Direct Load Apk框架介绍

Android插件化

张晓波

Android插件化

- 什么是插件
- 插件化原理
- 插件化核心
- Activity启动流程
- Service启动流程
- Receiver和Provider的处理
- 单进程与多进程
- 插件安全
- 与ApkPlug商业版对比
- TODO

什么是插件

- 1、一个免安装的apk文件，用户使用的时候感觉跟已安装应用的使用一样。
- 2、可以看作一个独立的业务模块，比如：查快递、话费充值、看新闻、定酒店、买电影票等；缺失某一插件对系统运行无影响。
- **3、任何一个商业apk，比如优酷、土豆视频、百度音乐、京东商城、小米商城、漫画岛、安卓壁纸、糗百等。**
- 4、和主业务框架松耦合的模块，和主业务框架剥离，可降低包大小，解决代码量过多引起的方法数超过虚拟机限制的问题。
- 5、可动态部署：使用的时候下载or加载，可动态升级。

欢迎使用直接运行apk框架!

五子棋

一款精致的五子棋游戏。

下载

军棋

相信大家一定玩过这是一款界面精美、功能丰富的军棋游戏

下载

今日头条

用户量过2.6亿的新闻阅读客户端

下载

糗百

一入糗百深似海，记忆碎片此中留

下载

别踩白块儿

全球累计超过 130,000,000（一亿三千万）次下载

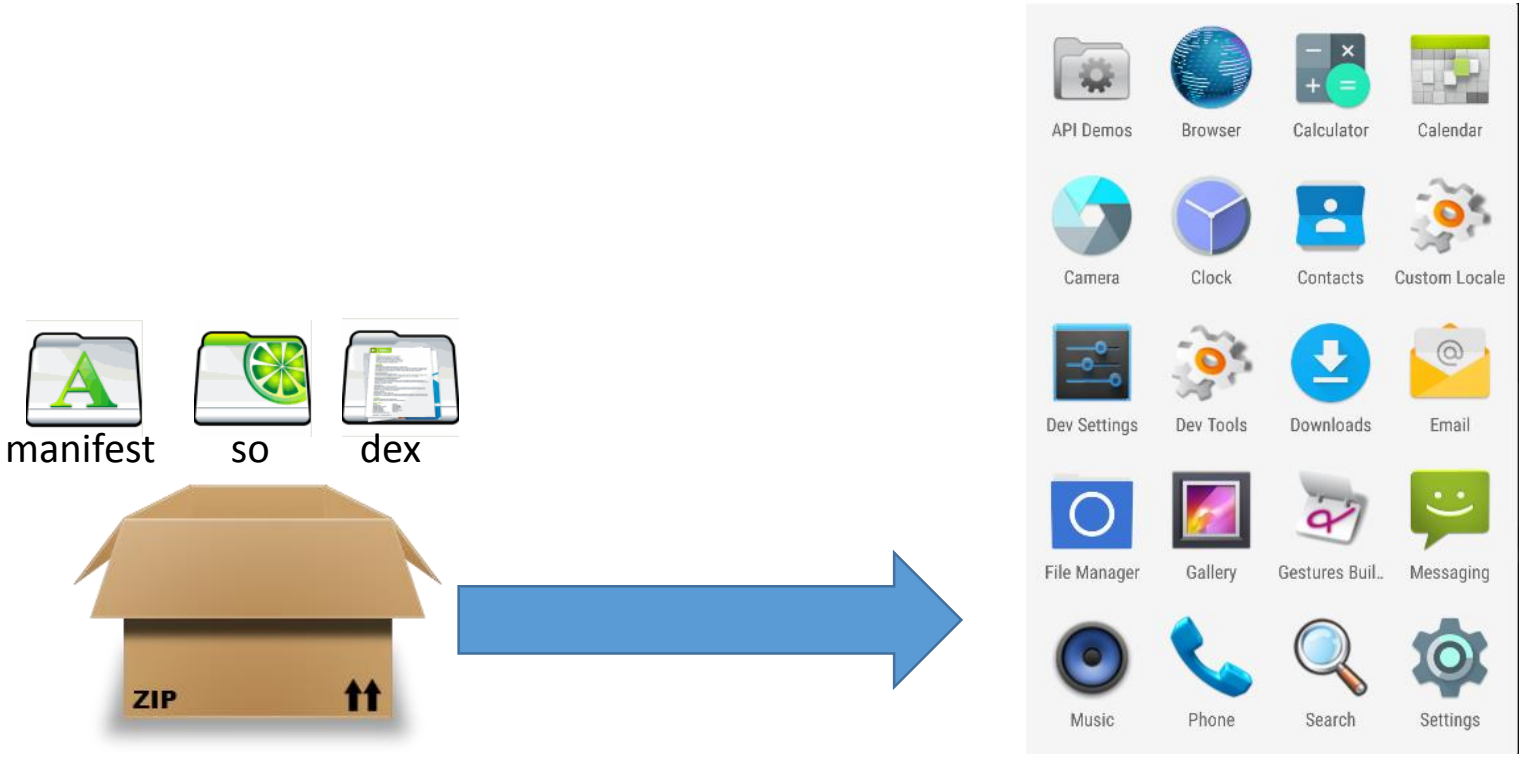
下载

墨迹天气

墨迹天气，只做最懂你的天气

下载

原理


manifest
so
dex
ZIP
