链滴

Android源码:Binder机制（一）

作者：wind

原文链接：https://ld246.com/article/1441808540482

来源网站：链滴

Android系统的IPC（进程间通信）的一种机制是Binder。其实做android开发的时候，遇到Binder的情况还蛮多的比如在继承Service的时候。
Binder采用我们常见的C/S架构，Server,ServiceManager,Client的关系如下：
1. Server ServiceManager注册服务
2. Client向ServiceManager查询服务
3. Client使用Server供的服务
所以Client要使用Service，首先在ServiceManager中查询该服务，然后再使用。
书上用了MediaServer的源码来解析Binder，入口代码如下：
<pre class="brush: java">int main(int argc,char** argv){
sp<ProcessState> proc(ProcessState::self());
sp<IServiceManager> sm=defaultServiceManager();

AudioFlinger::instantiate(); //服务的实例化

MediaPlayerService::instantiate();

CameraService::instantiate();

AudioPolicyService::instantiate();

ProcessState::self()->startThreadPool();

IPCThreadState::self()->joinThreadPool();

}</pre>

Binder作为IPC机制，少不了对象ProcessState。在ProcessState.cpp中，我们可以看到个直接和Binder通信相关的函数，open_driver()，这个函数的作用就是打开/dev/binder设备，与内的Binder驱动有了交互的通道。
那么defaultServiceManager又是干什么的呢？
看一下ServiceManager.cpp中的defaultServiceManager()，返回的是一个IServiceManager对象，
<pre class="brush: java">gDefaultServiceManager=interface_cast<IServiceManager>(ProcessState::self()->getContextObject(NULL));</pre>

看gretContextObject函数
<pre class="brush: java">sp<IBinder> ProcessState::getContextObject(const sp<IBinder>& caller){
if(supportsProcesses()){
return getStrongProxyForHandle(0);
}else{
return getContextObject(String16("default"),caller);
}
}</pre>

进入getStrongProxyForHandle(0)函数，这里的Handle在C++中叫做句柄，起到一个源标识的作用。
<pre class="brush: java">sp<IBinder> ProcessState::getStrongProxyForHandle(int32_t handle)
{
sp<IBinder> result;
AutoMutex _l(mLock);
handle_entry* e=lookupHandleLocked(handle);查找是否存在handle对应的资源
if(e!=null){
IBinder* b=e->binder;
if(b==null||!e->refs->attemptIncWeak(this)){
b=new BpBinder(handle);
e->binder=b;
if(b) e->refs=b->getWeakRefs();
result=b;
}else{

result.force_set(b);
e->refs->decWeak(this);
}
}
return result; //返回BpBinder对象
}</pre>
好的，这里出现了一个BpBinder，这里我们需要了解一下IBinder的衍生类，BpBinder和BBinde。
- BpBinder可以看做是客户端的Proxy(代理)，用来与Server交互。我们作为客户端，所以建BpBinder对象。
- BBinder则是Server端的代理，是BpBinder交互的直接对象。
好的，看
gDefaultServiceManager=interface_cast<IServiceManager>(ProcessState::self()->getContextObject(NULL));
这里将BpBinder对象通过interface_cast<IServieManager>转换成了IServiceManager对象。IServiceManager类，顾名思义就是服务的管理类包括addService，checkService一些管理Service的方法。
IServiceManager和主要类的关系下：

大概的Binder通信机制搞明白了，那么具体到传输的细节应该是什么样呢？

