Ingenic
Semiconductor

Linux recovery 部署说明

文档历史：

版本	作者	注释
1.0	张延明	新建

目录

一、产品板端

1. 资料说明

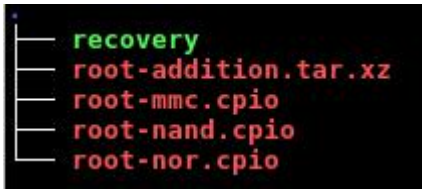
首先解压 linux-recovery.tar.xz

mkdir linux-recovery

tar -xvf linux-recovery.tar.xz -C linux-recovery

压缩包包含以下内容：

- client-side:包含了 recovery 应用和文件系统，如下图：



用户需要将 recovery 应用放到文件系统的/sbin/目录下，以 nand 为例：

先解开 root-nand.cpio

```
mkdir root-nand
cd root-nand
sudo cpio -idm < ../root-nand.cpio
cp -a ../recovery sbin/
```

重新打包

```
find ./ -print | cpio -H newc -ov > ../root-nand.cpio
```

root-nand.cpio 仅仅包含基本的 busybox、libc.so 和 ld.so 以及一些图片资源。recovery 应用所依赖的其他资源在运行时通过挂载 rootfs，从 roofs 拷贝过来, 详细参考 root-nand.cpio 中 sbin/envsetup.sh。这样可以减小 recovery.img 分区。rootfs-addition.tar.xz 包含了 recovery 应用运行所依赖的库和其他资源。**如果用户担心 rootfs 分区不能正常挂载（例如升级 rootfs 时突然断电等），那么就需要将 root-addition.tar.xz 里的内容按照已有的目录结构拷贝到 root-nand.cpio 中，sbin/envsetup.sh 便可以不用执行(/etc/init.d/S11env)，针对 rootfs-addition.tar.xz 可根据实际使用情况进行裁剪(例如不需要网络相关的库可删除)。这样 recovery 分区会变大些，但是只要 bootloader 分区没有被破坏，系统任何一个分区都可以升级/修复，因为 recovery.img 完全在内存里运行！**

recovery 应用的配置文件 recovery.conf 放在/etc 下，如下图：

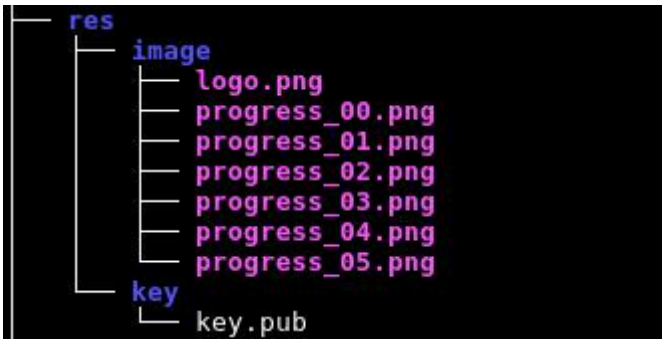
可配置的内容如下：

```
Version="2.0";

Application:
{
    Server:
    {
        ip="194.169.2.59";
        url="http://194.169.2.59:8008/data/recovery";
    };
};
```

其中，Version 字段配置 recovery 版本，不可修改，ip 字段配置 OTA 服务器的 IP 地址，url 字段配置升级包在服务器上的路径。注意，配置文件名和路径一定不能更改！

/res 存放公钥、Logo 以及进度条图片，图片只支持 PNG 格式，这些文件名和路径一定不能修改。如果用户更换进度条图片，文件名必须是 progress_XX.png，XX 必须是连续的数字。



res/image 可以存放名为 font.png 的字体图片，recovery 只支持 ASCII 码显示。字体图片如下：

图片高度必须为 96x2，高度为 96 个像素，上面一行为正常字体，下面一行为加粗字体。如果没有提供该文件则使用内置的 10x18 的字体。

- document: 包含说明文档
- resource: 包含了预置的一些数字证书和相应的私钥
- server-side: 包含了服务器端的资源，详细请见第二章服务器端
- tools: 包含了主机端的工具，如下图：

dumpkey.jar 用于从数字证书提取公钥，用法如下：

java -jar dumpkey.jar ../resource/security/testkey.x509.pem > key.pub

将生成的 key.pub 预置到 root-nand.cpio 中 res/key 目录下。

make_key 用于生成用户自定义的数字证书和相应的私钥，用法如下：

make_key releasekey '/C=CN/ST=Guangdong/L=Shenzhen/O=Company/OU=Department/CN=Your Name/emailAddress=YourE-mailAddress'

