**1、Service 是否在 main thread 中执行, service 里面是否能执行耗时的操作?**  
默认情况,如果没有显示的指 servic 所运行的进程, Service 和 activity 是运  
行在当前 app 所在进程的 main thread(UI 主线程)里面。  
service 里面不能执行耗时的操作(网络请求,拷贝数据库,大文件 )  
特殊情况 ,可以在清单文件配置 service 执行所在的进程 ,让 service 在另  
外的进程中执行  
<service  
android:name=*"com.baidu.location.f"*  
android:enabled=*"true"*  
android:process=*":remote"* >  
</service>

**2、Activity 怎么和 Service 绑定， 怎么在 Activity 中启动自己对应的 Service？**  
Activity 通过 bindService(Intent service, ServiceConnection conn, int  
flags)跟 Service 进行绑定，当绑定成功的时候 Service 会将代理对象通过回调  
的形式传给 conn，这样我们就拿到了 Service 提供的服务代理对象。  
在 Activity 中可以通过 startService 和 bindService 方法启动 Service。一  
般情况下如果想获取 Service 的服务对象那么肯定需要通过 bindService（）方  
法，比如音乐播放器，第三方支付等。如果仅仅只是为了开启一个后台任务那么  
可以使用 startService（）方法。

**3、请描述一下 Service 的生命周期**  
Service 有绑定模式和非绑定模式，以及这两种模式的混合使用方式。不同  
的使用方法生命周期方法也不同。

非 绑 定 模 式 ： 当 第 一 次 调 用 startService 的 时 候 执 行 的 方 法 依 次 为  
onCreate()、onStartCommand()，当 Service 关闭的时候调用 onDestory 方法。  
绑定模式：第一次 bindService（）的时候，执行的方法为 onCreate()、onBind(）解除绑定的时候会执行 onUnbind()、onDestory()。  
上面的两种生命周期是在相对单纯的模式下的情形。我们在开发的过程中还  
必须注意 Service 实例只会有一个，也就是说如果当前要启动的 Service 已经存  
在了那么就不会再次创建该 Service 当然也不会调用 onCreate（）方法。

一个 Service 可以被多个客户进行绑定，只有所有的绑定对象都执行了onBind（）方法后该 Service 才会销毁，不过如果有一个客户执行了 onStart()方法，那么这个时候如果所有的 bind 客户都执行了 unBind()该 Service 也不会  
销毁。

**4、什么是 IntentService？有何优点？**

一、IntentService 简介  
IntentService 是 Service 的子类，比普通的 Service 增加了额外的功能。  
先看 Service 本身存在两个问题：  
Service 不会专门启动一条单独的进程， Service 与它所在应用位于同一个进  
程中；  
Service 也不是专门一条新线程，因此不应该在 Service 中直接处理耗时的  
任务；  
二、IntentService 特征  
会创建独立的 worker 线程来处理所有的 Intent 请求；  
会创建独立的 worker 线程来处理 onHandleIntent()方法实现的代码，无需  
处理多线程问题；  
所有请求处理完成后，IntentService 会自动停止，无需调用 stopSelf()方法  
停止 Service；  
为 Service 的 onBind()提供默认实现，返回 null；  
为 Service 的 onStartCommand 提供默认实现， 将请求 Intent 添加到队列中；  
使用 IntentService本 人 写 了 一 个 IntentService 的 使 用 例 子 供 参 考 。 该 例 子 中 一 个MainActivity 一个 MyIntentService，这两个类都是四大组件当然需要在清单  
文件中注册。这里只给出核心代码：  
MainActivity.java:

**public void** click(View view){  
Intent intent = **new** Intent(**this**, MyIntentService.**class**);  
intent.putExtra("start", "MyIntentService");  
startService(intent);  
}

