Binder系列-开篇

Oct 31, 2015

> 基于Android 6.0的源码剖析

一、概述

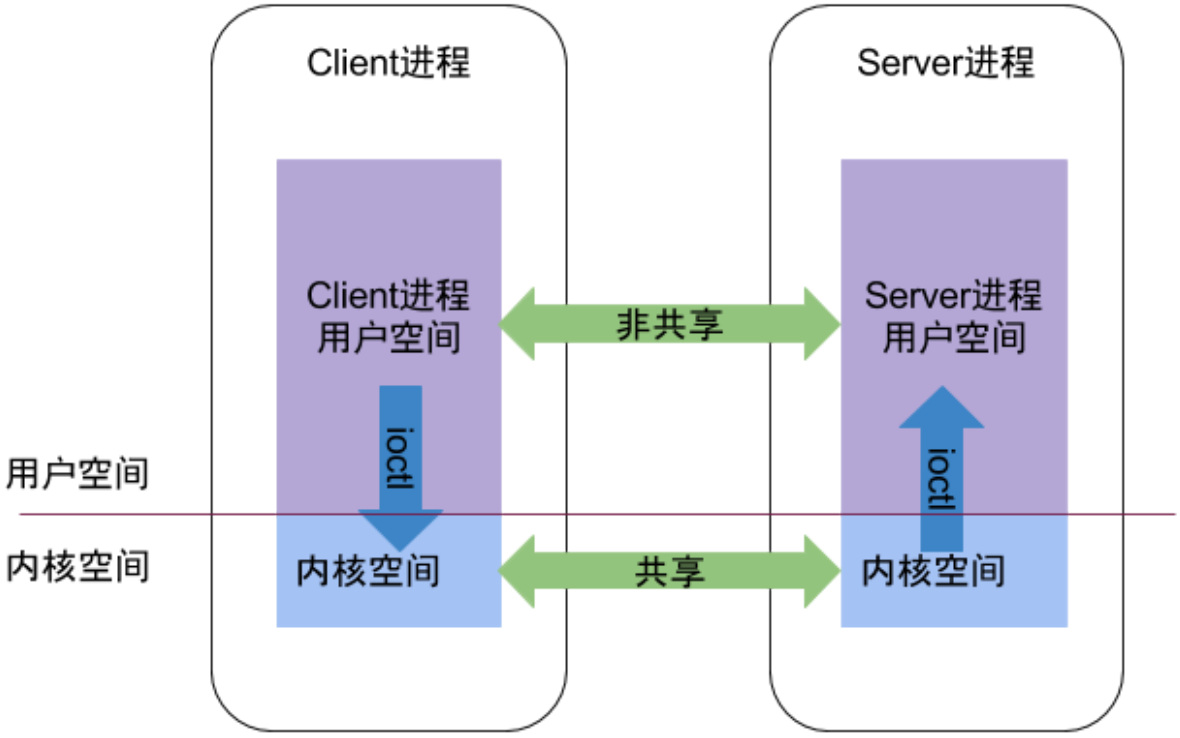
Android系统中，每个应用程序是由Android的 `Activity`、 `Service`、 `Broadcast` ， `ContentProvider` 这四剑客的中一个或多个组合而成。比如多个Activity可能运行在不同的进程中，那么针对这种跨进程Activity间的通信，Android采用的是Binder进程间通信机制。不仅于此，整个Android系统架构中，大量采用了Binder机制作为IPC（进程间通信）方案，当然也存在部分其他的IPC方式，比如Zygote通信便是采用socket。

Binder作为Android系统提供的一种IPC机制，无论从系统开发还是应用开发，都是Android系统中最重要的组成，也是最难理解的一块知识点。要深入了解Binder机制，最好的方法便是阅读源码，借用Linux鼻祖Linus Torvalds曾说过的一句话：Read The Fucking Source Code。

