题外话：由于同事咨询，口头讲解的话 1 对 1，但不方便多方传播及继承性，因此特意写此文章讲解一番。

1、android service 简要说明:

Android Service ： 又称为 Java Service ，是实现在框架层里的 Service ，使用 Java 语言编写。
Native Service ： 又称为 System Service ，是实现在 Runtime 层里的 Service 。使用 C++语言编写。

对于这两种 service 来说，两个对等 service 通讯都是利用 binder，只不过一种利用*.aidl，一种利用 IInterface
编写序列化代码而已，本质是一样的，本书先介绍 native service 的编写及两个 native service
如何通讯的过程：

2、native service 的特点

A、因为底层核心服务是 Android 框架里最接近 Linux/Driver 的部分。为了充分发挥硬件设备
的差异化特性，核心服务是让上层 Java 应用程序来使用 Driver/HW Device 特色的重要管道。
B、在开机过程中，就可以启动核心服务（ 例如汉字输入法服务等 ），让众多应用程序来共享之。
C、由于共享，所以能有效降低程序的大小及统一的接口变化 。

3、如何实现一个 native service

要点如下：

a、核心服务通常在独立的进程 (Process) 里执行。

b、必须提供 IBinder 接口，让其它程序可以进行跨进程的绑定 (Binding) 和呼叫。

c、因为共享，所以必须确保多线程安全 (Thread-safe) 。

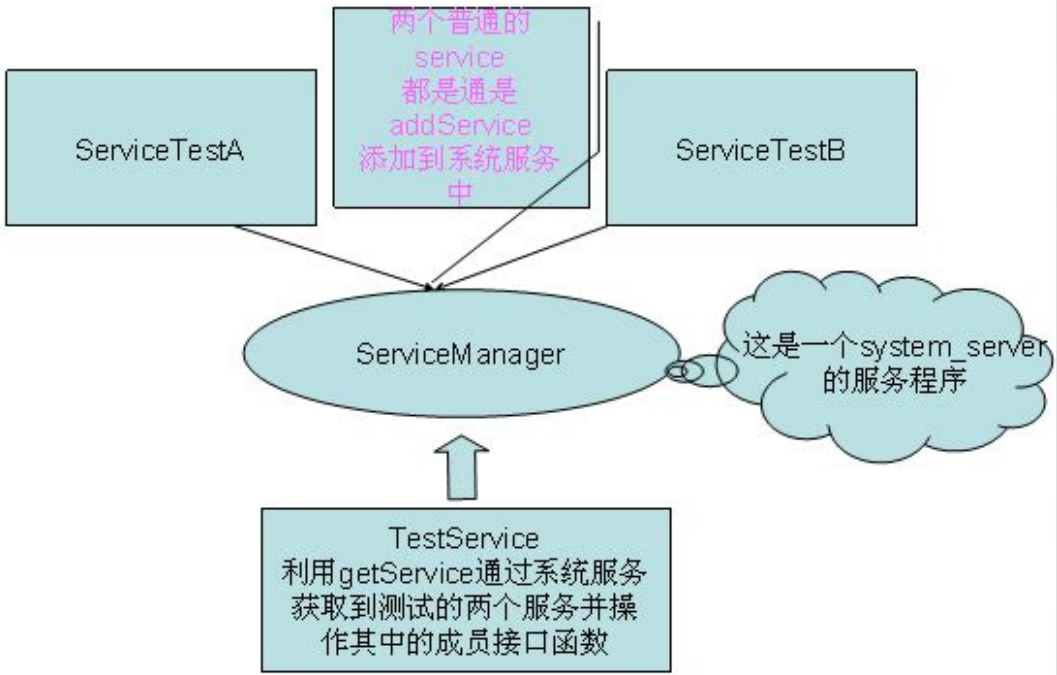
d、以 C++ 类别定义，诞生其对象，透过 SM 之协助，将该对象参考值传给 IServiceManager::addService() 函数，就加入到 Binder Driver 里了。

e、应用程序可透过 SM 之协助而远距绑定该核心服务，此时 SM 会回传 IBinder 接口给应用程序。

f、应用程序可透过 IBinder::transact() 函数来与核心服务互传数据。

下面以具体实例讲解一下具体的每个步骤如何实现完成

先说明一个测试例子的模块结构：


两个普通的
service
都是通是
addService
添加到系统服务
中
ServiceTestA
ServiceTestB
ServiceManager
这是一个system_server
的服务程序
TestService
利用getService通过系统服务
获取到测试的两个服务并操
作其中的成员接口函数