MyIntentService.java

**public class** MyIntentService **extends** IntentService {  
**private** String ex = "";  
**private** Handler mHandler = **new** Handler() {  
**public void** handleMessage(android.os.Message msg) {  
Toast.*makeText*(MyIntentService.**this**, "-e " + ex,  
Toast.*LENGTH\_LONG*).show();  
}  
};  
**public** MyIntentService(){  
**super**("MyIntentService");  
}  
@Override  
**public int** onStartCommand(Intent intent, **int** flags, **int** startId) {  
ex = intent.getStringExtra("start");  
**return super**.onStartCommand(intent, flags, startId);  
}  
@Override  
**protected void** onHandleIntent(Intent intent) {  
/\*\*  
\* 模拟执行耗时任务  
\* 该方法是在子线程中执行的，因此需要用到 handler 跟主线程进行通信  
\*/  
**try** {  
Thread.*sleep*(1000);  
} **catch** (InterruptedException e) {  
e.printStackTrace();  
}  
mHandler.sendEmptyMessage(0);  
**try** {  
Thread.*sleep*(1000);  
} **catch** (InterruptedException e) {  
e.printStackTrace();  
}  
}  
}

**5、说说 Activity、Intent、Service 是什么关系**

他们都是 Android 开发中使用频率最高的类。 其中 Activity 和 Service 都是  
Android 四大组件之一。 他俩都是 Context 类的子类 ContextWrapper 的子类，  
因此他俩可以算是兄弟关系吧。不过兄弟俩各有各自的本领，Activity 负责用户  
界面的显示和交互，Service 负责后台任务的处理。Activity 和 Service 之间可  
以通过 Intent 传递数据，因此可以把 Intent 看作是通信使者。

**6、Service 和 Activity 在同一个线程吗**

对于同一 app 来说默认情况下是在同一个线程中的，main Thread （UI  
Thread）。

**7、Service 里面可以弹吐司么**

可以的。弹吐司有个条件就是得有一个 Context 上下文，而 Service 本身就是  
Context 的子类， 因此在 Service 里面弹吐司是完全可以的。 比如我们在 Service  
中完成下载任务后可以弹一个吐司通知用户。

**8、什么是 Service 以及描述下它的生命周期。Service 有哪些启动方法，有什么区别，怎样停用 Service？**在 Service 的生命周期中，被回调的方法比 Activity 少一些，只有 onCreate,  
onStart, onDestroy,  
onBind 和 onUnbind。  
通常有两种方式启动一个 Service,他们对 Service 生命周期的影响是不一样的。  
1． 通过 startService  
Service 会经历 onCreate 到 onStart，然后处于运行状态，stopService  
的时候调用 onDestroy 方法。  
如果是调用者自己直接退出而没有调用 stopService 的话，Service 会一直  
在后台运行。  
2． 通过 bindService  
Service会运行onCreate， 然后是调用onBind，这个时候调用者和Service绑定在一起。调用者退出了，Srevice 就会调用 onUnbind->onDestroyed 方法。  
所谓绑定在一起就共存亡了。调用者也可以通过调用 unbindService 方法来停止服务，这时候 Srevice 就会调用 onUnbind->onDestroyed 方法。  
需要注意的是如果这几个方法交织在一起的话，会出现什么情况呢？  
一个原则是 Service 的 onCreate 的方法只会被调用一次，就是你无论多少次的startService 又 bindService，Service 只被创建一次。  
如果先是 bind 了，那么 start 的时候就直接运行 Service 的 onStart 方法，如果先是 start，那么 bind 的时候就直接运行 onBind 方法。

如果 service 运行期间调用了 bindService，这时候再调用 stopService 的话，service 是不会调用 onDestroy 方法的，service 就 stop 不掉了，只能调用UnbindService, service 就会被销毁如果一个 service 通过 startService 被 start 之后，多次调用 startService 的话，service 会多次调用 onStart 方法。多次调用 stopService 的话，service只会调用一次 onDestroyed 方法。  
如果一个 service 通过 bindService 被 start 之后， 多次调用 bindService 的话，service 只会调用一次 onBind 方法。  
多次调用 unbindService 的话会抛出异常。

**9、在 service 的生命周期方法 onstartConmand()可不可以执行网络操作？如何在 service 中执行网络操作？**

可以直接在 Service 中执行网络操作,在 onStartCommand()方法中可以执行网  
络操作

**10、请描述一下 BroadcastReceiver**  
BroadCastReceiver 是 Android 四大组件之一，主要用于接收系统或者 app 发  
送的广播事件。  
广播分两种：有序广播和无序广播。  
内部通信实现机制：通过 Android 系统的 Binder 机制实现通信。  
无序广播：完全异步，逻辑上可以被任何广播接收者接收到。优点是效率较高。  
缺点是一个接收者不能将处理结果传递给下一个接收者， 并无法终止广播 intent  
的传播。  
有序广播：按照被接收者的优先级顺序，在被接收者中依次传播。比如有三个广播接收者 A，B，C，优先级是 A > B > C。那这个消息先传给 A，再传给 B，最后传给 C。每个接收者有权终止广播，比如 B 终止广播，C 就无法接收到。此外A 接收到广播后可以对结果对象进行操作，当广播传给 B 时，B 可以从结果对象中取得 A 存入的数据。

