**Android应用启动流程**

Android系统源码每个版本都有不小的差异，以下探究基于Android API 29。

将回答以下问题

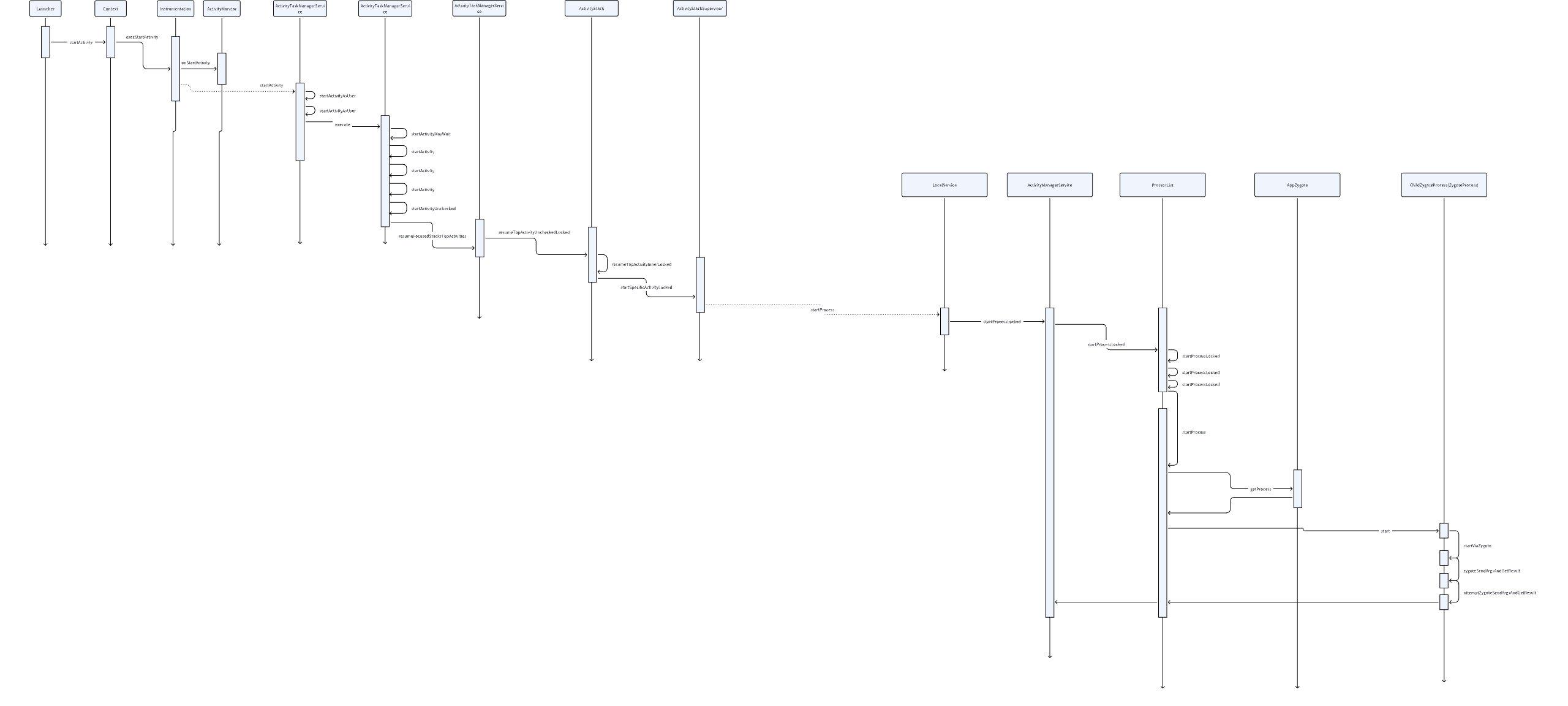
1. activity如何进行合法性检测
2. uid、pid、gid概念
3. 应用的uid、pid、gid如何设置的
4. 应用进程在哪里被启动
5. 应用资源加载时机
6. 应用application如何的创建
7. 应用间如何通过startActivity传数据的
8. 是否可以直接启动一个jar程序而不是启动apk

**从Launcher启动其它应用**

启动一个应用的方式：context.startActivity(..)

|  |
| --- |
| Kotlin Intent intent = mContext.getPackageManager().getLaunchIntentForPackage(pkgName); if (intent != null) {  mContext.startActivity(intent); } |

跟踪startActivity调用链：



Context.startActivity()

Instrumentation.execStartActivity()

ActivityTaskManager.getService().startActivity() // ActivityTaskManagerService

Instrumentation.*checkStartActivityResult() // 检测启动结果*

ActivityTaskManagerService.startActivityAsUser()

com.android.server.wm.ActivityStarter#execute()

com.android.server.wm.ActivityStarter#startActivityMayWait

com.android.server.wm.ActivityStarter#startActivity()

