Android进程间通信[IPC机制--Binder](http://www.cnblogs.com/xingfuzzhd/p/3428807.html)

文章来自 Android技术内幕 系统卷

转：<http://www.linuxidc.com/Linux/2011-08/40508.htm>

**什么是IPC机制以及IPC机制的种类**

在Linux中，是以进程为单位分配和管理资源的。出于保护机制，一个进程不能直接访问另一个进程的资源，也就是说，进程之间互相封闭。但是，一个复杂的应用系统中，通常会使用多个相关的进程来共同完成一项任务，因此要求进程之间必须能够互相通信，从而共享资源和信息。所以，操作系统内核必须提供进程间的通信机制（IPC）。

IPC机制种类：采用命名管道（name pipe），消息队列（message queue），信号（signal），内存共享（share memory）;

在Android终端上的应用软件的通信几乎看不到这些IPC通信方式，取而代之的是Binder方式。

Binder更简洁和快速，消耗的内存资源更小，Binder主要提供以下一些功能：

* 用驱动程序来推进进程间的通信。
* 通过共享内存来提高性能
* 为进程请求分配每个进程的线程池
* 针对系统中的对象引入了引用计数和跨进程的对象引用映射
* 进程间的同步调用

**初识Binder**

Binder是通过Linux的Binder Driver 来实现的，Binder操作类似于线程迁移（Thread migration），两个进程间通信看起来就像是一个进程进入另一个进程去执行代码，然后带着执行的结果返回。同时Binder机制是基于OpenBinder来实现的，是一个OpenBinder的Linux实现，android系统的运行都是将依赖Binder驱动。（OpenBinder组件架构是一个系统,主要提供一个高层抽象上传统的现代操作系统服务。当前实现运行在Linux,但是代码运行在一个不同的平台。） Binder的驱动原理 为了完成进程间通信，Binder采用了AIDL （Android Interface Definition Language）来描述进程间的接口。在实际的实现中，Binder是作为一个特殊的字符型设备而存在的，设备节点为/dev/binder,其实现遵循Linux设备驱动模型，实现代码主要涉及以下文件：

* kernel/driver/staging/binder.h
* kernel/driver/staging/binder.c

**Binder1驱动的实现**

上面我们已经对Binder驱动的原理进行了分析，在开始分析驱动的实现之前，我们还是通过一个例子说明Binder在实际应用中应该如何运用，以及它能帮我们解决什么样的问题。比如：A进程如果要使用B进程的服务，B进程首先要注册此服务，A进程通过Binder获取该服务的handle，通过这个handle ，A进程就可以使用该服务了，此外，你可以把handle理解成地址。A进程使用B进程的服务还意味着二者遵循相同的协议，这个协议反映在代码上就是二者要实现Ibinder接口。

1.“对象”与“引用” Binder不仅是Android 系统中的一个完善的IPC机制，它也可以被当作Andriod系统的一种RPC（远程过程调用）机制，那么我们就要记住Binder不仅可以与本地进程通信，还可以与远程进程通信；这里的本地进程就是我们所说的本地对象，而远程进程则使我们所说的远程服务的一个“引用”。(“引用”这个词并不是官方所描述的，而是笔者为了方便大家理解，将其称为引用) 那么这个本地“对象”与远程对象的“引用”有什么不同呢？ 本地“对象”表示本地进程的地址空间的一个地址，而远程对象的“引用”则是一个抽象的32位句柄。 它们之间的关系是互质的： 所有的进程本地对象都是本地进程的一个地址（address，ptr，binder），所有的远程进程的对象的“引用”都是一个句柄。

**Binder的构架与实现**

通过前面的学习，我们对Binder的整个工作流程有了一个深入 的认识，关于Binder的实现，还涉及有很多细节。 Binder的系统结构 在Android设计中，每一个Activity都是一个独立的进程，每个service也是一个独立的进程，而Activity要与Service进行通信，就是跨进程的通信，这时就需要使用Binder机制了。 那么在这里可以把Activity看作客户端，实际上也就是一个客户端与服务端之间的通信。 1.Binder机制的组成 Android的Binder机制就是一个C/S构架，客户端和服务端直接通过Binder交互数据，打开Binder写入数据，通过Binder读取数据，这样通讯就可以完成了。关于数据的读写则是由Binder的驱动完成的，除了Binder的驱动之外，整个机制还包括以下几个组成部分：

（1）Serice Manager Serice Manager主要负责管理Android系统中所有的服务，当客户端要与服务端进行通信时，首先就会通过Service Manager来查询和取得所需要交互的服务。当然，每个服务也都需要向service Manager注册自己提供的服务，以便能够提供客户端进行查询和获取。

（2）服务（server） 这里的服务即上面所说的服务端，通常也是Android的系统服务，通过Service Manager可以查询和获取某个server。

（3）客户端 这里的客户端一般是指Android系统上面的应用程序。它可以请求Server中的服务，比如Activity。

（4）服务代理 服务代理是指在客户端应用程序中生成的server代理（proxy）。从应用程序的角度来看，代理对象和本地对象没有差别，都可以调用其方法，方法都是同步的，并且返回相应的结果。服务代理也是Binder机制的核心模块。 2.Binder的系统构架 在Android源码中，有关Binder的实现在各个层析都有，主要的Binder库由本地原生代码实现，java和C++层都定义有同样功能的Binder接口，供应用程序使用，它们实际上都是调用原生Binder库的实现。Binder的系统构架如图：

其中，Binder驱动在前面已经介绍，它用于实现Binder的设备驱动，主要负责组织Binder的服务节点，调用Binder相关的处理线程，完成实际的Binder传输等，它位于Binder结构的最底层（即Linux内核层）。Binder Adapter 层是对Binder驱动的封装，主要用于操作Binder驱动，即应用程序不必直接接触Binder驱动程序，实现包括IPCThreadState.cpp 和ProcessState.cpp以及Parcel.cpp,以及Parcel.cpp中的部分内容。Binder核心库是Binder框架的核心实现，主体的客户端/服务端都分别有java和C++两种实现方案，主要供应程序使用，比如摄像头和多媒体等。它们通过Binder的核心库来实现。 Binder的机制和原理 作为Android系统的核心机制，Binder几乎贯穿整个Android系统，本节将从Binder所涉及的service Manager，服务，客户端，服务端（代理对象）等各个部分进行分析，在分析之前首先需要明确Binder的工作流程：

（1）客户端首先获得服务器的代理对象。所谓的代理对象实际上就是在客户端建立一个服务端的“引用”，该代理对象具有服务端的功能，使其在客户端访问服务端的方法就像访问本地方法一样。

（2）客户端通过调用服务器代理对象的方式向服务端发送请求。

（3）代理对象将用户请求通过Binder驱动发送到服务器进程。

（4）服务器进程处理用户请求，并通过Binder驱动返回处理结果给客户端的服务器代理对象。

（5）客户端收到服务器端的返回结果。 经过这样一个流程，Binder就完成了一次通信。课可以看出，这里与Binder通信的对象服务端就是服务，下面我们首先分析Android中的服务。