命令执行完后会在当前目录下生成数字证书和相应的私钥，如下图：

2. 编译 recovery.img

在配置好 root-nand.cpio 后下面需要编译 recovery.img

首先需要一份内核配置，该配置可以和正常启动的配置一样，也可以根据具体情况裁减一些(**针对 nor flash 的用户需尽可能的裁剪 kernel, 不用的文件系统和驱动和网络相关需全部去除**)，可以减小 kernel 镜像大小。recovery 内核配置中需要增加以下配置：

```
CONFIG_BLK_DEV_INITRD=y
CONFIG_INITRAMFS_SOURCE=root-nand.cpio 文件路径
CONFIG_RD_XZ=y
CONFIG_INITRAMFS_COMPRESSION_XZ=y
```

配置如下所示：

```
General setup    --->
    [*] Initial RAM filesystem and RAM disk (initramfs/initrd) support
     (../linux-recovery/client-side/root-nand.cpio) Initramfs source file(s)
    [ ]   Support initial ramdisks compressed using gzip
    [ ]   Support initial ramdisks compressed using bzip2
    [ ]   Support initial ramdisks compressed using LZMA
    [*]   Support initial ramdisks compressed using XZ
    [ ]   Support initial ramdisks compressed using LZO
          Built-in initramfs compression mode (XZ)    --->
```

在添加好 recovery 配置后，开始编译：
以 halley2_V20 为例：
    make halley2_sfcnand_recovery_defconfig
    make recovery
将生成的 recovery 烧录到 recovery 分区。

3. 添加 recovery 分区

参考烧录工具手册。

4. 编译 x-loader

recovery.img 生成后还需要 bootloader 支持引导 recovery.img。
下面编译 bootloader:
cd x-loader
vim include/configs/BOARD_NAME.h
在文件开头处添加#define CONFIG_RECOVERY
make
将生成的 bootloader 烧录到 bootloader 分区。

二、服务器端

本章主要描述如何在服务器端制作 ota 升级包，来用于系统升级。

1. 升级包制作程序说明

升级包制作程序目录结构如下图:

```
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ ls
customize.sh  Makefile  otapackage  pack.sh  partition.sh
```

partition.sh: 分区生成脚本，用于制作分区配置文件
customize.sh: 客户升级定制脚本，用于制作升级信息配置文件
pack.sh: 拆包脚本， 依赖于 partition.sh 和 customize.sh 生成的分区配置文件和升级配置文件
Makefile: 封装调用 pack.sh 进行拆包
otapackage: 升级包制作程序源码，所有脚本都依赖于它

它们的调用顺序如下:
第一步：先运行 partiton.sh 生成分区配置文件
第二步：再运行 customize.sh 生成客户定制升级文件
第三步：最后运行 make 来拆包

下面将对每一个步骤展开作逐一说明。

1.1 partition.sh

partition.sh 是分区生成脚本，用于制作分区配置文件，实际调用的主程序是 otapackage/customer/partition.py，该脚本只在系统分区表变更后使用，目的是生成和系统分区表一致分区配置文件。

在介绍使用方法之前，需要作以下声明

注意：由 partition.sh 生成的分区表必须和系统分区表匹配，否则升级前后，可能出现未知的后果。

使用方法主要分为以下三步:

第一步：查看帮助说明

```
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ ./partition.sh --help
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
```

partition.sh 后面需要添加一个存储介质参数，参数可以是 nor/nand/mmc 中的一种

第二步：按照帮助提示，输入信息，生成分区表配置文件

以下示例是在生成 nandflash 存储介质的分区配置文件

```
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ ./partition.sh nand
```

最终文件生成于 otapackage/customer/generated 目录下

```
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ ls otapackage/customer/generated/
partition_nand.conf
```