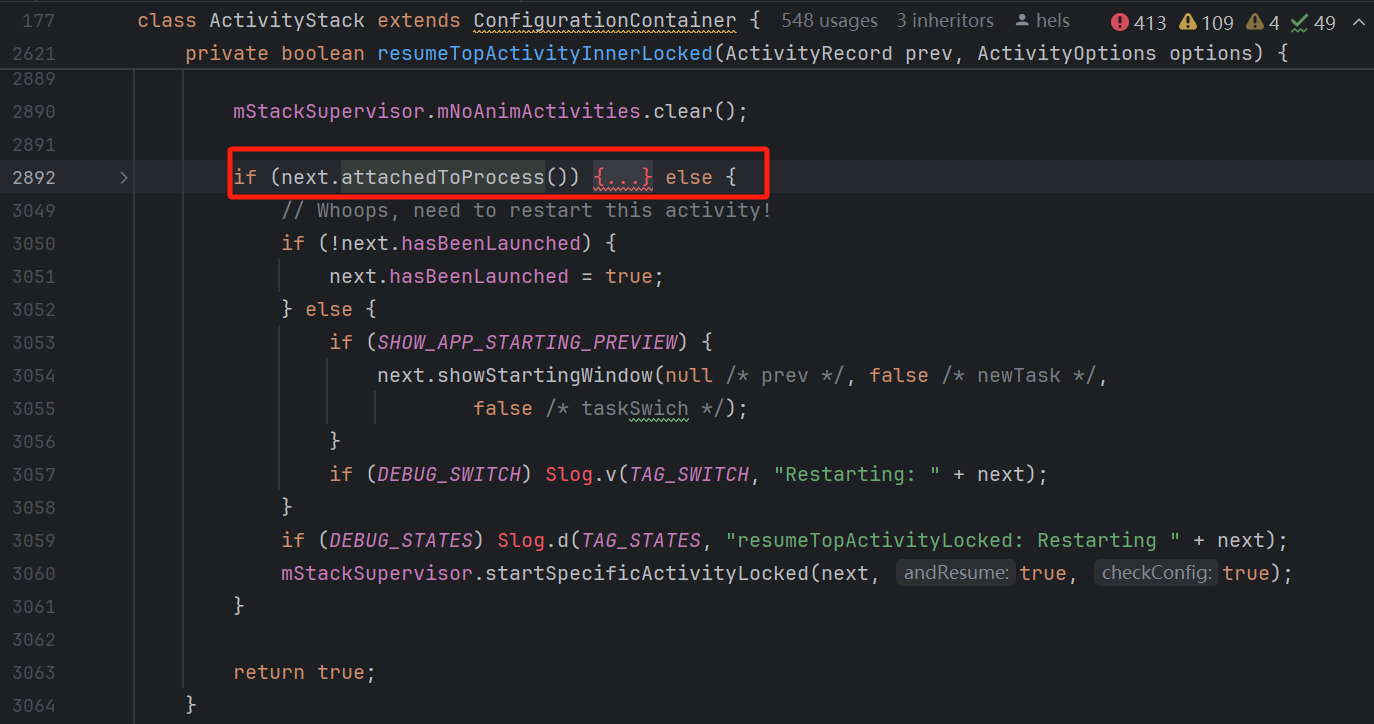
com.android.server.wm.ActivityStarter#startActivity()

com.android.server.wm.ActivityStarter#startActivity()

com.android.server.wm.ActivityStarter#startActivityUnchecked

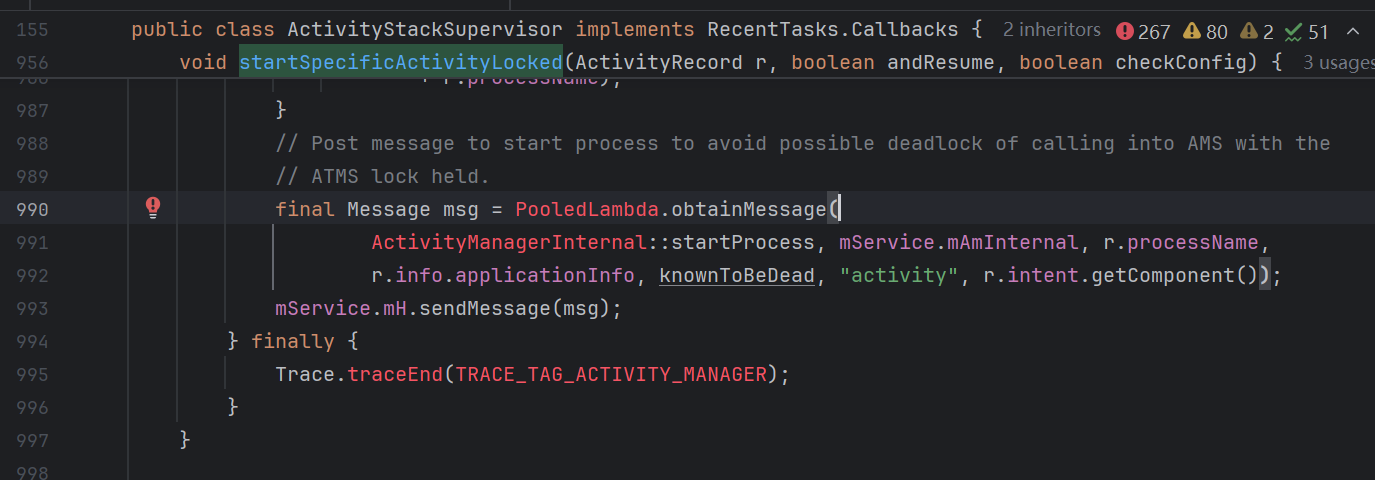
com.android.server.wm.RootActivityContainer#resumeFocusedStacksTopActivities

com.android.server.wm.RootActivityContainer#resumeFocusedStacksTopActivities()



