# PackageInstaller 原理简述

贾桂卿 56169

应用安装是智能机的主要特点，即用户可以把各种应用（如游戏等）安装到手机上，并可以对其进行卸载等管理操作。APK 是 Android Package 的缩写，即 Android 安装包。APK 是类似 Symbian Sis 或 Sisx 的文件格式。通过将 APK 文件直接传到 Android 模拟器或 Android 手机中执行即可安装。

Android 应用安装有如下四种方式

1. 系统应用安装－－开机时完成，没有安装界面
2. 网络下载应用安装－－通过 market 应用完成，没有安装界面
3. ADB 工具安装－－没有安装界面。
4. 第三方应用安装－－通过 SD 卡里的 APK 文件安装，有安装界面，由 packageinstaller.apk 应用处理安装及卸载过程的界面。

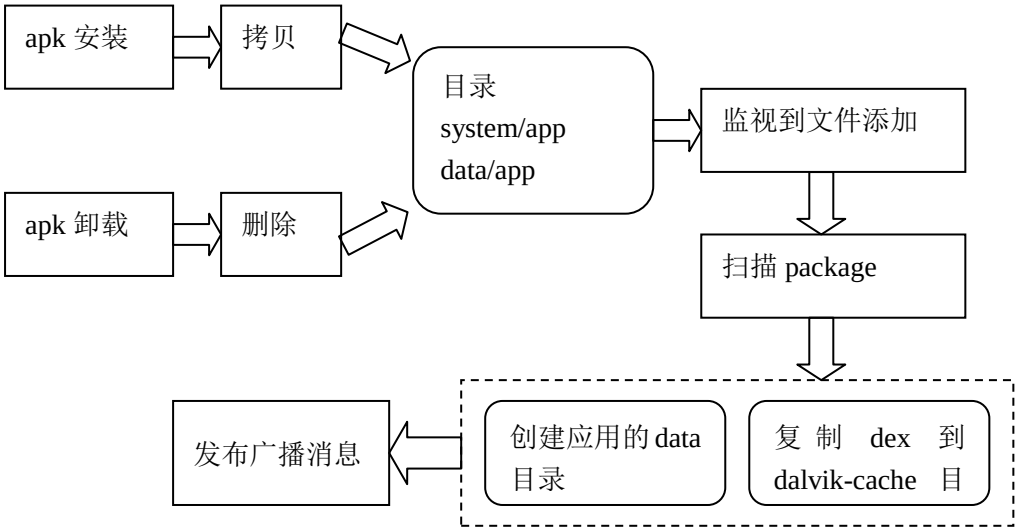
## 应用安装的流程及路径

应用安装涉及到如下几个目录：

| | |
|---|---|
| system/app | 系统自带的应用程序，无法删除 |
| data/app | 用户程序安装的目录，有删除权限。<br>安装时把 apk 文件复制到此目录 |
| data/data | 存放应用程序的数据 |
| Data/dalvik-cache | 将 apk 中的 dex 文件安装到 dalvik-cache 目录下(dex 文件是 dalvik 虚拟机的可执行文件，其大小约为原始 apk 文件大小的四分之一) |

