# TODO、

SharedPreferences源码解析

<https://juejin.im/post/5ad97691518825671c0e19d9>

输入法开关，怎么回事呢，如何选择的

ULocale.getDisplayName(locale.toLanguageTag(), displayULocale)显示原理

分辨率原理，如何吧手机变成平板

config\_supportAutoRotation原理

<**string-array translatable="false" name="config\_defaultPinnerServiceFiles"**>  
</**string-array**>

干啥用的

为何加到内寸：sys/fw/boot-fram.oat

Oat/arm64/servers.odex

Arm64/boot.oat

Arm64/boot-core-libart.oat

按键映射：data/keyboards/genre.kl

Thumbal 317

304 buttom\_a

Fallback 原理

<https://www.jianshu.com/p/641d0a984362?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation>

<permission android:name="android.permission.CAMERA"

android:permissionGroup="android.permission-group.CAMERA"

android:label="@string/permlab\_camera"

android:description="@string/permdesc\_camera"

android:protectionLevel="dangerous" />

**private boolean** showNextPermissionGroupGrantRequest() {  
 **final int** groupCount = **mRequestGrantPermissionGroups**.size();  
  
 **int** currentIndex = 0;  
 **for** (GroupState groupState : **mRequestGrantPermissionGroups**.values()) {  
 **if** (groupState.**mState** == GroupState.***STATE\_UNKNOWN***) {  
 CharSequence appLabel = **mAppPermissions**.getAppLabel();  
 Spanned message = Html.*fromHtml*(getString(R.string.***permission\_warning\_template***,  
 appLabel, groupState.**mGroup**.getDescription()), 0);

Android资源管理框架

<http://zuoquanxiong.com/Android%E8%B5%84%E6%BA%90%E7%AE%A1%E7%90%86%E6%A1%86%E6%9E%B6>

阅读ANDROID源码的一些姿势

<https://zhuanlan.zhihu.com/p/20564614>

android framework层源码解析，不定时更新哈

<https://github.com/yipianfengye/androidSource>

如何改写代码呢，overlay方式添加代码

Z自动化测试原理

编译apk，自动跳转到某一个界面

# Binder.**clearCallingIdentity**()机制

注意发送该广播前有如下操作：成对出现

　　//操作前 clear  
　　long ident = Binder.**clearCallingIdentity**();  
  
　　//do相关操作  
　　……  
  
　　//操作后restore  
　　Binder.**restoreCallingIdentity**(ident);   
  
　**通常这俩都是成对出现，具体的作用简单说下：这涉及到权限管理后面会讨论。**

      　　Binder.**clearCallingIdentity**()：

　　　　　　通过IPC binder调用来远端进程，当前进程会记录调用者的PID和UID，即通常使用的getCallingPid和getCallingUid，

  　　　　　 而会**clearCallingIdentity**把调用者PID和UID清除，将其设置为当前进程的PID和UID，并将原来的PID和UID作为返回值；

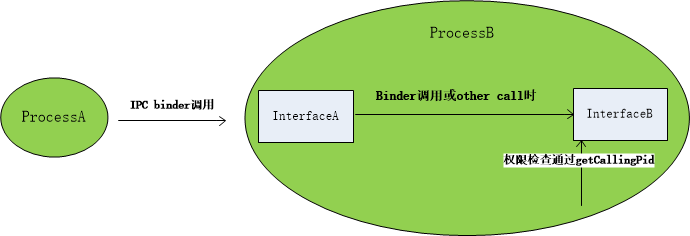
 　　　　　  PID和UID是保存在一个long型数中，通过移位计算。

           Binder.**restoreCallingIdentity**(ident)：恢复刚才清除的远端调用者的PID和UID。

**这样做有什么作用？**

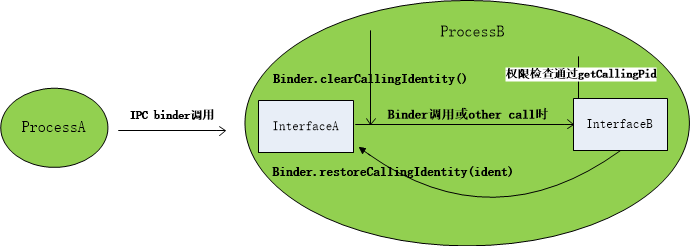
这涉及到权限管理，**clearCallingIdentity**接口注释，举了incoming call例子，看下原注释：

**大概的意思可以理解成这样：**

****

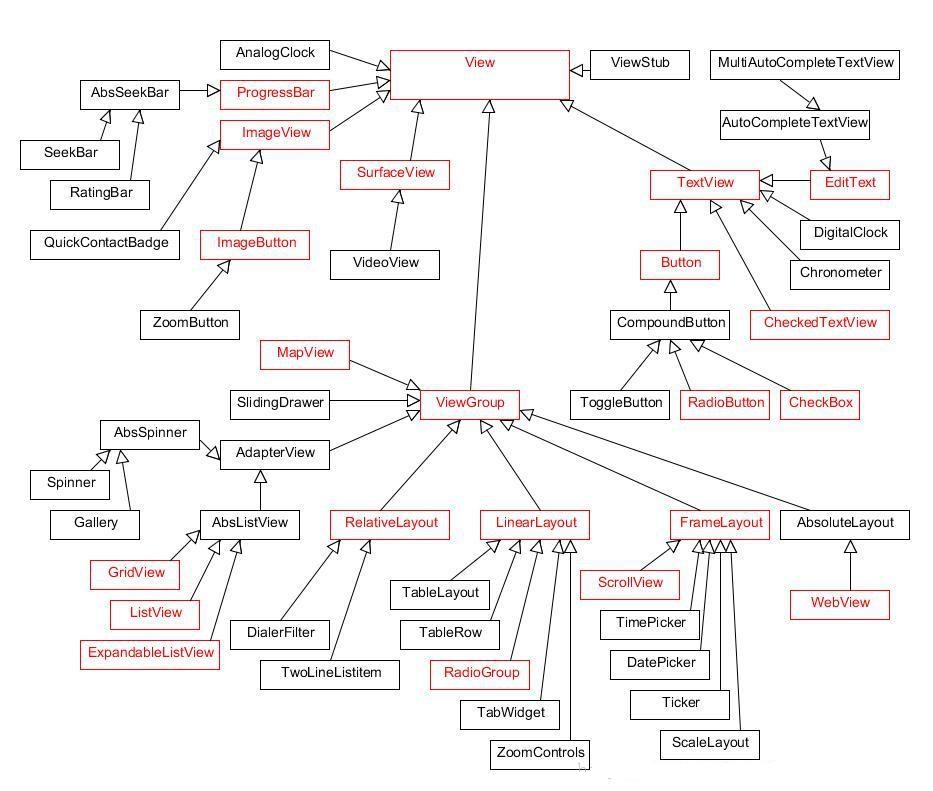
**在ProcessB中，InterfaceA调用InterfaceB时，InterfaceB中要做权限检查，通过getCallingPid，**

**这时拿到的PID是ProcessA的，权限不够肿么办。ProcessB的权限是够可以的：就可以如下面**



**代码里面很多这样的例子，具体原因请自行体会，贴一段源代码看看**

# View的继承关系图



# ViewRootImpl

# InputStage责任链机制