

dumpsys实现原理



Hly_Coder (/u/183339cdc7ae) [+ 关注](#)

2016.08.18 16:20* 字数 1005 阅读 1837 评论 5 喜欢 7

(/u/183339cdc7ae)

转载请标明出处：http://www.jianshu.com/users/183339cdc7ae/latest_articles
(http://www.jianshu.com/users/183339cdc7ae/latest_articles)

概述

dumpsys是一个android手机里面的可执行文件。

从名字就可以看出，主要是用于dump 当前android system的一些信息。比如

```
activity (当前系统中所有activity的堆栈关系)
alarm (当前系统中所有的Alarms)
```

等等，是一项分析手机问题，运行状态，使用情况等十分有效的手段。

查看所支持的dump选项

```
adb shell dumpsys -l
```

会列出所有可以dump的选项，比如想获取所有window相关的信息，可以使用命令

```
adb shell dumpsys window
```

实现逻辑

既然是一个可执行文件，必然是先找到其mk文

件：/frameworks/native/cmds/dumpsys/Android.mk

```
...  
LOCAL_SRC_FILES:= \  
    dumpsys.cpp  
...  
LOCAL_MODULE:= dumpsys  
include $(BUILD_EXECUTABLE)
```

dumpsys的源码结构其实很简单，只有一个dumpsys.cpp

/frameworks/native/cmds/dumpsys/dumpsys.cpp

```
int main(int argc, char* const argv[])  
{  
    ...  
    sp<IServiceManager> sm = defaultServiceManager();  
    ...  
    Vector<String16> services;  
    ...  
    services = sm->listServices();  
    ...  
    const size_t N = services.size();  
  
    for (size_t i=0; i<N; i++) {  
        sp<IBinder> service = sm->checkService(services[i]);  
        ...  
        int err = service->dump(STDOUT_FILENO, args);  
        ...  
    }  
  
    return 0;  
}
```

先通过defaultServiceManager()函数获得ServiceManager对象，然后根据dumpsys传进来的参数通过函数checkService来找到具体的service, 并执行该service的dump方法，达到dump service的目的。

gfxinfo实例讲解



因为笔者最近在研究手机跑2D/3D场景的性能评测，所以这里以dumpsys **gfxinfo**为例，说下它的大致流程。

具体服务

由上文可以知道，dumpsys的实现其实是根据参数来找到某个具体的service，然后执行其dump方法。

我们熟悉的系统service有ActivityManagerService（activity），

WindowManagerService（window）等，那gfxinfo具体对应的是哪个service呢？

在文

件：`/frameworks/base/services/core/java/com/android/server/am/ActivityManagerService.java`中有下面这段代码

```
public void setSystemProcess() {  
    ...  
    ServiceManager.addService("gfxinfo", new GraphicsBinder(this));  
    ...  
}
```

在ams里面，通过ServiceManager添加了一个name为gfxinfo的service叫GraphicsBinder，该service就是gfxinfo对应的service。

```
static class GraphicsBinder extends Binder {  
    ...  
    @Override  
    protected void dump(FileDescriptor fd, PrintWriter pw, String[] args) {  
        if (mActivityManagerService.checkCallingPermission(android.Manifest.permission.  
            != PackageManager.PERMISSION_GRANTED) {  
            pw.println("Permission Denial: can't dump gfxinfo from pid=" +  
                + Binder.getCallingPid() + ", uid=" + Binder.getCallingUid() +  
                + " without permission " + android.Manifest.permission.DUMP)  
            return;  
        }  
        mActivityManagerService.dumpGraphicsHardwareUsage(fd, pw, args);  
    }  
}
```



这里有一个权限检查，如果app想要dump系统信息必须要配置
android.Manifest.permission.DUMP权限，而该权限是三方app没办法使用的。

那作为一个三方app，我们有办法**绕过**该权限检查吗？这里按下不表，接着往后看。

好吧，其实可以绕过。详情 (<http://www.jianshu.com/p/da38861f0fa5>)

ActivityThread

当我们执行adb shell dumpsys gfxinfo的时候，其实最后执行的是ams中的dumpGraphicsHardwareUsage该方法。

```
final void dumpGraphicsHardwareUsage(FileDescriptor fd,
    PrintWriter pw, String[] args) {
    ArrayList<ProcessRecord> procs = collectProcesses(pw, 0, false, args);
    ...
    for (int i = procs.size() - 1 ; i >= 0 ; i--) {
        ProcessRecord r = procs.get(i);
        if (r.thread != null) {
            ...
            r.thread.dumpGfxInfo(tp.getWriteFd().getFileDescriptor(), args);
            ...
        }
    }
}
```

可以看出，最后真正做dump动作的，其实是r.thread这个对象，这里直接给出r.thread其实就是ActivityThread中的内部类：ApplicationThread，省略代码的追查，让主线更加清晰。

collectProcesses函数的主要作用就是：根据参数找到对应的进程信息



该参数就是命令行中gfxinfo后面传进来的参数。

如果gfxinfo后面没有跟参数，则表示获取所有的Process信息

如果gfxinfo后面跟 包名，则只获取指定报名的Process信息

文件：/frameworks/base/core/java/android/app/ActivityThread.java

```
public final class ActivityThread {  
    ...  
    private class ApplicationThread extends ApplicationThreadNative {  
        ...  
        @Override  
        public void dumpGfxInfo(FileDescriptor fd, String[] args) {  
            dumpGraphicsInfo(fd);  
            WindowManagerGlobal.getInstance().dumpGfxInfo(fd);  
        }  
        ...  
    }  
}
```

