[FAQ14102]L版本开机提示“Android正在升级或启动”

[DESCRIPTION]

[**导致该现象出现的可能**]：
1,L版本上预置apk不正确，导致32位与64位兼容性问题
2,如果您有重新push，再开机就会因为odex是旧的，导致重新提取。
3, 安装APK或开机过程中掉电
4, L1上设置关机闹钟，闹钟响起时择暂停，按Power键开机提示“android 正在启动”
5, apk对应的dex档被破坏/丢失，或OTA升级/APK/system Jar/包被更换与dex档不匹配
6,手机down bin档前没有format data image

7,debug手段

[SOLUTION]

一、L版本上预置apk不正确，导致32位与64位兼容性问题

（**现象最多**）预置apk的Android.mk配置不准确, 这会造成首次开机慢/android正在升级/apk报错class not found等异常，请重视！

1）、开机重新提取odex仅为arm的apk（说明此仅支持32位）：

I dex2oat : /system/bin/dex2oat --zip-fd=11 --zip-location=/system/app/***.apk --oat-fd=12 --oat-location=/data/dalvik-cache/arm/system@app@**@**.apk@classes.dex --instruction-set=arm --instruction-set-features=default --runtime-arg -Xms64m --runtime-arg -Xmx512m --swap-fd=13

开机重新提取odex为arm的原因是：预编译时，按照“[FAQ13573]L版本首次开机慢”开启宏后系统默认提取的是64bit的odex，所以造成开机后这些apk被判别为32bit(即该apk仅支持32位)时需要重新提取odex。

【Solution】：在Android.mk中设定LOCAL_MULTILIB :=32，则预编译时会提取该apk的32bit的odex

2）、开机重新提取odex仅为arm64的apk(无lib库)：

I dex2oat : /system/bin/dex2oat --zip-fd=11 --zip-location=/system/app/***.apk --oat-fd=12 --oat-location=/data/dalvik-cache/arm64/system@app@**@**.apk@classes.dex --instruction-set=arm64 --instruction-set-features=default --runtime-arg -Xms64m --runtime-arg -Xmx512m --swap-fd=13

开机重新提取odex为arm64的原因是：这类apk应该是有源码的（例如AtciService），但是其Android.mk中有定义其为32bit。所以预编译时会把其当作32bit的来处理，生成的odex存放在arm下。但是因为这个apk是没有lib的，所以开机后PMS会将其作为64bit来处理，所以会去arm64下找dex文件，没找到就会重新提取。但是因为L版本在提取odex后会删除原本存储的dex文件，所以重新提取就找不到dex文件然后报错。

【Solution】：删除LOCAL_MULTILIB :=32或LOCAL_32_BIT_ONLY=true

3）、开机重新提取odex既有arm又有arm64的apk：

【Solution】：在Android.mk中设定LOCAL_MULTILIB :=both，则预编译时会提取该apk的32bit和64bit的odex

4)、若存在解决不了的，可以对于这个apk的解决办法为预编译时跳过提取其的odex文件：

\build\core\dex_preopt_odex_install.mk中添加：

```
//mtk add begin
ifeq ($(LOCAL_MODULE),MODULE_NAME)
LOCAL_DEX_PREOPT:=
endif

//mtk add end

built_odex:=
installed_odex:=
....
```

MODULE_NAME写为需要替换的apk的module name(注意添加位置、避免编译报错)

那么现在有些apk因为报错（要么自身兼容问题，要么android.mk配置不对），采取预编译时跳过提取措施也是可以的，但这会增加了首次开机的些许时间(不宜添加过多)，即编译阶段不去提取odex，而是首次开机时区提取odex。

二、如果您有重新push，再开机就会因为odex是旧的，导致重新提取。

出现这种情况一般是系统中一些apk仍然需要提取odex文件，比如预编译了odex的apk，如果您有重新push，再开机就会因为odex是旧的，导致重新提取。注：L版本上不支持adb push，重启后才会生效。

三、安装APK或开机过程中掉电

请避免拔电池操作（L1版本）

Google在L1做了change, 开机时异常断电有机会造成file corruption（如果是某app的oat file corruption，会造成app无法启动）

所以Google设计一套方法，只要上一次是异常断电，下一次开机就”全部重建”ART optimization files(*.oat)，因此

1. 会显示Application is starting…
2. 重部重建oat file，所以开机时间会延长

【关键log】: Incomplete boot detected. Pruning dalvik cache

四、L1版本上设置关机闹钟，闹钟响起时择暂停，按Power键开机提示“android 正在启动”

如有碰到请至PactchManagerSystem系统申请Patch ID: ALPS02057885

注：如L1上"android正在启动"等同之前的"android正在升级"

五、apk对应的dex档被破坏/丢失，或OTA升级/APK/system Jar包被更换与dex档不匹配

六、手机down bin档前没有format data image

七、[debug手段]：

总结打trace的方式，这样可以定位出当前这个提示是哪个地方的原因（适用于L1版本）。

首先，@PackageManagerService.java：

1.打开DEBUG_DEXOPT的开关

2.performBootDexOpt()添加trace（“//mtk”标记

```
public void performBootDexOpt() {
enforceSystemOrRoot("Only the system can request dexopt be performed");

// Before everything else, see whether we need to fstrim.
try {
IMountService ms = PackageHelper.getMountService();
if (ms != null) {
final boolean isUpgrade = isUpgrade();
Slog.d(TAG + "_Debug","isUpgrade = " + isUpgrade);//mtk
boolean doTrim = isUpgrade;

......

if (doTrim) {
if (!isFirstBoot()) {
try {
Slog.d(TAG + "_Debug","showBootMessage called1...");//mtk
ActivityManagerNative.getDefault().showBootMessage(
mContext.getResources().getString(
R.string.android_upgrading_fstrim), true);
} catch (RemoteException e) {
}
}
ms.runMaintenance();
}
} else {
Slog.e(TAG, "Mount service unavailable!");
}
} catch (RemoteException e) {
// Can't happen; MountService is local
}

final ArraySet<PackageParser.Package> pkgs;
```

3. performBootDexOpt(PackageParser.Package pkg, int curr, int total) 添加trace（“//mtk”标记

```
private void performBootDexOpt(PackageParser.Package pkg, int curr, int total) {
if (DEBUG_DEXOPT) {
Log.i(TAG, "Optimizing app " + curr + " of " + total + ": " + pkg.packageName);
}
if (!isFirstBoot()) {
try {
Slog.d(TAG + "_Debug","showBootMessage called2...");//mtk
ActivityManagerNative.getDefault().showBootMessage(
```

```
mContext.getResources().getString(R.string.android_upgrading_apk,
curr, total), true);
} catch (RemoteException e) {
}
}
```

其次，SystemServer.java中startOtherServices()添加如下trace（“//mtk”标记

```
try {
mPackageManagerService.performBootDexOpt();
} catch (Throwable e) {
reportWtf("performing boot dexopt", e);
}

try {
Slog.d("PackageManager" + "_Debug","showBootMessage called3...");//mtk
ActivityManagerNative.getDefault().showBootMessage(
context.getResources().getText(
com.android.internal.R.string.android_upgrading_starting_apps),
false);
} catch (RemoteException e) {
}
```

再次，WindowManagerService.java中打开DEBUG_BOOT debug 开关

以上，总共是6个地方的改动。

[FAQ13573]L版本首次开机慢

[FAQ14105]L版本打开WITH_DEXPREOPT宏后首次开机仍慢

[FAQ13232]L 预置apk

[FAQ13697]L 版本如何将第三方so库打包到apk