)(fillxfill) sim=#20 ty=1 fl=#1910100 pfl=0x8 fmt=-2 wanim=0x1030295 vsysui=0x400}

Requested w=1280 h=720 mLayoutSeq=187

mHasSurface=false mShownFrame=[0.0,0.0][1280.0,720.0] isReadyForDisplay()=false

WindowStateAnimator{4275a2a8 com.cyanogenmod.trebuchet/com.cyanogenmod.trebuchet.Launcher}:

mWallpaperX=0.5 mWallpaperY=0.5

mWallpaperXStep=0.25 mWallpaperYStep=1.0

Window #4 Window{4245bfb8 u0 com.android.systemui.ImageWallpaper}:

mDisplayId=0 mSession=Session{425e8270 2092:u0a10046} mClient=android.os.BinderProxy@42647aa8

mOwnerUid=10046 mShowToOwnerOnly=true

mAttrs=WM.LayoutParams{(0,0)(2015x1280) gr=#800033 ty=2013 fl=#318 fmt=2 wanim=0x10301ec}

Requested w=2015 h=1280 mLayoutSeq=191

mIsImWindow=false mIsWallpaper=true mIsFloatingLayer=true mWallpaperVisible=false

Offsets x=-368 y=-280

mHasSurface=true mShownFrame=[-368.0,-280.0][1647.0,1000.0] isReadyForDisplay()=false

WindowStateAnimator{42745db0 com.android.systemui.ImageWallpaper}:

Surface: shown=false layer=21000 alpha=1.0 rect=(-368.0,-280.0) 2015.0 x 1280.0

mWallpaperX=0.5 mWallpaperY=0.5

mWallpaperXStep=0.25 mWallpaperYStep=1.0

DisplayContents:

Display: mDisplayId=0

init=1280x720 160dpi cur=1280x720 app=1280x720 rng=720x695-1280x1255

layoutNeeded=false

mCurConfiguration={1.0 ?mcc?mnc zh\_CN ldltr sw720dp w1280dp h695dp 160dpi xlrg long land finger qwerty/v/v dpad/v s.4 themeResource=null}

mCurrentFocus=Window{42688fc8 u0 com.youku.tv/com.youku.tv.ui.activity.AllViedosActivity}

mFocusedApp=AppWindowToken{4277fca8 token=Token{427490e8 ActivityRecord{4277a408 u0 com.youku.tv/.ui.activity.AllViedosActivity}}}

mInTouchMode=true mLayoutSeq=8017

## 其他

4、查看后台services信息：

adb shell service list

5、 adb shell dumpsys notification

获取通知信息

## REF

[dumpsys命令用法](http://gityuan.com/2016/05/14/dumpsys-command/)

# dumpsys SurfaceFlinger

用于查看叠加画面显示的层数。用法：

dumpsys SurfaceFlinger [--list、--latency、--latency-clear]

dumpsys SurfaceFlinger后面的参数通过-h，查询不到，我是在SurfaceFlinger服务的dump实现查看到。

用法

dumpsys SurfaceFlinger --list

com.android.systemui.ImageWallpaper

**com.tencent.mm/com.tencent.mm.plugin.webview.ui.tools.WebViewUI**

...

第二行即为显示最前面的应用程序。当出现黑屏的时候，可以快速定位目前运行的是哪个应用程序。

SurfaceFlinger的dump信息详解

## 特殊宏的打开

一般dump的第一行都是这样的:

1. Build configuration: [sf] [libui] [libgui]

这说明其实没有打开任何特殊的宏,实际上,如果一下特殊宏打开,第一行log会打印出来:

1. FRAMEBUFFER\_FORCE\_FORMAT,HAS\_CONTEXT\_PRIORITY,NEVER\_DEFAULT\_TO\_ASYNC\_MODE,TARGET\_DISABLE\_TRIPLE\_BUFFERING,DONT\_USE\_FENCE\_SYNC

一般情况下,这些宏一个都不会打开.

## Sync机制

第二行一般是这样的:

1. Sync configuration: [using: EGL\_ANDROID\_native\_fence\_sync EGL\_KHR\_wait\_sync]

这行其实打印了目前使用的sync机制,这个值源于这段逻辑:

1. if (useNativeFenceSync()) {
2. mString.append(" EGL\_ANDROID\_native\_fence\_sync");
3. }
4. if (useFenceSync()) {
5. mString.append(" EGL\_KHR\_fence\_sync");
6. }
7. if (useWaitSync()) {
8. mString.append(" EGL\_KHR\_wait\_sync");
9. }

注意,一二是互斥的,三可以与一二共存.

## DispSync参数

第三行是打印的是Vsync相关的参数:

1. DispSync configuration: app phase 0 ns, sf phase 0 ns, present offset 0 ns (refresh 16666667 ns)

这些参数我们还是比较熟悉的,有意思的是打印这些参数时候使用的语法:

1. result.appendFormat("app phase %" PRId64 " ns, sf phase %" PRId64 " ns, "
2. "present offset %d ns (refresh %" PRId64 " ns)",
3. vsyncPhaseOffsetNs, sfVsyncPhaseOffsetNs, PRESENT\_TIME\_OFFSET\_FROM\_VSYNC\_NS,
4. mHwc->getRefreshPeriod(HWC\_DISPLAY\_PRIMARY));

PRId64的用法很独特,这是一种跨平台的打印64位整数的做法:

1. printf("%" PRId64 "\n", value);
2. *// 相当于64位的：*
3. printf("%" "ld" "\n", value);
4. *// 或32位的：*
5. printf("%" "lld" "\n", value);

## layer的dump

接下来就是很长的一段layer的dump,一般以这样一句话开始:

1. Visible layers (count = 9)

count的值来源于layersSortedByZ中layer的数量.   
接下来就进入各个layer的dump,我们参考代码并以launcher所在的layer为例来解释下各行的意义:

1. + Layer 0xb3f92000 (com.sec.android.app.launcher/com.android.launcher2.Launcher) id=87

0xb3f92000指向当前layer对象的值,括号中是当前layer的名称,id是创建layer时产生的序列号.

