**武汉纺织大学数计学院**

**《移动平台应用开发》课程设计报告**

**成 绩：**

**学 号：1804280415**

**姓 名： 李雄壮**

**班 级： 物联网11804**

**指导教师： 熊炜**

**报告日期：2020 年 12月4日**

[课程设计前提： 3](#_Toc59013046)

[课程的目的： 3](#_Toc59013047)

[Service是什么？ 3](#_Toc59013048)

[Local Service（本地服务） 4](#_Toc59013049)

[Service与Activity之间通信的原理： 4](#_Toc59013050)

[Service启动方式： 4](#_Toc59013051)

[第一种启动方式： 4](#_Toc59013052)

[第二种启动方式： 5](#_Toc59013053)

[关键代码详解： 5](#_Toc59013054)

[1.bindService： 5](#_Toc59013055)

[2.unbindService： 6](#_Toc59013056)

[2.1unbindService流程： 6](#_Toc59013057)

[3.ServiceConnection： 6](#_Toc59013058)

[3.1原理步骤 6](#_Toc59013059)

[3.2实现前提： 6](#_Toc59013060)

[3.2.1：实现ServiceConnection： 6](#_Toc59013061)

[流程图： 7](#_Toc59013062)

[7](#_Toc59013063)

[大致流程： 8](#_Toc59013064)

[界面截图： 9](#_Toc59013065)

[10](#_Toc59013066)

[结果截图： 11](#_Toc59013067)

# 课程设计前提：

## 课程的目的：

绑定本地Service并与之通信。

## Service是什么？

Service是一个应用组件，它用来在后台完成一个时间跨度比较大的工作且没有关联任何界面

一个Service可以完成下面工作：

访问网络

播放音乐

文件IO操作

大数据的数据库操作

…

服务的特点：

Service在后台运行，不用与用户进行交互

即使应用退出，服务也不会停止。

在默认情况下，Service运行在应用程序进程的主线程（UI线程）中，如果要在Service中处理一些网络连接等耗时的操作，那么应该将这些任务放在分线程中处理，避免阻塞用户界面。

Local Service（本地服务）：

Service对象与Service的启动者在同个进程中运行，两者的通信是进程内通信

## Service与Activity之间通信的原理：

在开发中，经常把一些不需要与用户进行交互的工作放在service中来完成，service运行在后台，这样有些人可能会产生错觉，以为service是运行在新线程中，其实不然，service也运行在主线程中，因此不能在service中进行耗时操作，否则会报ANR异常，但是可以在service中新开线程来进行耗时操作，比如下载等等。

## Service启动方式：

### 第一种启动方式：

一般在onStart()方法中启动service（使用方法startService()或者bindService()），在onStop方法中停止service（使用方法stopService()或者unbindService()）。

在Activity中启动Service intent = new Intent(this, MyService.class);startService(intent);停止Service：stopservice(intent)

### 第二种启动方式：

1. 设计接口，通过Ibinder实现接口来对Activity暴露方法供其调用。

定义一个接口：public interface MyServiceInterface

实行接口并重写接口的抽象方法：public class MyBinder extends Binder implements MyServiceInterface

然后在连接成功使调用onServiceConnected方法

然后通过得到接口的实现类实例(MyServiceInterface) service;，向上转型使用接口的所有方法。

1. 在onBinder()中返回Binder，而在继承Binder的类中直接返回当前service对象，通过该service对象来调用service中公开的方法。

# 关键代码详解：

## 1.bindService：

bindService(service, conn, flags);

\* service ：意图

\* conn ：activity和服务的连接通道

\* flags ： BIND\_AUTO\_CREATE 连接的时候如果有服务的对象则复用，如果服务对象不存在，则创建一个服务的对象

　　\* 完整生命周期：onCreate()-->onBind()-->onUnbind()-->onDestroy()

　　\* 绑定服务：onCreate()-->onBind()

　　\* 解绑服务：onUnbind()-->onDestroy()

\* 特点：

　　1. 服务只能被绑定一次，多次绑定无效

　　2. 服务只能被解绑一次，多次解绑会抛出异常

　　3. activity和绑定的Service是同生共死

　　4. 绑定服务调用服务里的方法

\* 区别：

　　\* start开启服务可以长期运行在后台

\* bind绑定服务可以调用服务里的方法

## 2.unbindService：

### 2.1unbindService流程：

client（客户端）执行unbindServie的流程：

client与Service解除绑定时，Service先检测是否还与其他client（客户端）与其连接→

如果没有，Service执行onUnbind方法，然后在执行onDestroy方法

如果有，Service不会执行onUnbind和onDestroy方法

## 3.ServiceConnection：

3.1原理步骤：应用组件(客户端)可以调用bindService()绑定到一个service．Android系统之后调用service的onBind()方法，它返回一个用来与service交互的IBinder．