API Demos
Browser
Calculator
Calendar
Camera
Clock
Contacts
Custom Locale
Dev Settings
Dev Tools
Downloads
Email
File Manager
Gallery
Gestures Buil..
Messaging
Music
Phone
Search
Settings

- APK引导容器

插件化核心

1、**Java反射：**插件化的核心技术，用来获取或黑掉系统服务

2、**DynamicProxy：**在一个接口的每个方法调用前后干点坏事

3、**PluginClassLoader：**插件的类加载器，基于插件dex、so的解压路径生成

4、**LoadedApk：**apk数据结构，为插件四大组件提供运行必要的信息，比如ClassLoader、Resources

5、**BaseContext：**包装插件Context（为插件提供各种路径支持：外部存储路径、外部缓存路径等等，绕过IMountService的mkdirs操作，并为插件路径访问提供统一管理）

6、**GuardActivityManagerProxy：**ActivityManagerNative和ActivitymanagerProxy之间的桥梁

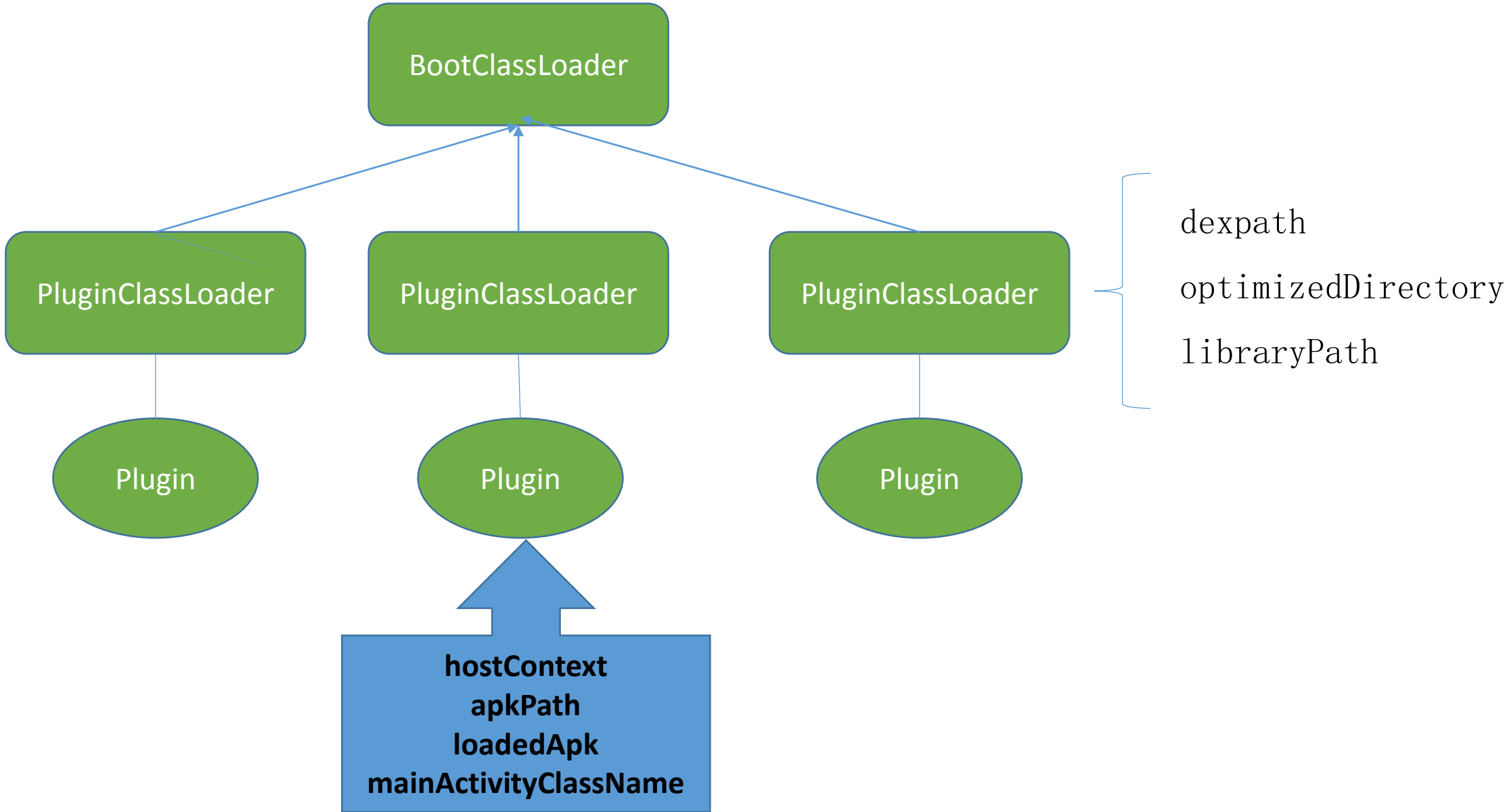
7、**ActivityThread.mH.mCallback：**拦截由AMS发来的指令，并做处理（主要针对Activity）