### 区域信息

1. Region transparentRegion (this=0xb3f92164, count=1)
2. [ 0, 0, 0, 0]
3. Region visibleRegion (this=0xb3f92008, count=1)
4. [ 0, 0, 1440, 2560]

接下来的两段是两个Region的dump,每个region可能包含多个区域,所以这里count也可能不等于1.   
前两行的值来源于activeTransparentRegion,表示的是这个layer里面透明区域的大小.   
后两行值来源于visibleRegion,表示可见区域的大小.

### 基本信息

1. layerStack= 0, z= 21010, pos=(0,0), size=(1440,2560), crop=(0, 0,1440,2560), isOpaque=0,
2. invalidate=0, alpha=0xff, flags=0x00000000, tr=[1.00, 0.00][0.00, 1.00]
3. client=0xb11160c0

上面这段dump源自这段代码:

1. result.appendFormat( " "
2. "layerStack=%4d, z=%9d, pos=(%g,%g), size=(%4d,%4d), crop=(%4d,%4d,%4d,%4d), "
3. "isOpaque=%1d, invalidate=%1d, "
4. "alpha=0x%02x, flags=0x%08x, tr=[%.2f, %.2f][%.2f, %.2f]\n"
5. " client=%p\n",
6. s.layerStack, s.z, s.transform.tx(), s.transform.ty(), s.active.w, s.active.h,
7. s.active.crop.left, s.active.crop.top,
8. s.active.crop.right, s.active.crop.bottom,
9. isOpaque(s), contentDirty,
10. s.alpha, s.flags,
11. s.transform[0][0], s.transform[0][1],
12. s.transform[1][0], s.transform[1][1],
13. client.get());

* layerStack表示这个layer是保存在哪个layerstack中(不同的display是有不同的layerstack的,这点可以通过一个连接HDMI时的layerstack很容易确认).
* z表示Z轴坐标,z值越大,layer越靠上.
* pos的值是layer左上角的位置,这个值比较特殊的是ImageWallpaper这个layer的pos值,因为ImageWallpaper的大小大于屏幕大小,所以ImageWallpaper的pos值在屏幕的外面(note4是pos=(-560,0)).
* size自然是layer的大小
* crop代表裁剪区域,这点依然是对于壁纸很明显,因为壁纸layer大小大于屏幕,必须涉及到需要裁剪一部分显示在屏幕上,因此它的裁剪区域是crop=( 560, 0,2000,2560).
* isOpaque代表是否是不透明的,只有完全不透明的layer这个值才是1,比如壁纸,像状态栏和launcher他们都是0,代表不是完全不透明的
* invalidate表示这个layer的数据是失效的,这个值绝大多数情况下都是0.因为我们看到的一般都是绘制好的有效的数据.一种情况下这值特别频繁的多见为1,就是刚刚锁屏(解锁)时.因为突然锁屏,会导致绘制的内容和要显示的内容完全不同,导致layer的各种数据要重新计算,所以将layer置为失效.
* alpha表示了这张layer的透明度,这个值跟isOpaque是有区别的.isOpaque表示了这个layer可以是透明的,也就是没有显示数据的地方,可以透明;而alpha表示透明度,也即是有数据的地方也可以因为透明度而收到影响产生透明的效果.
* flag值含义丰富,它是众多flag或出来的结果,影响它值的包括:

1. enum {
2. eLayerHidden = 0x01, *// SURFACE\_HIDDEN in SurfaceControl.java*
3. eLayerOpaque = 0x02, *// SURFACE\_OPAQUE*
4. eLayerTransparent = 0x200, *// SURFACE\_TRANSPARENT*
5. };
6. enum {
7. ePositionChanged = 0x00000001,
8. eLayerChanged = 0x00000002,
9. eSizeChanged = 0x00000004,
10. eAlphaChanged = 0x00000008,
11. eMatrixChanged = 0x00000010,
12. eTransparentRegionChanged = 0x00000020,
13. eVisibilityChanged = 0x00000040,
14. eLayerStackChanged = 0x00000080,
15. eCropChanged = 0x00000100,
16. */\* SRIB : Smg Surface Animator : State that will indicate animation change \*/*
17. e3DAnimationChanged = 0x00001000,
18. */\* SRIB : Smg Surface Animator : Change End\*/*
19. eOpacityChanged = 0x00000200,
20. *// { SRUK-SFBLUR*
21. eTranslucentRegionChanged = 0x00000400,
22. *// SRUK-SFBLUR }*
23. eTransparencyChanged = 0x80000000,
24. };
25. enum { *// (keep in sync with Surface.java)*
26. eHidden = 0x00000004,
27. eDestroyBackbuffer = 0x00000020,
28. eSecure = 0x00000080,
29. eNonPremultiplied = 0x00000100,
30. eOpaque = 0x00000400,
31. eProtectedByApp = 0x00000800,
32. eProtectedByDRM = 0x00001000,
33. eCursorWindow = 0x00002000,
34. */\* SISO Changes for Internal\_Only - Start \*/*
35. eFXInternalDisplay = 0x80000000,
36. */\* SISO Changes for Internal\_Only - End \*/*
37. eFXSurfaceNormal = 0x00000000,
38. eFXSurfaceDim = 0x00020000,
39. eFXSurfaceMask = 0x000F0000,
40. *// begin of app fw : fixed orientation window*
41. eFixedOrientation = 0x40000000,
42. *// end of app fw*
43. *// begin of MDM remote control*
44. eNoRemoteControl = 0x08000000,
45. *// end of MDM remote control*
46. };

所有的这些值都可能影响layer的状态,涉及不同模块不同功能,这里不再展开.