serviceTestA 是一个普通的过程，提供两个整数的剩法及除法运算

serviceTestB 是一个普通的过程，提供两个整数的加法及减法运算

TestService 是一个测试进程的程序，主要验证两个服务进程的接口函数，其中的代码可以放在任何一个进程进行访问调用

a、编写服务进程

serviceTestA.h 头文件定义：

```
#ifndef __SERVICE_TEST_A__
#define __SERVICE_TEST_A__

#include <utils/RefBase.h>
#include <binder/IInterface.h>
#include <binder/Parcel.h>
#include <utils/threads.h>
```

```
namespace android{
//继承 BBinder 类，从而提供 IBinder 接口
class serviceTestA:public BBinder
{
public:
          serviceTestA();
virtual ~serviceTestA();
static int instantiate(); //建立唯一类实例
virtual status_t onTransact(uint32_t, const Parcel&, Parcel*, uint32_t);

private:
// protected by mLock 多线程安全
                              mutable Mutex mLock;
};
}
```

```
#endif /* __SERVICE_TEST_A__ */
```

serviceTestA.cpp 实现文件：

```
#include <cutils/log.h>
#include <cutils/properties.h>
#include <binder/IServiceManager.h>
#include <binder/IPCThreadState.h>
#include <serviceTestA/serviceTestA.h>

namespace android {

enum{
CALCULATE_MUL_NUM = 0,
```

```
CALCULATE_DIV_NUM ,
};
    int serviceTestA::instantiate() {
LOGI("serviceTestA instantiate");

int r = defaultServiceManager()->addService(String16("service.TestA"),
new serviceTestA());
LOGI("serviceTestA r = %d/n", r);
return r;
}

serviceTestA::serviceTestA() {
LOGI("serviceTestA created");
}
```

```
serviceTestA::~serviceTestA(){
LOGI("serviceTestA destroyed");
}

status_t serviceTestA::onTransact(uint32_t code, const Parcel&data, Parcel*reply, uint32_t flags){
LOGI("serviceTestA::onTransact code = %d",code);
Mutex::Autolock _l(mLock);
switch(code){
case CALCULATE_MUL_NUM:{
int a = data.readInt32();
int b = data.readInt32();
int sum = a * b ;
LOGI("sum mul value = %d",sum);
reply->writeInt32(sum);
return NO_ERROR;
```

```
}break;
case CALCULATE_DIV_NUM:{
int a = data.readInt32();
int b = data.readInt32();
int sum = a / b ;
LOGI("sum div value = %d",sum);
reply->writeInt32(sum);
return NO_ERROR;
}break;
default:
return BBinder::onTransact(code, data, reply, flags);
}
return 0;
}
}
```

Android.mk 文件：

```
LOCAL_PATH:= $(call my-dir)

include $(CLEAR_VARS)

LOCAL_SRC_FILES:= serviceTestA.cpp

LOCAL_SHARED_LIBRARIES:= libutils libutils libbinder

LOCAL_C_INCLUDES := $(TOP)/frameworks/base/include

LOCAL_MODULE:= libServiceTestA

LOCAL_PRELINK_MODULE:= false
```

include $(BUILD_SHARED_LIBRARY)

这里生成 libServiceTestA 动态库，方例升级服务程序

编写独立的进程程序：

它的用途是：诞生一个 serviceTestA 类别之对象，然后将该对象参考存入 Binder Driver 里。

```
#include <sys/types.h>
#include <unistd.h>
#include <cutils/log.h>
#include <binder/IServiceManager.h>
```

```
#include <binder/IPCThreadState.h>
#include <serviceTestA/serviceTestA.h>

using namespace android;

int main(int argc,char *argv[]){
sp<ProcessState> proc(ProcessState::self());
sp<IServiceManager> sm = defaultServiceManager();
LOGI("ServiceManager: %p", sm.get());
serviceTestA::instantiate(); // 这是重点。。。。
ProcessState::self()->startThreadPool();
IPCThreadState::self()->joinThreadPool();
return 0;
}
```

Android.mk 文件：

```
LOCAL_PATH:= $(call my-dir)

include $(CLEAR_VARS)

LOCAL_SRC_FILES:= main.cpp

LOCAL_SHARED_LIBRARIES:= libutils libServiceTestA

LOCAL_MODULE:= serviceTestA

include $(BUILD_EXECUTABLE)
```


这里最重要的是调用：serviceTestA::instantiate();

其先执行到 new serviceTestA() ，就诞生一个 serviceTestA 类别之对象；

接着，呼叫 defaultServiceManager() 函数取得 SM 的 IServiceManager 接口；

再呼叫 IServiceManager::addService() 将该对象参考存入 Binder Driver 里，并且同时存入到 ServiceManager 中注册并管理，如此其它进程才能通过 ServiceManager::getService 找到相应服务进程