8、**GuardInstrumentation：**替换ActivityThread的instrumentation，为插件的Application、Activity启动提供正确的context和生命周期回调

9、**GuardPackageManager：**拦截PackageManager服务，以便能让插件以为自己是已经安装了

10、**GuardNotification：**拦截插件的Notification请求，中转到宿主，并由宿主代为执行

PluginClassLoader---加载原理



插件化核心

1、**Java反射：**插件化的核心技术，用来获取或黑掉系统服务

2、**DynamicProxy：**在一个接口的每个方法调用前后干点坏事

3、**PluginClassLoader：**插件的类加载器，基于插件dex、so的解压路径生成

4、**LoadedApk：**apk数据结构，为插件四大组件提供运行必要的信息，比如ClassLoader、Resources

5、**BaseContext：**包装插件Context（为插件提供各种路径支持：外部存储路径、外部缓存路径等等，绕过IMountService的mkdirs操作，并为插件路径访问提供统一管理）

6、**GuardActivityManagerProxy：**ActivityManagerNative和ActivitymanagerProxy之间的桥梁

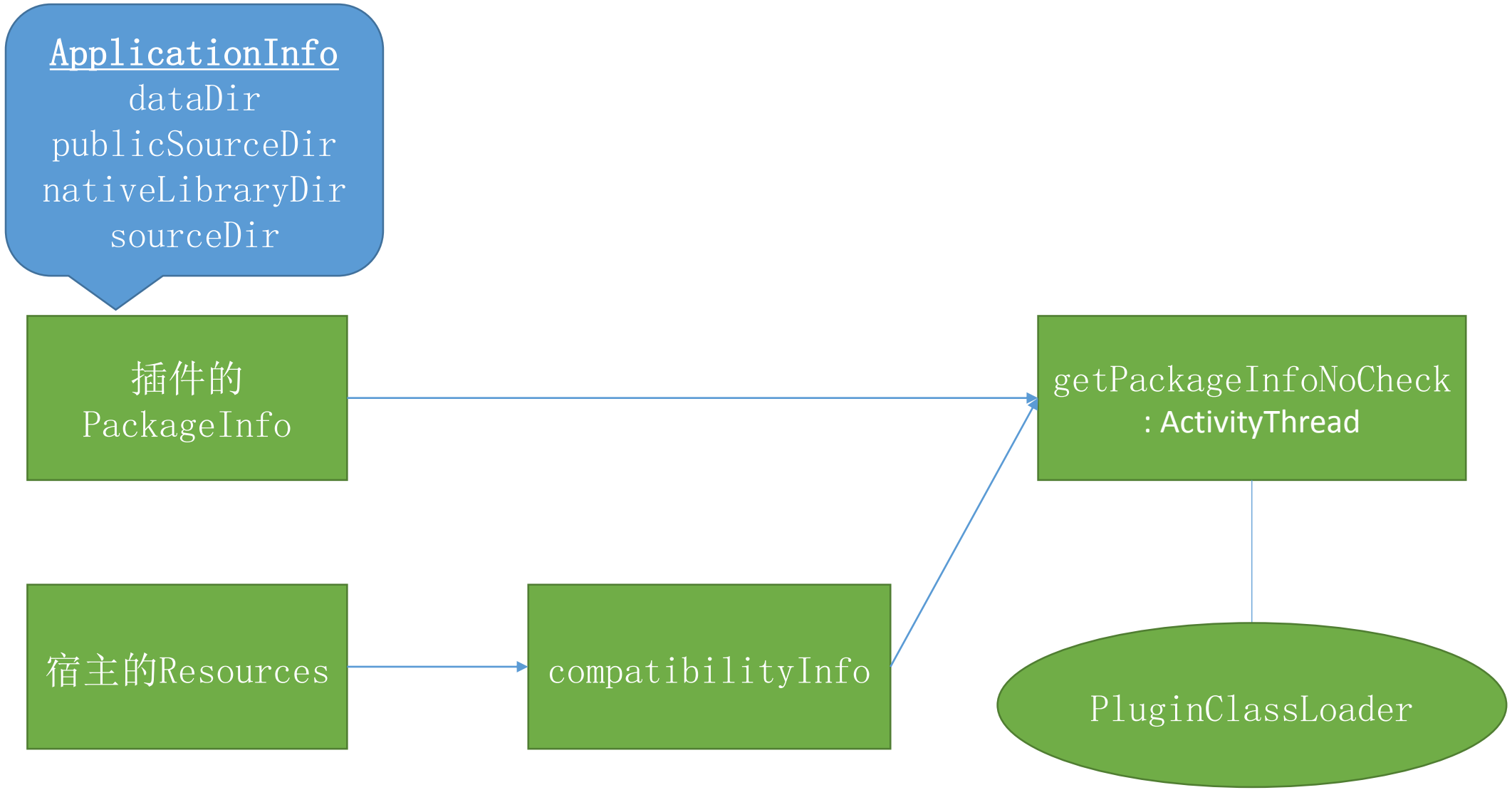
7、**ActivityThread.mH.mCallback：**拦截由AMS发来的指令，并做处理（主要针对Activity）

8、**GuardInstrumentation：**替换ActivityThread的instrumentation，为插件的Application、Activity启动提供正确的context和生命周期回调

9、**GuardPackageManager：**拦截PackageManager服务，以便能让插件以为自己是已经安装了

10、**GuardNotification：**拦截插件的Notification请求，中转到宿主，并由宿主代为执行

插件LoadedApk生成过程



插件化核心

1、**Java反射：**插件化的核心技术，用来获取或黑掉系统服务

2、**DynamicProxy：**在一个接口的每个方法调用前后干点坏事

3、**PluginClassLoader：**插件的类加载器，基于插件dex、so的解压路径生成

4、**LoadedApk：**apk数据结构，为插件四大组件提供运行必要的信息，比如ClassLoader、Resources

5、**BaseContext：**包装插件Context（为插件提供各种路径支持：外部存储路径、外部缓存路径等等，绕过IMountService的mkdirs操作，并为插件路径访问提供统一管理）

