|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D:\02-资料分类\04-文档模板\扉页图1-new.jpg | | | | |
|  | AndroidR\_Native内存泄漏调试指南 | | |  |
| 文档版本 V1.0 |  | |
| 发布日期 2020-07-17 |  | |
|  | | | | |
|  | 紫光展锐科技有限公司 | |  |  |

|  |
| --- |
| 版权所有 © 紫光展锐科技有限公司。保留一切权利。  本文件所含数据和信息都属于紫光展锐所有的机密信息，紫光展锐保留所有相关权利。本文件仅为信息参考之目的提供，不包含任何明示或默示的知识产权许可，也不表示有任何明示或默示的保证，包括但不限于满足任何特殊目的、不侵权或性能。当您接受这份文件时，即表示您同意本文件中内容和信息属于紫光展锐机密信息，且同意在未获得紫光展锐书面同意前，不使用或复制本文件的整体或部分，也不向任何其他方披露本文件内容。紫光展锐有权在未经事先通知的情况下，在任何时候对本文件做任何修改。紫光展锐对本文件所含数据和信息不做任何保证，在任何情况下，紫光展锐均不负责任何与本文件相关的直接或间接的、任何伤害或损失。  请参照交付物中说明文档对紫光展锐交付物进行使用，任何人对紫光展锐交付物的修改、定制化或违反说明文档的指引对紫光展锐交付物进行使用造成的任何损失由其自行承担。紫光展锐交付物中的性能指标、测试结果和参数等，均为在紫光展锐内部研发和测试系统中获得的，仅供参考，若任何人需要对交付物进行商用或量产，需要结合自身的软硬件测试环境进行全面的测试和调试。非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。 |



|  |
| --- |
| 紫光展锐科技有限公司 |

前 言

概述

本文简述android native层的内存泄漏的调试方法。

读者对象

本文档适用于需要在android native层调试的内存泄漏的同事，及其他对此感兴趣的同事。

缩略语

|  |  |  |
| --- | --- | --- |
| 缩略语 | 英文全名 | 中文解释 |
|  |  |  |
|  |  |  |

变更信息

| 文档版本 | 发布日期 | 作者 | 修改说明 |
| --- | --- | --- | --- |
| V1.0 | 2019-07-17 | Jasson.zhang | 初稿 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

关键字

Malloc debug、AndroidQ、内存泄漏。

目 录

[1 Bionic调试开关 4](#_Toc45894476)

[1.1 打开malloc debug开关 4](#_Toc45894477)

[1.2 开关参数介绍 5](#_Toc45894478)

[1.2.1 libc.debug.malloc.program 5](#_Toc45894479)

[1.2.2 libc.debug.malloc.options backtrace 5](#_Toc45894480)

[2 使用dumpheap获取调用栈 8](#_Toc45894481)

[2.1 用dumpheap抓取调用栈 8](#_Toc45894482)

[2.2 调用栈解析 9](#_Toc45894483)

[2.2.1 解析脚本 9](#_Toc45894484)

[2.2.2 使用方法 10](#_Toc45894485)

[2.2.3 使用举例 10](#_Toc45894486)

[2.3 调用栈分析举例 10](#_Toc45894487)

[3 使用dumpsys获取调用栈 12](#_Toc45894488)

[3.1 使用dumpsys调试native程序 12](#_Toc45894489)

[3.1.1 dumpsys超时补丁 12](#_Toc45894490)

[3.1.2 步骤 13](#_Toc45894491)

[3.2 以 media.player 为例 13](#_Toc45894492)

[3.2.1 相关命令 13](#_Toc45894493)

[3.2.2 调用栈举例 14](#_Toc45894494)

[3.3 支持dumpsys –m命令 15](#_Toc45894495)

[3.3.1 支持dumpsys -m 的补丁 15](#_Toc45894496)

[3.3.2 操作步骤 16](#_Toc45894497)

[3.3.3 调用栈举例 16](#_Toc45894498)

# Bionic调试开关

Android Bionic提供了检测Native代码内存泄漏的库，设置libc.debug.malloc属性后，程序调用malloc() 接口会被替换成leak\_malloc()接口，目的是分配更多的内存用来记录调用malloc的call\_back等信息，方便定位泄漏点，但该机制会影响性能，也不适合做到正式版本里。

Bionic提供了检测Native代码内存申请的库，设置libc.debug.malloc属性后，通过android自带的dumpheap命令获取

## 打开malloc debug开关

libc.debug.malloc

libc 中 malloc debug 的 backtrace 功能，可以在内存申请的时候，将申请内存的函数调用栈地址保存在buffer中。我们可以通过分析这些调用关系来查找内存泄漏点。

* 我们可以通过 setprop libc.debug.malloc.options backtrace 来打开这个backtrace功能。
* 我们可以通过setprop libc.debug.malloc.program 来指定应用。

例如，我们需要分析cameraserver 的Native层内存分配，我们可以使用以下命令行：

|  |
| --- |
| sudo adb root //获取root权限  adb shell stop  adb shell setprop libc.debug.malloc.options backtrace // 打开过滤功能  adb shell setprop libc.debug.malloc.program cameraserver //指定过滤模块  adb shell start |

## 开关参数介绍

我们主要是通过libc.debug.malloc.options backtrace 和 libc.debug.malloc.program 这两个设置，来打开开关，下面对这两个设置进行简单说明：

### libc.debug.malloc.program

如果只设置libc.debug.malloc.options backtrace，不设置libc.debug.malloc. program的情况下，会记录所有进程内存申请的 backtrace，极大降低系统响应速度。

因此我们可以根据需求，只打开需要检测的进程来减少记录的进程，提高提高系统响应速度。

相关应用，可以找system/bin/下的可执行文件的名称。

如果需要分析基于zygote程序，需要根据需要打开app\_process

|  |
| --- |
| adb shell setprop libc.debug.malloc.program app\_process  adb shell setprop libc.debug.malloc.program app\_process32  adb shell setprop libc.debug.malloc.program app\_process64 |