* 接下来的一组tr数据代表屏幕的旋转和缩放程度.大多数的layer实际上是不需要旋转和缩放的,因为他们定义的大小就是跟屏幕一致的,所以他们的这组数据是[1.00, 0.00][0.00, 1.00],实际上如果你使用这组数据来做矩阵变换的话,矩阵是不会发生变化的.   
  需要旋转的比较典型的场景是照相机.横着拿相机时它的layer的变换矩阵是[-1.00, 0.00][-0.00, -1.00],也就是旋转180°.   
  这个值的来源是上层调用setMatrix函数设置的.
* client含义比较简单,值的来源是创建layer时,对应的SurfaceSession中mNativeClient.这东西也是跟SurfaceSession一一对应的,也就是跟SurfaceFlinger连接时一一对应的.从这个值我们可以判断,client值相同的layer,必然来自同一个进程(因为他们是由同一个连接创建出来的).

### 4.3 buffer信息

1. format= 1, activeBuffer=[1440x2560:1664, 1], queued-frames=0, mRefreshPending=0
2. mTexName=38 mCurrentTexture=2
3. mCurrentCrop=[0,0,0,0] mCurrentTransform=0
4. mAbandoned=0
5. -BufferQueue mMaxAcquiredBufferCount=1, mDequeueBufferCannotBlock=0,
6. default-size=[1440x2560], default-format=1, transform-hint=00, FIFO(0)={}
7. [00:0xb110e100] state=FREE , 0xb3eb1ec0 [1440x2560:1664, 1]
8. [01:0xb3ec7000] state=FREE , 0xb620d060 [1440x2560:1664, 1]
9. >[02:0xb110e200] state=ACQUIRED, 0xb1111100 [1440x2560:1664, 1]

#### 4.3.1 数据格式

首先是数据的format,值的来源是layer创建时赋予的,当然我们如果追溯的话,可以追溯到WindowManagerService创建SurfaceControl的过程,值也是创建时指定的.值的定义如下:

1. enum {
2. *//*
3. *// these constants need to match those*
4. *// in graphics/PixelFormat.java & pixelflinger/format.h*
5. *//*
6. PIXEL\_FORMAT\_UNKNOWN = 0,
7. PIXEL\_FORMAT\_NONE = 0,
8. *// logical pixel formats used by the SurfaceFlinger -----------------------*
9. PIXEL\_FORMAT\_CUSTOM = -4,
10. *// Custom pixel-format described by a PixelFormatInfo structure*
11. PIXEL\_FORMAT\_TRANSLUCENT = -3,
12. *// System chooses a format that supports translucency (many alpha bits)*
13. PIXEL\_FORMAT\_TRANSPARENT = -2,
14. *// System chooses a format that supports transparency*
15. *// (at least 1 alpha bit)*
16. PIXEL\_FORMAT\_OPAQUE = -1,
17. *// System chooses an opaque format (no alpha bits required)*
18. *// real pixel formats supported for rendering -----------------------------*
19. PIXEL\_FORMAT\_RGBA\_8888 = HAL\_PIXEL\_FORMAT\_RGBA\_8888, *// 4x8-bit RGBA*
20. PIXEL\_FORMAT\_RGBX\_8888 = HAL\_PIXEL\_FORMAT\_RGBX\_8888, *// 4x8-bit RGB0*
21. PIXEL\_FORMAT\_RGB\_888 = HAL\_PIXEL\_FORMAT\_RGB\_888, *// 3x8-bit RGB*
22. PIXEL\_FORMAT\_RGB\_565 = HAL\_PIXEL\_FORMAT\_RGB\_565, *// 16-bit RGB*
23. PIXEL\_FORMAT\_BGRA\_8888 = HAL\_PIXEL\_FORMAT\_BGRA\_8888, *// 4x8-bit BGRA*
24. PIXEL\_FORMAT\_RGBA\_5551 = 6, *// 16-bit ARGB*
25. PIXEL\_FORMAT\_RGBA\_4444 = 7, *// 16-bit ARGB*
26. PIXEL\_FORMAT\_sRGB\_A\_8888 = HAL\_PIXEL\_FORMAT\_sRGB\_A\_8888, *// 4x8-bit sRGB + A*
27. PIXEL\_FORMAT\_sRGB\_X\_8888 = HAL\_PIXEL\_FORMAT\_sRGB\_X\_8888, *// 4x8-bit sRGB, no A*
28. };

其实只有下面的值是真实可用的,其余值在SurfaceFlinger创建时会被转换:

1. switch (format) {
2. case PIXEL\_FORMAT\_TRANSPARENT:
3. case PIXEL\_FORMAT\_TRANSLUCENT:
4. format = PIXEL\_FORMAT\_RGBA\_8888;
5. break;
6. case PIXEL\_FORMAT\_OPAQUE:
7. format = PIXEL\_FORMAT\_RGBX\_8888;
8. break;
9. }

其实当前常见的format也就是这几种.

1. HAL\_PIXEL\_FORMAT\_RGBA\_8888 = 1,
2. HAL\_PIXEL\_FORMAT\_RGBX\_8888 = 2,
3. HAL\_PIXEL\_FORMAT\_RGB\_888 = 3,
4. HAL\_PIXEL\_FORMAT\_RGB\_565 = 4,
5. HAL\_PIXEL\_FORMAT\_BGRA\_8888 = 5,

0代表未知格式.   
常见的layer中,dimlayer一般是0,大多数layer是1,壁纸是2,照相机的预览数据是4,视频播放也是4.

