# Binder机制

### 知识储备

在简介Binder前，先了解一下Linux的基础知识

### 1 进程空间划分

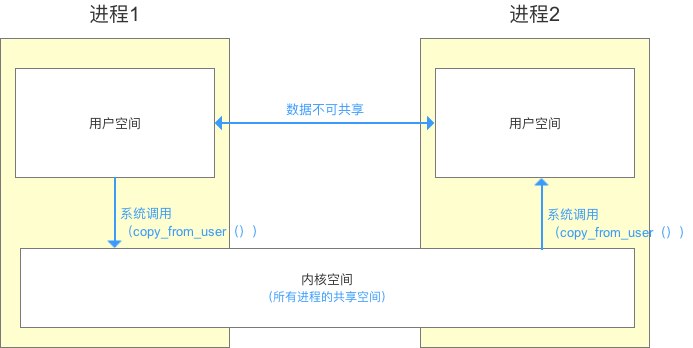
一个进程空间分为 用户空间 & 内核空间（Kernel），即把进程内 用户 & 内核 隔离开来

**二者区别：**

* 进程间，用户空间的数据不可共享，所以用户空间 = 不可共享空间
* 进程间，内核空间的数据可共享，所以内核空间 = 可共享空间，所有进程共用1个内核空间

进程内 用户空间 & 内核空间 进行交互 需通过 系统调用，主要通过函数：

* copy\_from\_user（）：将用户空间的数据拷贝到内核空间
* copy\_to\_user（）：将内核空间的数据拷贝到用户空间



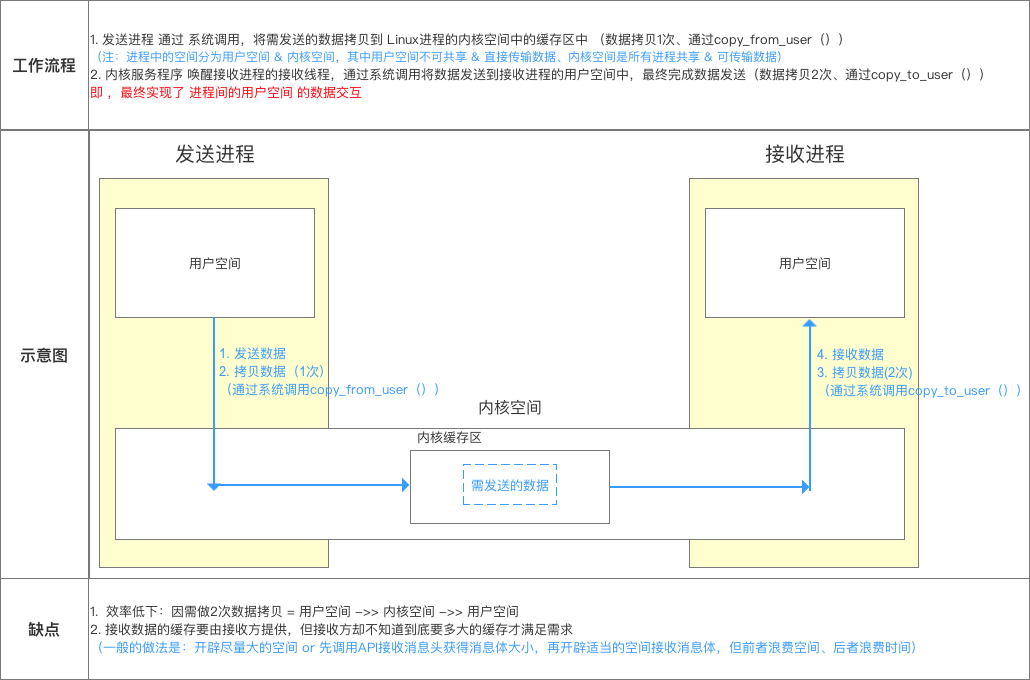
### 进程隔离 & 跨进程通信(IPC)

**进程隔离：**  
为了保证 安全性 & 独立性，一个进程 不能直接操作或者访问另一个进程，即Android的进程是相互独立、隔离的

**跨进程通信（ IPC ） ：**

即进程间需进行数据交互、通信

**跨进程通信的基本原理如下图：**

****

* Binder的作用则是：连接 两个进程，实现了mmap()系统调用，主要负责 创建数据接收的缓存空间 & 管理数据接收缓存
* 传统的跨进程通信需拷贝数据2次，但Binder机制只需1次，主要是使用到了内存映射，具体下面会详细说明

### Binder机制

**Binder机制简介：**

在Android系统中，每一个应用程序都运行在独立的进程中，这也保证了当其中一个程序出现异常而不会影响另一个应用程序的正常运转。在许多情况下，我们activity都会与各种系统的service打交道，很显然，我们写的程序中activity与系统service肯定不是同一个进程，但是它们之间是怎样实现通信的呢？

