1 用户指南

1.1 ota 升级包是什么

ota 升级功能是君正开发的一套在线升级程序。ota 升级包服务于 ota 升级程序，ota 升级程序开始的第一步就是解析校验 ota 升级包，只有升级包验证成功才会进行升级动作，因此对于用户来说，只有熟悉 ota 升级包的生成过程，才能保证正确升级。

1.2 如何制作 ota 升级包

ota 升级包制作程序的组成文件关系如下图

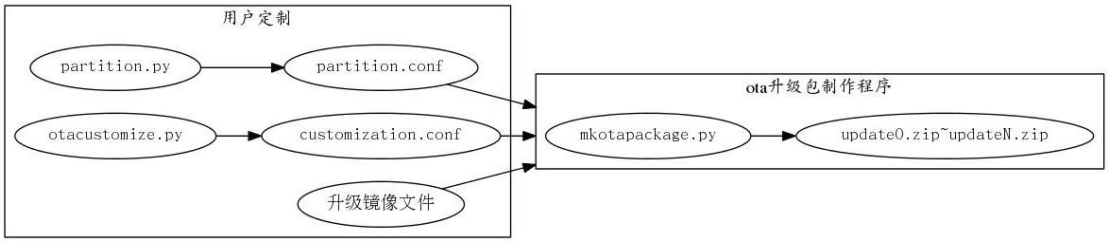

图 1-1 组成文件关系图

ota 升级包制作程序 mkotapackage.py 依赖于 partition.conf 和 customization.conf 这两个配置文件，成功运行 mkotapackage.py 将生成 update0.zip~updateN.zip 个升级包。

备注: N 的大小依赖于升级方式的选配。

partition.conf 主要定义了系统分区表结构，这个分区表必须和终端机器的当前分区配置一致，否则将升级失败，该文件一般只需要配置一次即可，具体设置请参考后续章节”1.4 如何配置”。

customization.conf 主要定义了镜像名称、镜像类型、镜像大小、要烧写的位置以及升级方式，该文件是用户定制部分的核心，具体设置请参考后续章节”1.4 如何配置”。

升级镜像文件是将要烧写的一组镜像。

mkotapackage.py 是 ota 升级包制作主程序，运行该脚本当识别到正确的 partition.conf、customization.conf 以及升级镜像文件后，即可生成 update.zip 升级包。

1.3 如何使用 ota 升级包

ota 升级包的部署位置如下图

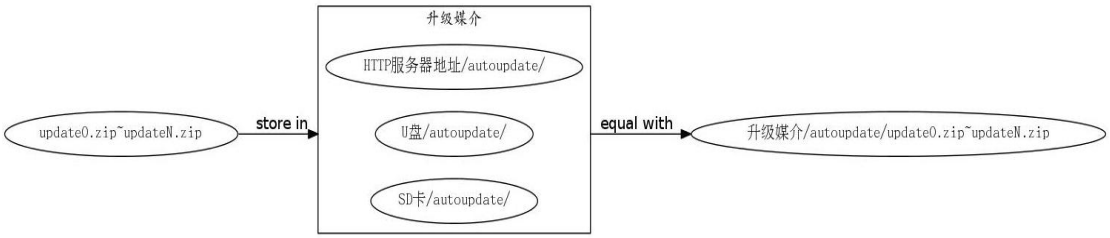

图 1-2 升级包部署

将所有升级包 update0.zip~updateN.zip 部署在升级媒介的指定位置，统一位置为 autoupdate, ota 升级程序捕捉到升级包后开始自动升级。

1.4 如何配置

1.4.1 配置 partition.conf

ota 升级程序获取到配置后，第一步是要验证 partition.conf 配置文件是否和系统当前配置匹配，partition.conf 的布局结构如下图

```
[storageinfo]
mediumtype=nor
capacity=16MB
[partition]
item1=uboot,0x0,0x40000,mtdblock0
item2=kernel,0x40000,0x300000,mtdblock1
item3=rootfs,0x360000,0xca0000,mtdblock2
```