根据实际情况选择app\_process，app\_process32或者app\_process64。

由于Java应用都是通过app\_process运行，因此打开后会跟踪所有的java应用，由于java应用都运行在java虚拟机中，因此java虚拟机相关native层的操作都会被跟踪，应用打开和操作会变慢。

system\_srver也是基于zygote的程序，因此要分析system\_srver 安装上面方法根据实际情况设置app\_process 即可。

由于system\_srver中包含主要的系统功能，因此打开后可以感到系统明显变慢。

### libc.debug.malloc.options backtrace

libc.debug.malloc.options backtrace 后面可以跟参数，例如，libc.debug. malloc. options backtrace = 8 ，这个参数表示的是记录调用栈的级数，如果不跟参数，每次内存申请 backtrace 默认记录16级调用栈，最大可以设置为256。

例如：

|  |
| --- |
| 256 bytes ( 256 bytes \* 1 allocations )  #00 pc 00011680 /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::Parcel::continueWrite(unsigned int)+471)  #01 pc 000103a8 /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::IPCThreadState::IPCThreadState()+175)  #02 pc 00016700 /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::IPCThreadState::self()+35)  #03 pc 00017c16 /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::BpHwBinder::BpHwBinder(int)+109)  #04 pc 000183a6 /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::ProcessState::getStrongProxyForHandle(int)+57)  #05 pc 000153f6 /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::ProcessState::getContextObject(android::sp<android::hardware::IBinder> const&)+3)  #06 pc 0001e668 /system/lib/vndk-sp-28/libhidltransport.so (android::hardware::defaultServiceManager1\_1()+219)  #07 pc 0001fb3c /system/lib/vndk-sp-28/libhidltransport.so (android::hardware::details::getRawServiceInternal(std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char>> const&, std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char>> const&, bool, bool)+31)  #08 pc 0001127a /system/lib/vndk-28/android.hardware.camera.provider@2.4.so (\_ZN7android8hardware7details18getServiceInternalINS0\_6camera8provider4V2\_418BpHwCameraProviderENS5\_15ICameraProviderEvvEENS\_2spIT0\_EERKNSt3\_\_112basic\_stringIcNSB\_11char\_traitsIcEENSB\_9allocatorIcEEEEbb+129)  #09 pc 000113c2 /system/lib/vndk-28/android.hardware.camera.provider@2.4.so (android::hardware::camera::provider::V2\_4::ICameraProvider::getService(std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char>> const&, bool)+5)  #10 pc 00000af2 /vendor/bin/hw/android.hardware.camera.provider@2.4-service  #11 pc 00000a82 /vendor/bin/hw/android.hardware.camera.provider@2.4-service  #12 pc 0008c78c /system/lib/libc.so (\_\_libc\_init+47)  #13 pc 000009f0 /vendor/bin/hw/android.hardware.camera.provider@2.4-service  #14 pc 00000000 <unknown>  #15 pc 00000000 <unknown> |

打开 backtrace 后，会影响性能，使操作变慢，我们可以通过减少记录层级来提高响应速度。

比如我们设置为只记录8级调用栈

|  |
| --- |
| setprop libc.debug.malloc.options backtrace = 8 |

那么上面的调用栈就会变为

|  |
| --- |
| 256 bytes ( 256 bytes \* 1 allocations )  #00 pc 00011680 /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::Parcel::continueWrite(unsigned int)+471)  #01 pc 000103a8 /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::IPCThreadState::IPCThreadState()+175)  #02 pc 00016700 /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::IPCThreadState::self()+35)  #03 pc 00017c16 /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::BpHwBinder::BpHwBinder(int)+109)  #04 pc 000183a6 /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::ProcessState::getStrongProxyForHandle(int)+57)  #05 pc 000153f6 /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::ProcessState::getContextObject(android::sp<android::hardware::IBinder> const&)+3)  #06 pc 0001e668 /system/lib/vndk-sp-28/libhidltransport.so (android::hardware::defaultServiceManager1\_1()+219)  #07 pc 0001fb3c /system/lib/vndk-sp-28/libhidltransport.so (android::hardware::details::getRawServiceInternal(std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char>> const&, std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char>> const&, bool, bool)+31) |

有时根据调试的需要，我们也可以记录更多级的调用栈，比如下面这个调用栈这种情况，

|  |
| --- |
| 9488 bytes ( 9488 bytes \* 1 allocations )  #00 pc 00003bd4 /system/lib/vndk-28/libcamera\_metadata.so (clone\_camera\_metadata+43)  #01 pc 000a4252 /vendor/lib/hw/camera.sp9832e.so (android::hardware::camera::common::V1\_0::helper::CameraMetadata::operator=(camera\_metadata const\*)+37)  #02 pc 00069b14 /vendor/lib/hw/camera.sp9832e.so (sprdcamera::SprdCamera3Setting::getStaticMetadata(int, camera\_metadata\*\*)+55)  #03 pc 0008266e /vendor/lib/hw/camera.sp9832e.so (sprdcamera::SprdCamera3PageTurn::getCameraInfo(camera\_info\*)+37)  #04 pc 000825d6 /vendor/lib/hw/camera.sp9832e.so (sprdcamera::SprdCamera3PageTurn::get\_camera\_info(int, camera\_info\*)+29)  #05 pc 0001e976 /vendor/lib/hw/android.hardware.camera.provider@2.4-impl-sprd.so (android::hardware::camera::common::V1\_0::helper::CameraModule::getCameraInfo(int, camera\_info\*)+273)  #06 pc 0001eb50 /vendor/lib/hw/android.hardware.camera.provider@2.4-impl-sprd.so (android::hardware::camera::common::V1\_0::helper::CameraModule::getDeviceVersion(int)+51)  #07 pc 0000fe3e /vendor/lib/camera.device@3.2-impl-sprd.so (android::hardware::camera::device::V3\_2::implementation::CameraDevice::CameraDevice(android::sp<android::hardware::camera::common::V1\_0::helper::CameraModule>, std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char>> const&, android::SortedVector<std::\_\_1::pair<std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char>>, std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char>>>> const&)+205)  #08 pc 00004996 /vendor/lib/camera.device@3.3-impl-sprd.so (android::hardware::camera::device::V3\_3::implementation::CameraDevice::CameraDevice(android::sp<android::hardware::camera::common::V1\_0::helper::CameraModule>, std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char>> const&, android::SortedVector<std::\_\_1::pair<std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char>>, std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char>>>> const&)+61)  #09 pc 00013a0e /vendor/lib/hw/android.hardware.camera.provider@2.4-impl-sprd.so (android::hardware::camera::provider::V2\_4::implementation::CameraProvider::getCameraDeviceInterface\_V3\_x(android::hardware::hidl\_string const&, std::\_\_1::function<void (android::hardware::camera::common::V1\_0::Status, android::sp<android::hardware::camera::device::V3\_2::ICameraDevice> const&)>)+677)  #10 pc 00010b22 /system/lib/vndk-28/android.hardware.camera.provider@2.4.so (android::hardware::camera::provider::V2\_4::BnHwCameraProvider::\_hidl\_getCameraDeviceInterface\_V3\_x(android::hidl::base::V1\_0::BnHwBase\*, android::hardware::Parcel const&, android::hardware::Parcel\*, std::\_\_1::function<void (android::hardware::Parcel&)>)+209)  #11 pc 00010eaa /system/lib/vndk-28/android.hardware.camera.provider@2.4.so (android::hardware::camera::provider::V2\_4::BnHwCameraProvider::onTransact(unsigned int, android::hardware::Parcel const&, android::hardware::Parcel\*, unsigned int, std::\_\_1::function<void (android::hardware::Parcel&)>)+497)  #12 pc 0001656c /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::BHwBinder::transact(unsigned int, android::hardware::Parcel const&, android::hardware::Parcel\*, unsigned int, std::\_\_1::function<void (android::hardware::Parcel&)>)+51)  #13 pc 0000f73e /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::IPCThreadState::executeCommand(int)+805)  #14 pc 0000f30a /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::IPCThreadState::getAndExecuteCommand()+121)  #15 pc 0000fbc6 /system/lib/vndk-sp-28/libhwbinder.so (android::hardware::IPCThreadState::joinThreadPool(bool)+189) |