#### 4.3.2 activeBuffer

* activeBuffer的前两项表示了当前正在显示的buffer的宽和高.
* 第三项表示Stride.这个值很有意思,我们发现他有时候是等于宽的,有时候是大于宽的,我们先来看下这个值的解释.   
  The number of *pixels* that a line in the buffer takes in memory. This may be >= width.   
  我们知道内存申请使用是需要成块对齐的,也就是说不是说使我们申请多大的内存,就会给我们多大的内存,因为涉及到对齐,所以很可能这个内存实际上是大于我们的需要的.(暂时没有仔细研究,有待确认)像有些marvell型号,内存是按照64位对齐的,那么我们申请一个100宽的buffer,系统就会给我们留出128的buffer大小供我们使用.
* 第四项并没有什么特殊,表示format,跟前面的format应该是一致的.

#### 4.3.3 queued-frames 新的帧的数量

queued-frames的含义是是否有新的帧,如果当前没有新的帧,这个值是0.   
一般在画面持续变化时(照相预览,视频播放,窗口滑动,游戏),这个值会是1.表示有新的一帧.   
偶尔也可以见到这个值是2(这个值应该最大就是2,因为只有三个缓冲区).

#### 4.3.4 mRefreshPending刷新卡住了吗?

mRefreshPending几乎所有的常见情况下都是0,因为这个参数代表了一个layer执行了Invalidate却没有完成Refresh,除非发生错误这显然不可能.

### 4.4 SurfaceFlingerConsumer的dump

接下来开始对消费者进行dump,SurfaceFlingerConsumer是GLConsumer子类,所以这里实际上是调用了GLConsumer的dumpLocked函数.   
先来看下代码:

1. result.appendFormat(
2. "%smTexName=%d mCurrentTexture=%d\n"
3. "%smCurrentCrop=[%d,%d,%d,%d] mCurrentTransform=%#x\n",
4. prefix, mTexName, mCurrentTexture, prefix, mCurrentCrop.left,
5. mCurrentCrop.top, mCurrentCrop.right, mCurrentCrop.bottom,
6. mCurrentTransform);

它会对应打印出来这一段信息:

1. mTexName=38 mCurrentTexture=2
2. mCurrentCrop=[0,0,0,0] mCurrentTransform=0

#### 4.4.1 材质名称

mTexName的值来源是在消费者被创建时,我们知道最常见的创建消费者的时候是Layer::onFirstRef时会调用:

1. mSurfaceFlingerConsumer = new SurfaceFlingerConsumer(consumer, mTextureName);

创建一个消费者,有两个参数,其中mTextureName是我们目前关注的,如果追溯来源你会发现mTextureName的值来源于glGenTextures,这个函数的实现依赖平台,参考ligagl,它是这样的:

1. *// generate unique (shared) texture names*
2. c->surfaceManager->getToken(n, textures);

还是继续回来看SurfaceFlingerConsumer的创建:

1. SurfaceFlingerConsumer(const sp<IGraphicBufferConsumer>& consumer,
2. uint32\_t tex)
3. : GLConsumer(consumer, tex, GLConsumer::TEXTURE\_EXTERNAL, false, false),
4. mTransformToDisplayInverse(false)
5. GLConsumer::GLConsumer(const sp<IGraphicBufferConsumer>& bq, uint32\_t tex,
6. uint32\_t texTarget, bool useFenceSync, bool isControlledByApp) :
7. ConsumerBase(bq, isControlledByApp),
8. mCurrentTransform(0),
9. mCurrentScalingMode(NATIVE\_WINDOW\_SCALING\_MODE\_FREEZE),
10. mCurrentFence(Fence::NO\_FENCE),
11. mCurrentTimestamp(0),
12. mCurrentFrameNumber(0),
13. mDefaultWidth(1),
14. mDefaultHeight(1),
15. mFilteringEnabled(true),
16. mTexName(tex),
17. mUseFenceSync(useFenceSync),
18. mTexTarget(texTarget),
19. mEglDisplay(EGL\_NO\_DISPLAY),
20. mEglContext(EGL\_NO\_CONTEXT),
21. mCurrentTexture(BufferQueue::INVALID\_BUFFER\_SLOT),
22. mAttached(true)

我们现在可以看出来mTexName的值来源于前面创建的材质名称.   
mCurrentTexture的初始值是INVALID\_BUFFER\_SLOT,也就是-1,后面会在updateAndReleaseLocked时被更改,值的来源是使用的BufferItem的mBuf值,也就是mSlot,这应该是使用的buffer数组的slot值,这个变量的合理取值只有0,1,2三个值(mSlot is the slot index of this buffer ,default INVALID\_BUFFER\_SLOT).

#### 4.4.2 mCurrentCrop裁剪区域

mCurrentCrop的值来源同样是updateAndReleaseLocked调用时被赋值,值的来源是BufferItem的mCrop值.这个值基本一直都是0,只有在视频播放和照相机时会被设置(值的来源有待更深入的研究, mCrop is the current crop rectangle for this buffer slot).

#### 4.4.3 mCurrentTransform 旋转相关

mCurrentTransform的值和前面我们说过的tr值很类似. (mTransform is the current transform flags for this buffer slot. refer to NATIVE\_WINDOW\_TRANSFORM\_\* in ).   
它也跟旋转有关,我们来看下window.h中的定义:

1. */\* parameter for NATIVE\_WINDOW\_SET\_BUFFERS\_TRANSFORM \*/*
2. enum {
3. */\* flip source image horizontally \*/*
4. NATIVE\_WINDOW\_TRANSFORM\_FLIP\_H = HAL\_TRANSFORM\_FLIP\_H ,
5. */\* flip source image vertically \*/*
6. NATIVE\_WINDOW\_TRANSFORM\_FLIP\_V = HAL\_TRANSFORM\_FLIP\_V,
7. */\*rotate source image 90 degrees clock-wise, is applied after TRANSFORM\_FLIP\_{H|V} \*/*
8. NATIVE\_WINDOW\_TRANSFORM\_ROT\_90 = HAL\_TRANSFORM\_ROT\_90,
9. */\* rotate source image 180 degrees \*/*
10. NATIVE\_WINDOW\_TRANSFORM\_ROT\_180 = HAL\_TRANSFORM\_ROT\_180,
11. */\* rotate source image 270 degrees clock-wise \*/*
12. NATIVE\_WINDOW\_TRANSFORM\_ROT\_270 = HAL\_TRANSFORM\_ROT\_270,
13. */\* transforms source by the inverse transform of the screen it is displayed onto. This*
14. *\* transform is applied last \*/*
15. NATIVE\_WINDOW\_TRANSFORM\_INVERSE\_DISPLAY = 0x08
16. };
17. enum {
18. */\* flip source image horizontally (around the vertical axis) \*/*
19. HAL\_TRANSFORM\_FLIP\_H = 0x01,
20. */\* flip source image vertically (around the horizontal axis)\*/*
21. HAL\_TRANSFORM\_FLIP\_V = 0x02,
22. */\* rotate source image 90 degrees clockwise \*/*
23. HAL\_TRANSFORM\_ROT\_90 = 0x04,
24. */\* rotate source image 180 degrees \*/*
25. HAL\_TRANSFORM\_ROT\_180 = 0x03,
26. */\* rotate source image 270 degrees clockwise \*/*
27. HAL\_TRANSFORM\_ROT\_270 = 0x07,
28. */\* don't use. see system/window.h \*/*
29. HAL\_TRANSFORM\_RESERVED = 0x08,
30. };

### 4.5 ConsumerBase(消费者)的dump

子类GLConsumer dump完毕,调用了它的父类的dump函数,基本就是调用了IGraphicBufferConsumer的dump函数.   
生产者消费者这套体系我们之前以前讲过,这里我们就不再详细展开,如果不清楚看下Layer::onFirstRef这个函数就明白了,消费者这个值来自于BufferQueue::createBufferQueue的创建,其中创建了新的BufferQueueConsumer做为消费者.

子类GLConsumer dump完毕,调用了它的父类的dump函数,基本就是调用了IGraphicBufferConsumer的dump函数.   
生产者消费者这套体系我们之前以前讲过,这里我们就不再详细展开,如果不清楚看下Layer::onFirstRef这个函数就明白了,消费者这个值来自于BufferQueue::createBufferQueue的创建,其中创建了新的BufferQueueConsumer做为消费者.

1. void BufferQueue::createBufferQueue(sp<IGraphicBufferProducer>\* outProducer,
2. sp<IGraphicBufferConsumer>\* outConsumer,
3. const sp<IGraphicBufferAlloc>& allocator) {
4. sp<BufferQueueCore> core(new BufferQueueCore(allocator));
5. sp<IGraphicBufferProducer> producer(new BufferQueueProducer(core));
6. sp<IGraphicBufferConsumer> consumer(new BufferQueueConsumer(core));
7. \*outProducer = producer;
8. \*outConsumer = consumer;

当然BufferQueueConsumer的dump函数啥也没写,就调用了BufferQueueCore的dump函数.

打印出来的信息一般是这样的:

1. -BufferQueue mMaxAcquiredBufferCount=1, mDequeueBufferCannotBlock=0,
2. default-size=[1440x2560], default-format=1, transform-hint=00, FIFO(0)={}
3. [00:0xb110e100] state=FREE , 0xb3eb1ec0 [1440x2560:1664, 1]
4. [01:0xb3ec7000] state=FREE , 0xb620d060 [1440x2560:1664, 1]
5. >[02:0xb110e200] state=ACQUIRED, 0xb1111100 [1440x2560:1664, 1]

下面我们按照代码顺序详细解释一下:

#### 4.5.1 队列中的buffer

我们之前在解释queued-frames的含义时已经说过,在画面持续变化时(照相预览,视频播放,窗口滑动,游戏),queued-frames值会是1.表示有新的一帧.   
对应的,有新的frames自然需要有在队列中等待的buffer,对应这段代码会把这个队列打印出来:

1. Fifo::const\_iterator current(mQueue.begin());
2. while (current != mQueue.end()) {
3. fifo.appendFormat("%02d:%p crop=[%d,%d,%d,%d], "
4. "xform=0x%02x, time=%#" PRIx64 ", scale=%s\n",
5. current->mSlot, current->mGraphicBuffer.get(),
6. current->mCrop.left, current->mCrop.top, current->mCrop.right,
7. current->mCrop.bottom, current->mTransform, current->mTimestamp,
8. BufferItem::scalingModeName(current->mScalingMode));
9. ++current;
10. }

对应打印出来的dump信息是这样的:

1. 02:0xb631e480 crop=[0,0,0,0], xform=0x07, time=0xc4d5da9b1e0, scale=FREEZE

* 02是mSlot的值,crop是裁剪区域,xform是旋转,这三个我们上面已经讲过,这里不再展开.
* time是这个buffer被queue的时间(mTimestamp is the current timestamp for this buffer slot. This gets to set by queueBuffer each time this slot is queued. This value is guaranteed to be monotonically increasing for each newly acquired buffer.).
* scale是缩放模式,一般取值如下:

1. enum {
2. */\* the window content is not updated (frozen) until a buffer of*
3. *\* the window size is received (enqueued)*
4. *\*/*
5. NATIVE\_WINDOW\_SCALING\_MODE\_FREEZE = 0,
6. */\* the buffer is scaled in both dimensions to match the window size \*/*
7. NATIVE\_WINDOW\_SCALING\_MODE\_SCALE\_TO\_WINDOW = 1,
8. */\* the buffer is scaled uniformly such that the smaller dimension*
9. *\* of the buffer matches the window size (cropping in the process)*
10. *\*/*
11. NATIVE\_WINDOW\_SCALING\_MODE\_SCALE\_CROP = 2,
12. */\* the window is clipped to the size of the buffer's crop rectangle; pixels*
13. *\* outside the crop rectangle are treated as if they are completely*
14. *\* transparent.*
15. *\*/*
16. NATIVE\_WINDOW\_SCALING\_MODE\_NO\_SCALE\_CROP = 3,
17. };

