**Android的通信，整理如下**

**进程内通信**

1. Intent
2. Broadcast
3. Extends binder
4. Messager
5. Static method

**进程间通信**

1. Intent
2. Broadcast
3. Messager
4. AIDL

**进程间共享数据**

1. ContentProvider

**通信的实现可以归类为下面三种**

1. 启动的时候通过Intent传值，比如调用StartActivity，StartService。只是启动时可用，不能中途通信。(Intent)
2. Broadcast，使用简单。(Broadcast)
3. 通过BindService建立通信，能中途通信。 (Extends binder，Messager,，AIDL)

参阅资料

Activity与Service通信的方式有三种：

<http://handsomeliuyang.iteye.com/blog/1289481>

[startservice与bindservice混合使用问题](http://blog.csdn.net/fantcy/article/details/8547172)

<http://blog.csdn.net/fantcy/article/details/8547172>

[android中的category](http://blog.csdn.net/ygc87/article/details/7480695)

<http://blog.csdn.net/ygc87/article/details/7480695>

[android activity详解](http://blog.csdn.net/nkmnkm/article/details/7103907) 好多详解 四大组件详解 全讲

<http://blog.csdn.net/nkmnkm/article/details/7101178>

**Intent**

Android为了屏蔽进程的概念，利用不同的组件[Activity、Service]来表示进程之间的通信！

组件间通信的核心机制是Intent，通过Intent可以开启一个Activity或Service，不论这个Activity或Service是属于当前应用还是其它应用的！

Intent包含两部分：

1、目的[action]

2、内容[category、data]区分性数据或内容性数据

Intent类型：

1、显式--直接指定消息目的地，只适合同一进程内的不同组件之间通信

new Intent(this,Target.class)

2、隐式--AndroidMainifest.xml中注册，一般用于跨进程通信

new Intent(String action)

FLAG\_ACTIVITY\_NEW\_TASK标记效果是一样的了。注意如果试图从非activity的非正常途径启动一个activity，比如从一个service中启动一个activity，则intent比如要添加FLAG\_ACTIVITY\_NEW\_TASK标记（编者按：activity要存在于activity的栈中，而非activity的途径启动activity时必然不存在一个activity的栈，所以要新起一个栈装入启动的activity）。简而言之，跳转到的activity根据情况，可能压在一个新建的栈中。

Intent 不能传送超过40k的东西，否则会报错误android FAILED Binder Transaction。

要从其他地方调用Activity

<category android:name=*"android.intent.category.DEFAULT"* />

不能只有

<action android:name=*"android.intent.action.MAIN"* />

跨进程通过Intent调用Activity和Service，需要打开android:exported="true"。

Service的话需要设置权限，例如android:permission="com.sky.login" ， 否则会报警告Exported service does not require permission。

参阅资料

Android中 maybe missing INTERNET permission 错误解决

<http://my.oschina.net/okimbin/blog/109912>

intent百度百科

<http://baike.baidu.com/view/2456737.htm>

**Broadcast**

Android中是如何实现Broadcast机制

Broadcast机制是基于一种注册方式的，Broadcast Receiver将其特征描述并注册在系统中。根据注册时机，可以分为两类，网上有人称之为冷注册和热注册。

• 冷注册，就是Broadcast Receiver的相关信息写在配置文件中，系统会负责在相关事件发生的时候及时通知到该Broadcast Receiver。这种模式适合于这样的场景：某事件发生 -> 通知Broadcast -> 启动相关处理应用。比如，监听来电、邮件、短信之类的，都隶属于这种模式。APK启动的时候，就开始监听。

• 热注册，顾名思义，注册这样的事情都是由应用自己来处理的，通常是在OnResume事件中通过registerReceiver进行注册，在OnPause等事件中通过unregisterReceiver反注册，通过这种方式使其能够在运行期间保持对相关事件的关注。比如，一款优秀的词典软件，可能会有在运行期间关注网络状况变化的需求，使其可以在有廉价网络的时候优先使用网络查询词汇，在其他情况下，首先通过本地词库来查词。而这样的监听，只需要在其工作状态下保持就好，不运行的时候，管你是天大的网路变化，与我何干。其模式可以归结为：启动应用 -> 监听事件 -> 发生时进行处理。