如果我们想要看到更早的调用栈，就需要增加调用栈级别。

例如：如我们记录64级调用栈。

|  |
| --- |
| setprop libc.debug.malloc.options backtrace = 64 |

需要注意的是，记录更多级的调用栈，会导致响应速度，变得更慢。

# 使用dumpheap获取调用栈

对于基于zygote的进程，google 提供了 dumpheap 命令进行抓取native 代码的调用栈。

## 用dumpheap抓取调用栈

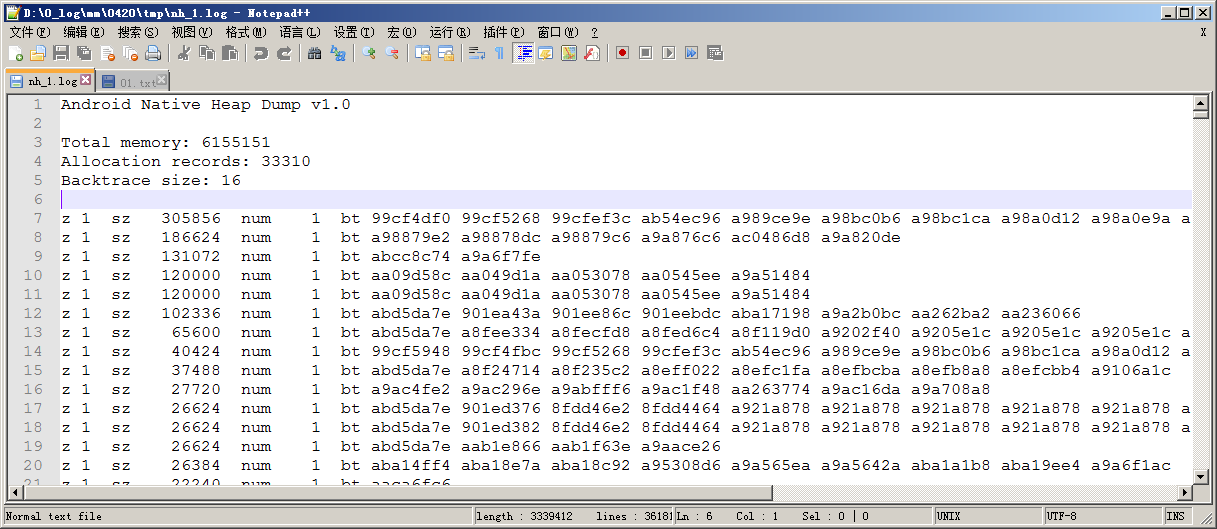
我们首先要打开libc.debug.malloc.options，将其配置为backtrace，方法如前面章节，这里不重复介绍，打开malloc 的 debug 开关之后，我们可以通过am dumpheap命令进行dump

|  |
| --- |
| adb shell am dumpheap -n <PID\_TO\_DUMP> /data/local/tmp/nativeheap.txt |

其中：

* –n ：参数表示dump Native栈
* <PID\_TO\_DUMP> : 表示要dump的进程的 PID
* /data/local/tmp/nativeheap.txt : 表示将dump出的栈所放置的文件