#### 4.5.2 BufferQueue的基本默认信息

接下来的一段代码会打印BufferQueue的一些基本信息:

1. result.appendFormat("%s-BufferQueue mMaxAcquiredBufferCount=%d, "
2. "mDequeueBufferCannotBlock=%d, default-size=[%dx%d], "
3. "default-format=%d, transform-hint=%02x, FIFO(%zu)={%s}\n",
4. prefix, mMaxAcquiredBufferCount, mDequeueBufferCannotBlock,
5. mDefaultWidth, mDefaultHeight, mDefaultBufferFormat, mTransformHint,
6. mQueue.size(), fifo.string());

一般会打印如下:

1. -BufferQueue mMaxAcquiredBufferCount=1, mDequeueBufferCannotBlock=0, default-size=[1920x1080],
2. default-format=4, transform-hint=04

##### 4.5.2.1 允许同时acquire的buffer的数量

mMaxAcquiredBufferCount是允许同时acquire的buffer的数量,解释如下:

1. *// mMaxAcquiredBufferCount is the number of buffers that the consumer may*
2. *// acquire at one time. It defaults to 1, and can be changed by the consumer*
3. *// via setMaxAcquiredBufferCount, but this may only be done while no*
4. *// producer is connected to the BufferQueue. This value is used to derive*
5. *// the value returned for the MIN\_UNDEQUEUED\_BUFFERS query to the producer.*

基本这个值只能是1,不再深究.

##### 4.5.2.2 dequeueBuffer是否允许被block

1. *// mDequeueBufferCannotBlock indicates whether dequeueBuffer is allowed to*
2. *// block. This flag is set during connect when both the producer and*
3. *// consumer are controlled by the application.*
4. bool mDequeueBufferCannotBlock;

mDequeueBufferCannotBlock几乎总是为0,除非一个应用同时控制了生产者和消费者,这很罕见.

##### 4.5.2.3 buffer default-size默认buffer大小

这两个值的来源应该是BufferQueueConsumer::setDefaultBufferSize函数(不是特别确定,因为这段代码写的不好,严重破坏了封装性).

用处是这样的: mDefaultHeight holds the default height of allocated buffers. It is used in dequeueBuffer if a width and height of 0 are specified.

##### 4.5.2.4 mDefaultBufferFormat默认格式

mDefaultBufferFormat很简单,format含义可以参考前面的解释.   
mDefaultBufferFormat can be set so it will override the buffer format when it isn’t specified in dequeueBuffer.

##### 4.5.2.5 mTransformHint

同样用于旋转.   
mTransformHint is the transform probably applied to buffers of this window. this is only a hint, actual transform may differ.

#### 4.5.3 各个Buffer的信息

接下来是打印BufferSlot中各个buffer的信息,一般打印如下:

1. [00:0xb651d780] state=QUEUED , 0xb6321240 [1080x1920:1152, 1]
2. [01:0xb1513200] state=FREE , 0xb65189c0 [1080x1920:1152, 1]
3. >[02:0xb651d080] state=ACQUIRED, 0xb6518330 [1080x1920:1152, 1]

是由下面的代码打印出来的.

1. for (int s = 0; s < maxBufferCount; ++s) {
2. const BufferSlot& slot(mSlots[s]);
3. const sp<GraphicBuffer>& buffer(slot.mGraphicBuffer);
4. result.appendFormat("%s%s[%02d:%p] state=%-8s", prefix,
5. (slot.mBufferState == BufferSlot::ACQUIRED) ? ">" : " ",
6. s, buffer.get(),
7. BufferSlot::bufferStateName(slot.mBufferState));
8. if (buffer != NULL) {
9. result.appendFormat(", %p [%4ux%4u:%4u,%3X]", buffer->handle,
10. buffer->width, buffer->height, buffer->stride,
11. buffer->format);
12. }
13. result.append("\n");
14. }
15. }

ACQUIRED的buffer前面会打印>,表示这是当前在显示的buffer.   
state表示buffer的状态,取值包括DEQUEUED,QUEUED,FREE,ACQUIRED.我们知道ACQUIRED是在显示的,DEQUEUED是在绘制的,QUEUED绘制完成还未显示的,free是未使用的.   
后面的大小,stride,和format前面都讲过了,这里不再说明.

至此,layer的dump已经说明完毕,我们继续分析Displays的dump.

## Displays的dump.

### 设备名称

首先DisplayDevice是设备的名字,这个可以调用接口设置,但是比较常见的值一般有:Built-in Screen,HDMI Screen,Virtual Screen,wfdservice等等.

### 设备类型

type则是一个枚举值:

1. enum DisplayType {
2. DISPLAY\_ID\_INVALID = -1,
3. DISPLAY\_PRIMARY = HWC\_DISPLAY\_PRIMARY,
4. DISPLAY\_EXTERNAL = HWC\_DISPLAY\_EXTERNAL,
5. DISPLAY\_VIRTUAL = HWC\_DISPLAY\_VIRTUAL,
6. NUM\_BUILTIN\_DISPLAY\_TYPES = HWC\_NUM\_PHYSICAL\_DISPLAY\_TYPES,
7. };
8. enum {
9. HWC\_DISPLAY\_PRIMARY = 0,
10. HWC\_DISPLAY\_EXTERNAL = 1, *// HDMI, DP, etc.*
11. HWC\_DISPLAY\_VIRTUAL = 2, *// wfdservice*
12. HWC\_NUM\_PHYSICAL\_DISPLAY\_TYPES = 2,
13. HWC\_NUM\_DISPLAY\_TYPES = 3,
14. };

### layerStack

layerStack是存储layer的容器,我们知道每个display只会有一个layerstack来存储他要显示的layer,但是不同的display可以使用同一个layerStack,也可以使用不同的layerStack.   
上面我们贴的这个就是两个display使用了不同的layerstack,因为他们显示的内容不一样(电视播放幻灯片).  
后续我们可以研究下什么情况下会导致layerstack切换.