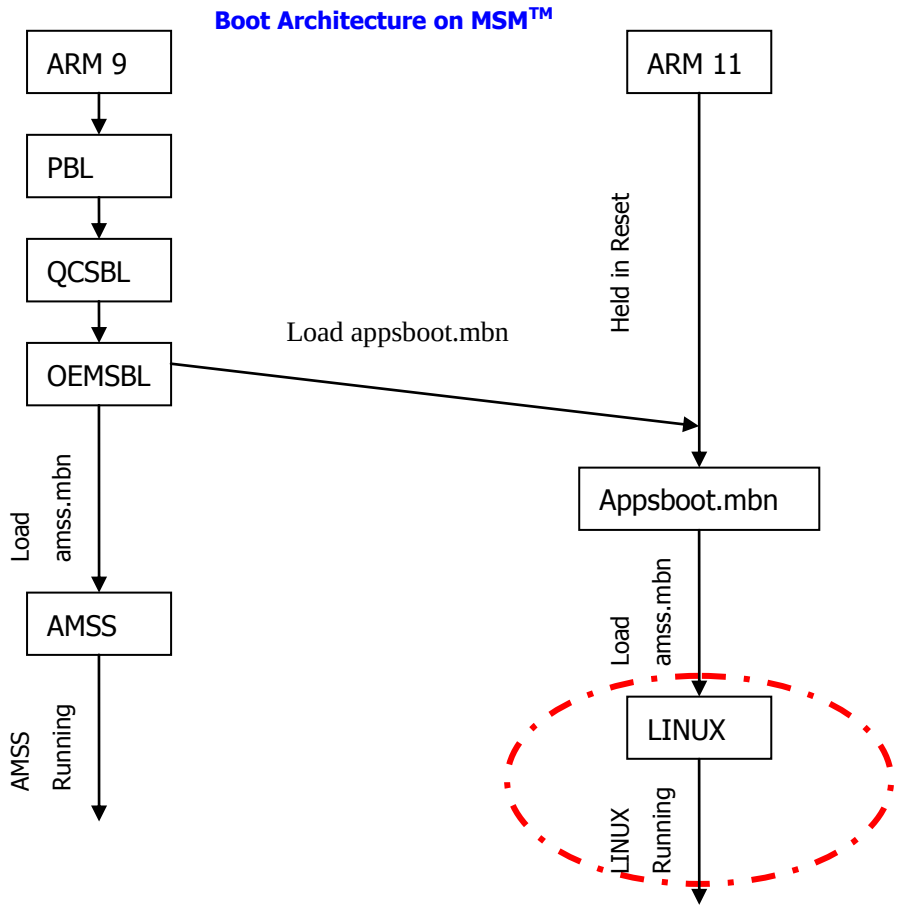
安装过程：复制 APK 安装包到 data/app 目录下，解压并扫描安装包，把 dex 文件(Dalvik 字节码)保存到 dalvik-cache 目录，并 data/data 目录下创建对应的应用数据目录。

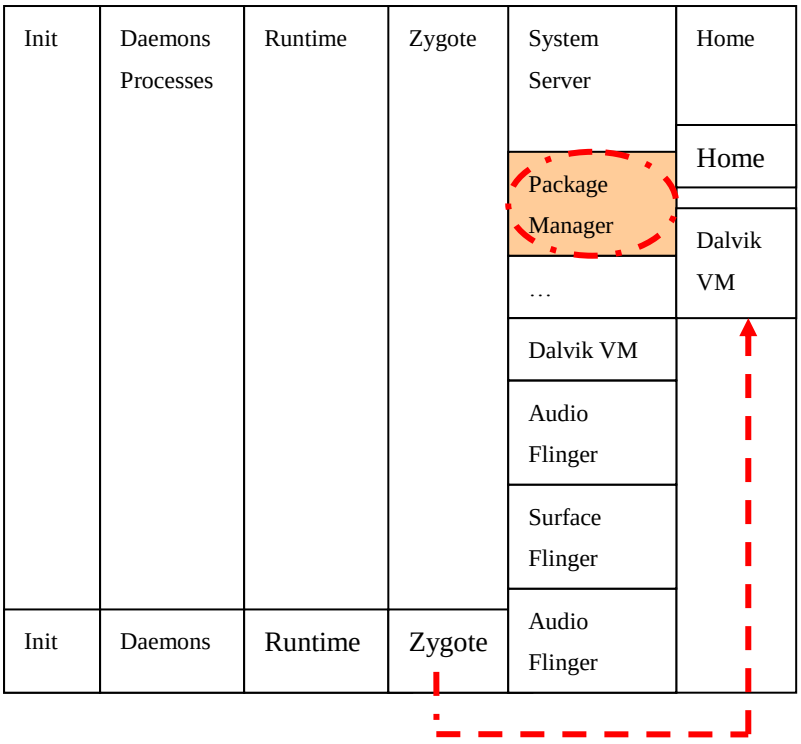
卸载过程：删除安装过程中在上述三个目录下创建的文件及目录。




## 一、系统应用安装：

PackageManager Service 处理各种应用的安装，卸载，管理等工作，开机时由 systemServer 启动此服务(源文件路径：android\frameworks\base\services\java\com\android\server\ PackageManagerService.java)