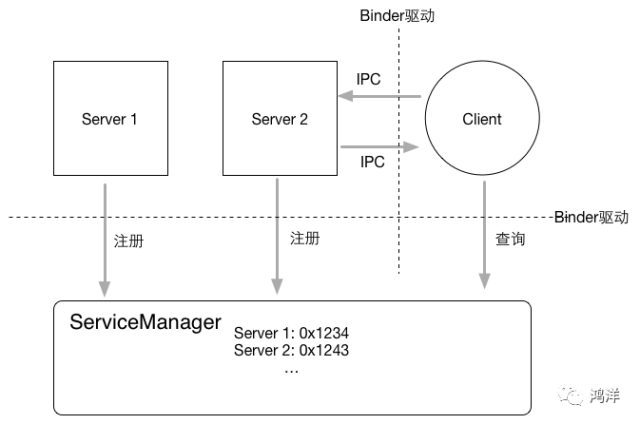
所以Binder是android中一种实现进程间通信（IPC）的方式之一。

**Binder的实现：**

* 首先，Binder分为Client和Server两个进程。

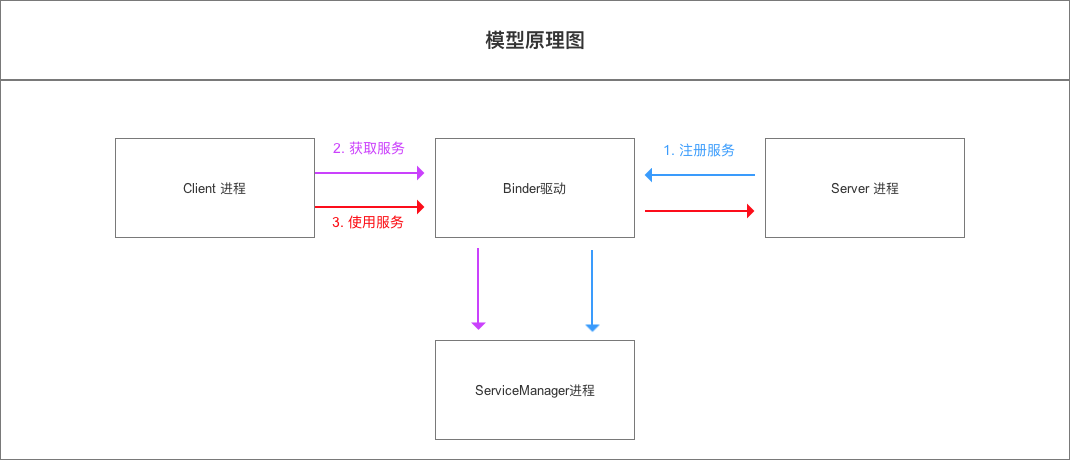
注意，Client和Server是相对的。谁发消息，谁就是Client，谁接收消息，谁就是Server。