### 屏幕方向

* orient表示屏幕方向
* 后面括号里面的type,是和我们上面说的设备类型完全不同的东西,这个值是由Transform::type算出来的.   
  基本是下面这些值与或非出来的:

1. enum type\_mask {
2. IDENTITY = 0,
3. TRANSLATE = 0x1,
4. ROTATE = 0x2,
5. SCALE = 0x4,
6. UNKNOWN = 0x8
7. };

### powerMode

powerMode表示了屏幕当前的状态,它有以下取值:

1. enum {
2. */\* The display is turned off (blanked). \*/*
3. HWC\_POWER\_MODE\_OFF = 0,
4. */\* The display is turned on and configured in a low power state*
5. *\* that is suitable for presenting ambient information to the user,*
6. *\* possibly with lower fidelity than normal but greater efficiency. \*/*
7. HWC\_POWER\_MODE\_DOZE = 1,
8. */\* The display is turned on normally. \*/*
9. HWC\_POWER\_MODE\_NORMAL = 2,
10. */\* The display is configured as in HWC\_POWER\_MODE\_DOZE but may*
11. *\* stop applying frame buffer updates from the graphics subsystem.*
12. *\* This power mode is effectively a hint from the doze dream to*
13. *\* tell the hardware that it is done drawing to the display for the*
14. *\* time being and that the display should remain on in a low power*
15. *\* state and continue showing its current contents indefinitely*
16. *\* until the mode changes.*
17. *\**
18. *\* This mode may also be used as a signal to enable hardware-based doze*
19. *\* functionality. In this case, the doze dream is effectively*
20. *\* indicating that the hardware is free to take over the display*
21. *\* and manage it autonomously to implement low power always-on display*
22. *\* functionality. \*/*
23. HWC\_POWER\_MODE\_DOZE\_SUSPEND = 3,
24. };

常见的取值有0和2,代表屏幕熄灭和普通情况.   
目前还没看到1和3的情况.

### 其他一些参数

* 设备大小由eglQuerySurface得来,不展开.
* ANativeWindow代表要渲染的本地窗口,这个不同的display之间应该肯定不同.
* flips代表屏幕翻页的次数,其实也就是SurfaceFlinger调用doComposition的次数,也就是屏幕画面更新的次数
* hwcId需要注意的是,如果一个设备不是HWC合成的,这个值会是负数.需要指出的是,这个值不受开关overlay的影响,也就是说如果这个设备是支持HWC的,应该就不会是负数.目前来看,只有开发者选项模拟二级显示出现的display这个会是负数.
* mIsSecure是屏幕自身的属性,mSecureLayerVisible应该会跟播放DRM之类的场景相关
* activeConfig目前看到的总是0,还不清楚作用
* numLayers是这个display上可见的layer的数量
* v,f,s分别代表三个大小:Viewport,Frame,Scissor.

## 5555 display信息的dump

首先会打印当前display的数量,数量基于mDisplays的大小,这个容器在SurfaceFlinger初始化时会生成数据,后面根据收到不同的消息在handleTransactionLocked函数中也会调整.   
正常情况下是1,也就是只有一个display(Built-in Screen),当设备连接了HDMI或者使用了屏幕共享等功能时,会有额外的display加入,比如下面这个:

1. Displays (2 entries)
2. + DisplayDevice: HDMI Screen
3. type=1, hwcId=1, layerStack=6, (1920x1080), ANativeWindow=0xb4d94d08, orient= 0 (type=00000000), flips=1173, isSecure=1,
4. secureVis=0, powerMode=2, activeConfig=0, numLayers=1
5. v:[0,0,1920,1080], f:[0,0,1920,1080], s:[0,0,1920,1080],transform:[[1.000,0.000,-0.000][0.000,1.000,-0.000][0.000,0.000,1.000]]
6. mAbandoned=0
7. -BufferQueue mMaxAcquiredBufferCount=2, mDequeueBufferCannotBlock=0, default-size=[1920x1080], default-format=1, transform-hint=00,
8. FIFO(0)={}
9. [00:0xb6418c80] state=FREE , 0xb43ed880 [1920x1080:1920, 1]
10. [01:0xb43cb300] state=FREE , 0xb640d970 [1920x1080:1920, 1]
11. >[02:0xb43cb280] state=ACQUIRED, 0xb43ed830 [1920x1080:1920, 1]
12. + DisplayDevice: Built-in Screen
13. type=0, hwcId=0, layerStack=0, (1080x1920), ANativeWindow=0xb4d94608, orient= 0 (type=00000000), flips=3140, isSecure=1,
14. secureVis=0, powerMode=2, activeConfig=0, numLayers=2
15. v:[0,0,1080,1920], f:[0,0,1080,1920], s:[0,0,1080,1920],transform:[[1.000,0.000,-0.000][0.000,1.000,-0.000][0.000,0.000,1.000]]

这个是连接了HDMI后的数据.

## 7 EventThread的dump

EventThread的dump信息:

1. VSYNC state: disabled
2. soft-vsync: disabled
3. numListeners=33,
4. events-delivered: 18546

## 8 HWC的dump

HWC的dump从这句话开始:

1. h/w composer state:
2. h/w composer present and enabled

其中present和enable与否由以下参数决定:

1. result.appendFormat(" h/w composer %s and %s\n",
2. hwc.initCheck()==NO\_ERROR ? "present" : "not present",
3. (mDebugDisableHWC || mDebugRegion || mDaltonize
4. || mHasColorMatrix) ? "disabled" : "enabled");

只要init成功,就是present;而是否enable则由众多debug选项决定.   
mDebugDisableHWC我们很熟悉,就是我们在开发者选项里勾选的禁用硬件叠加.   
mDebugRegion是开发者选项里面的开发者选项中的显示屏幕更新,也可以通过属性debug.sf.showupdates来控制.   
mDaltonize被1014号命令控制,暂时先不研究.   
mHasColorMatrix被1015号命令控制.