绑定是异步的．bindService()会立即返回，它不会返回IBinder给客户端．要接收IBinder，客户端必须创建一个ServiceConnection的实例并传给bindService()．ServiceConnection包含一个回调方法，系统调用这个方法来传递要返回的IBinder．

*注：只有activities,services,和contentproviders可以绑定到一个service—你不能从一个broadcastreceiver绑定到service．*

### 3.2实现前提：

从客户端绑定到一个service，必须：

#### 3.2.1：实现ServiceConnection：

实现必须重写两个回调方法：

onServiceConnected()：

系统调用这个来传送在service的onBind()中返回的IBinder．

onServiceDisconnected()：

**Android**系统在同service的连接意外丢失时调用这个．比如当service崩溃了或被强杀了．当客户端解除绑定时，这个方法不会被调用．

２调用bindService()，传给它ServiceConnection的实现．

３当系统调用你的onServiceConnected()方法时，你就可以使用接口定义的方法们开始调用service了．

４要与service断开连接，调用unbindService()．

　　当客户端被销毁，它将从service解除绑定，但是必须总是在你完成与service的交互时或当你的activity暂停于是service在不被使用时可以关闭.

## 流程图：

客户端Client Service

Service是否创建

调用bindService

否

onCreate()

IBinder

onServiceConnected（）

onBind()

Client与Service完成绑定

Service运行

调用unbindService()

是否还有其他client与Service相连

Client与Service解除绑定

否

onUnbind()

## 

onDestory

## 大致流程：

按下计算阶乘按钮后，绑定服务。

通过bindService()绑定到服务的客户端：public IBinder onBind(Intent intent)

return new Mybinder();

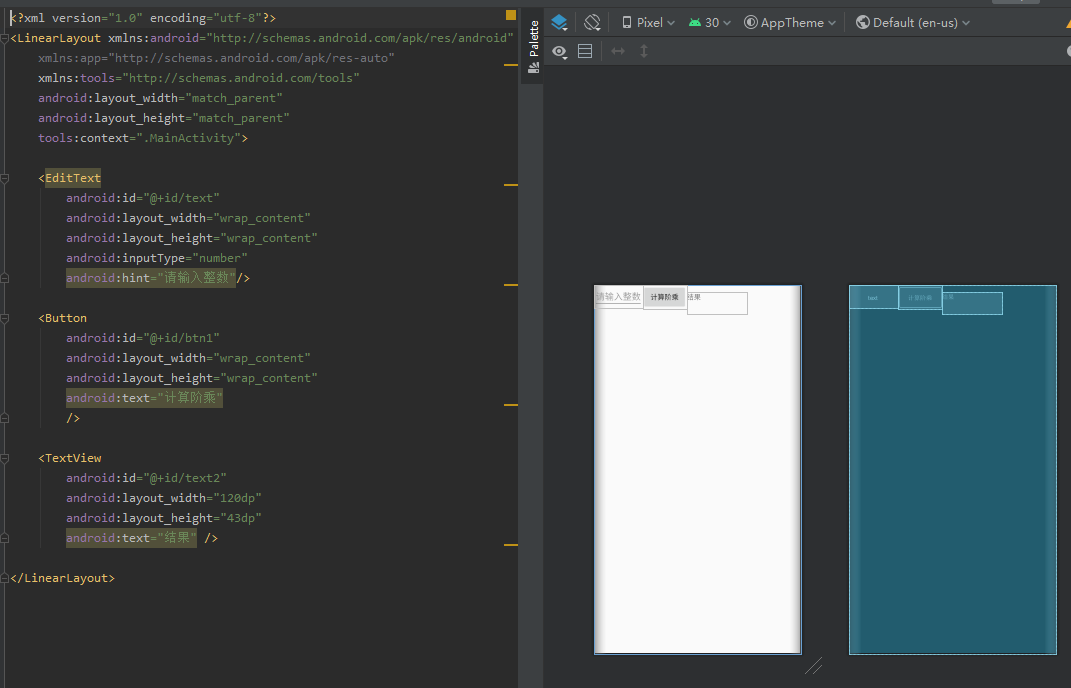
然后在OnServiceConnected中通过MyService中Mybinder中的getService方法获得 yService实例，运行在MyService中的公开方法getFactorial()得到阶乘