* 其次，我们看下面这个图（摘自田维术的博客），基本说明白了Binder的组成解构：



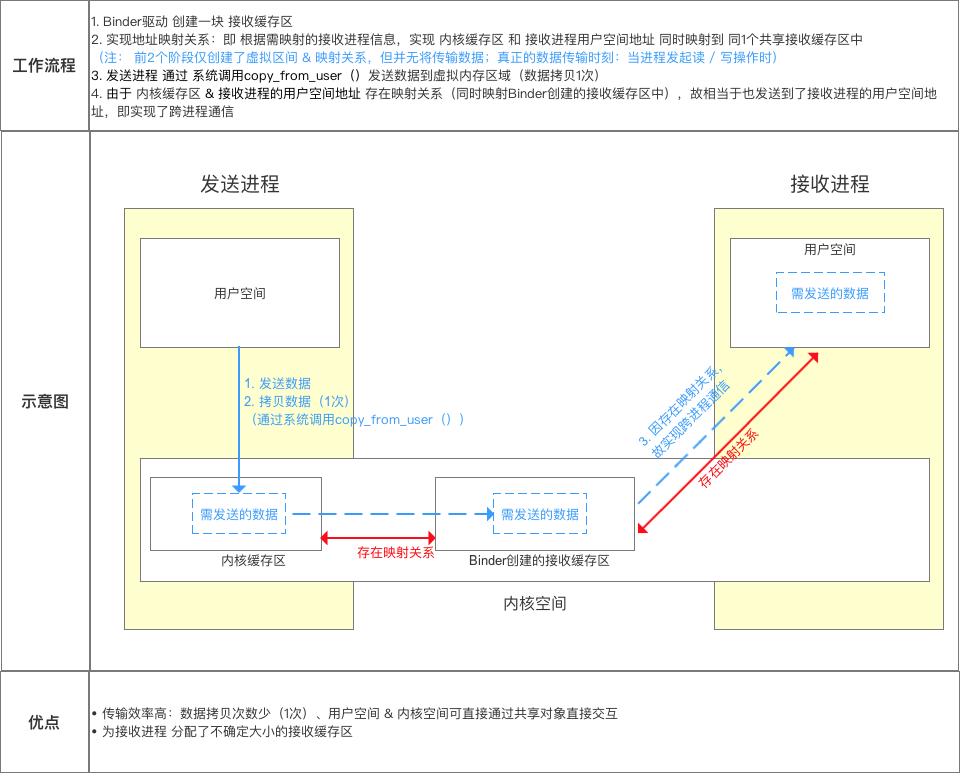
图中的IPC就是进程间通信的意思，图中的主要成员如下：

* Client进程：使用服务的进程，Android客户端
* Server进程：提供服务的进程，服务器端
* ServiceManager：管理service的注册和查询(将字符形式的binder名字，转换成Client中对该Binder的引用)
* Binder驱动：一种虚拟设备驱动，是连接Service、Client、SM的桥梁，通过内存映射传递进程间的数据，通过Binder线程池实现线程控制，由Binder驱动自身管理，binder驱动持有每个server进程在内核空间中的Binder实体，并给Client进程提供Binder实体的引用