下面这个是dumpheap命令来抓取到的 nativeheap。



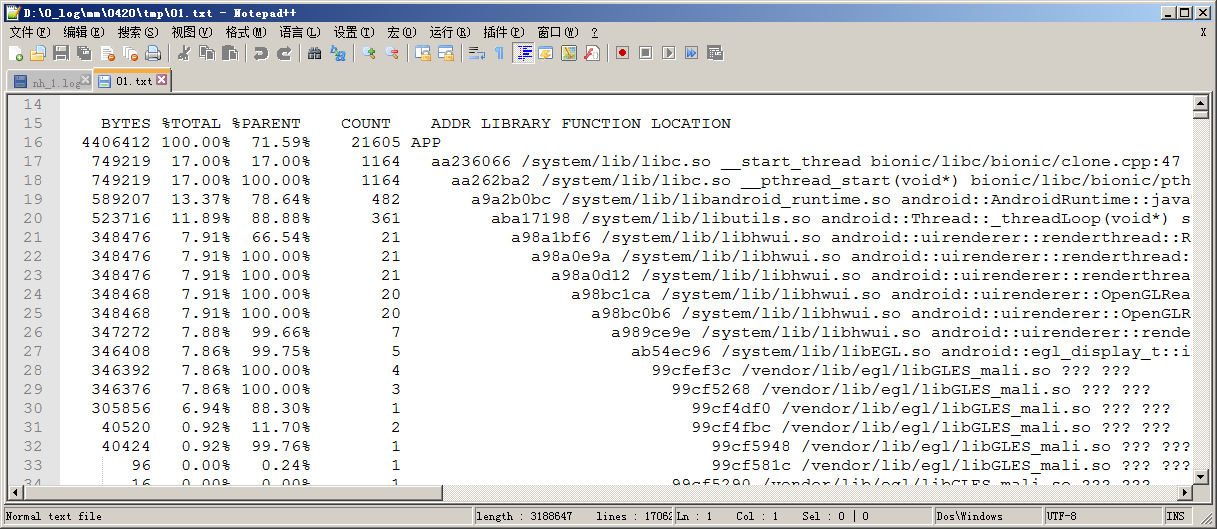
## 调用栈解析

### 解析脚本

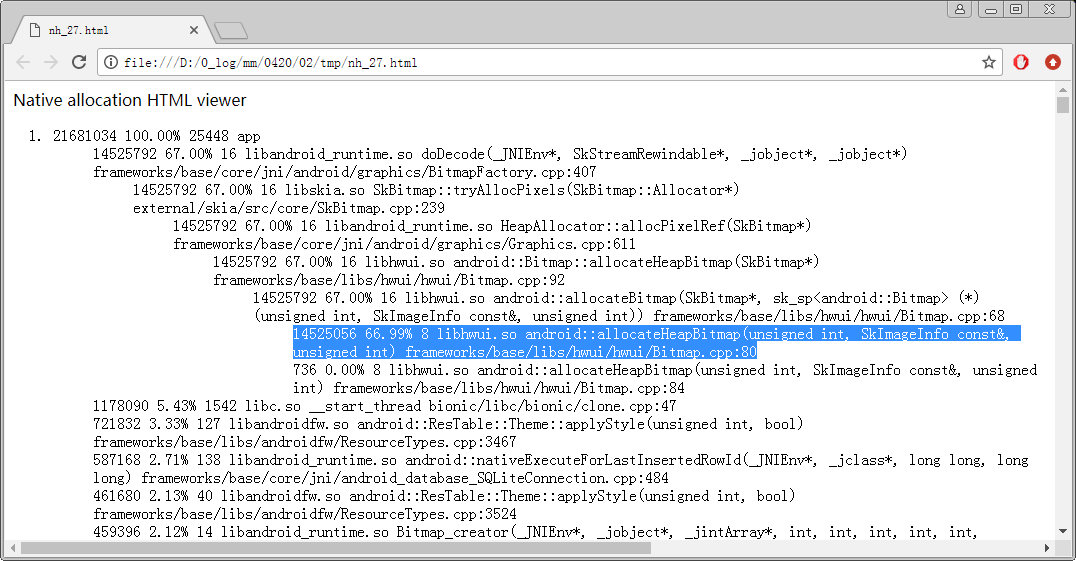
dumpheap命令抓取到的 nativeheap，可以通过google原生脚本native\_heapdump\_viewer.py解析出调用栈。

脚本放在development/scripts/native\_heapdump\_viewer.py

该脚本可以将其解析成文本格式。



也能将其转换成更易读的html格式



### 使用方法

比较简单的方法是将 脚本和 待解析log 拷贝到 symbols 同级目录

|  |
| --- |
| $ tree  ├── **native\_heapdump\_viewer.py** // 脚本  ├── heap.txt // 待解析log  └── **symbols** // 该版本对应的symbols  └── system  ├── bin  ├── etc  ├── fake-libs  ├── lib  └── xbin |

然后直接执行：

python native\_heapdump\_viewer.py 待解析文件 > 输出文件

这里需要注意：需要使用 python2.7 的版本，实测发现使用python3会报错。

### 使用举例

举例解析成文本格式

|  |
| --- |
| $ python native\_heapdump\_viewer.py heap.txt > heap\_dec.txt |

举例解析成html格式

|  |
| --- |
| $ python native\_heapdump\_viewer.py --html heap.txt > heap\_dec.html |

