Android 10

OTA开发使用指南

1.0

2020.02.27

文档履历

版本号	日期	制/修订人	内容描述
1.0	2020.02.27		

目录

1. 前言

1.1 编写目的

本文档目的是让系统开发人员了解 OTA 升级的概念与整体架构，掌握 OTA 升级的流程以及使用方法，并了解 Android OTA 的定制化设计。

1.2 适用范围

本模块适用于 A133/100 Android 10 系统。

1.3 相关人员

系统开发人员。

1.4 专业术语

- OTA: (over-the-air) 空中下载技术。指 Android 系统提供的标准软件升级方式，即通过无线网络下载更新包并无损失地升级系统，而无需通过有线方式进行连接。
- boot: boot 分区，包含 Linux 内核以及最小的 root 文件系统。它负责挂载 system 分区以及其他分区。
- system: system 分区，包含 AOSP(Android Open Source Project) 的系统应用程序以及库文件。正常操作下，该分区是以只读形式挂载。它的内容只能够在 OTA 升级过程中改变。
- recovery: 包含第二个 Linux 系统，包括 Linux 内核以及名为 recovery 的二进制可执行文件，recovery 的作用是用于读取更新包并将其内容更新至其他分区。
- vendor: vendor 分区包含在 AOSP(Android Open Source Project) 中不包含源码的系统程序以及库文件，该分区以只读形式挂载，它的内容只有在 OTA 升级才会被改变。
- cache: 用于暂时保存少数应用程序以及存储下载的 OTA 升级包。其他程序如使用该分区保存文

件有可能会丢失。OTA 升级可能导致该分区数据被完全清除。同时 cache 分区保存 OTA 升级中的日志文件和 command 文件以及升级文件的备份。

- data：用于存储 ota 包，在特殊情况下可更改 data 分区的内容。

2. OTA 升级流程

2.1 OTA 运行原理

Android 平台提供 Google diff arithmetic 差分机制，升级包支持完整升级以及差分升级，OTA 运行原理图如下所示：

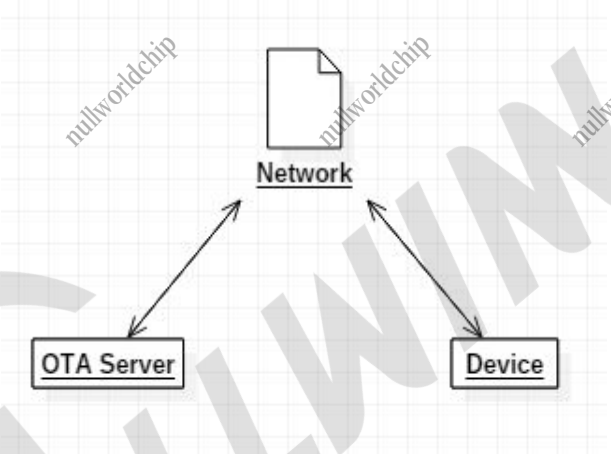

图 1: OTA 运行原理图

1. OTA Server 负责对更新包进行上传，下载以及版本的管理。
2. 开发者在修改 Android 系统后，通过差分制作工具制作出差分包，并使用客户端进行更新包上传和版本管理。
3. 设备通过 wifi 网络进行连接和下载，最后完成更新工作。

2.2 OTA 升级流程介绍

2.2.1 非安全 OTA 升级流程

典型的 OTA 升级流程具体可以分为如下步骤：

1. 设备会周期性检查 OTA 服务器，并确认更新包的可升级性。用户也可以通过将更新包放入内部存储，或者以 U 盘，SD 卡，移动硬盘的形式进行升级。
2. 更新包下载到 cache 或者 data 分区，加密签名将会利用 system/etc/security/otacerts.zip 文件进行验证，通过验证后才能进行 OTA 升级。
3. 设备重启进入到 recovery 模式，至此 recovery 分区中的内核代替了 boot 分区进行启动。
4. recovery 二进制程序被 init 进程启动, 并在 cache/recovery/command 中找到已经下载的更新包。
5. recovery 利用在/res/keys 的公钥验证更新包，假如验证成功进入下一步，否则将信息保存到 cache/recovery/目录下的日志文件，并重启设备。
6. 更新包的数据被解压并用于更新 boot,system 或者有需要时更新 vendor 分区。此时 recovery 的更新补丁保存在 system 分区当中。boot0，uboot 也在该过程进行更新。
7. 设备重新启动，并分为以下两步进行：
    - 更新的 boot 分区被装载，它会挂载并启动 system 分区的可执行程序。
    - 在正常启动的过程，系统将会判断目标 recovery 分区是否存在，假如存在，则 recovery 分区通过打补丁方式进行更新。

