**做android开发时，发现一个关于android退出时不能彻底关闭的问题，比如：一个程序里new 出了N多个Thread，这样在退出程序的可能不能完全关闭，最后发现，只用finish()方法，有时候不能彻底退出，个人感觉还是要在适当的地方加上：System.exit(0);**

**-----------------------------------------------------------------------------**

**前言:**

**finish()：结束当前Activity，不会立即释放内存。遵循android内存管理机制。**

**exit()：结束当前组件如Activity，并立即释放当前Activity所占资源。**

**killProcess()：结束当前组件如Activity，并立即释放当前Activity所占资源。**

**restartPackage()：结束整个App，包括service等其它Activity组件。**

**1. finish()方法**

**该方法可以结束当前Activity，但是如果你的App有很多Activity 的话，使用该方法显得有点捉襟见肘了。**

**另外，还有一个方法finishActivity (int requestCode) ,关于这个方法，先看看sdk的api说明吧！**

**public voidfinishActivity (int requestCode)**

**Since: APILevel 1**

**Force finishanother activity that you had previously started withstartActivityForResult(Intent, int).**

**ParametersrequestCode The request code of the activity that you had given tostartActivityForResult(). If there are multiple activities started with thisrequest code, they will all be finished.**

**也许你会这样理解 ，Activity1 通过方法startActivityForResult (Intent, int) 启动Activity2，然后在Activity2 中通过方法finishActivity (intrequestCode)来结束Activity1，但是很不幸运，不是这样的。不信你可以Demo一把！**

**上面文档说得很明白，该方法强制关闭通过方法startActivityForResult (Intent, int)启动的Activity，也就是说在Activity1 中的(重写)方法onActivityResult(intrequestCode, int resultCode, Intent data) 来接收Activity2返回的结果，必须在Activity1 中调用finishActivity(int requestCode)来结束Activity2。（一般在onActivityResult 方法调用该方法结束Activity2）。**

**Force finishanother activity that you had previously started withstartActivityForResult(Intent, int).**

**Parameters**

**还有，下面两个方法，可以参阅文档以及源码研究一下。**

**finishActivityFromChild(Activity child, int requestCode)**

**finishFromChild(Activity child)**

**2. killProcess**

**通过调用android.os.Process 的相关方法，结束App，示例如下：**

**btn\_exit.setOnClickListener(new Button.OnClickListener() {**

**@Override**

**public voidonClick(View v) {**

**android.os.Process.killProcess(android.os.Process.myPid());**

**}**

**});**

**3. exit**

**我们知道，Java 的exit(int code) 方法可以退出程序，通过查看该方法源码，知道它实际上是调用下面的方法：**

**Runtime.getRuntime().exit(code);**

**示例代码，如下所示：**

**btn\_exit.setOnClickListener(new Button.OnClickListener() {**

**@Override**

**public voidonClick(View v) {**

**System.exit(0);//正常退出App**

**}**

**});**

**接下来，我们研究一下这个方法。java.lang.System这个类的该方法jdk说明：**

**exit**

**public staticvoid exit(int status)**

**终止当前正在运行的Java 虚拟机。参数用作状态码；根据惯例，非0 的状态码表示异常终止。**

**该方法调用Runtime 类中的exit 方法。该方法永远不会正常返回。**

**调用System.exit(n) 实际上等效于调用：**

**Runtime.getRuntime().exit(n)**

**参数：**

**status - 退出状态。**

**抛出：**

**SecurityException - 如果安全管理器存在并且其checkExit 方法不允许以指定状态退出。**

**另请参见：**

**Runtime.exit(int)**

**也就是说，参数为非0值的话是异常退出程序。参数为0的话，就是正常退出。**

**看RunTime这个类的该方法源码：**

**public voidexit(int status) {**

**SecurityManager security = System.getSecurityManager();**

**if (security!= null) {**

**security.checkExit(status);**

**}**

**Shutdown.exit(status);**

**}**

**其api说明：**

**exit**

**public voidexit(int status)**

**通过启动虚拟机的关闭序列，终止当前正在运行的Java 虚拟机。此方法从不正常返回。可以将变量作为一个状态码；根据惯例，非零的状态码表示非正常终止。**

**虚拟机的关闭序列包含两个阶段。在第一个阶段中，会以某种未指定的顺序启动所有已注册的关闭钩子(hook)（如果有的话），并且允许它们同时运行直至结束。在第二个阶段中，如果已启用退出终结，则运行所有未调用的终结方法。一旦完成这个阶段，虚拟机就会暂停。**

**如果在虚拟机已开始其关闭序列后才调用此方法，那么若正在运行关闭钩子，则将无限期地阻断此方法。如果已经运行完关闭钩子，并且已启用退出终结(on-exit finalization)，那么此方法将利用给定的状态码（如果状态码是非零值）暂停虚拟机；否则将无限期地阻断虚拟机。**

**System.exit 方法是调用此方法的一种传统而便捷的方式。**

**参数：**

**status - 终止状态。按照惯例，非零的状态码表明非正常终止。**

**抛出：**

**SecurityException - 如果安全管理器存在，并且其checkExit 方法不允许存在指定的状态**

**另请参见：**

**SecurityException, SecurityManager.checkExit(int),addShutdownHook(java.lang.Thread), removeShutdownHook(java.lang.Thread),runFinalizersOnExit(boolean), halt(int)**