第三步：更改分区表

更改方法分为两种：

1) 更改源程序文件

文件位置: otapackage/customer/config.py

举例说明 nand 的配置更改，其余存储介质配置更改类似

```
# nand flash configuation
re_nand_tag = 'nand'
re_nand_device_size = '128MB'
re_nand_partition = (
                    ("uboot", "0x0", "0x100000", "mtdblock0"),
                    ("kernel", "0x100000", "0x800000", "mtdblock1"),
                    ("oeminfo", "0x900000", "0x80000", "mtdblock2"),
                    ("recovery", "0x980000", "0x800000", "mtdblock3"),
                    ("rootfs", "0x1180000", "0x2800000", "mtdblock4"),
                    ("data", "0x3980000", "0x4680000", "mtdblock5"),
)
```

re_nand_device_size 字段是当前 flash 的容量大小， 单位是 MB， 当前示例使用 128MB 的 nandflash

re_nand_partition 是程序默认分区表，运行 partition.sh 时生成配置文件的分区信息来自于这里， 每一行格式为”分区名称”,“分区起始地址”,“分区大小”,“分区类型”。其中分区类型一栏的尾标命名规则是当前分区号减 1，例如 uboot 分区是所在第一个分区，分区号为 1, 则分区类型起名为 mtdblock0。

请用户根据需要自行更改，**需要特别注意的是，分区表的分区排列顺序必须从上到下要按照地址递增顺序排列，否则程序不予支持。**

2) 更改生成的配置文件

配置文件命名规则是 partition_XXX.conf, XXX 表示存储介质。

已经生成的配置文件如下图

```
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ cat otapackage/customer/generated/partition_nand.conf
[storageinfo]
mediumtype=nand
capacity=128MB
[partition]
item1=uboot,0x0,0x100000,mtdblock0
item2=kernel,0x100000,0x800000,mtdblock1
item3=oeminfo,0x900000,0x80000,mtdblock2
item4=recovery,0x980000,0x800000,mtdblock3
item5=rootfs,0x1180000,0x2800000,mtdblock4
item6=data,0x3980000,0x4680000,mtdblock5
```

capacity 字段请用户根据存储介质大小配置，partiton 字段的每一行格式为” itemN = 分区名称，分区起始地址，分区大小，分区类型”。其中 itemN 命名规则是 item 加分区号，分区类型一栏的尾标命名规则是当前分区号减 1，例如 uboot 分区是所在第一个分区，分区号为 1, 也就是 item1, 分区类型起名为 mtdblock0。

请用户根据需要自行更改，**需要特别注意的是，分区表的分区排列顺序必须从上到下要按照地址递增顺序排列，否则程序不予支持。**

1.2 customize.sh

customize.sh 是客户升级定制脚本，用于制作升级信息配置文件， 实际调用的主程序是 otapackage/customer/customize.py。 升级信息文件存储的是用户每次升级时设置的升级信息，主要包括要升级的目标介质、每个升级镜像名称、类型、升级位置、以及是否对镜像文件拆包等信息。

在介绍使用方法之前，需要作以下声明

注意：cusomize.sh 设定的升级地址必须和分区表匹配，设置时请避免出现升级地址越界或升级地址重叠问题。

使用方法主要分为以下几步:

第一步：查看帮助说明

```
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ ./customize.sh --help
usage: customize.py [-h] [-c DEVCTL] mediumtype imgcnt

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

mediumtype: 存储介质参数，参数可以是 nor/nand/mmc 中的一种

imgcnt: 指定要升级的镜像个数

devctl: 设备控制选项，暂时保留

第二步：按照帮助提示，运行脚本

升级目标存储介质为 nand，有 4 个镜像需要升级，运行后，将产生一系列自助向导，请按照向导提示输入每个镜像信息，如下图：

```
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ ./customize.sh nand 4
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]: 0x100000
image2 updatemode [must be one in ('full', 'slice')]: slice
image3 name: system.ubi
image3 type [must be one in ('normal', 'ubifs', 'jffs2', 'cramfs', 'yaffs2')]: ubifs
image3 offset [hexadecimal is better]: 0x1180000
image3 updatemode [must be one in ('full', 'slice')]: slice
image4 name: data.ubi
image4 type [must be one in ('normal', 'ubifs', 'jffs2', 'cramfs', 'yaffs2')]: ubifs
image4 offset [hexadecimal is better]: 0x3980000
image4 updatemode [must be one in ('full', 'slice')]: slice
```

image name: 需要和要升级的镜像名称完全匹配，上面示例对应的镜像名称如下：