3. OTA 模块使用说明

3.1 OTA 的升级范围

原生 Android 提供的 Recovery 升级程序只支持更新 system 分区、recovery 分区及 boot 分区。除此之外，我们根据产品特点，给 Recovery 扩展了一些专有功能，以满足 BSP 的更新需要。

分区类型	是否支持	是否原生升级内容
Boot 分区更新	√	√
System 分区更新	√	√
Vendor 分区更新	√	√
Recovery 分区更新	√	√
Env 分区更新	√	×
Bootloader 分区更新	√	×
Boot0/Uboot 升级	√	×
sys_config.fex 更新	√	×
board.dts 更新	√	×
sys_partition.fex 更新	×	×

值得注意的是，BSP 中的关于模块的大部分配置都集中在 board.dts 中，如果需要更新 board.dts 的配置，就必须通过更新整个内核。如果希望自己实现制作 ota 包的脚本，可以参考以下目录的脚本文件：android/device/softwinner/common/vendorsetup.sh

3.2 制作 OTA 包步骤

使用 OTA 前首先需要区分三个包：

- TargetFile: 包含制作时当前编译版本的 system 分区，boot 分区，recovery 分区等内容，可用于制作 OTA 完整包和差分包。
- OTA 完整包：包含本次升级版本的所有内容，可以从之前各个版本直接升级到当前的版本。制作完整包需要当前版本的 TargetFile。

- OTA 差分包：包含本次升级版本和之前特定一个版本的升级内容，只适用于之前特定一个版本升级到当前版本。制作差分包需要之前特定版本的 TargetFile 和当前版本的 TargetFile。

3.2.1 制作 OTA 完整包

3.2.1.1 制作 OTA 完整包命令

打包过程：

```
$ source build/envsetup.sh
$ lunch
$ make -j8
$ pack4dist[-d][-v]
```

如果需要对固件进行签名，把相关签名文件放入 android/vendor/security 目录，流程不变。对于安全固件，需要加上-v 参数启用安全系统校验。

```
$pack4dist 非安全
$pack4dist -d 卡打印升级包
$pack4dist  -v 安全
$pack4dist  -d -v 安全卡打印升级包
```

使用上述打包过程 pack4dist 后会自动生成目标文件包（target-files-package）路径为：$OUT/obj/PACKAGING/target_files_intermediates/$TARGET_PRODUCT-target_files.zip。若包含签名目标文件包，则路径为：$OUT/signed_target_files.zip 注：生成的 target_files.zip 文件需要与固件一同保存，用于后续生成 OTA 包。

3.2.1.2 pack4dist 命令执行过程

pack4dist 后会自动生成目标文件包是因为封装了如下命令：

1.TargetFile 签名

制作带签名的 OTA 升级包的流程如下：

```
$./build/tools/releasetools/sign_target_files_apks -d [key_path]
[unsigned_target_file.zip] [signed_target_file.zip]
```

[key_path] 为存放 key 文件夹的路径，（如果没有签名文件，则默认生成不签名的 ota 包）需要包括 4 个 key 分别是 media，platform，releasekey，shared，具体包含以下文件：media.pem，media.x509.pem，platform.pk8，releasekey.pem，releasekey.x509.pem，shared.pk8，media.pk8，platform.pem，platform.x509.pem，releasekey.pk8，shared.pem ，shared.x509.pem [unsigned_target_file.zip] 表示上一步生成的没有签名的 TargetFile, [signed_target_file.zip] 表示命令输出得到的经过签名的 TargetFile,

2. 从签名过的 TargetFile 得到镜像（boot.img,system.img 和 recovery.img)

```
$ ./build/tools/releasetools/img_from_target_files
   [signed_target_file.zip]  [img.zip]
```

[signed_target_file.zip] 表示经过签名的 TargetFile, [img.zip] 表示命令输出得到的镜像压缩包。

3. 解压 img.zip, 得到的 boot.img,system.img 和 recovery.img 复制到 out/target/product/[device]/下面，重新 pack 得到可烧录的固件，是签名过的固件。

4. 生成 ota 包完整包

```
$./build/tools/releasetools/ota_from_target_files --block
   [target_file.zip] [ota_full.zip]
```

[target_file.zip] 表示最终的 TargetFile

[ota_full.zip] 表示命令输出得到的 OTA 完整包

3.2.2 制作 OTA 差分包

```
$./build/tools/releasetools/ota_from_target_files  -i
   [target_file_v1.zip] [target_file_v2.zip] ota_inc.zip
```

