Author: “Bobby Wang [wbo4958@gmail.com](mailto:wbo4958@gmail.com)”

ActivityManagerService是android系统中非常重要的模块，

AMS在SystemServer初始化的流程

一、 AMS在SystemServer初始化顺序

1. AMS初始化及启动

mActivityManagerService = mSystemServiceManager.startService(  
 ActivityManagerService.Lifecycle.**class**).getService();  
mActivityManagerService.setSystemServiceManager(mSystemServiceManager);

安装installer  
mActivityManagerService.setInstaller(installer);

初始化电源管理

mActivityManagerService.initPowerManagement();

//将SystemServer准备成系统进程

mActivityManagerService.setSystemProcess();

//设置 Component用法manager

mActivityManagerService.setUsageStatsManager(  
 LocalServices.*getService*(UsageStatsManagerInternal.**class**));

//主要是为ActivityThread安装系统provider  
mActivityManagerService.installSystemProviders();

//设置window manager,

mActivityManagerService.setWindowManager(wm);

networkPolicy = **new** NetworkPolicyManagerService(  
 context, mActivityManagerService,  
 (IPowerManager)ServiceManager.*getService*(Context.***POWER\_SERVICE***),  
 networkStats, networkManagement);

**if** (safeMode) {  
 mActivityManagerService.enterSafeMode();  
 *// Disable the JIT for the system\_server process* VMRuntime.getRuntime().disableJitCompilation();  
} **else** {  
 *// Enable the JIT for the system\_server process* VMRuntime.getRuntime().startJitCompilation();  
}

**if** (safeMode) {  
 mActivityManagerService.showSafeModeOverlay();  
}

mActivityManagerService.systemReady(**new** Runnable() {

@Override  
**public void** run() {  
 Slog.*i*(***TAG***, "Making services ready");  
 mSystemServiceManager.startBootPhase(  
 SystemService.***PHASE\_ACTIVITY\_MANAGER\_READY***);  
  
 **try** {  
 mActivityManagerService.startObservingNativeCrashes();  
 } **catch** (Throwable e) {  
 reportWtf("observing native crashes", e);  
 }

…

}

二、 AMS初始化与启动

在SystemServer里

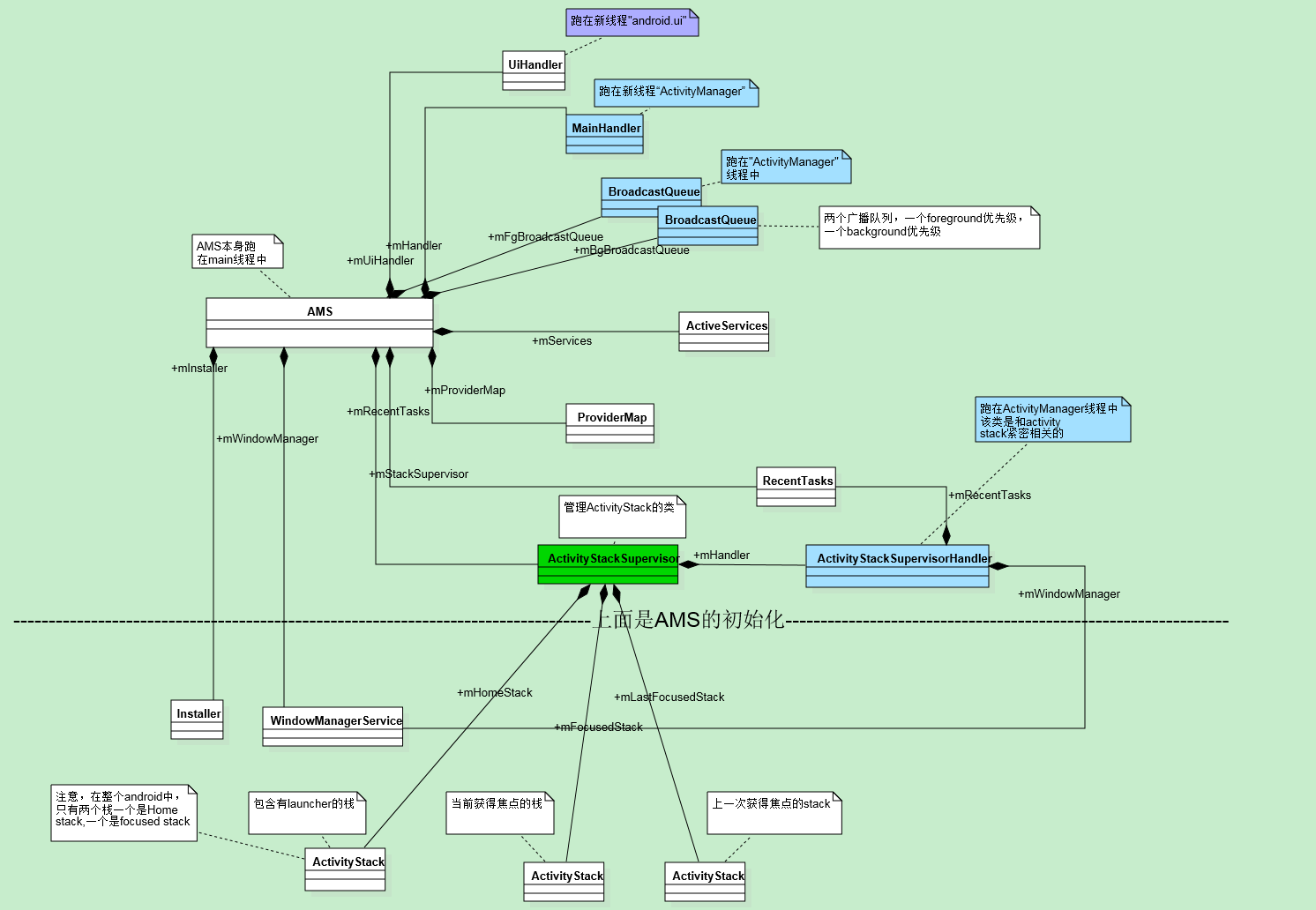
mActivityManagerService = mSystemServiceManager.startService(  
 ActivityManagerService.Lifecycle.**class**).getService();  
mActivityManagerService.setSystemServiceManager(mSystemServiceManager);