图 1-3 分区配置信息

存储信息(storagetype)和分区表(partition)是必须要根据系统配置指定，存储信息包含存储媒介类型(mediumtype)和容量大小(capacity)，分区表的列分布类型包含分区名称、分区首地址、分区大小以及分区所属块，每个部分的指定请严格按照系统分区表来定义。

建议使用 partition.py 自动生成分区表，该脚本使用方法如下图

```
torence@torence-ThinkPad-X1-Carbon:~/Work/jz/ota$ python -m otapackage/customer/partition -h
usage: partition.py [-h] mediumtype

positional arguments:
  mediumtype  The medium type to update on. Must be one in ('mmc', 'nand',
              'nor')

optional arguments:
  -h, --help  show this help message and exit
torence@torence-ThinkPad-X1-Carbon:~/Work/jz/ota$ python -m otapackage/customer/partition nor
torence@torence-ThinkPad-X1-Carbon:~/Work/jz/ota$ ls -l otapackage/customer/generated/partition.conf
```

图 1-4 partition.py 使用方法

根据存储介质的不同，可选配的参数可以是 nor, nand 或 mmc，指定配置后将自动生成默认分区配置文件 partition.conf，生成位置在 otapackage/customer/generated

partition.py 脚本对应的默认分区配置信息位于 otapackage/customer/config.py

1.4.2 配置 customization.conf

该文件是用户配置的核心文件， 升级包制作工具是根据该配置文件识别烧写镜像的，包含以下可配置信息，如图:

```
[update]
imgcnt=3
[image1]
name=x-loader-pad-with-sleep-lib.bin
type=normal
offset=0x0
updatemode=full
[image2]
name=xImage
type=normal
offset=0x40000
updatemode=slice
[image3]
name=system.jffs2
type=jffs2
offset=0x360000
updatemode=slice
```

图 1-5 customization.conf

升级信息(update)包含要升级的镜像数目(imgcnt)以及每个镜像的配置信息(imageN)，镜像配置信息包括镜像名称(name)、镜像类型(type)、烧写的位置(offset)以及升级方式(updatemode)，其中镜像名称需要和待烧录的镜像文件名称匹配；镜像类型的设置规则为非文件系统镜像类型是 normal，文件系统镜像需要根据不同文件系统类型指定，目前支持的文件系统有 ubifs、jffs2, cramfs 和 yaffs2；烧写的位置请严格根据分区配置来指定，以防止烧录过程中失败；升级方式支持全量升级(full)和拆包升级(slice)两种方式，请根据需要设置。

镜像名称请严格按照镜像文件名称指定，图 1-5 配置信息对应的镜像文件如下图

```
torence@torence-ThinkPad-X1-Carbon:~/Work/jz/ota$ ls -l otapackage/image
total 14432
-rw-r--r-- 1 torence torence 11730944 Oct  6 11:19 system.jffs2
-rw-rw-r-- 1 torence torence  3018816 Oct  6 11:19 xImage
-rw-rw-r-- 1 torence torence    24576 Oct  6 11:19 x-loader-pad-with-sleep-lib.bin
```