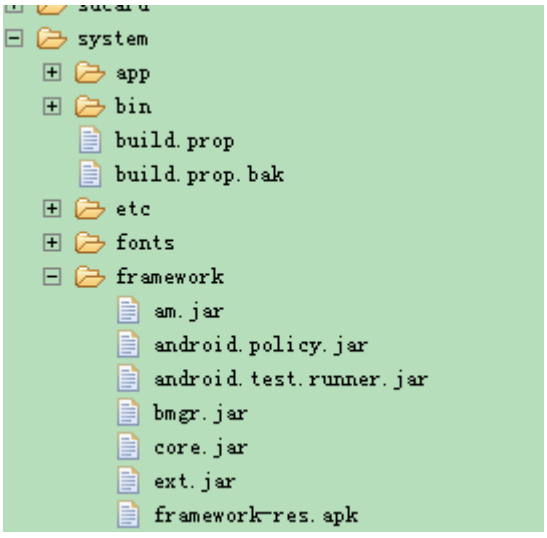

Boot Architecture on MSM™
ARM 9
PBL
QCSBL
OEMSBL
Load amss.mbn
AMSS
AMSS Running
Load appsboot.mbn
Held in Reset
ARM 11
Appsboot.mbn
Load amss.mbn
LINUX
LINUX Running




PackageManager Service 服务启动的流程：

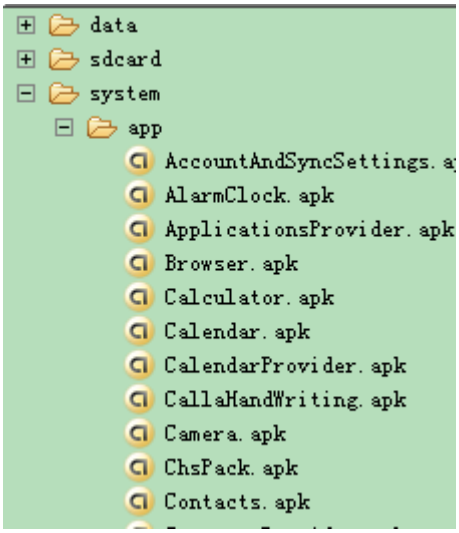
1. 首先扫描安装"system\framework"目录下的 jar 包

1. scanDirLI(mFrameworkDir, PackageParser.PARSE_IS_SYSTEM,
scanMode | SCAN_NO_DEX);

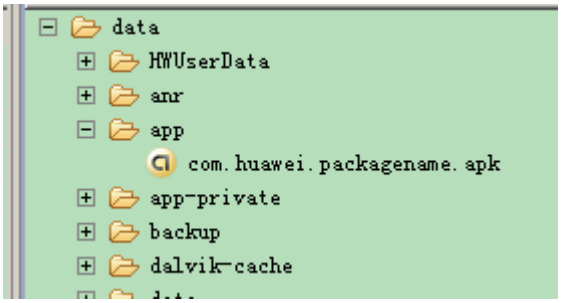



2.第二步扫描安装"system\app"目录下的各个系统应用

scanDirLI(mSystemAppDir, PackageParser.PARSE_IS_SYSTEM, scanMode);




3.第三步扫描"data\app"目录，即用户安装的第三方应用
scanDirLI(mAppInstallDir, 0, scanMode);




4.第四步扫描" data\app-private"目录，即安装 DRM 保护的 APK 文件（目前没有遇到过此类的应用）。
scanDirLI(mDrmAppPrivateInstallDir, 0, scanMode | SCAN_FORWARD_LOCKED);