## 调用栈分析举例

通过解析结果分析 native层泄漏

首先，我们使用dumpheap命令每半小时抓取相关进程（如system\_server）的 NativeDumpHeap，

接着，使用native\_heapdump\_viewer.py 脚本逐一解析。

然后，我们对解析后的调用栈进行比对：

|  |
| --- |
| 9261256 100.00% 27128 app  2073692 22.39% 2 libandroid\_runtime.so doDecode(\_JNIEnv\*, SkStreamRewindable\*, \_jobject\*, \_jobject\*) frameworks/base/core/jni/android/graphics/BitmapFactory.cpp:407  2073692 22.39% 2 libskia.so SkBitmap::tryAllocPixels(SkBitmap::Allocator\*) external/skia/src/core/SkBitmap.cpp:239  2073692 22.39% 2 libandroid\_runtime.so HeapAllocator::allocPixelRef(SkBitmap\*) frameworks/base/core/jni/android/graphics/Graphics.cpp:611  2073692 22.39% 2 libhwui.so android::Bitmap::allocateHeapBitmap(SkBitmap\*) frameworks/base/libs/hwui/hwui/Bitmap.cpp:92  2073692 22.39% 2 libhwui.so android::allocateBitmap(SkBitmap\*, sk\_sp<android::Bitmap> (\*)(unsigned int, SkImageInfo const&, unsigned int)) frameworks/base/libs/hwui/hwui/Bitmap.cpp:68  **2073600 22.39% 1** libhwui.so android::allocateHeapBitmap(unsigned int, SkImageInfo const&, unsigned int) frameworks/base/libs/hwui/hwui/Bitmap.cpp:80  10322433 100.00% 24975 app  4147384 40.18% 4 libandroid\_runtime.so doDecode(\_JNIEnv\*, SkStreamRewindable\*, \_jobject\*, \_jobject\*) frameworks/base/core/jni/android/graphics/BitmapFactory.cpp:407  4147384 40.18% 4 libskia.so SkBitmap::tryAllocPixels(SkBitmap::Allocator\*) external/skia/src/core/SkBitmap.cpp:239  4147384 40.18% 4 libandroid\_runtime.so HeapAllocator::allocPixelRef(SkBitmap\*) frameworks/base/core/jni/android/graphics/Graphics.cpp:611  4147384 40.18% 4 libhwui.so android::Bitmap::allocateHeapBitmap(SkBitmap\*) frameworks/base/libs/hwui/hwui/Bitmap.cpp:92  4147384 40.18% 4 libhwui.so android::allocateBitmap(SkBitmap\*, sk\_sp<android::Bitmap> (\*)(unsigned int, SkImageInfo const&, unsigned int)) frameworks/base/libs/hwui/hwui/Bitmap.cpp:68  4147200 40.18% 2 libhwui.so android::allocateHeapBitmap(unsigned int, SkImageInfo const&, unsigned int) frameworks/base/libs/hwui/hwui/Bitmap.cpp:80  15891498 100.00% 24885 app  8294768 52.20% 8 libandroid\_runtime.so doDecode(\_JNIEnv\*, SkStreamRewindable\*, \_jobject\*, \_jobject\*) frameworks/base/core/jni/android/graphics/BitmapFactory.cpp:407  8294768 52.20% 8 libskia.so SkBitmap::tryAllocPixels(SkBitmap::Allocator\*) external/skia/src/core/SkBitmap.cpp:239  8294768 52.20% 8 libandroid\_runtime.so HeapAllocator::allocPixelRef(SkBitmap\*) frameworks/base/core/jni/android/graphics/Graphics.cpp:611  8294768 52.20% 8 libhwui.so android::Bitmap::allocateHeapBitmap(SkBitmap\*) frameworks/base/libs/hwui/hwui/Bitmap.cpp:92  8294768 52.20% 8 libhwui.so android::allocateBitmap(SkBitmap\*, sk\_sp<android::Bitmap> (\*)(unsigned int, SkImageInfo const&, unsigned int)) frameworks/base/libs/hwui/hwui/Bitmap.cpp:68  8294400 52.19% 4 libhwui.so android::allocateHeapBitmap(unsigned int, SkImageInfo const&, unsigned int) frameworks/base/libs/hwui/hwui/Bitmap.cpp:80  24331394 100.00% 27265 app  10368460 42.61% 10 libandroid\_runtime.so doDecode(\_JNIEnv\*, SkStreamRewindable\*, \_jobject\*, \_jobject\*) frameworks/base/core/jni/android/graphics/BitmapFactory.cpp:407  10368460 42.61% 10 libskia.so SkBitmap::tryAllocPixels(SkBitmap::Allocator\*) external/skia/src/core/SkBitmap.cpp:239  10368460 42.61% 10 libandroid\_runtime.so HeapAllocator::allocPixelRef(SkBitmap\*) frameworks/base/core/jni/android/graphics/Graphics.cpp:611  10368460 42.61% 10 libhwui.so android::Bitmap::allocateHeapBitmap(SkBitmap\*) frameworks/base/libs/hwui/hwui/Bitmap.cpp:92  10368460 42.61% 10 libhwui.so android::allocateBitmap(SkBitmap\*, sk\_sp<android::Bitmap> (\*)(unsigned int, SkImageInfo const&, unsigned int)) frameworks/base/libs/hwui/hwui/Bitmap.cpp:68  10368000 42.61% 5 libhwui.so android::allocateHeapBitmap(unsigned int, SkImageInfo const&, unsigned int) frameworks/base/libs/hwui/hwui/Bitmap.cpp:80  22538843 100.00% 25687 app  14515844 64.40% 14 libandroid\_runtime.so doDecode(\_JNIEnv\*, SkStreamRewindable\*, \_jobject\*, \_jobject\*) frameworks/base/core/jni/android/graphics/BitmapFactory.cpp:407  14515844 64.40% 14 libskia.so SkBitmap::tryAllocPixels(SkBitmap::Allocator\*) external/skia/src/core/SkBitmap.cpp:239  14515844 64.40% 14 libandroid\_runtime.so HeapAllocator::allocPixelRef(SkBitmap\*) frameworks/base/core/jni/android/graphics/Graphics.cpp:611  14515844 64.40% 14 libhwui.so android::Bitmap::allocateHeapBitmap(SkBitmap\*) frameworks/base/libs/hwui/hwui/Bitmap.cpp:92  14515844 64.40% 14 libhwui.so android::allocateBitmap(SkBitmap\*, sk\_sp<android::Bitmap> (\*)(unsigned int, SkImageInfo const&, unsigned int)) frameworks/base/libs/hwui/hwui/Bitmap.cpp:68  **14515200 64.40% 7** libhwui.so android::allocateHeapBitmap(unsigned int, SkImageInfo const&, unsigned int) frameworks/base/libs/hwui/hwui/Bitmap.cpp:80  21681034 100.00% 25448 app  14525792 67.00% 16 libandroid\_runtime.so doDecode(\_JNIEnv\*, SkStreamRewindable\*, \_jobject\*, \_jobject\*) frameworks/base/core/jni/android/graphics/BitmapFactory.cpp:407  14525792 67.00% 16 libskia.so SkBitmap::tryAllocPixels(SkBitmap::Allocator\*) external/skia/src/core/SkBitmap.cpp:239  14525792 67.00% 16 libandroid\_runtime.so HeapAllocator::allocPixelRef(SkBitmap\*) frameworks/base/core/jni/android/graphics/Graphics.cpp:611  14525792 67.00% 16 libhwui.so android::Bitmap::allocateHeapBitmap(SkBitmap\*) frameworks/base/libs/hwui/hwui/Bitmap.cpp:92  14525792 67.00% 16 libhwui.so android::allocateBitmap(SkBitmap\*, sk\_sp<android::Bitmap> (\*)(unsigned int, SkImageInfo const&, unsigned int)) frameworks/base/libs/hwui/hwui/Bitmap.cpp:68  14525056 66.99% 8 libhwui.so android::allocateHeapBitmap(unsigned int, SkImageInfo const&, unsigned int) frameworks/base/libs/hwui/hwui/Bitmap.cpp:80 |