Native-dumpGraphicsInfo实现

dumpGraphicsInfo是一个native函数实现在

/frameworks/base/core/jni/android_view_GLES20Canvas.cpp

```
static void  
android_app_ActivityThread_dumpGraphics(JNIEnv* env, jobject clazz, jobject javaFile  
#ifdef USE_OPENGL_RENDERER  
    int fd = jniGetFDFromFileDescriptor(env, javaFileDescriptor);  
    android::uirenderer::RenderNode::outputLogBuffer(fd);  
#endif // USE_OPENGL_RENDERER  
}
```

/frameworks/base/libs/hwui/RenderNode.cpp

```
void RenderNode::outputLogBuffer(int fd) {
    DisplayListLogBuffer& logBuffer = DisplayListLogBuffer::getInstance();
    ...
    FILE *file = fdopen(fd, "a");

    fprintf(file, "\nRecent DisplayList operations\n");
    logBuffer.outputCommands(file);

    String8 cachesLog;
    Caches::getInstance().dumpMemoryUsage(cachesLog);
    fprintf(file, "\nCaches:\n%s", cachesLog.string());
    ...
}
```

可以看出是打印了一些RenderNode的信息，包括memory和cache。

Native-ThreadedRenderer实现

/frameworks/base/core/java/android/view/WindowManagerGlobal.java

```
public void dumpGfxInfo(FileDescriptor fd) {
    ...
    HardwareRenderer renderer =
        root.getView().mAttachInfo.mHardwareRenderer;
    if (renderer != null) {
        renderer.dumpGfxInfo(pw, fd);
    }
    ...
}
```

HardwareRenderer是一个抽象类，具体的实现是在

/frameworks/base/core/java/android/view/ThreadedRenderer.java

```
@Override
void dumpGfxInfo(PrintWriter pw, FileDescriptor fd) {
    pw.flush();
    nDumpProfileInfo(mNativeProxy, fd);
}
```



nDumpProfileInfo也是一个native函数，也是去抓取一些graphic信息，不再做详细讲解。
到这里，dumpsys gfxinfo的流程就基本讲解完了。

总结

dumpsys的实现其实是通过serviceManager拿到对应的service信息，然后执行该service的dump函数。
需要注意的是：每个service都有一个权限检查，需要系统app才可以dump。

 Android子系统 (/nb/5205876)

[举报文章](#) © 著作权归作者所有



Hly_Coder (/u/183339cdc7ae)


写了 24281 字，被 53 人关注，获得了 90 个喜欢
(/u/183339cdc7ae)

+ 关注

程序猿。

如果觉得我的文章对您有用，请随意赞赏。您的支持将鼓励我继续创作！

赞赏支持

 喜欢 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-like-button)

7







更多分享

(http://cwb.assets.jianshu.io/notes/images/5332962/weibo/image_C



登录 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-comment-form)

发表评论



5条评论

只看作者

按喜欢排序 按时间正序 按时间倒序



风儿李 (/u/6e8e452dd456)

2楼 · 2016.08.22 08:25

(/u/6e8e452dd456)

是用vc6.0写代码

👍 赞 💬 回复

Hly_Coder (/u/183339cdc7ae) : @风儿李 (/users/6e8e452dd456) 什么意思哦

2016.08.22 08:55 💬 回复

✎ 添加新评论



unfind (/u/d930e7f92d23)

3楼 · 2016.08.22 13:56

(/u/d930e7f92d23)

原来是这个样子，每次敲dumpsys命令，都不知道这是什么意思。现在终于知道原理了，。

👍 赞 💬 回复

Hly_Coder (/u/183339cdc7ae) : @unfind (/users/d930e7f92d23) 🙄 多看看背后的原理

2016.08.22 14:04 💬 回复

✎ 添加新评论



邵翔宇 (/u/5ea4beb37efa)

4楼 · 2017.03.01 15:33

(/u/5ea4beb37efa)

您好我是小米工程师 我最近也研究这个 方便加下微信交流吗 shaoxy1992 备注写下dumpsys 嘿嘿方便备注

👍 赞 💬 回复



被以下专题收入，发现更多相似内容



Android知识 (/c/3fde3b545a35?utm_source=desktop&utm_medium=notes-included-collection)



Android... (/c/5139d555c94d?utm_source=desktop&utm_medium=notes-included-collection)



Android... (/c/ddfd0f9bb992?utm_source=desktop&utm_medium=notes-included-collection)



首页投稿 (/c/bDHhpK?utm_source=desktop&utm_medium=notes-included-collection)



@IT·互联网 (/c/V2CqjW?utm_source=desktop&utm_medium=notes-included-collection)



源码解析 (/c/2d79ad5a0124?utm_source=desktop&utm_medium=notes-included-collection)