通过我在MainActivity中定义的textview控件显示结果：

Int factorial = myService.getFactorial();

Tx.setText(String.valueOf(factorial));

# 界面截图：



EditText:用户在活动中输入一个整数。

Button：用户单击按钮绑定服务，用服务中定义的方法计算阶乘

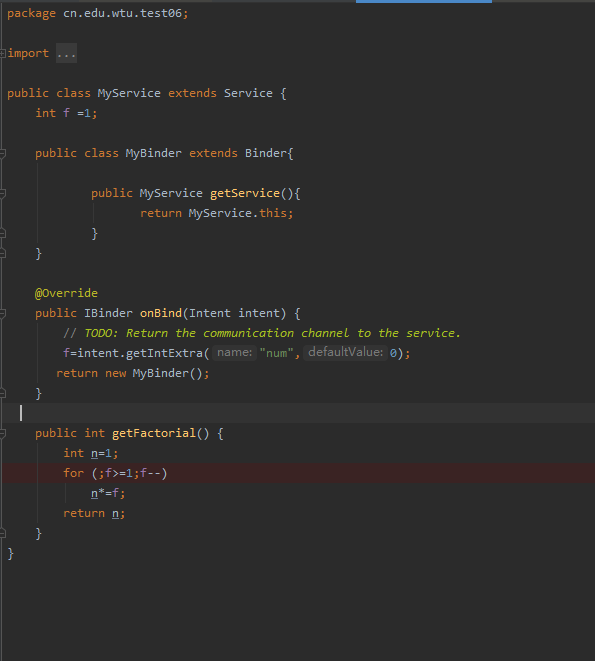
TextView：从Myservice中显示返回的结果

## 

因为绑定服务只能绑定一次，而且我的是通过按钮(计算阶乘)绑定服务，通过Intent传递用户输入的值intent.putExtra(“)，如果要想多次计算阶乘，需要在单击计算后，解绑一次，如此反复循环，可计算多次。

在计算一次阶乘后，使用handler用作于异步消息的处理，当得到阶乘不为0时，发出一个空消息：sendEmptyMessage(int what),

在初始化Handler对象时重写的handleMessage方法来接收Message并进行unbindService(connection)。

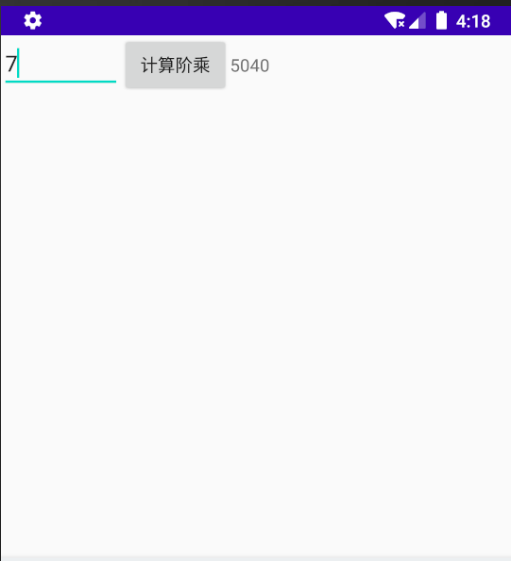


MyService ：

首先定义一个接受Activity传过来的值的整数变量，赋初值，然后通过Mybinder继承Binder

里面定义的成员方法，返回Myservice实例，然后通过Myservice实例使用类中公有方法(计算阶乘)，然后返回计算结果。

## 结果截图：



# 心得体会：

出于性能优化考虑，Android的UI操作并不是线程安全的，这意味着如果有多个线程并发操作UI组件，则可能导致线程安全问题。为了解决这个问题，Android制定了一条简单的规则：只允许UI线程修改Activity里的UI组件。

当一个程序第一次启动时，Android会同时启动一条主线程（Main Thread），主线程主要负责处理与UI相关的事件，如用户的按键事件、用户接触屏幕的事件及屏幕绘图事件，并把相关的事件分发到对应的组件进行处理。

所以耗时的任务不应该放在主线程去做，只需要调用即可。