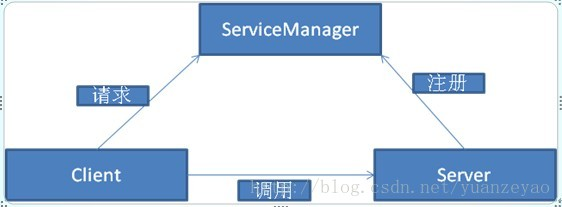
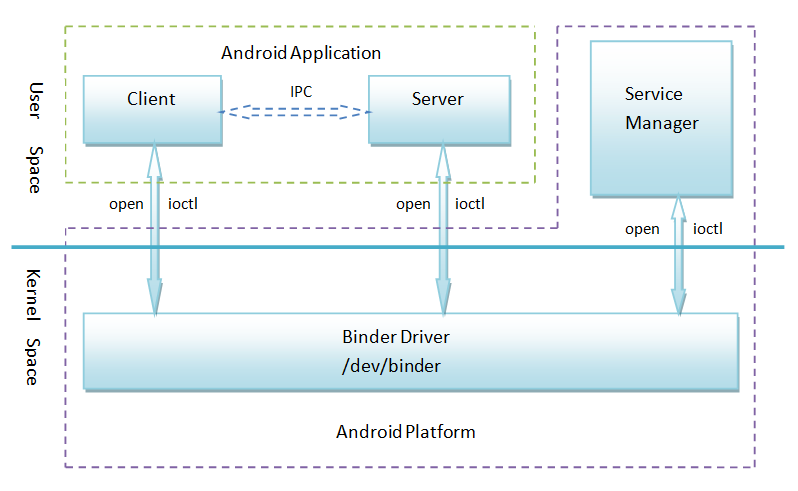
**Binder**机制

****参考：<https://www.jianshu.com/p/a709756c1609>

<https://www.jianshu.com/p/b5cc1ef9f917>

<https://blog.csdn.net/AndroidStudyDay/article/details/93749470>

<https://www.jianshu.com/p/225ccd39a667>

****

1. server通过open\_driver()打开“dev/binder”的虚拟设备，从而打开binder驱动、获得binder，便可以通过binder进行通信。主线程运行startThreadPool，工作线程运行joinThreadPool。  
   通过defaultServiceManager()得到了ServiceManager对象，并通过ServiceManager的addService()方法将server注册到ServiceManager中。
2. ServiceManager集中管理service，通过binder\_open()打开一个Binder，告诉Binder驱动程序自己是Binder上下文管理者（即守护线程），binder\_looper()处理addService()消息。
3. client通过open\_driver()打开“dev/binder”的虚拟设备，从而打开binder驱动、获得binder，通过defaultServiceManager()得到了ServiceManager对象，通过ServiceManager的getService()查询到server信息生成Binder并保存到mRemote变量中，通过Server.transact()方法将消息发送给Server。
4. Client、Server和Service Manager实现在用户空间中且分别运行在独立的进程中，Binder驱动程序实现在内核空间中
5. Binder驱动程序提供设备文件/dev/binder与用户空间交互，Client、Server和Service Manager通过open和ioctl文件操作函数与Binder驱动程序进行通信
6. Service Manager是成为Android进程间通信（IPC）机制Binder守护进程的过程 打开/dev/binder文件：open("/dev/binder", O\_RDWR);
   1. 打开/dev/binder文件：open("/dev/binder", O\_RDWR);
   2. 建立128K内存映射：mmap(NULL, mapsize, PROT\_READ, MAP\_PRIVATE, bs->fd, 0);
   3. 通知Binder驱动程序它是守护进程：binder\_become\_context\_manager(bs);
   4. 进入循环等待请求的到来：binder\_loop(bs, svcmgr\_handler);
7. Client如果想要获得Service Manager远程接口，却不必通过进程间通信机制来获得，因为Service Manager远程接口是一个特殊的Binder引用，它的引用句柄一定是0。