参阅资料

BroadcastReceiver mReceiver=new BroadcastReceiver(){

Public void onReceive(Context context,Intent intent){

//收到Broadcast会跑到这里

}

};

IntentFilter mfilter=new IntentFilter();

mfilter.addAction(Intent.ACTION\_SCREEN\_OFF);//后附上常见的Action

registerReceiver(mReceiver,mfilter);

Android Broadcast机制深入解析

<http://sunsz.iteye.com/blog/1611129>

Android之Broadcast, BroadcastReceiver（广播）

<http://www.cnblogs.com/playing/archive/2011/03/23/1992030.html>

精通android之Broadcast Receiver

<http://blog.csdn.net/baolong47/article/details/6621900>

[Android Broadcast 和 BroadcastReceiver的权限限制](http://blog.csdn.net/javensun/article/details/7334230)

<http://blog.csdn.net/javensun/article/details/7334230>

**Extends binder**

如果不需要进行不同应用程序间的并发通信(IPC)，you should create your interface by implementing a Binder。

需要实现binder的子类，并返回这个子类对象。

参阅资料

Activity与Service通信的方式有三种：

<http://handsomeliuyang.iteye.com/blog/1289481>

**Messager**

或者你想进行IPC，但不需要处理多线程的，则implement your interface using a Messenger。

垮线程发送Messager 需要发送实现了Parcelable 接口的类，比如String要打包成Bundle发送，否则回报错java.lang.RuntimeException: Can't marshal non-Parcelable objects across processes.

参阅资料

Activity与Service通信的方式有三种：

<http://handsomeliuyang.iteye.com/blog/1289481>

**AIDL**

AIDL (Android Interface Definition Language) 是一种IDL 语言，用于生成可以在Android设备上两个进程之间进行进程间通信(interprocess communication, IPC)的代码。如果在一个进程中（例如Activity）要调用另一个进程中（例如Service）对象的操作，就可以使用AIDL生成可序列化的参数，跨越进程的界限，进行通信。

AIDL IPC机制是面向接口的，像COM或Corba一样，但是更加轻量级。它是使用代理类在客户端和实现端传递数据。它只适用于Activity与Service之间的通信，类似于远程方法调用，就像是C/S模式的访问，通过定义AIDL接口文件来定义一个IPC接口，Server端实现IPC接口，Client端调用IPC接口的本地代理。

其他种通信方式，是基于AIDL实现的。

选择AIDL的使用场合

官方文档特别提醒我们何时使用AIDL是必要的：只有你允许客户端从不同的应用程序为了进程间的通信而去访问你的service，以及想在你的service处理多线程。

建立AIDL服务的步骤　　建立AIDL服务要比建立普通的服务复杂一些，具体步骤如下：

（1）在Eclipse Android工程的Java包目录中建立一个扩展名为aidl的文件。该文件的语法类似于Java代码，但会稍有不同。详细介绍见实例的内容。

（2）如果aidl文件的内容是正确的，ADT会自动生成一个Java接口文件（\*.java）。

（3）建立一个服务类（Service的子类）。

（4）实现由aidl文件生成的Java接口。

（5）在AndroidManifest.xml文件中配置AIDL服务，尤其要注意的是，<action>标签中android:name的属性值就是客户端要引用该服务的ID，也就是Intent类的参数值。

ADIL支持的数据类型：

java语言原语（int ,boolean,float,char等）

String和Charsequence

List(包含类的)对象，其中每一个元素都是支持的类型，接收类将总是接收实例化为ArrayList的list对象

Map(不包含类的)对象，其中每一个key和元素都是支持的类型，接收对象将总是接收实例化为HashMap的Map对象，

AIDL生成的接口，这些类型是需要import语句，

实现Parcelable接口的类，这些类型是需要import语句

AIDL中的对象，必须实现Parcelable接口。

在别的APK中使用AIDL，需要将AIDL拷贝过去，并且放在跟原AIDL一致的包内。实现Parcelable接口的类，也需要拷贝过去，放在原来的包下，并且要声明一个.aidl文件，在主AIDL文件中Import此对象。