图 1-6 与 customization.conf 匹配的升级镜像

建议使用 cutomize.py 根据提示设置生成 customization.conf , 该脚本使用方法如下图

```
torence@torence-ThinkPad-X1-Carbon:~/Work/jz/ota$ python -m otapackage/customer/customize -h
usage: customize.py [-h] [-c DEVCTL] imgcnt

positional arguments:
  imgcnt                The total count of images you want to update, must be
                        number but not character

optional arguments:
  -h, --help            show this help message and exit
  -c DEVCTL, --devctl DEVCTL
                        The special flag for device control. Optional value is
                        listed as follows. --devctl=1: Chip erase will be
                        issued fisrtly before the update main sequence run
torence@torence-ThinkPad-X1-Carbon:~/Work/jz/ota$ python -m otapackage/customer/customize 3
image1 name: x-loader-pad-with-sleep-lib.bin
image1 type [must be one in ('normal', 'ubifs', 'jffs2', 'cramfs', 'yaffs2')]: normal
image1 offset [hexadecimal is better]: 0x0
image1 updatemode [must be one in ('full', 'slice')]: full
image2 name: xImage
image2 type [must be one in ('normal', 'ubifs', 'jffs2', 'cramfs', 'yaffs2')]: normal
image2 offset [hexadecimal is better]: 0x40000
image2 updatemode [must be one in ('full', 'slice')]: slice
image3 name: system.jffs2
image3 type [must be one in ('normal', 'ubifs', 'jffs2', 'cramfs', 'yaffs2')]: jffs2
image3 offset [hexadecimal is better]: 0x360000
image3 updatemode [must be one in ('full', 'slice')]: slice
torence@torence-ThinkPad-X1-Carbon:~/Work/jz/ota$ ls -l otapackage/customer/generated/customization.conf
```

图 1-7 cutomize.py 使用方法

首先指定要升级的镜像个数，上面的例子为 3，再根据提示分别设置 image1~3 的配置信息。

2 升级包详解

2.1 升级包内容

通过 mkotapackage.py 制作生成的升级包内容如下图:

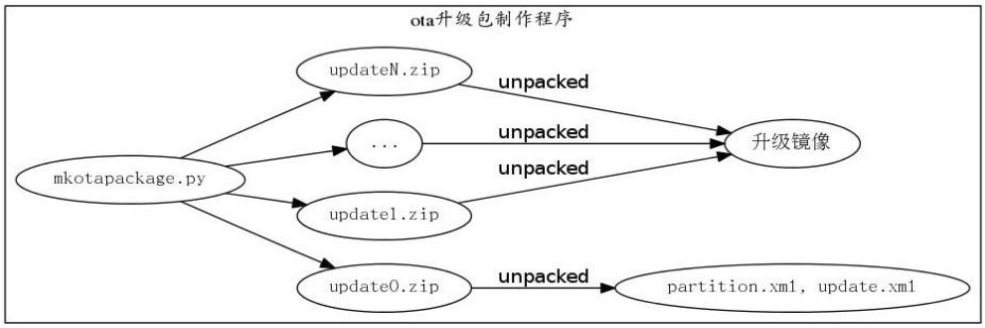

图 2-1 升级包内容简介

ota 升级程序首先从 update0.zip 提取出 partition.xml 和 update.xml 配置文件，再从 update1~N.zip 中提取出镜像文件，镜像文件的数据排布由升级方式决定。

2.1.1 partition.xml 内容

partition.xml 是直接从 partition.conf 转换生成的，内容如下

```
−<device>
  −<devinfo>
      <type>nor</type>
      <capacity>16MB</capacity>
    </devinfo>
  −<partition>
    −<item>
        <name>uboot</name>
        <offset>0x0</offset>
        <size>0x40000</size>
        <blockname>mtdblock0</blockname>
      </item>
    −<item>
        <name>kernel</name>
        <offset>0x40000</offset>
        <size>0x300000</size>
        <blockname>mtdblock1</blockname>
      </item>
    −<item>
        <name>rootfs</name>
        <offset>0x360000</offset>
        <size>0xca0000</size>
        <blockname>mtdblock2</blockname>
      </item>
    </partition>
  </device>
```

图 2-2 partition.xml

2.1.2 update.xml 内容

update.xml 作为升级包与升级程序沟通的媒介 ,是 ota 升级包制作的核心部分，其中定义总共要升级的镜像数以及每个镜像的描述信息， 镜像描述信息包括镜像名称、类型、写入位置、整包大小以及升级方式，升级方式分为整包升级和拆包升级，在拆包升级时，标明了分片大小以及分片数量。