上述四个选项有任意一个打开,就composer的状态就会使disable.

接下来进入HWComposer的dump函数:   
- 首先打印的是hwc的version,目前一般0103或者0104居多.   
- mDebugForceFakeVSync的值源自属性debug.sf.no\_hw\_vsync,打开之后会使用纯软件模拟VSync信号.   
- 接下来是打印各个display的一些基本信息:

1. Display[0] configurations (\* current):
2. \* 0: 1440x2560, xdpi=508.000000, ydpi=516.062988, secure=1 refresh=16666667

这些基本信息取自对应display设备的DisplayData的config信息.

接下来就开始打印各个layer的信息,这个也是我们最常见到的layer信息.   
依然是分display打印的,分别打印每个display的可见layer,通过getVisibleLayersSortedByZ获取的.

1. type | handle | hint | flag | tr | blnd | format | source crop(l,t,r,b) | frame | dirtyRect | name
2. ------------+----------+----------+----------+----+-------+----------+-----------------------------------+---------------------------+-------------------
3. HWC | aed1c650 | 0002 | 0000 | 00 | 0100 | ? 00000011 | 0.0, 0.0, 1920.0, 1080.0 | 0, 0, 960, 540 | [ 0, 0, 1920, 1080] | SurfaceView
4. HWC | aed1c470 | 0000 | 0000 | 00 | 0105 | RGBA\_8888 | 0.0, 0.0, 2560.0, 1440.0 | 0, 0, 960, 540 | [ 0, 0, 2560, 1440] | SurfaceView
5. FB TARGET | b3ec5240 | 0000 | 0000 | 00 | 0105 | RGBA\_8888 | 0.0, 0.0, 960.0, 540.0 | 0, 0, 960, 540 | [ 0, 0, 0, 0] | HWC\_FRAMEBUFFER\_TARGET

逐次来看下:   
- type表示合成类型,可取的值包括以下几种: “GLES”,”HWC”,”BKGND”, “FB TARGET”,”SIDEBAND”,”HWC\_CURSOR”,”FB\_BLIT”,”UNKNOWN”.(更详细的一些解释可以参见hwcomposer.h)   
- handle是个标识符,可以跟某个buffer的handle对应.(Handle of buffer to compose. This handle is guaranteed to have been allocated from gralloc using the GRALLOC\_USAGE\_HW\_COMPOSER usage flag. If the layer’s handle is unchanged across two consecutive prepare calls and the HWC\_GEOMETRY\_CHANGED flag is not set for the second call then the HWComposer implementation may assume that the contents of the buffer have not changed. )   
- hint is bit mask set by the HWC implementation during (\*prepare)().It is preserved between (\*prepare)() calls, unless the HWC\_GEOMETRY\_CHANGED flag is set, in which case it is reset to 0.   
下面是hint的可能取值.

1. */\**
2. *\* hwc\_layer\_t::hints values*
3. *\* Hints are set by the HAL and read by SurfaceFlinger*
4. *\*/*
5. enum {
6. */\**
7. *\* HWC can set the HWC\_HINT\_TRIPLE\_BUFFER hint to indicate to SurfaceFlinger*
8. *\* that it should triple buffer this layer. Typically HWC does this when*
9. *\* the layer will be unavailable for use for an extended period of time,*
10. *\* e.g. if the display will be fetching data directly from the layer and*
11. *\* the layer can not be modified until after the next set().*
12. *\*/*
13. HWC\_HINT\_TRIPLE\_BUFFER = 0x00000001,
14. */\**
15. *\* HWC sets HWC\_HINT\_CLEAR\_FB to tell SurfaceFlinger that it should clear the*
16. *\* framebuffer with transparent pixels where this layer would be.*
17. *\* SurfaceFlinger will only honor this flag when the layer has no blending*
18. *\**
19. *\*/*
20. HWC\_HINT\_CLEAR\_FB = 0x00000002
21. };

* flag的常见取值只有0和1,1代表这个layer由SF处理.如果我们关闭硬件叠加,可以看到layer的这个值都是1.都是这个值不能跟type画等号,在layer众多的时候,layer都是GLES类型,但是只有很少的layer的flag是1(原因有待研究,1的详细解释是这样的:HWC\_SKIP\_LAYER is set by SurfaceFlnger to indicate that the HAL shall not consider this layer for composition as it will be handled by SurfaceFlinger ).
* tr是transform的缩写,但是注意这里的tr和前面单独layer dump时的tr或transform的值这不能划等号的,这里的tr的含义是transformation to apply to the buffer during composition.
* blnd的含义如下:

1. enum {
2. */\* no blending \*/*
3. HWC\_BLENDING\_NONE = 0x0100,
4. */\* ONE / ONE\_MINUS\_SRC\_ALPHA \*/*
5. HWC\_BLENDING\_PREMULT = 0x0105,
6. */\* SRC\_ALPHA / ONE\_MINUS\_SRC\_ALPHA \*/*
7. HWC\_BLENDING\_COVERAGE = 0x0405
8. };

* format前面说过很多次了,这里不再说了.

## REF

[SurfaceFlinger的dump信息详解](http://lee_3do.leanote.com/post/SurfaceFlinger%E7%9A%84dump%E4%BF%A1%E6%81%AF%E8%AF%A6%E8%A7%A3)

[引发的问题：如果只是简单的将视频小窗口层输出到video层，壁纸和桌面混合输出到Graphic层，就会导致video被完全遮挡，因为video层在最底下。](http://www.chongchonggou.com/g_492198516.html)

# 3.性能分析

## Top

root@zs600b:/ # top -m 10

-t表示查看线程

CTRL+C结束统计，