```
torence@torence-ThinkPad-X1-Carbon:~/Work/git/manhhantu/image_nand$ ls
data.ubi  system.ubi  xImage  x-loader-pad-with-sleep-lib.bin
```

image type: 每个镜像的类型，如果是文件系统镜像就按照文件系统类型来命名，当前支持 ubifs、jffs2、cramfs 以及 yaffs2 共 4 种文件系统类型。对于非文件系统镜像名称为 normal

image offset: 镜像烧录位置

image updatemode: 是否拆包，full 表示全量升级，slice 表示拆包，拆包大小默认为 1MB。

第三步：查看生成的配置文件

已经生成的配置文件，命名规则是 customization_XXX.conf, XXX 表示存储介质。

```
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ cat otapackage/customer/generated/customization_nand.conf
[update]
mediumtype=nand
imgcnt=4
[image1]
name=x-loader-pad-with-sleep-lib.bin
type=normal
offset=0x0
updatemode=full
[image2]
name=xImage
type=normal
offset=0x100000
updatemode=slice
[image3]
name=system.ubi
type=ubifs
offset=0x1180000
updatemode=slice
[image4]
name=data.ubi
type=ubifs
offset=0x3980000
updatemode=slice
```

这个配置文件对比上面的自助生成过程，显得清晰明了，具体字段含义不再赘述。

客户每次升级可以直接更改该配置文件，不必使用繁琐的自助程序。如果直接更改该配置文件，请注意 imgcnt 字段和 image 标签总数要匹配，并且 image 标签不要有重复。

并且每个 image 标签排列顺序必须从上到下要按照地址递增顺序排列，否则程序不予支持。

1.3 Makefile

makefile 会调用 otapackage 主程序来拆包，otapackage 实际依赖分区配置文件 partition_XXX.conf 和升级信息配置文件 customization_XXX.conf、key 文件和镜像。

执行 make 之前需要检查以上依赖信息是否存在，因此流程总共分为两步

第一步：检查依赖文件是否存在

1) 配置文件

我们的配置文件如下:

```
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ ls otapackage/customer/generated/
customization_nand.conf  partition_nand.conf
```

这些配置文件的生成请参考章节“1.1 partition.sh”和“1.2 customize.sh”

2) key 文件

默认的 key 文件位置如下图：

```
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ ls ../resource/security/
media.pk8  media.x509.pem  platform.pk8  platform.x509.pem  README  shared.pk8  shared.x509.pem  testkey.pk8  testkey.x509.pem
```

我们选配的 key 文件已经在 Makefile 中指定

RES_SRC := $(TOPDIR)/../resource

SECURITY := $(RES_SRC)/security

KEYNAME := testkey

从上面看我们选用的 key 是 testkey，实际匹配 testkey.x509.pem 和 testkey.pk8。

3) 镜像

otapackage 程序识别的镜像位置是“镜像目录/XXX”,XXX 表示存储介质，可以是 nor,nand,mmc 中的任意一种，如果没有目录“镜像目录/XXX”需要自行创建。

Makefile 中指定镜像目录在”当前目录/image”下

TOPDIR := $(shell pwd)/

IMAGE_SRC ?= $(TOPDIR)/image

这里要注意的是这里我们要升级 nand 存储介质，因此需要在 image 下创建 nand 目录之后把镜像放入该目录下，如下图:

```
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ mkdir -p image/nand
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ cp -arv /home/torence/Work/git/manhhantu/image_nand/* image/nand/
‘/home/torence/Work/git/manhhantu/image_nand/data.ubi’ -> ‘image/nand/data.ubi’
‘/home/torence/Work/git/manhhantu/image_nand/system.ubi’ -> ‘image/nand/system.ubi’
‘/home/torence/Work/git/manhhantu/image_nand/xImage’ -> ‘image/nand/xImage’
‘/home/torence/Work/git/manhhantu/image_nand/x-loader-pad-with-sleep-lib.bin’ -> ‘image/nand/x-loader-pad-with-sleep-lib.bin’
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ ls image/nand/
data.ubi  system.ubi  xImage  x-loader-pad-with-sleep-lib.bin
```

第二步：执行 make