注: 所有大小和位置描述的单位都是字节

update.xml 的数据结构如下图

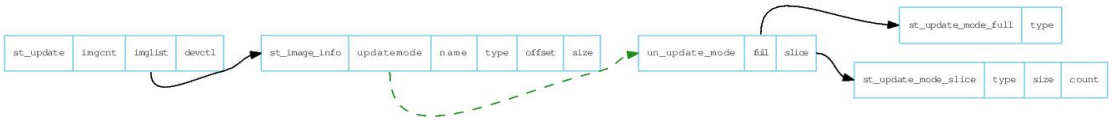

图 2-3 updata.xml 数据结构

完整内容如下图

```
−<update>
   <devctl>0</devctl>
  −<imagelist count="3">
   −<image>
      <name>x-loader-pad-with-sleep-lib.bin</name>
      <type>normal</type>
      <offset>0x0</offset>
      <size>24576</size>
    −<updatemode>
       <type>0x200</type>
      </updatemode>
    </image>
   −<image>
      <name>xImage</name>
      <type>normal</type>
      <offset>0x40000</offset>
      <size>3018816</size>
    −<updatemode>
       <type>0x201</type>
       <size>1048576</size>
       <count>3</count>
      </updatemode>
    </image>
   −<image>
      <name>system.jffs2</name>
      <type>jffs2</type>
      <offset>0x360000</offset>
      <size>11730944</size>
    −<updatemode>
       <type>0x201</type>
       <size>1048576</size>
       <count>12</count>
      </updatemode>
    </image>
  </imagelist>
</update>
```

图 2-4 update.xml

其中镜像类型<image><type>的定义如下

普通镜像(‘normal’)、ubi 文件系统(‘ubifs’)、jffs2 文件系统(‘jffs2’), cramfs 文件系统(‘cramfs’)和 yaffs2 文件系统(‘yaffs2’)

镜像升级方式<updatemode><type>的枚举定义如下

UPGRADE_MODE_FULL=0x200,UPGRADE_MODE_SLICE=0x201

update.xml 文件的等效 C 语言结构体描述如下:

```
struct st_update_mode_full {
    int type; //升级方式类型
};
struct st_update_mode_slice {
    int type; //升级方式类型
    unsigned int size; //分片大小
    unsigned int count; //分片数量
};
union un_update_mode {
    struct st_update_mode_full full;      //全量升级
    struct st_update_mode_slice slice; //拆包/分片升级
}
struct st_image_info {
    char *name; //镜像文件名称
    char *type;          //镜像类型
    unsigned int offset;    //写入位置
    unsigned int size; //镜像大小
    union un_update_mode updatemode;    //升级方式
};
struct st_update {
    int devctl;          // 1: nand scrub
    int imgcnt;          //镜像数量
    struct st_image_info *imglist;    //镜像描述列表
};
```

其中 devctl 字段是考虑为后续特殊操作预留的字段，目前只当 devctl=0 或 1 时生效，当 devctl=0 时，对于每个待升级的镜像都是先擦除镜像覆盖区域后升级镜像，镜像没有覆盖的区域不会被擦除；当 devctl=1 时，表示升级时先擦除整片存储介质，再升级各个镜像。

