**第三节 Android Binder**

进程间通信，是基于C/S 架构，如果进程A 主动连接进程B，那么进程A 称为 Client 端，进程B 称为 Server 端，反之亦然。进程A 访问进程B是通过中间的一个内存(共享内存)，即Android专门用于进程间通信的内存。

如何通信呢？ Stub是跟服务端交互的，Proxy是跟客户端交互的。

假设将服务端看成一个Service，一旦Service启动，就会在共享内存中注册(分配)一个IBinder的引用，这个引用保存了对该服务的描述(注意不是该服务的对象，不可直接调用，因为不在同一个进程)，如类名、包名、成员变量。

第一个概念：对于进程间通信，我们客户端去连接服务端，这个服务端一旦启动，它就会在共享内存中分配一个IBinder引用，IBinder引用实际上就是对它的描述，告诉你我现在注册了一个可以用来进程间通信的服务。那么具体是怎么通信的？

Stub的服务端表示什么意思？进程间通信实际上最基本的原理是数据的读和写。

Proxy 是用来向这个IBinder 去写数据的，Stub是通过这个IBinder 去 向Stub回调

整个过程：

进程A 向 进程B 写数据：

通过进程A的Proxy去连接IBinder，然后向IBinder写数据，IBinder再通过进程B的Stub来反馈进程B.

ActivityManager

Stub

——

Proxy

Stub

——

Proxy

服务端

进程B

IBinder

CPP

客户端

进程A

Android 进程

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 优点 | 缺点 | 适用场景 |
| Intent | 简单易用 | 只能传输Bundle所支持的数据类型 | 四大组件间的进程间通信 |
| 文件共享 | 简单易用 | 不适合高并发 | 简单的数据共享，无高并发场景 |
| AIDL | 功能强大，支持一对多并发实时通信 | 使用稍微复杂，需注意线程同步 | 复杂的进程间调用，Android 中最常用 |
| Messenger | 比AIDL稍微简单易用些 | 比AIDL功能弱，只支持一对多串行实时通信 | 简单的进程间通信 |
| ContentProvider | 强大的数据共享能力，可通过Call方法扩展 | 受约束的AIDL，主要对外提供数据线的CRUD操作 | 进程间的大量数据共享 |
| RemoteViews | 在跨进程访问UI方面有奇效 | 比较小众的通信方式 | 某些特殊的场景 |
| Socket | 跨主机，通信范围广 | 只能传输原始的字节流 | 常用于网络通信中 |

传统跨进程通信：

传统跨进程的弊端：

**手写 Android 的 IPC 进程通信框架，实现远程方法调度**

不仅能支持跨进程，还能跨APP.

进程的概念：

进程A 与进程 B 不一样的内存空间

每个进程拥有自己独立的资源，拥有私有的地址空间。

Android App 使用进程的优缺点

缺点：

静态成员和单例模式完全失效

线程同步机制失效

Application 会多次 重建

优点：

不能影响主业务的代码稳定运行

不能占用主业务的进程内存

不受主业务进程生命周期影响，独立存在和运行

要满足这三个需求下，不由就会想到在应用内开辟一个进程单独推送业务使用，

因为其特点明显：

独立进程运行出现了崩溃和异常而退出并不会影响其他进程运行

独立进程意味着有系统开辟的独立内存空间，不会和其他进程产生占用内存而影响其内存分配问题。

独立进程的启动和退出可以完全不依赖用户对应用的使用，可以独立启动。

需求：打破进程限制，实现自由的调用服务端方法

服务端进程

需要给其他进程调度的对象注册到缓存中心

缓存中心：登记： 类名---class 类名----方法 类名-----对象

## 第八章：Android Handler 源码分析