通过上面的log我们可以看出，随着时间的增长，该调用栈的数量从1个增长到7个，内存从，约2M增长到约14.5M，而其他调用栈内存没有增长，因此该调用栈内存占比由22%增长到67%。

接下来我们就可以针对该调用栈进行分析，分析其不断增长的原因。

# 使用dumpsys获取调用栈

对于Servicemanager启动的服务，我们可以通过dumpsys -m命令获取native层的调用栈

我们也可以通过抓log的方式进行离线的调试。

## 使用dumpsys调试native程序

我们可以通过设置Bionic 的debuglveve为 backtrace 然后使用dumpsys -m获取log 进行分析。

### dumpsys超时补丁

由于dump有10s超时限制，超时后会报错导致dump失败，所以在执行dumpsys之前需要将dumpsys的超时时间设置为-1,永不超时。

* 修改 frameworks/native/cmds/dumpsys/dumpsys.cpp 文件

之前的修改方法不同通过编译器的检验，已不适用。需要按照如下方法修改：

<http://review.source.unisoc.com/gerrit/#/c/691820/>

|  |
| --- |
| **/sprdroidr\_trunk/frameworks/native/cmds/dumpsys$ git diff**  diff --git a/cmds/dumpsys/dumpsys.cpp b/cmds/dumpsys/dumpsys.cpp  index a427c8d..e956e91 100644  --- a/cmds/dumpsys/dumpsys.cpp  +++ b/cmds/dumpsys/dumpsys.cpp  @@ -418,7 +418,7 @@ status\_t Dumpsys::writeDump(int fd, const String16& serviceName, std::chrono::mi  auto time\_left\_ms = [end]() {  auto now = std::chrono::steady\_clock::now();  auto diff = std::chrono::duration\_cast<std::chrono::milliseconds>(end - now);  - return std::max(diff.count(), 0LL);  + return std::min(diff.count(), -1LL);  };    int rc = TEMP\_FAILURE\_RETRY(poll(&pfd, 1, time\_left\_ms())); |

之后，将编译好的可执行文件dumpsys push到手机system/bin/下。

### 步骤

* 首先，根据模块实际情况合入前面dumpsys超时的补丁。
* 然后，按照之前的方法设置Bionic 的debuglveve为 backtrace
* 然后重启服务，之后通过dumpsys 打印调用栈

## 以 media.player 为例

对于mediaplayer，原生支持“dumpsys –m”操作，可以输出native memory dump，我们以mediaplayer为例讲述dumpsys –m的使用。

### 相关命令

|  |
| --- |
| //首先参照前面章节stop后打开backtrace开关并start之后，等待，直至启动完成出现画面  //以 media.player 为例，杀掉media.player 相关服务  $ **adb shell ps -A |grep mediaserver**  media 15530 1 57776 15852 binder\_thread\_read 0 S mediaserver  **$ adb shell kill -9 15530**  **$ adb shell ps -A |grep mediaserver**  media 17860 1 51396 15352 binder\_thread\_read 0 S mediaserver  //抓取media.player 的dump  **$ adb shell dumpsys media.player -m > dump.log**  Can't find service: media.player //出现该提示说明服务还没有完全启动成功，需要多等一会儿再试  **$ adb shell dumpsys media.player -m > dump.log**  **$ cat dump.log** |

### 调用栈举例

我们可以看到：打印的结果中，已经将相关的调用栈进行了解析。

|  |
| --- |
| $ adb shell dumpsys media.player -m > dump.log  $ cat dump.log  0 registered handlers:  No media recorder client  Files opened and/or mapped:  Dumping memory:  347739 bytes in 1804 allocations  ABI: 'arm'  15440 bytes ( 15440 bytes \* 1 allocations )  #00 pc 0002b4ee /apex/com.android.runtime/lib/bionic/libc.so (malloc+33)  #01 pc 0009b3e0 /apex/com.android.runtime/lib/bionic/libc.so  #02 pc 0009f6ac /apex/com.android.runtime/lib/bionic/libc.so  #03 pc 0009b81a /apex/com.android.runtime/lib/bionic/libc.so (localtime+25)  #04 pc 0000945e /system/lib/libresourcemanagerservice.so (android::ServiceLog::add(android::String8 const&)+77)  #05 pc 000054ca /system/lib/libresourcemanagerservice.so (android::ResourceManagerService::removeResource(int, long long, bool)+53)  #06 pc 000068a6 /system/lib/libresourcemanagerservice.so (android::ResourceManagerService::removeClient(int, long long)+29)  #07 pc 00095b30 /system/lib/libstagefright.so (android::MediaCodec::ResourceManagerServiceProxy::removeClient()+51)  #08 pc 0009d4ac /system/lib/libstagefright.so (android::MediaCodec::onMessageReceived(android::sp<android::AMessage> const&)+6039)  #09 pc 0000ee44 /system/lib/libstagefright\_foundation.so (android::AHandler::deliverMessage(android::sp<android::AMessage> const&)+23)  #10 pc 000113d6 /system/lib/libstagefright\_foundation.so (android::AMessage::deliver()+81)  #11 pc 0000f5ee /system/lib/libstagefright\_foundation.so (android::ALooper::loop()+525)  #12 pc 0000dd94 /system/lib/libutils.so (android::Thread::\_threadLoop(void\*)+303)  #13 pc 0000d84c /system/lib/libutils.so  #14 pc 000a9c9a /apex/com.android.runtime/lib/bionic/libc.so  #15 pc 00062b6c /apex/com.android.runtime/lib/bionic/libc.so  13928 bytes ( 6964 bytes \* 2 allocations )  #00 pc 0002b4ee /apex/com.android.runtime/lib/bionic/libc.so (malloc+33)  #01 pc 000352f4 /system/lib/libc++.so (operator new(unsigned int)+11)  #02 pc 000517fc /system/lib/libaudioclient.so (android::BnAudioFlingerClient::onTransact(unsigned int, android::Parcel const&, android::Parcel\*, unsigned int)+55)  #03 pc 000330cc /system/lib/libbinder.so (android::BBinder::transact(unsigned int, android::Parcel const&, android::Parcel\*, unsigned int)+143)  #04 pc 00038e96 /system/lib/libbinder.so (android::IPCThreadState::executeCommand(int)+625)  #05 pc 00038b66 /system/lib/libbinder.so (android::IPCThreadState::getAndExecuteCommand()+97)  #06 pc 000391f2 /system/lib/libbinder.so (android::IPCThreadState::joinThreadPool(bool)+37)  #07 pc 00000884 /system/bin/mediaserver  #08 pc 0005be8a /apex/com.android.runtime/lib/bionic/libc.so (\_\_libc\_init+65)  #09 pc 000007e4 /system/bin/mediaserver  #10 pc 00000000 <unknown>  2808 bytes ( 2808 bytes \* 1 allocations )  #00 pc 0002b4ee /apex/com.android.runtime/lib/bionic/libc.so (malloc+33)  #01 pc 000352f4 /system/lib/libc++.so (operator new(unsigned int)+11)  #02 pc 00037458 /system/lib/libcodec2\_vndk.so  #03 pc 00037c5c /system/lib/libcodec2\_vndk.so  #04 pc 00000000 <unknown> |