2.2 升级包生成规则

升级包生成规则的定义是打包程序的核心，如下图

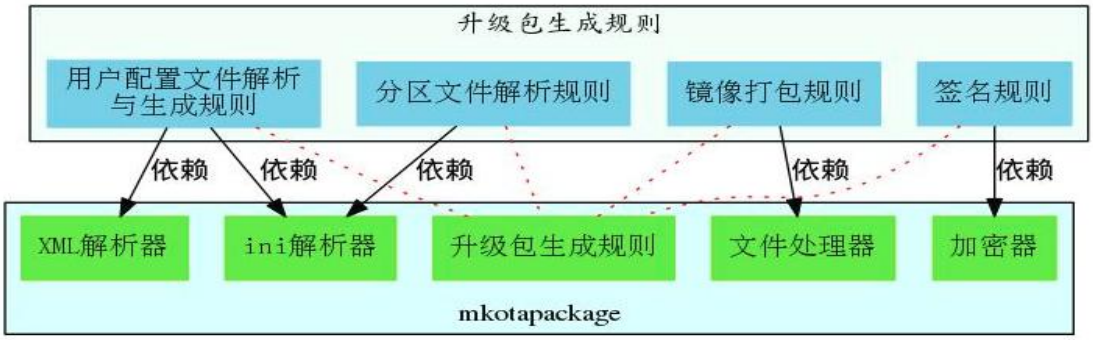

图 2-5 mkotapackage 结构图

mkotapackage 程序流程图如下

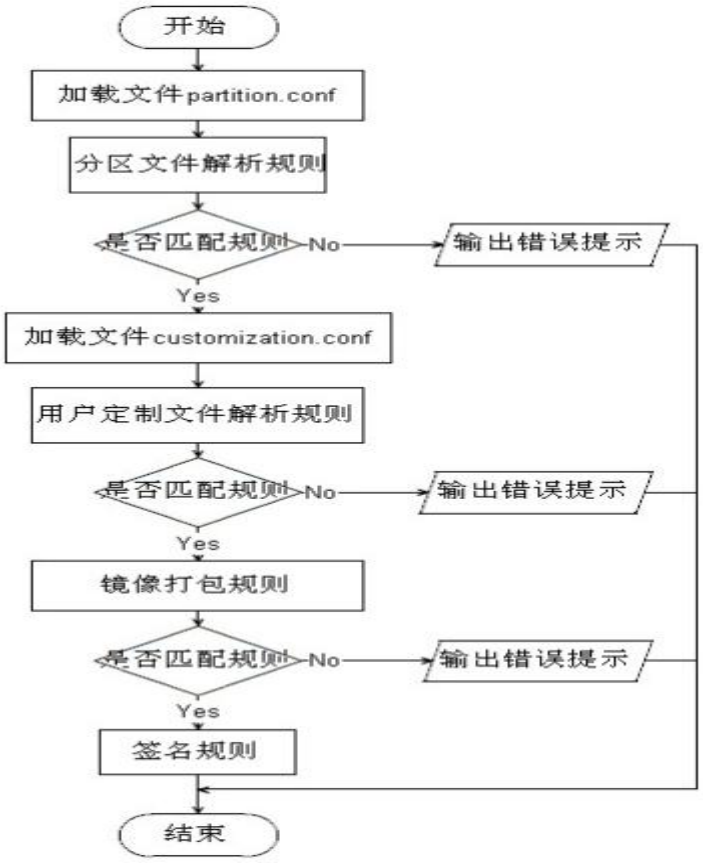