6、**GuardActivityManagerProxy：**ActivityManagerNative和ActivitymanagerProxy之间的桥梁

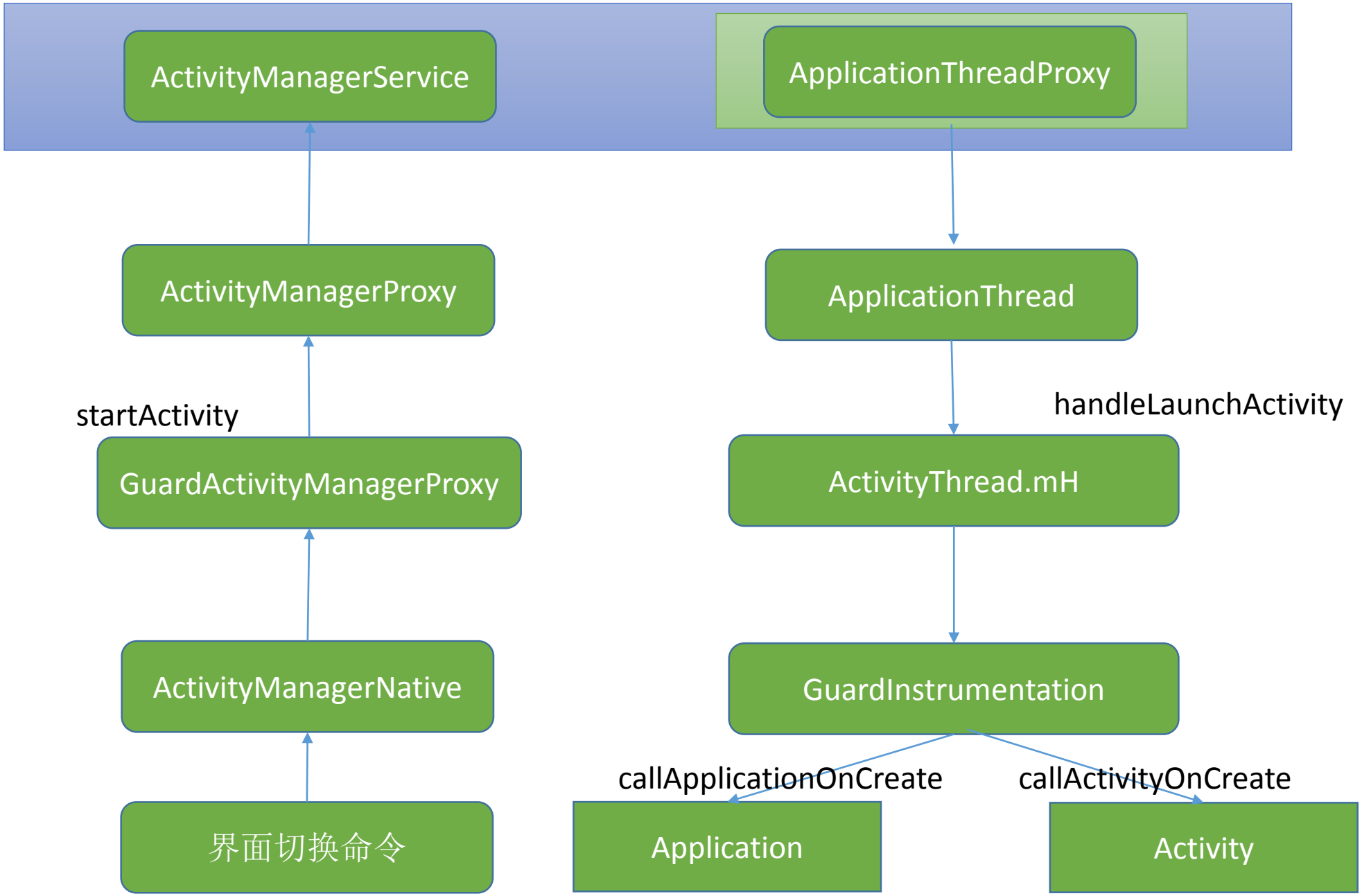
7、**ActivityThread.mH.mCallback：**拦截由AMS发来的指令，并做处理（主要针对Activity）

8、**GuardInstrumentation：**替换ActivityThread的instrumentation，为插件的Application、Activity启动提供正确的context和生命周期回调