[target_file_v1.zip] 表示经过签名的版本 v1 的 TargetFile

[target_file_v2.zip] 表示经过签名的版本 v2 的 TargetFile

[ota_inc.zip] 表示命令输出得到的 OTA 差分包

注意：

1. 该差分包仅对指定的前一版本固件有效。
2. 制作一个完整包，也会生成当前版本的一个 target-file 文件包。

4. 使用 OTA 包升级

4.1 从应用升级

将 ota 包放到外部存储或者内部存储中在桌面通过以下方式找到升级入口，选择 ota 包进行升级

1. 系统语言为英文时:Settings–>Device Preferences–>About–>System update

2. 系统语言为中文时: 设置–> 设置偏好设置–> 关于–> 系统更新

4.2 Recovery 升级

4.2.1 Apply update from ADB

1.将固件放在PC端，如：E:/update.zip。
2.进入Recovery。
3.选择Apply update from ADB。
4.打开cmd，并输入adb sideload E:/update.zip。
5.等待打印Install from ADB complete.升级完成。
6.选择reboot system now重启并进入android。

4.2.2 Apply update from TFcard or USB

1.将固件放入TF卡或U盘中。
2.进入Recovery。
3.插入TF卡或U盘
4.在Recovery菜单中选择Apply update from TFcard **or** USB
5.找到升级包的路径并选择开始升级。
6.等待打印Install from SD card complete.升级完成。
7.选择reboot system now重启并进入android。

5. FAQ

5.1 升级注意事项

5.1.1 OTA 不能改变分区数目及其大小

Recovery 只是一个运行在 Linux 上的一个普通应用程序，它并没有能力对现有分区表进行调整，所以第一次量产时就要将分区的数目和大小确定清楚，杜绝后续升级调整分区数目及其大小的想法，OTA 不能改变分区数目和分区的大小。

5.1.2 cache 分区的大小确定

原生 Recovery 机制中，因为 Recovery 内的分区挂载路径与 Android 的分区挂载路径并不完全相同，所以在 Android 上层传入更新包地址时，必须要保证这个包路径在 Recovery 和 Android 系统都是相同的。

能够读写的分区中只有 cache 分区和 data 分区会被 Recovery 和 Android 系统同时挂载，这意味着需要将包放这两个分区中，Recovery 才能识别。所以 Google 原生策略中，当在外部储存选择一个升级包时，都默认复制到 cache 分区中。所以在划分分区时需要注意要分配 cache 分区足够大的空间，否则可能出现无法容纳更新包而导致无法升级的问题。

5.1.3 misc 分区需要有足够的权限被读写

misc 分区是 Recovery 与 Android 之间的桥梁，如果 misc 分区的读写权限过高，会导致上层应用无法对其写入数据，则会令 Recovery 功能异常。检验此功能是否存在问题时，请确保 misc 分区的设备节点/dev/block/xxx 和其软链接/dev/block/by-name/misc 有足够的权限被读写。

```
root@android:/dev/block/by-name# ls -l
lrwxrwxrwx root     root   2000-01-02 07:16 misc -> /dev/block/mmcblk0p6(misc分区软链接)
...
root@android:/dev/block # ls -l
```

```
brw-rw---- 1 system root 179,   7 2019-09-16 10:15 /dev/block/mmcblk0p6
.....
```

5.2 制作 OTA 包常见问题和注意事项

1.TargetFile 和固件是否匹配的区分方法在 Android 设备执行 adb pull /system/build.prop 会得到这个固件的 build.prop 文件对于 TargetFile，解压出来查看 SYSTEM/build.prop，对比这两个 build.prop 如果一致，表示这个固件和 TargetFile 是匹配的.

2. 差分包升级失败

在升级差分包时，会检验经过修改的 apk 或代码部分。解压缩差分包后，升级脚本的位置在 (/META-INF/com/google/android/updater-script) 目录下，对于修改过的部分，差分升级会首先检查升级前版本的 SHA 值以及升级后的 SHA 值，只有匹配后，才能够升级成功。因此客户可以判断升级前的文件或 APK 的 SHA 值是否准确，从而判断差分升级前的固件与升级差分包是否匹配。以某文件的差分升级为例子，在 updater-script 脚本中的形式如下：

```
apply_patch_check([文件名],[升级后SHA值]，[升级前SHA值]) || abort("...");
```

由此可以使用命令 shasum 对差分升级的文件进行 SHA 查询。差分升级需要使用升级前的 targetfile 以及升级后的 targetfile，如升级文件 system/bin/A.apk, 那么可以解压升级前后的 targetfile, 并查看 system/bin/A.apk 的值与升级脚本的 SHA 值是否匹配，假如匹配，则能够保证该文件差分升级成功。否则该文件不能差分升级成功，其原因归根到底是升级前的固件与生成差分包的升级前的 targetfile 不对应。

6. Declaration

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner. The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This document neither states nor implies warranty of any kind, including fitness for any particular application.tates nor implies warranty of any kind, including fitness for any particular application.