图 2-6 mkotapackage 流程图

下面逐一分析各个规则定义，首先分析分区文件解析规则，如下图

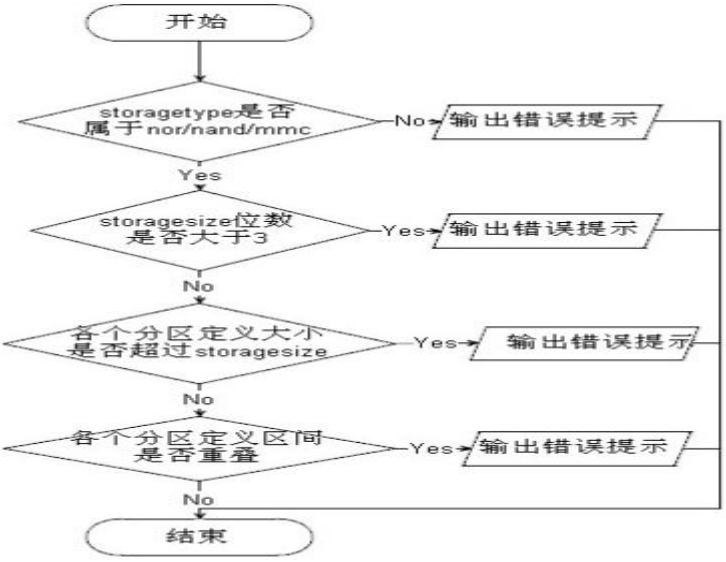

图 2-7 分区文件解析流程图

用户定制文件解析程序流程图如下

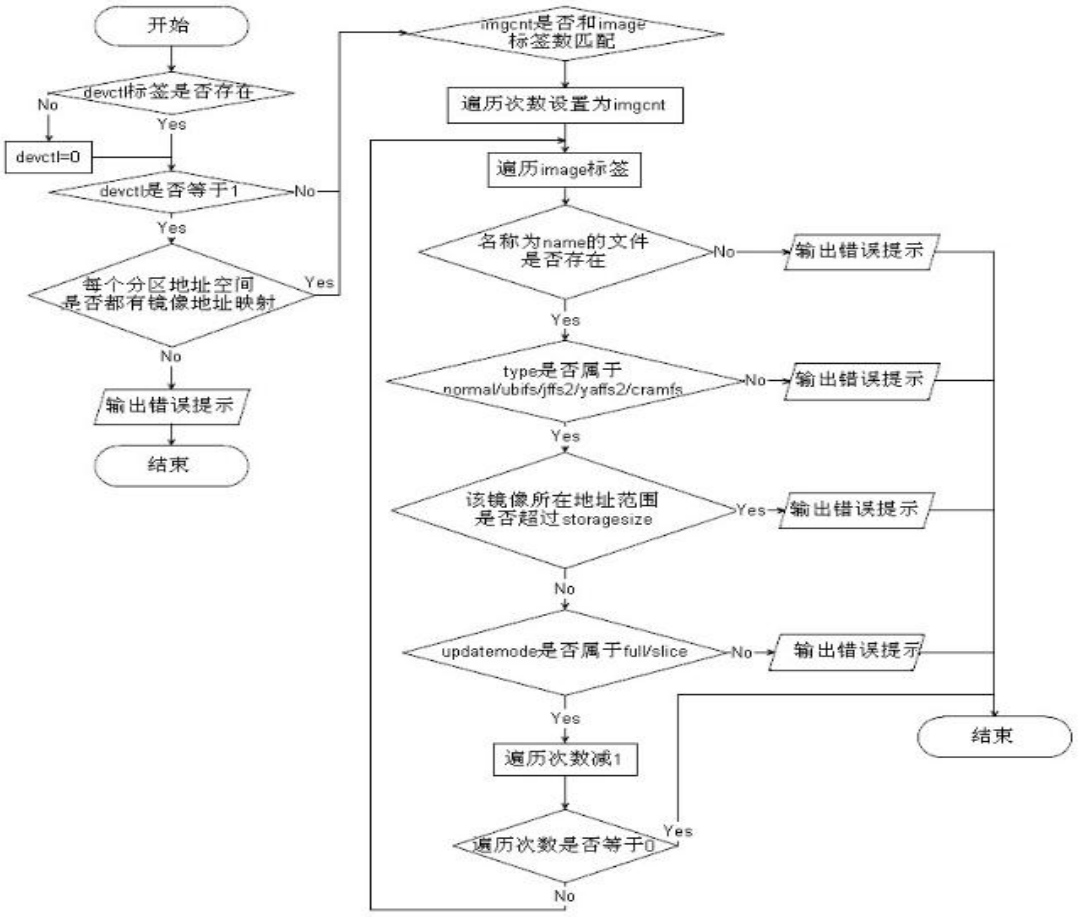