二、 Binder

2.1 IPC原理

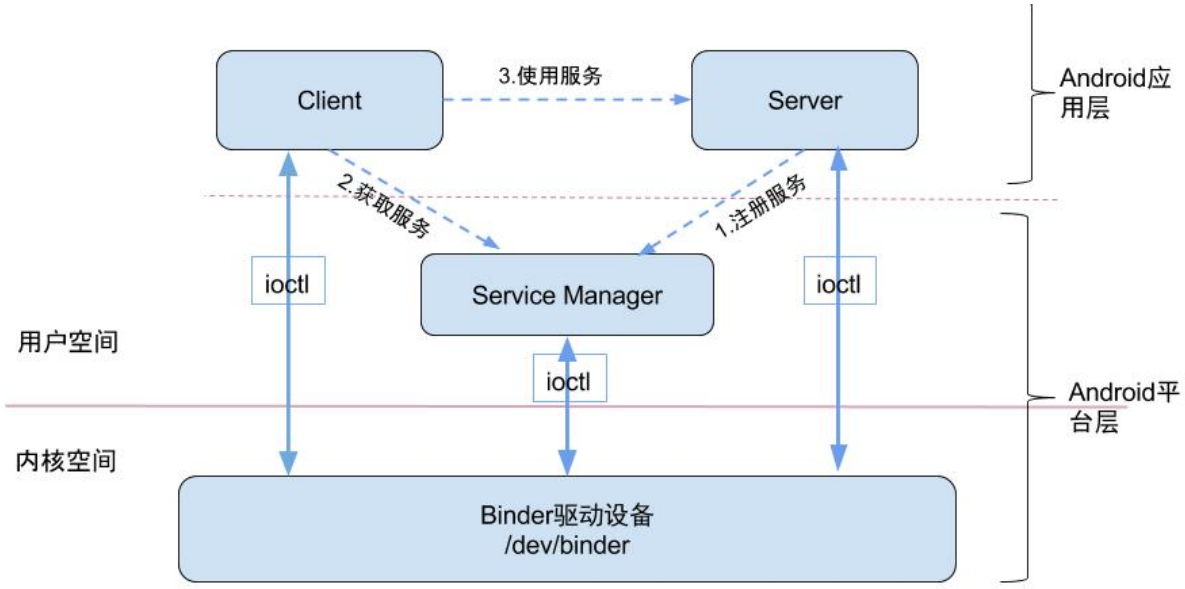
从进程角度来看IPC机制



每个Android的进程，只能运行在自己进程所拥有的虚拟地址空间。对应一个4GB的虚拟地址空间，3GB是用户空间，1GB是内核空间，当然内核空间的大小是可以通过参数配置调整的。其中用户空间，不同进程之间彼此是不能共享的，而内核空间却是可共享的。那么Client进程向Server进程通信，需要依赖两个进程可共享的内核空间。Client端与Server端进程的交互可通过ioctl等方法来进行通信交互。

2.2 Binder原理

Binder通信采用c/s架构，从组件视角来说，包含Client、Server、ServiceManager以及binder驱动，其中ServiceManager用于管理系统中的各种服务。



从图中可以看出无论是注册服务，还是获取服务都离不开Service Manager（注：此处的Service Manager是指Native层的ServiceManager，后面还讲到framework层的Service Manager，与其功能不尽完全相同）。ServiceManager是整个Binder通信机制的大管家，是Android进程间通信机制Binder的守护进程，要掌握Binder机制，首先需要了解系统是如何首次启动Service Manager (http://www.yuanhh.com/2015/11/07/binder-start-sm/)。当Service Manager启动之后，Client端和Server端通信时都需要先获取Service Manager (http://www.yuanhh.com/2015/11/08/binder-get-sm/)接口，才能开始通信服务。

图中Client/Server/ServiceManage之间的相互通信都是基于Binder机制。既然基于Binder机制通信，那么同样也是C/S架构，则每个步骤都有相应的客户端与服务端。有了Service Manager，便可以开始注册和获取服务。