在通过 Context.sendOrderedBroadcast(intent, receiverPermission,resultReceiver, scheduler, initialCode, initialData, initialExtras)时我们可以指定 resultReceiver 广播接收者，这个接收者我们可以认为是最终接收者，通常情况下如果比他优先级更高的接收者如果没有终止广播， 那么他的 onReceive会被执行两次，第一次是正常的按照优先级顺序执行，第二次是作为最终接收者接收。如果比他优先级高的接收者终止了广播，那么他依然能接收到广播。在我们的项目中经常使用广播接收者接收系统通知，比如开机启动、sd 挂载、低电量、外播电话、锁屏等。如果我们做的是播放器，那么监听到用户锁屏后我们应该将我们的播放之暂停等。

**11、在 manifest 和 代 码 中 如 何 注 册 和 使 用  
BroadcastReceiver**

在清单文件中注册广播接收者称为静态注册，在代码中注册称为动态注册。  
静态注册的广播接收者只要 app 在系统中运行则一直可以接收到广播消息，动  
态注册的广播接收者当注册的Activity或者Service销毁了那么就接收不到广播  
了。  
静态注册：在清单文件中进行如下配置  
<receiver android:name=*".BroadcastReceiver1"* >  
<intent-filter>  
<action android:name=*"android.intent.action.CALL"* >  
</action>  
</intent-filter>  
</receiver>

动态注册：在代码中进行如下注册

receiver = **new** BroadcastReceiver();  
IntentFilter intentFilter = **new** IntentFilter();  
intentFilter.addAction(CALL\_ACTION);  
**context**.registerReceiver(receiver, intentFilter);

**12、注册广播接收者两种方式的区别，及优缺点**

1)第一种是常驻型（静态注册），也就是说当应用程序关闭后，如果有信息广播来，程序也会被系统调用自动运行。  
2)第二种不是常驻型广播（动态注册），也就是说广播跟随程序的生命周期。  
注册的方法有两种，一种是静态注册，一种是动态注册。  
动态注册优点：在 Android 的广播机制中，动态注册的优先级是要高于静态注册优先级的，因此在必要的情况下，我们是需要动态注册广播接收器的。  
静态注册优点：动态注册广播接收器还有一个特点，就是当用来注册的 Activity 关掉后，  
广播也就失效了。同时反映了静态注册的一个优势，就是无需担忧广播接收器是否被关闭，只要设备是开启状态，广播接收器就是打开着的。

**13、BroadCastReceiver 的生命周期**  
a. 广播接收者的生命周期非常短暂的，在接收到广播的时候创建，  
onReceive()方法结束之后销毁；  
b. 广播接收者中不要做一些耗时的工作，否则会弹出 Application No  
Response 错误对话框；  
c. 最好也不要在广播接收者中创建子线程做耗时的工作， 因为广播接收者被  
销毁后进程就成为了空进程，很容易被系统杀掉；  
d. 耗时的较长的工作最好放在服务中完成；

**14、Android 引入广播机制的用意**  
a. 从 MVC 的角度考虑(应用程序内) 其实回答这个问题的时候还可以这样问，  
android 为什么要有那 4 大组件，现在的移动开发模型基本上也是照搬的  
web 那一套 MVC 架构，只不过是改了点嫁妆而已。android 的四大组件本  
质上就是为了实现移动或者说嵌入式设备上的 MVC 架构，它们之间有时候  
是一种相互依存的关系，有时候又是一种补充关系，引入广播机制可以方便  
几大组件的信息和数据交互。  
b. 程序间互通消息(例如在自己的应用程序内监听系统来电)  
c. 效率上(参考 UDP 的广播协议在局域网的方便性)

d. 设计模式上(反转控制的一种应用，类似监听者模式)

**15、四大组件中哪些是通过意图激活的？**

Activity、Service、BroadcastReceiver，必须通过有 Intent 激活。