图 2-8 用户定制文件解析流程图

镜像打包规则流程如下

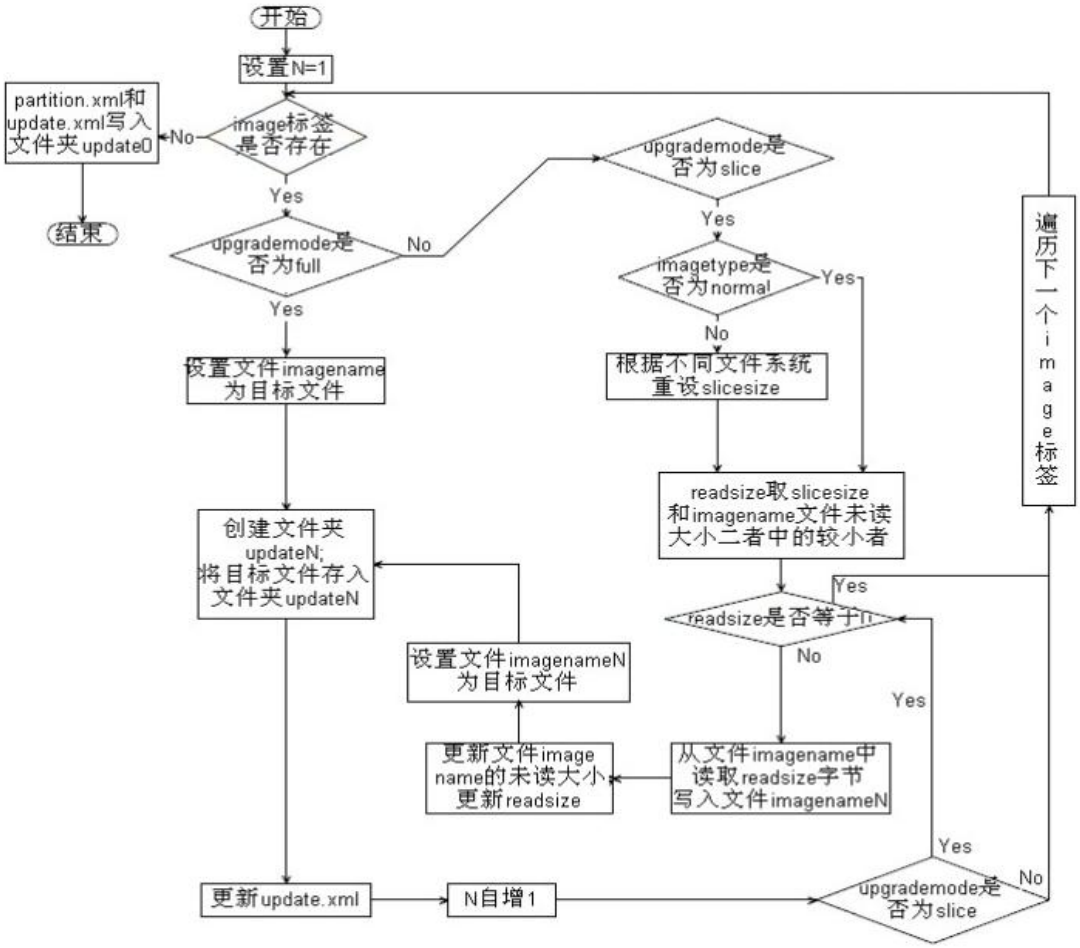

图 2-9 镜像打包流程图

签名流程如下

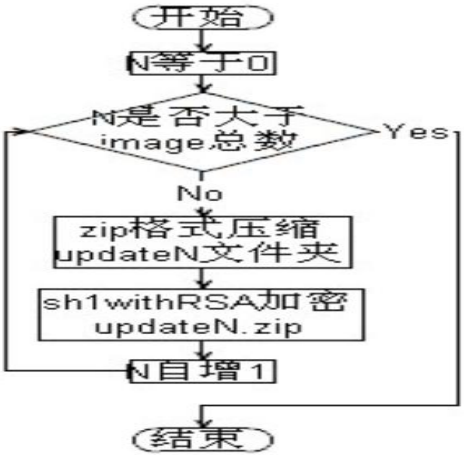

图 2-10 签名流程图