## 支持dumpsys –m命令

由于部分模块（例如：Cameraserver模块）默认不支持dumpsys -m 的操作，因此我们以Cameraserver为例讲述支持dumpsys -m 的方法。

### 支持dumpsys -m 的补丁

Cameraserver默认不支持dumpsys -m 的操作，需要通过修改代码实现该功能。

修改方法可参考gerrit：

<http://review.source.unisoc.com/gerrit/#/c/691899/>

|  |
| --- |
| [platform/frameworks/av.git] / services / camera / libcameraservice / CameraService.cpp  diff --git a/services/camera/libcameraservice/CameraService.cpp b/services/camera/libcameraservice/CameraService.cpp  index 399f292..eeb1106 100644 (file)  --- a/services/camera/libcameraservice/CameraService.cpp  +++ b/services/camera/libcameraservice/CameraService.cpp  @@ -27,7 +27,7 @@  #include <sys/types.h>  #include <inttypes.h>  #include <pthread.h>  -  +#include <mediautils/MemoryLeakTrackUtil.h>  #include <android/hardware/ICamera.h>  #include <android/hardware/ICameraClient.h>    @@ -3668,6 +3668,10 @@ status\_t CameraService::dump(int fd, const Vector<String16>& args) {  std::string s = info.ToString(/\*log\_contents\*/ true);  write(fd, s.c\_str(), s.size());  }  + } else if (args[i] == String16("-m")) {  + dprintf(fd,"\nDumping start");  + std::string s = dumpMemoryAddresses(100 /\* limit \*/);  + write(fd, s.c\_str(), s.size());  }  }  return NO\_ERROR; |

这里需要特别注意到的是：

头文件 MemoryLeakTrackUtil.h 位于mediautils 目录下，与之前版本目录不同。

### 操作步骤

操作步骤与前面相同

* 合入相关补丁后，
* stop后打开backtrace开关并start之后，等待，直至启动完成出现画面
* 然后重启Cameraserver服务，
* 之后使用adb shell dumpsys media.camera -m命令进行抓取