以上代码同理，serviceTestB 服务进程也一样的这样建立，不再复述。

b、测试服务进程

testService.cpp 编写：

```
#include <cutils/log.h>
#include <cutils/properties.h>
#include <binder/IServiceManager.h>
#include <binder/IPCThreadState.h>
#include <serviceTestA/serviceTestA.h>
#include <serviceTestB/serviceTestB.h>
```

```
using namespace android;


enum{
CALCULATE_ADD_NUM = 0,
CALCULATE_SUB_NUM ,
};


enum{
CALCULATE_MUL_NUM = 0,
CALCULATE_DIV_NUM ,
};
    int main(int argc,char *argv[]){
sp<IBinder> TestAbinder;
sp<IBinder> TestBbinder;
```

```
Parcel data, reply;
int sum=0;

LOGI("testService main is call...");

sp<IServiceManager> sm = defaultServiceManager();
    while(1){
TestAbinder = sm->getService(String16("service.TestA"));
LOGE("TestA::getAddService %p/n",sm.get());
if (TestAbinder == 0) {
LOGE("TestAService not published, waiting...");
usleep(1000000);
continue;
}
else{
```

```
LOGI("TestA::getAddService success...");
break;
}
}
    while(1){
TestBbinder = sm->getService(String16("service.TestB"));
LOGE("TestB::getAddService %p/n",sm.get());
if (TestBbinder == 0) {
LOGE("TestBService not published, waiting...");
usleep(1000000);
continue;
}
else{
LOGI("TestB::getAddService success...");
break;
```

```
}
}
    //测试两个 service 中的函数
data.writeInt32(1000);
data.writeInt32(200);
LOGI("BpAddService::create remote()->transact()/n");
TestAbinder->transact(CALCULATE_MUL_NUM,data,&reply);
sum = reply.readInt32();
LOGI("CALCULATE_ADD_NUM value = %d",sum);
    data.writeInt32(1000);
data.writeInt32(200);
LOGI("BpAddService::create remote()->transact()/n");
TestAbinder->transact(CALCULATE_DIV_NUM,data,&reply);
sum = reply.readInt32();
LOGI("CALCULATE_SUB_NUM value = %d",sum);
```

```
    data.writeInt32(1000);
data.writeInt32(200);
LOGI("BpAddService::create remote()->transact()/n");
TestBbinder->transact(CALCULATE_ADD_NUM,data,&reply);
sum = reply.readInt32();
LOGI("CALCULATE_MUL_NUM value = %d",sum);

    data.writeInt32(1000);
data.writeInt32(200);
LOGI("BpAddService::create remote()->transact()/n");
TestBbinder->transact(CALCULATE_SUB_NUM,data,&reply);
sum = reply.readInt32();
LOGI("CALCULATE_DIV_NUM value = %d",sum);

    return 0;
}
```

这里最重要的就是通过 defaultServiceManager 得到默认的 sm,然后通过 getService 得到 sp<IBinder>对象,即可操作相应服务进程的接口函数，整个过程还是相当清晰的。

最后附上测试的结果打印：

```
    # ./TestService
./TestService
# logcat
logcat
--------- beginning of /dev/log/main
I/               ( 1379): testService main is call...
E/               ( 1379): TestA::getAddService 0xa680/n
I/               ( 1379): TestA::getAddService success...
E/               ( 1379): TestB::getAddService 0xa680/n
```

```
I/               ( 1379): TestB::getAddService success...
I/               ( 1379): BpAddService::create remote()->transact()/n
I/               ( 1371): serviceTestA::onTransact code = 0
I/               ( 1371): sum mul value = 200000
I/               ( 1379): CALCULATE_MUL_NUM value = 200000
I/               ( 1379): BpAddService::create remote()->transact()/n
I/               ( 1371): serviceTestA::onTransact code = 1
I/               ( 1371): sum div value = 5
I/               ( 1379): CALCULATE_DIV_NUM value = 5
I/               ( 1379): BpAddService::create remote()->transact()/n
I/               ( 1374): serviceTestB::onTransact code = 0
I/               ( 1374): sum add value = 1200
I/               ( 1379): CALCULATE_ADD_NUM value = 1200
I/               ( 1379): BpAddService::create remote()->transact()/n
I/               ( 1374): serviceTestB::onTransact code = 1
I/               ( 1374): sum sub value = 800
```

I/   (1379): CALCULATE_SUB_NUM value = 800

结果表明完全正确