安装应用的过程
1.scanDirLI(File dir, int flags, int scanMode)　遍历安装指定目录下的文件

2.scanPackageLI(File scanFile,
File destCodeFile, File destResourceFile, int parseFlags,
int scanMode)　　安装 package 文件

3.scanPackageLI(
File scanFile, File destCodeFile, File destResourceFile,
PackageParser.Package pkg, int parseFlags, int scanMode)
通过解析安装包 parsePackage 获取到安装包的信息结构

4.mInstaller.install(pkgName, pkg.applicationInfo.uid,
pkg.applicationInfo.uid);　实现文件复制的安装过程
（源文件路径：frameworks\base\cmds\installd\installd.install）

```
struct cmdinfo cmds[] = {
    { "ping",           0, do_ping },
    { "install",        3, do_install },
    { "dexopt",         3, do_dexopt },
    { "movedex",        2, do_move_dex },
    { "rmdex",          1, do_rm_dex },
    { "remove",         1, do_remove },
    { "freecache",      1, do_free_cache },
    { "rmcache",        1, do_rm_cache },
    { "protect",        2, do_protect },
    { "getsize",        3, do_get_size },
    { "rmuserdata",     1, do_rm_user_data },
};
```

## 二、从 market 上下载应用：

Google Market 应用需要使用 gmail 账户登录才可以使用，选择某一应用后，开始下载安装包，此过程中，在手机的信号区有进度条提示，下载完成后，会自动调用 Packagemanager 的接口安装，调用接口如下：

public void installPackage(final Uri packageURI, final IPackageInstallObserver observer, final int flags)

final Uri packageURI：文件下载完成后保存的路径

final IPackageInstallObserver observer：处理返回的安装结果

final int flags：安装的参数，从 market 上下载的应用，安装参数为-r (replace)

installPackage 接口函数的安装过程：

1.public void installPackage(
final Uri packageURI, final IPackageInstallObserver observer, final int flags,
final String installerPackageName)

final String installerPackageName:安装完成后此名称保存在 settings 里，一般为 null,不是关键参数

2.File tmpPackageFile = copyTempInstallFile(packageURI, res);

把 apk 文件复制到临时目录下的临时文件

3.private void installPackageLI(Uri pPackageURI,
int pFlags, boolean newInstall, String installerPackageName,
File tmpPackageFile, PackageInstalledInfo res)

解析临时文件，获取应用包名 pkgName = PackageParser.parsePackageName(
tmpPackageFile.getAbsolutePath(), 0);

4.判断如果带有参数 INSTALL_REPLACE_EXISTING，则调用 replacePackageLI(pkgName,
tmpPackageFile,
destFilePath, destPackageFile, destResourceFile,
pkg, forwardLocked, newInstall, installerPackageName,
res)

5.如果没有，则调用 installNewPackageLI(pkgName,
tmpPackageFile,
destFilePath, destPackageFile, destResourceFile,

pkg, forwardLocked, newInstall, installerPackageName,
res);

6.private PackageParser.Package scanPackageLI(
File scanFile, File destCodeFile, File destResourceFile,
PackageParser.Package pkg, int parseFlags, int scanMode)