用通俗易懂的语言详细解说android AIDL

<http://www.androidkaifa.com/thread-142-1-1.html>

android进程间通信：使用AIDL

<http://blog.csdn.net/saintswordsman/article/details/5130947>

理解Android系统的进程间通信原理(一）----RPC中的代理模式

<http://blog.csdn.net/stonecao/article/details/6425019>

Android 中Parcelable的作用

<http://blog.sina.com.cn/s/blog_78e3ae430100pxba.html>

[Android Parcelable的使用](http://blog.csdn.net/zyc13701469860/article/details/6429934)

<http://blog.csdn.net/zyc13701469860/article/details/6429934>

**ContentProvider**

1.适用场景

1) ContentProvider为存储和读取数据提供了统一的接口

2) 使用ContentProvider，应用程序可以实现数据共享

3) android内置的许多数据都是使用ContentProvider形式，供开发者调用的(如视频，音频，图片，通讯录等)

2.相关概念介绍

1）ContentProvider简介

当应用继承ContentProvider类，并重写该类用于提供数据和存储数据的方法，就可以向其他应用共享其数据。虽然使用其他方法也可以对外共享数据，但数据访问方式会因数据存储的方式而不同，如：采用文件方式对外共享数据，需要进行文件操作读写数据；采用sharedpreferences共享数据，需要使用sharedpreferences API读写数据。而使用ContentProvider共享数据的好处是统一了数据访问方式。

2）Uri类简介

Uri uri = Uri.parse("content://com.changcheng.provider.contactprovider/contact")

在Content Provider中使用的查询字符串有别于标准的SQL查询。很多诸如select, add, delete, modify等操作我们都使用一种特殊的URI来进行，这种URI由3个部分组成， “content://”, 代表数据的路径，和一个可选的标识数据的ID。以下是一些示例URI:

content://media/internal/images 这个URI将返回设备上存储的所有图片

content://contacts/people/ 这个URI将返回设备上的所有联系人信息

content://contacts/people/45 这个URI返回单个结果（联系人信息中ID为45的联系人记录）

　　尽管这种查询字符串格式很常见，但是它看起来还是有点令人迷惑。为此，Android提供一系列的帮助类（在android.provider包下），里面包含了很多以类变量形式给出的查询字符串，这种方式更容易让我们理解一点，因此，如上面content://contacts/people/45这个URI就可以写成如下形式：

　　Uri person = ContentUris.withAppendedId(People.CONTENT\_URI, 45);

AndroidManifest.xml

<provider

android:name=*"com.sky.ServiceContentProvider"*

android:authorities=*"com.sky.login"*

android:enabled=*"true"*

android:exported=*"true"*

android:grantUriPermissions=*"true"*

android:multiprocess=*"true"*

android:readPermission=*"com.sky.login"*

android:writePermission=*"com.sky.login"* >

</provider>

用于共享数据库，用法跟数据库差不多，有读写权限。

参阅资料

contentprovider的学习实例总结

<http://www.cnblogs.com/chenglong/articles/1892029.html>

Android之ContentProvider总结

<http://www.cnblogs.com/qingblog/archive/2012/07/02/2572977.html>

官方

<http://developer.android.com/guide/topics/providers/content-provider-basics.html>

**Static method**

静态方法，比如常见的单例，获得对象，即可进行通信。

传递数据可能的方式

1. Intent传递小于40K的数据。
2. 通过Indtent传递大数据的本地文件地址，或者网络的url。
3. 使用远程调用的形式，比如AIDL，将数据存在一个公共的中间内存块，再取出来。
4. 通过ContentProvider存取。