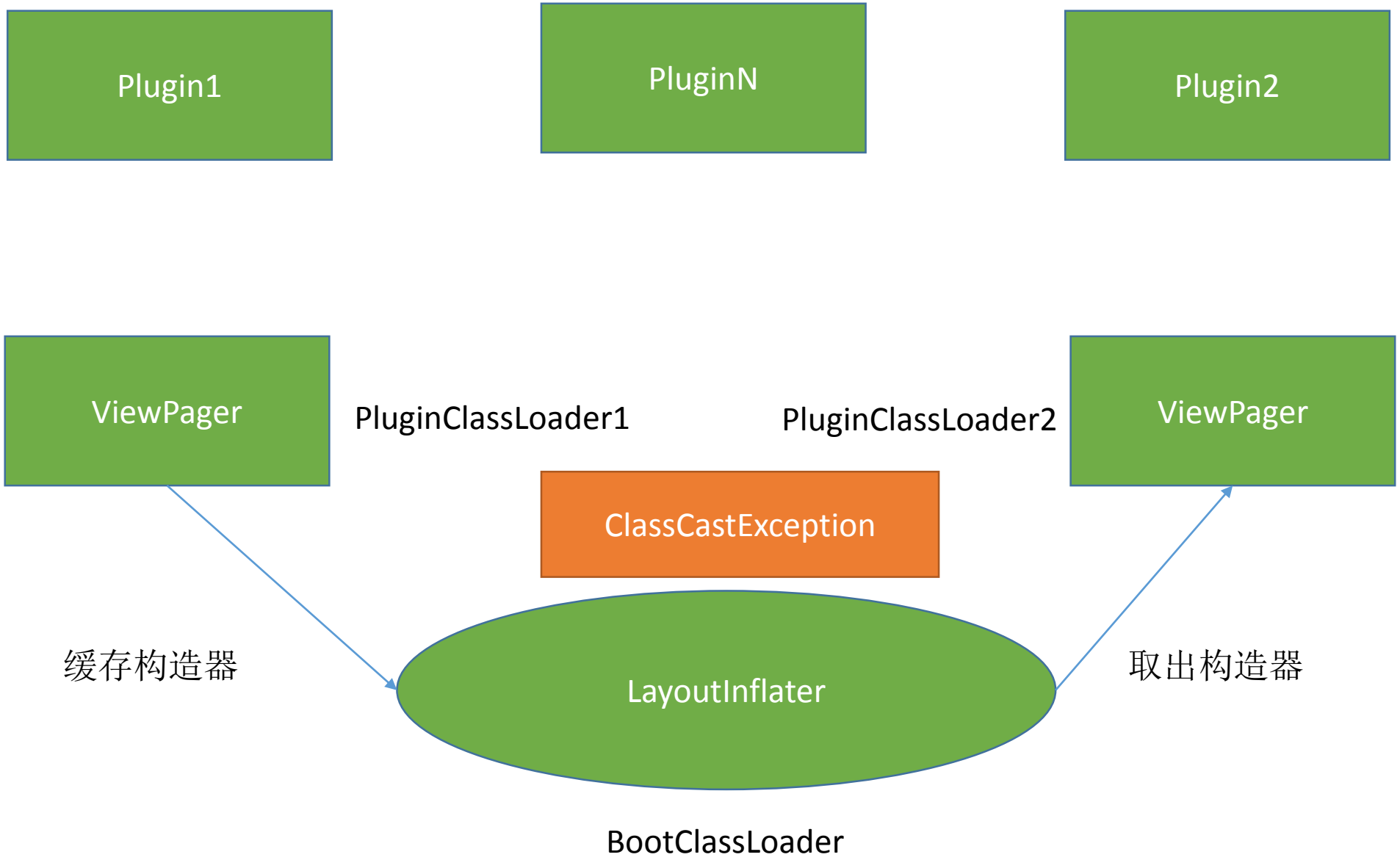
9、**GuardPackageManager：**拦截PackageManager服务，以便能让插件以为自己是已经安装了

10、**GuardNotification：**拦截插件的Notification请求，中转到宿主，并由宿主代为执行

Activity启动流程---hack原理



Activity启动流程---单进程插件切换的坑



Activity启动流程---位置定位遇到的坑

LocationManager

(mService : ILocationManager)

- getLastLocation
- requestLocationUpdates
- removeUpdates
- requestGeofence
- removeGeofence

动态代理拦截，packageName传入宿主的包名

（注意API版本变化引起的参数位置变化）

ContextWrapper

ContextThemeWrapper

mBase

getSystemService

Activity启动流程---Toast发送遇到的坑

NotificationManager
(sService : INotificationManager)

enqueueToast

参数1：动态拦截，pkg传入宿主的包名

参数2：动态拦截，记录callback

（android.app.ITransientNotification callback）

可以随时取消toast显示

NotificationManager

插件Notification拦截和此过程类似，支持宿主转发插件的文本Notification，需要hack ActivityManagerProxy的getIntentSender

插件化-Java层访问私有目录的坑

动态代理，拦截libcore.io.Os，将私有目录重定向为宿主所分配给插件的目录

(/data/data/plugin_package_name/...