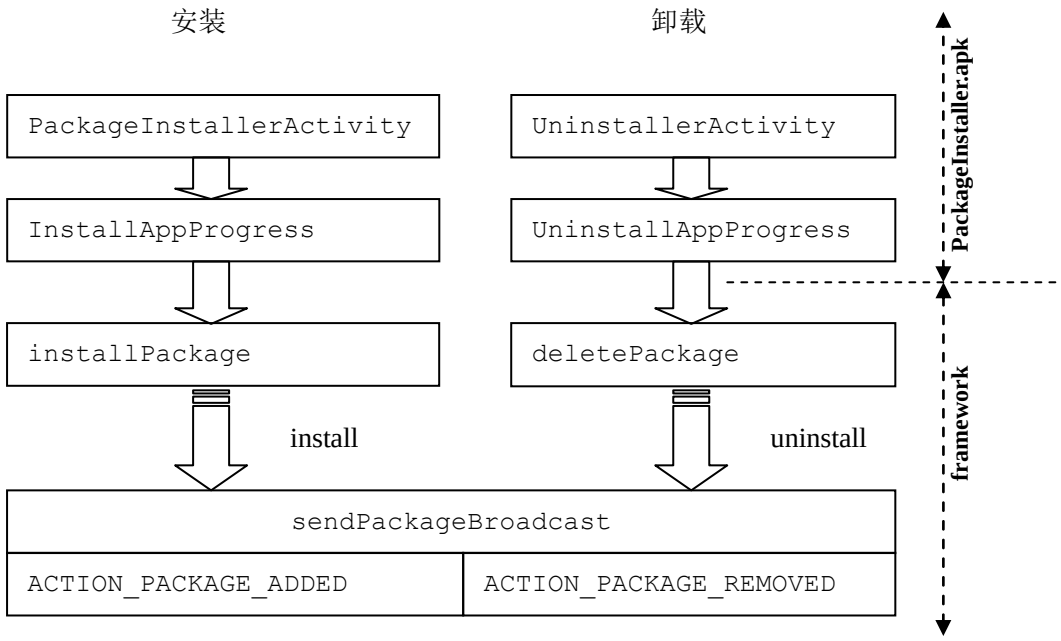
scanPackageLI 以后的流程，与开机时的应用安装流程相同。

## 三、从 ADB 工具安装

Android Debug Bridge (adb) 是 SDK 自带的管理设备的工具，通过 ADB 命令行的方式也可以为手机或模拟器安装应用，其入口函数源文件为 pm.java
(源文件路径：android\frameworks\base\cmds\pm\src\com\android\commands\pm\pm.java)
ADB 命令行的形式为 adb install <path_to_apk> ,还可以带安装参数如："-l" "-r" "-i" "-t"
函数 runInstall()中判断参数
"-l"－－INSTALL_FORWARD_LOCK
"-r"—— INSTALL_REPLACE_EXISTING
"-i" ——installerPackageName
"-t"——INSTALL_ALLOW_TEST
我们常用的参数为-r，表示覆盖安装手机上已安装的同名应用。从 market 上下载的应用，也是直接传入这个参数安装的。
runInstall 与 market 调用同样的接口完成应用安装。
public void installPackage(android.net.Uri packageURI, android.content.pm.IPackageInstallObserver observer, int flags, java.lang.String installerPackageName)

## 四、第三方应用安装——通过 SD 卡里的 APK 文件安装

把 APK 安装包保存在 SD 卡中，从手机里访问 SD 卡中的 APK 安装包，点击就可以启动安装界面，系统应用 Packageinstaller.apk 处理这种方式下的安装及卸载界面流程，如下图：




PackageInstallerActivity 负责解析包，判断是否是可用的 Apk 文件
创建临时安装文件/data/data/com.android.packageinstaller/files/ApiDemos.apk
并启动安装确认界面 startInstallConfirm，列出解析得到的该应用基本信息。如果手机上已安装有同名应用，则需要用户确认是否要替换安装。
确认安装后，启动 InstallAppProgress，调用安装接口完成安装。
pm.installPackage(mPackageURI, observer, installFlags);

## 其它：

1. PackageManagerService.java 的内部类 AppDirObserver 实现了监听 app 目录的功能：当把某个 APK 拖到 app 目录下时，可以直接调用 scanPackageLI 完成安装。
2.手机数据区目录“data/system/packages.xml”文件中，包含了手机上所有已安装应用的基本信息，如安装路径，申请的 permission 等信息。


Name
data
HWUserData
anr
app
app-private
backup
dalvik-cache
data
data_test
dontpanic
gps-test
local
lost+found
misc
property
remote-api-tests
rpc-tests
system
accounts.db
appwidgets.xml
batterystats.bin
entropy.dat
packages.xml
registered_services

完。