**该方法又是调用Shutdown这个类的exit()方法。**

**static voidexit(int status) {**

**booleanrunMoreFinalizers = false;**

**synchronized(lock) {**

**if (status !=0) runFinalizersOnExit = false;**

**switch (state){**

**case RUNNING:/\* Initiate shutdown \*/**

**state = HOOKS;**

**break;**

**case HOOKS: /\*Stall and halt \*/**

**break;**

**caseFINALIZERS:**

**if (status !=0) {**

**/\* Haltimmediately on nonzero status \*/**

**halt(status);**

**} else {**

**/\*Compatibility with old behavior:**

**\* Run morefinalizers and then halt**

**\*/**

**runMoreFinalizers = runFinalizersOnExit;**

**}**

**break;**

**}**

**}**

**if(runMoreFinalizers) {**

**runAllFinalizers();**

**halt(status);**

**}**

**synchronized(Shutdown.class) {**

**/\* Synchronizeon the class object, causing any other thread**

**\* thatattempts to initiate shutdown to stall indefinitely**

**\*/**

**sequence();**

**halt(status);**

**}**

**}**

**其中，runAllFinalizers()是一个本地方法：**

**JNIEXPORT voidJNICALL**

**Java\_java\_lang\_Shutdown\_runAllFinalizers(JNIEnv \*env, jclass ignored)**

**{**

**jclass cl;**

**jmethodID mid;**

**if ((cl =(\*env)->FindClass(env, "java/lang/ref/Finalizer"))**

**&&(mid = (\*env)->GetStaticMethodID(env, cl,**

**"runAllFinalizers", "()V"))) {**

**(\*env)->CallStaticVoidMethod(env, cl, mid);**

**}**

**}**

**System.exit()的参数是把退出原因返回给系统, 一般来说可以是任何的整数 。**

**0表示正常退出,1表示非正常 。**

**最后说一下finish()与exit方法的区别：**

**finish()是Activity的类方法，仅仅针对Activity，当调用finish()时，只是将活动推向后台，并没有立即释放内存，活动的资源并没有被清理；当调用System.exit(0)时，退出当前Activity并释放资源（内存），但是该方法不可以结束整个App如有多个Activty或者有其他组件service等不会结束。**

**其实android的机制决定了用户无法完全退出应用，当你的application最长时间没有被用过的时候，android自身会决定将application关闭了。**

**4. restartPackage方法**

**ActivityManager manager = (ActivityManager)getSystemService(Context.ACTIVITY\_SERVICE);**

**manager.restartPackage(getPackageName());**

**首先需要创建ActivityManager对象，然后调用restartPackage()方法（如果有兴趣的话，可以看源码）。**

**注意：getPackageName获得当前应用包名称，如mark.zhang**

**使用这种方式来退出App，需要权限：**

**<uses-permissionandroid:name="android.permission.RESTART\_PACKAGES" />**

**更加详细的说明，如下：**

**voidandroid.app.ActivityManager.restartPackage(String packageName)**

**Have thesystem perform a force stop of everything associated with the given applicationpackage. All processes that share its uid will be killed, all services it hasrunning stopped, all activities removed, etc. In addition, aIntent.ACTION\_PACKAGE\_RESTARTED broadcast will be sent, so that any of itsregistered alarms can be stopped, notifications removed, etc.**

**You must holdthe permission android.Manifest.permission.RESTART\_PACKAGES to be able to callthis method.**

**Parameters:**

**packageNameThe name of the package to be stopped.**

**可以看出，相同的UID的进程会被kill，还会停止相关的服务以及移除所有的Activity，并且会发送一个广播。**

**注意一个问题：在android2.2之后，该方法不可以将应用程序结束，需要使用ActivityManager类的下面这个方法：**

**public voidkillBackgroundProcesses (String packageName)**

**api 文档说的很清楚：**

**public voidrestartPackage (String packageName)**

**Since: APILevel 3**

**This method isdeprecated.**

**This is nowjust a wrapper for killBackgroundProcesses(String); the previous behavior hereis no longer available to applications because it allows them to break otherapplications by removing their alarms, stopping their services, etc.**

**另外，需要使用权限：**

**<uses-permissionandroid:name="android.permission.KILL\_BACKGROUND\_PROCESSES"/>**

**但是不管你怎么样折腾，还是无法退出App，呜呼哀哉！这里给出一个方法：**

**int currentVersion= android.os.Build.VERSION.SDK\_INT;**

**if(currentVersion > android.os.Build.VERSION\_CODES.ECLAIR\_MR1) {**

**Intent startMain = new Intent(Intent.ACTION\_MAIN);**

**startMain.addCategory(Intent.CATEGORY\_HOME);**

**startMain.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);**

**startActivity(startMain);**

**System.exit(0);**

**} else {//android2.1**

**ActivityManager am = (ActivityManager)getSystemService(ACTIVITY\_SERVICE);**

**am.restartPackage(getPackageName());**

**}**

**5. 小结**

**finish()：结束当前Activity，不会立即释放内存。遵循android内存管理机制。**

**exit()：结束当前组件如Activity，并立即释放当前Activity所占资源。**

**killProcess()：结束当前组件如Activity，并立即释放当前Activity所占资源。**

**restartPackage()：结束整个App，包括service等其它Activity组件。**

**特别注意：除了finish()方法可以调用Activity的生命周期方法如onStop()、onDestroy()，其余三种退出App均不会调用Activity的生命周期方法。除非，在调用这几个方法之前或者之后主动调用Activity的生命周期方法。如：**

**System.exit(int);**

**onDestroy();**