```
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ make
pack start on 'nand'
  adding: update000/ (stored 0%)
  adding: update000/update.xml (deflated 73%)
  adding: update000/device.xml (deflated 73%)
  adding: update001/ (stored 0%)
  adding: update001/x-loader-pad-with-sleep-lib.bin (deflated 71%)
  adding: update002/ (stored 0%)
  adding: update002/xImage_001 (deflated 1%)
  adding: update003/ (stored 0%)

.....

  adding: update038/data.ubi_015 (deflated 13%)
  adding: update039/ (stored 0%)
  adding: update039/data.ubi_016 (deflated 12%)
  adding: update040/ (stored 0%)
  adding: update040/data.ubi_017 (deflated 41%)
  adding: update041/ (stored 0%)
  adding: update041/data.ubi_018 (deflated 72%)
```

最终生成的镜像位于以下位置:

```
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ ls out/
global.xml  nand
torence@torence-ThinkPad-X1-Carbon:~/Work/git/linux-recovery/server$ ls out/nand/
update000.zip  update004.zip  update008.zip  update012.zip  update016.zip  update020.zip  update024.zip  update028.zip  update032.zip  update036.zip  update040.zip
update001.zip  update005.zip  update009.zip  update013.zip  update017.zip  update021.zip  update025.zip  update029.zip  update033.zip  update037.zip  update041.zip
update002.zip  update006.zip  update010.zip  update014.zip  update018.zip  update022.zip  update026.zip  update030.zip  update034.zip  update038.zip
update003.zip  update007.zip  update011.zip  update015.zip  update019.zip  update023.zip  update027.zip  update031.zip  update035.zip  update039.zip
```

该位置可以在 makefile 中修改, 修改位置如下：

OUTDIR := $(TOPDIR)/out

请把 out 目录下的所有信息进一步部署到服务器或存储介质后进行下一步升级动作。

2. 格式化镜像

本章将针对分区格式化需求，讲解如何制作特殊镜像用于格式化分区。

主要讲解 ubifs 和 jffs2 两种文件系统的格式化镜像制作方法。

2.1 ubifs

在当前文件系统中任意选择一个位置，制作格式化镜像, 过程如下

~/work/x1000/recovery_testseed$ mkdir fs_format

~/work/x1000/recovery_testseed$ ls -l fs_format

#注意：如果要用 ubi 文件系统格式化分区，这个文件夹里面的内容可以为空

~/work/x1000/recovery_testseed$ mkfs.ubifs -e 0x1f000 -c 1024 -m 0x800 -d fs_format -o formator.ubi

~/work/x1000/recovery_testseed$ ls -l formator.ubi

-rw-rw-r-- 1 torence torence 1777664 Nov 2010:56 formator.ubi

生成 formator.ubi 后，重新执行“1.2 customize.sh”和“1.3 Makefile”两步来制作有效升级包。

2.2 jffs2

在当前文件系统中任意选择一个位置，制作格式化镜像, 过程如下

~/work/x1000/recovery_testseed$ mkdir fs_format

~/work/x1000/recovery_testseed$ ls -l fs_format/ #注意:这个文件夹内至少应该有一个文件，文件内容可以为空

-rw-rw-r-- 1 torence torence 0 Nov 8 12:19 format

~/work/x1000/recovery_testseed$ mkfs.jffs2 -e 0x8000 -p 0xc80000 -d fs_format -o formator.jffs2

~/work/x1000/recovery_testseed$ ls -l formator.jffs2

-rw-r--r-- 1 torence torence 32768 Nov 8 12:22 formator.jffs2

生成 formator.jffs2 后， 重新执行“1.2 customize.sh”和“1.3 Makefile”两步来制作有效升级包

三、QA

Q：如何更换的数字证书？
A：参考 1.1 章节 make_key 和 dumpkey.jar 的用法

Q：如何更换 logo 和升级进度条以及字体？
A：参考 1.1 章节关于资源的描述

Q：如何编译 recovery 镜像？
A：参考 1.1 章节关于编译的描述

Q：如何配置 OTA 服务器地址？
A：参考 1.1 章节关于 recovery.conf 的描述

Q：如何使用 SD 卡或者 U 盘进行本地升级？
A：将升级包放在 SD 卡或 U 盘中 recovery-update 目录下，插入 SD 卡或 U 盘，进入 recovery 模式，执行 recovery 应用

问题反馈：

ZhangYanMing<yanming.zhang@ingenic.com>