to

/data/data/host_package_name/app_data/plugin_package_name/…)

```
public interface Os {

public FileDescriptor accept(FileDescriptor fd, InetSocketAddress peerAddress) throws ErrnoException, SocketExcepti
        public boolean access(String path, int mode) throws ErrnoException;

public InetAddress[] android_getaddrinfo(String node, StructAddrinfo hints, int netId) throws GaiException;

public void bind(FileDescriptor fd, InetAddress address, int port) throws ErrnoException, SocketException;
        public void chmod(String path, int mode) throws ErrnoException;
        public void chown(String path, int uid, int gid) throws ErrnoException;
        public void close(FileDescriptor fd) throws ErrnoException;

public void connect(FileDescriptor fd, InetAddress address, int port) throws ErrnoException, SocketException;
        public FileDescriptor dup(FileDescriptor oldFd) throws ErrnoException;
        public FileDescriptor dup2(FileDescriptor oldFd, int newFd) throws ErrnoException;
        public String[] environ();
        public void execv(String filename, String[] argv) throws ErrnoException;
        public void execve(String filename, String[] argv, String[] envp) throws ErrnoException;
        public void fchmod(FileDescriptor fd, int mode) throws ErrnoException;
```

Activity启动流程---支持Unity、Cocos

1、抽取manifest配置

```
<uses-feature android:glEsVersion=“0x20000" />
<uses-feature android:name="android.hardware.touchscreen" android:required="false" />
<uses-feature android:name="android.hardware.touchscreen.multitouch" android:required="false" />
 <uses-feature android:name="android.hardware.touchscreen.multitouch.distinct" android:required="false" />
 <meta-data android:name="unityplayer.UnityActivity" android:value="true" />
```

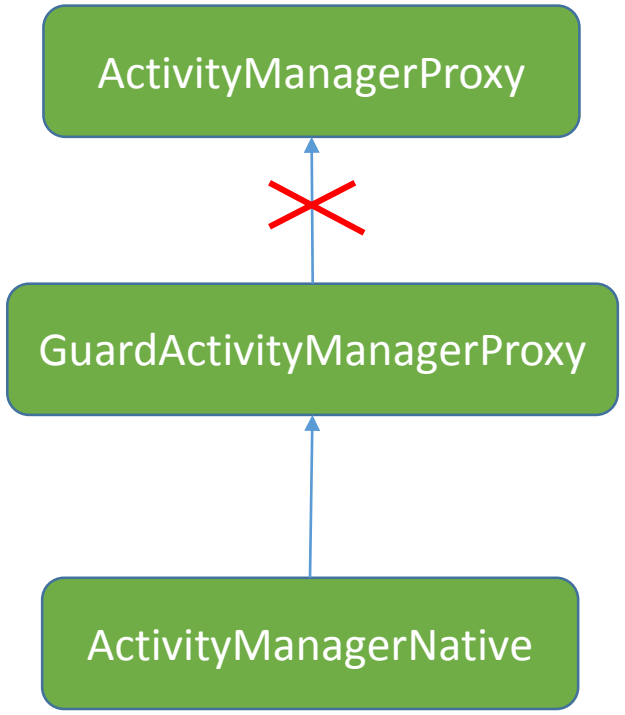
2、modify ApplicationInfo. nativeLibraryDir

3、包装base context

```
@Override
public File getObbDir() {
        // 用于unity
}

@Override
public File[] getObbDirs() {
        // 用于unity
}
```