判断进程是否存在

com.android.server.wm.ActivityStackSupervisor#startSpecificActivityLocked// 进程不存在逻辑



给ams发送消息，交由ams处理

com.android.server.am.ActivityManagerService.LocalService#startProcess // ActivityManagerInternal由AMS中的LocalService实现

com.android.server.am.ActivityManagerService#startProcessLocked

com.android.server.am.ProcessList#startProcessLocked//

com.android.server.am.ProcessList#startProcessLocked// 判断是同步启动还是异步启动进程，设置虚拟机运行flag（是否可调试），开发中经常使用android:sharedUserId="android.uid.system"，其中就是这下面这里生效

|  |
| --- |
| Java gids[0] = UserHandle.getSharedAppGid(UserHandle.getAppId(uid)); gids[1] = UserHandle.getCacheAppGid(UserHandle.getAppId(uid)); gids[2] = UserHandle.getUserGid(UserHandle.getUserId(uid)); |

|  |
| --- |
| Java // the per-user SELinux context must be set if (TextUtils.*isEmpty*(app.info.seInfoUser)) {  Slog.wtf(ActivityManagerService.*TAG*, "SELinux tag not defined",  new IllegalStateException("SELinux tag not defined for "  + app.info.packageName + " (uid " + app.uid + ")")); } final String seInfo = app.info.seInfo  + (TextUtils.*isEmpty*(app.info.seInfoUser) ? "" : app.info.seInfoUser); // Start the process. It will either succeed and return a result containing // the PID of the new process, or else throw a RuntimeException. // 新进程启动位置 final String entryPoint = "android.app.ActivityThread";  return startProcessLocked(hostingRecord, entryPoint, app, uid, gids,  runtimeFlags, mountExternal, seInfo, requiredAbi, instructionSet, invokeWith,  startTime); |

com.android.server.am.ProcessList#startProcess//启动进程，这里判断进程由哪个zygote启动

webviewZygote或者AppZygote还是普通的ZygoteProcess

com.android.server.am.ProcessList#createAppZygoteForProcessIfNeeded// 创建AppZygote对象

// 和AppZygote进行交互的类，AppZygote进程预加载了常用库，提高了应用启动速度

android.os.AppZygote

// 和AppZygote进程进行通信，现在不直接调用Zygote中的方法，而是通过android.os.ZygoteProcess.ZygoteState封装类来进行通信

android.os.ZygoteProcess

android.os.ZygoteProcess#preloadApp // 预加载应用

android.os.ZygoteProcess#attemptZygoteSendArgsAndGetResult// 发送启动进程的请求，并获取启动结果

com.android.internal.os.Zygote#forkAndSpecialize

com.android.internal.os.Zygote#nativeForkAndSpecialize// zogote native 创建进程

|  |
| --- |
| Java static int forkAndSpecialize(int uid, int gid, int[] gids, int runtimeFlags,  int[][] rlimits, int mountExternal, String seInfo, String niceName, int[] fdsToClose,  int[] fdsToIgnore, boolean startChildZygote, String instructionSet, String appDataDir,  boolean isTopApp, String[] pkgDataInfoList, String[] whitelistedDataInfoList,  boolean bindMountAppDataDirs, boolean bindMountAppStorageDirs) {  ZygoteHooks.preFork();   int pid = *nativeForkAndSpecialize*(  uid, gid, gids, runtimeFlags, rlimits, mountExternal, seInfo, niceName, fdsToClose,  fdsToIgnore, startChildZygote, instructionSet, appDataDir, isTopApp,  pkgDataInfoList, whitelistedDataInfoList, bindMountAppDataDirs,  bindMountAppStorageDirs);  // 以下代码会被执行两次，一次是调用进程，一次是新进程，新进程中，pid未被赋值所以等于0  if (pid == 0) {  // Note that this event ends at the end of handleChildProc,  Trace.traceBegin(Trace.TRACE\_TAG\_ACTIVITY\_MANAGER, "PostFork");   // 设置新进程的网络权限  if (gids != null && gids.length > 0) {  NetworkUtils.setAllowNetworkingForProcess(*containsInetGid*(gids));  }  }   // Set the Java Language thread priority to the default value for new apps.  Thread.*currentThread*().setPriority(Thread.*NORM\_PRIORITY*);   ZygoteHooks.postForkCommon();  return pid; } |