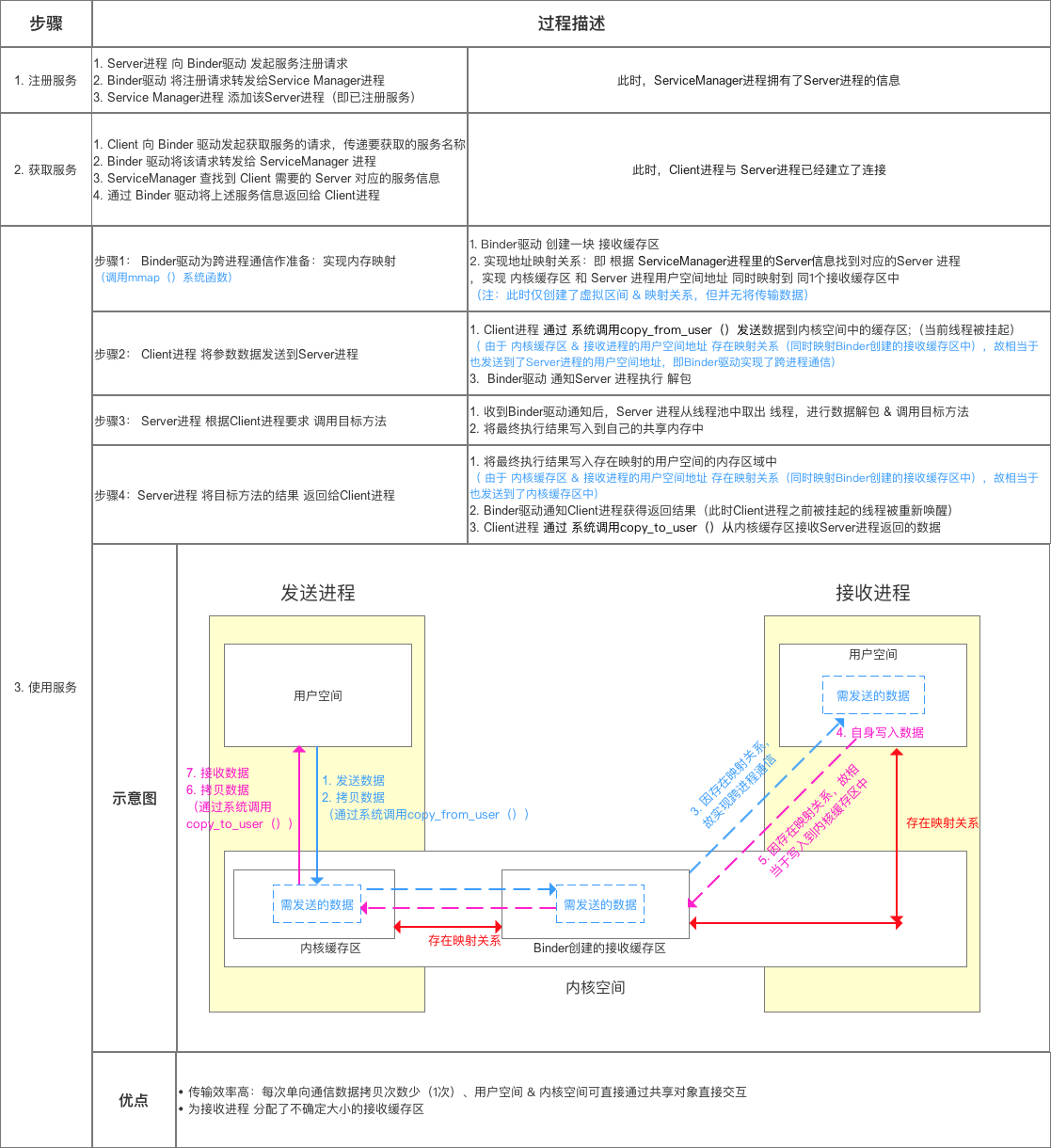


**Binder驱动的核心原理：**

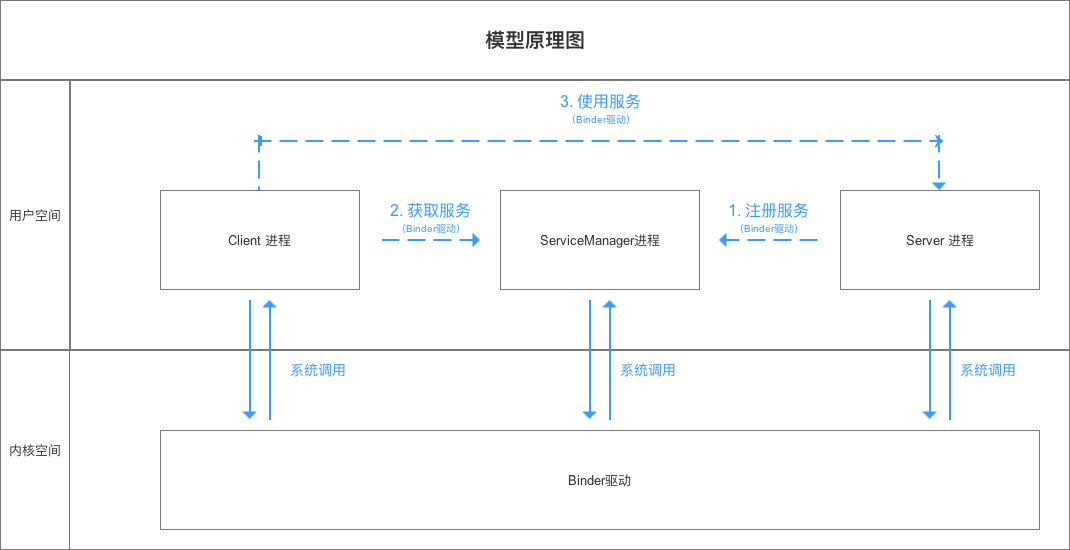
内存映射



Binder驱动的完整过程如下：



完整流程：

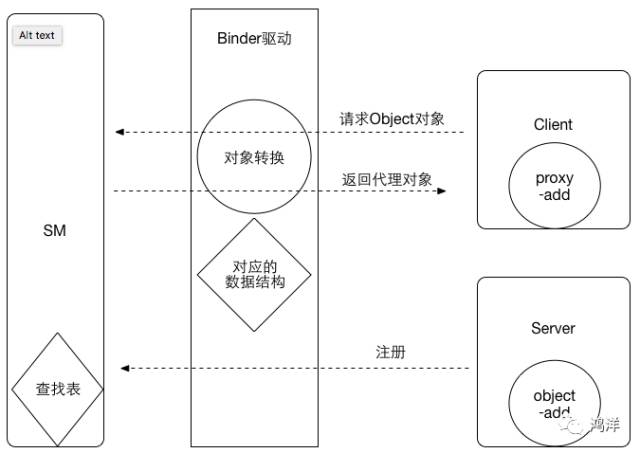


图中的ServiceManager，负责把Binder Server注册到一个容器中

有人把ServiceManager比喻成电话局，存储着每个住宅的座机电话，还是很恰当的。张三给李四打电话，拨打电话号码，会先转接到电话局，电话局的接线员查到这个电话号码的地址，因为李四的电话号码之前在电话局注册过，所以就能拨通；没注册，就会提示该号码不存在。

对照着Android Binder机制，对着上面这图，张三就是Binder Client，李四就是Binder Server，电话局就是ServiceManager，电话局的接线员在这个过程中做了很多事情，对应着图中的Binder驱动.

接下来我们看Binder通信的过程，还是摘自田维术博客的一张图：



我们看到，Client想要直接调用Server的add方法，是不可以的，因为它们在不同的进程中，这时候就需要Binder来帮忙了。

* 首先是Server在SM这个容器中注册。
* 其次，Client想要调用Server的add方法，就需要先获取Server对象， 但是SM不会把真正的Server对象返回给Client，而是把Server的一个代理对象返回给Client，也就是Proxy。
* 然后，Client调用Proxy的add方法，SM会帮他去调用Server的add方法，并把结果返回给Client。