Service启动流程



startService

stopService

stopServiceToken

bindService

unBindService

ServiceRunner

模拟正常的启动或绑定Service的过程

Receiver和Provider的处理

静态receiver：在Application创建时动态注册

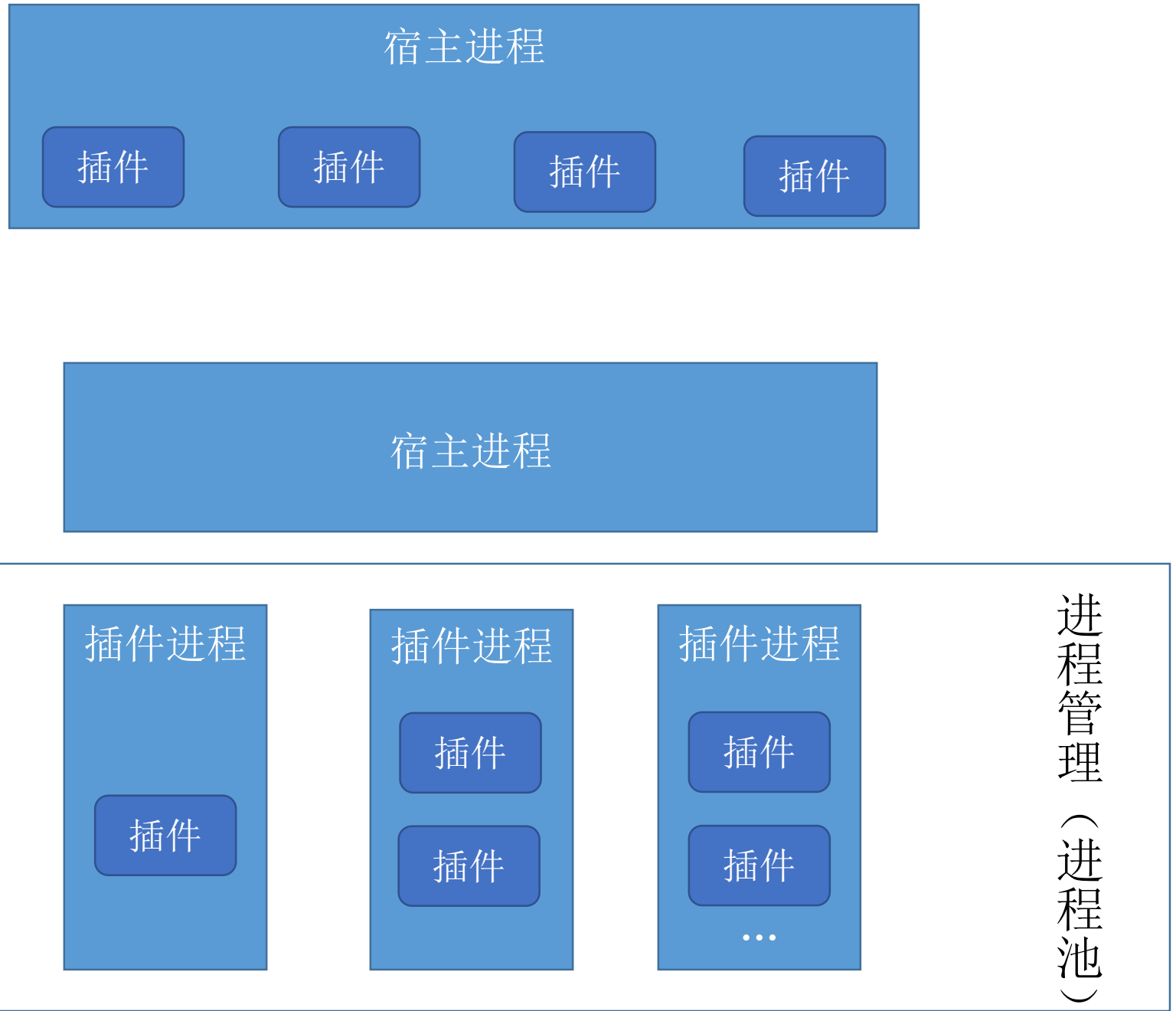
Content Provider：hack ActivityManagerNative getContentProvider方法

hack ContentResolver mPackageName

同进程多个插件之间可共享provider

注意applicationInfo要正确赋值

单进程与多进程



插件安全

插件完整性

插件可信性

运行期校验

兼容特性

- 2.3.x测试通过
- 3.x测试通过
- 4.x测试通过
- 4.4测试通过
- 5.0测试通过
- 5.1.1测试通过
- 云测试无一一例外全部PASS
- 支持Dalvik环境
- 支持ART环境

功能	apkplug商业版本	Direct Load Apk
插件权限	需要在宿主配置	需要在宿主配置
Activity组件	支持	支持
Activity跳转特效	不支持	支持
纯Action方式访问Activity	不支持	支持
Service组件	支持	支持
Broadcast组件	支持Manifest.xml	支持
ContentProvider	支持	支持
插件间访问ContentProvider	支持	支持
ContentProvider数据监听notifyChange	不支持	支持
WebView访问插件资源(assets)	不支持	支持
PackageName	Context.getPackageName()返回的是宿主包名	Context.getPackageName()返回的是插件包名
主题皮肤切换	支持	支持
插件异常隔离机制	支持	支持
Service	不需要宿主配置	支持
Broadcast	支持Manifest.xml中直接配置	支持
ContentProvider	支持	支持
插件异常隔离机制	支持	支持
创建插件Activity快捷方式	支持	支持
安装普通APK	支持	支持

TODO

- 对自动升级的支持：除了现有的插件热更新，未来思考对插件框架本身实现自动更新
- 对动态性的支持：插件框架可以支持用户定制插件，允许灵活控制整个应用系统
- 对插件仓库的支持：将插件发布到插件仓库，插件开发者可一键发布应用或更新