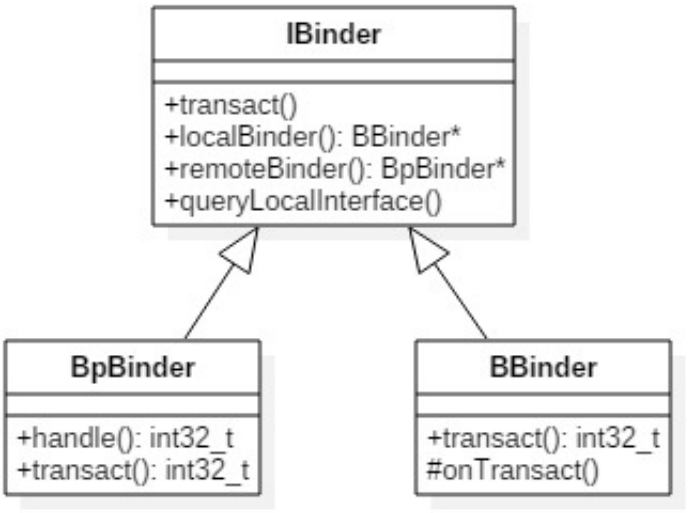
1. **注册服务(addService) (http://www.yuanhh.com/2015/11/14/binder-add-service/)**：Server进程要先注册Service到ServiceManager。该过程：Server是客户端，ServiceManager是服务端。
2. **获取服务(getService) (http://www.yuanhh.com/2015/11/15/binder-get-service/)**：Client进程使用某个Service前，须先向ServiceManager中获取相应的Service。该过程：Client是客户端，ServiceManager是服务端。
3. **使用服务**：Client根据得到的Service信息建立与Service所在的Server进程通信的通路，然后就可以直接与Service交互。该过程：client是客户端，server是服务端。

图中的Client,Server,Service Manager之间交互都是虚线表示，是由于它们彼此之间不是直接交互的，而是都通过与Binder驱动 (http://www.yuanhh.com/2015/11/01/binder-driver/)进行交互的，从而实现IPC通信方式。其中Binder驱动是内核空间，Client,Server,Service Manager属于用户空间。

另外，Binder驱动和Service Manager可以看做是Android平台的基础架构，而Client和Server是Android的应用层，开发人员只需自定义实现client、Server端，借助Android的基本平台架构便可以直接进行IPC通信。文章Binder系列7 —— 如何使用Binder (http://www.yuanhh.com/2015/11/22/binder-use/)分别从Native层和framework层的两个视角来阐述**系统工程师**可如何实现自己的Binder通信。文章Binder系列8 —— 如何使用AIDL (http://www.yuanhh.com/2015/11/23/binder-aidl/)是从App视角来阐述**应用开发工程师**可如何在自己的App中应用Binder通信的。

2.3 C/S模式

BpBinder(客户端)和BBinder(服务端)都是Android中Binder通信相关的代表，它们都从IBinder类中派生而来，关系图如下：



- client端：BpBinder.transact()来发送事务请求；
- server端：BBinder.transact()会接收到相应事务。

三、 提纲

在后续的Binder源码分析过程中所涉及的源码，会有部分的精简，主要是去掉所有log输出语句，已减少代码篇幅过于长，Binder系列文章的提纲如下：

- Binder系列1 —— Binder驱动 (http://www.yuanhh.com/2015/11/01/binder-driver/)
- Binder系列2 —— 启动Service Manager (http://www.yuanhh.com/2015/11/07/binder-start-sm/)
- Binder系列3 —— 获取Service Manager (http://www.yuanhh.com/2015/11/08/binder-get-sm/)
- Binder系列4 —— 注册服务(addService) (http://www.yuanhh.com/2015/11/14/binder-add-service/)
- Binder系列5 —— 获取服务(getService) (http://www.yuanhh.com/2015/11/15/binder-get-service/)
- Binder系列6 —— framework层分析 (http://www.yuanhh.com/2015/11/21/binder-framework/)
- Binder系列7 —— 如何使用Binder (http://www.yuanhh.com/2015/11/22/binder-use/)
- Binder系列8 —— 如何使用AIDL (http://www.yuanhh.com/2015/11/23/binder-aidl/)
- Binder系列9 —— 总结 (http://www.yuanhh.com/2015/11/28/binder-summary/)

喜欢

0条评论 最新 最早 最热

还没有评论，沙发等你来抢

嘿嘿参北斗哇 (http://www.baidu.com/p/嘿嘿参北斗哇) 帐号管理


(http://duoshuo.com/settings/avatar/)

说点什么吧...

分享到: 发布

多说 (http://duoshuo.com)

gityuan@gmail.com (mailto:gityuan@gmail.com) · Github

(https://github.com/yuanhuihui) · 天道酬勤 · © 2015 Yuanhh · Jekyll

(https://github.com/jekyll/jekyll) theme by HyG (https://github.com/Gaohaoyang)