### 调用栈举例

如下dump 信息：

|  |
| --- |
| $ adb shell dumpsys media.camera –m  Dumping start541212 bytes in 1635 allocations  ABI: 'arm'  65056 bytes ( 32528 bytes \* 2 allocations )  #00 pc 00005f0e /system/lib/libcamera\_metadata.so (allocate\_camera\_metadata+33)  #01 pc 0002cbd0 /system/lib/libcamera\_client.so (android::CameraMetadata::resizeIfNeeded(unsigned int, unsigned int)+75)  #02 pc 0002ce3a /system/lib/libcamera\_client.so (android::CameraMetadata::updateImpl(unsigned int, void const\*, unsigned int)+153)  #03 pc 000bddec /system/lib/libcameraservice.so (android::CameraProviderManager::ProviderInfo::DeviceInfo3::addPreCorrectionActiveArraySize()+119)  #04 pc 000c2db2 /system/lib/libcameraservice.so (android::CameraProviderManager::ProviderInfo::DeviceInfo3::DeviceInfo3(std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&, unsigned long long, std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&, unsigned short, android::hardware::camera::common::V1\_0::CameraResourceCost const&, android::sp<android::CameraProviderManager::ProviderInfo>, std::\_\_1::vector<std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> >, std::\_\_1::allocator<std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > > > const&, android::sp<android::hardware::camera::device::V3\_2::ICameraDevice>)+685)  #05 pc 000c0776 /system/lib/libcameraservice.so (std::\_\_1::unique\_ptr<android::CameraProviderManager::ProviderInfo::DeviceInfo, std::\_\_1::default\_delete<android::CameraProviderManager::ProviderInfo::DeviceInfo> > android::CameraProviderManager::ProviderInfo::initializeDeviceInfo<android::CameraProviderManager::ProviderInfo::DeviceInfo3>(std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&, unsigned long long, std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&, unsigned short)+465)  #06 pc 000bfcae /system/lib/libcameraservice.so (android::CameraProviderManager::ProviderInfo::addDevice(std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&, android::hardware::camera::common::V1\_0::CameraDeviceStatus, std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> >\*)+493)  #07 pc 000bef2a /system/lib/libcameraservice.so (android::CameraProviderManager::ProviderInfo::initialize(android::sp<android::hardware::camera::provider::V2\_4::ICameraProvider>&, unsigned long long)+1749)  #08 pc 000b9e16 /system/lib/libcameraservice.so (android::CameraProviderManager::addProviderLocked(std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&)+277)  #09 pc 000b9c46 /system/lib/libcameraservice.so (android::CameraProviderManager::initialize(android::wp<android::CameraProviderManager::StatusListener>, android::CameraProviderManager::ServiceInteractionProxy\*)+237)  #10 pc 000a2896 /system/lib/libcameraservice.so (android::CameraService::enumerateProviders()+757)  #11 pc 000a2326 /system/lib/libcameraservice.so (android::CameraService::onFirstRef()+117)  #12 pc 000009ba /system/bin/cameraserver  #13 pc 000008ca /system/bin/cameraserver  #14 pc 0005be8a /apex/com.android.runtime/lib/bionic/libc.so (\_\_libc\_init+65)  #15 pc 00000870 /system/bin/cameraserver  32528 bytes ( 32528 bytes \* 1 allocations )  #00 pc 00005f0e /system/lib/libcamera\_metadata.so (allocate\_camera\_metadata+33)  #01 pc 0002cbd0 /system/lib/libcamera\_client.so (android::CameraMetadata::resizeIfNeeded(unsigned int, unsigned int)+75)  #02 pc 0002ce3a /system/lib/libcamera\_client.so (android::CameraMetadata::updateImpl(unsigned int, void const\*, unsigned int)+153)  #03 pc 000bddec /system/lib/libcameraservice.so (android::CameraProviderManager::ProviderInfo::DeviceInfo3::addPreCorrectionActiveArraySize()+119)  #04 pc 000c2db2 /system/lib/libcameraservice.so (android::CameraProviderManager::ProviderInfo::DeviceInfo3::DeviceInfo3(std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&, unsigned long long, std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&, unsigned short, android::hardware::camera::common::V1\_0::CameraResourceCost const&, android::sp<android::CameraProviderManager::ProviderInfo>, std::\_\_1::vector<std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> >, std::\_\_1::allocator<std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > > > const&, android::sp<android::hardware::camera::device::V3\_2::ICameraDevice>)+685)  #05 pc 000c0776 /system/lib/libcameraservice.so (std::\_\_1::unique\_ptr<android::CameraProviderManager::ProviderInfo::DeviceInfo, std::\_\_1::default\_delete<android::CameraProviderManager::ProviderInfo::DeviceInfo> > android::CameraProviderManager::ProviderInfo::initializeDeviceInfo<android::CameraProviderManager::ProviderInfo::DeviceInfo3>(std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&, unsigned long long, std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&, unsigned short)+465)  #06 pc 000bfcae /system/lib/libcameraservice.so (android::CameraProviderManager::ProviderInfo::addDevice(std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&, android::hardware::camera::common::V1\_0::CameraDeviceStatus, std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> >\*)+493)  #07 pc 000beec8 /system/lib/libcameraservice.so (android::CameraProviderManager::ProviderInfo::initialize(android::sp<android::hardware::camera::provider::V2\_4::ICameraProvider>&, unsigned long long)+1651)  #08 pc 000b9e16 /system/lib/libcameraservice.so (android::CameraProviderManager::addProviderLocked(std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&)+277)  #09 pc 000b9c46 /system/lib/libcameraservice.so (android::CameraProviderManager::initialize(android::wp<android::CameraProviderManager::StatusListener>, android::CameraProviderManager::ServiceInteractionProxy\*)+237)  #10 pc 000a2896 /system/lib/libcameraservice.so (android::CameraService::enumerateProviders()+757)  #11 pc 000a2326 /system/lib/libcameraservice.so (android::CameraService::onFirstRef()+117)  #12 pc 000009ba /system/bin/cameraserver  #13 pc 000008ca /system/bin/cameraserver  #14 pc 0005be8a /apex/com.android.runtime/lib/bionic/libc.so (\_\_libc\_init+65)  #15 pc 00000870 /system/bin/cameraserver  55264 bytes ( 27632 bytes \* 2 allocations )  #00 pc 00005f0e /system/lib/libcamera\_metadata.so (allocate\_camera\_metadata+33)  #01 pc 0002cbd0 /system/lib/libcamera\_client.so (android::CameraMetadata::resizeIfNeeded(unsigned int, unsigned int)+75)  #02 pc 0002ce3a /system/lib/libcamera\_client.so (android::CameraMetadata::updateImpl(unsigned int, void const\*, unsigned int)+153)  #03 pc 000bddec /system/lib/libcameraservice.so (android::CameraProviderManager::ProviderInfo::DeviceInfo3::addPreCorrectionActiveArraySize()+119)  #04 pc 000c2db2 /system/lib/libcameraservice.so (android::CameraProviderManager::ProviderInfo::DeviceInfo3::DeviceInfo3(std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&, unsigned long long, std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&, unsigned short, android::hardware::camera::common::V1\_0::CameraResourceCost const&, android::sp<android::CameraProviderManager::ProviderInfo>, std::\_\_1::vector<std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> >, std::\_\_1::allocator<std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > > > const&, android::sp<android::hardware::camera::device::V3\_2::ICameraDevice>)+685)  #05 pc 000c0776 /system/lib/libcameraservice.so (std::\_\_1::unique\_ptr<android::CameraProviderManager::ProviderInfo::DeviceInfo, std::\_\_1::default\_delete<android::CameraProviderManager::ProviderInfo::DeviceInfo> > android::CameraProviderManager::ProviderInfo::initializeDeviceInfo<android::CameraProviderManager::ProviderInfo::DeviceInfo3>(std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&, unsigned long long, std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&, unsigned short)+465)  #06 pc 000bfcae /system/lib/libcameraservice.so (android::CameraProviderManager::ProviderInfo::addDevice(std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&, android::hardware::camera::common::V1\_0::CameraDeviceStatus, std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> >\*)+493)  #07 pc 000bef2a /system/lib/libcameraservice.so (android::CameraProviderManager::ProviderInfo::initialize(android::sp<android::hardware::camera::provider::V2\_4::ICameraProvider>&, unsigned long long)+1749)  #08 pc 000b9e16 /system/lib/libcameraservice.so (android::CameraProviderManager::addProviderLocked(std::\_\_1::basic\_string<char, std::\_\_1::char\_traits<char>, std::\_\_1::allocator<char> > const&)+277)  #09 pc 000b9c46 /system/lib/libcameraservice.so (android::CameraProviderManager::initialize(android::wp<android::CameraProviderManager::StatusListener>, android::CameraProviderManager::ServiceInteractionProxy\*)+237)  #10 pc 000a2896 /system/lib/libcameraservice.so (android::CameraService::enumerateProviders()+757)  #11 pc 000a2326 /system/lib/libcameraservice.so (android::CameraService::onFirstRef()+117)  #12 pc 000009ba /system/bin/cameraserver  #13 pc 000008ca /system/bin/cameraserver  #14 pc 0005be8a /apex/com.android.runtime/lib/bionic/libc.so (\_\_libc\_init+65)  #15 pc 00000870 /system/bin/cameraserver |