AMS作为一个系统级的service, 自然而然应该由SystemServiceManager进行统一的管理与track, 只不过AMS中用一个静态的LifeCycle类，该类继续于SystemService

类，因此android将AMS间接交给SystemServiceManager去管理、track. 因此在mSystemServiceManager.startService(AMS.Lifecycle.class)后，实际上执行了

1. new Lifecycle(context); -> onStart();

**public static final class** Lifecycle **extends** SystemService {  
 **private final** ActivityManagerService mService;  
  
 **public** Lifecycle(Context context) {  
 **super**(context);  
 mService = **new** ActivityManagerService(context);  
 }  
  
 @Override  
 **public void** onStart() {  
 mService.start();  
 }  
  
 **public** ActivityManagerService getService() {  
 **return** mService;  
 }  
}



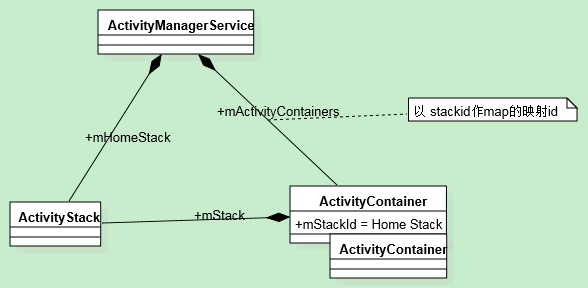
//设置window manager,

mActivityManagerService.setWindowManager(wm);

//创建home stack, 并映射到 default display

createStackOnDisplay(***HOME\_STACK\_ID***, Display.***DEFAULT\_DISPLAY***);  
mHomeStack = mFocusedStack = mLastFocusedStack = getStack(***HOME\_STACK\_ID***);

//默认初始化 home stack, focused, last focused stack.



三、AMS.systemReady(Runnable goingCallback)

systemReady(Runnable goingCallback) {

…

**if** (goingCallback != **null**) goingCallback.run();

}

//启动 persistent app

List apps = AppGlobals.*getPackageManager*().  
 getPersistentApplications(***STOCK\_PM\_FLAGS***);  
**if** (apps != **null**) {  
 **int** N = apps.size();  
 **int** i;  
 **for** (i=0; i<N; i++) {  
 ApplicationInfo info  
 = (ApplicationInfo)apps.get(i);  
 **if** (info != **null** &&  
 !info.packageName.equals("android")) {  
 addAppLocked(info, **false**, **null /\* ABI override \*/**);  
 }  
 }  
}

startHomeActivityLocked(mCurrentUserId, "systemReady");

mStackSupervisor.resumeTopActivitiesLocked();