Android 系统编译及客制化指南

Table of Contents

1. 代码管理及版本控制

一、首次使用 Git

在首次使用 Git 获取所需代码前需要进行以下操作以设置 Git 权限和用户信息

1、获取用于 Git 连接认证 SSH 密钥文件，此文件由负责 Git 远程仓库管理同事创建并提供

2、使用终端工具（建议使用 Xshell 或 putty）连接到编译服务器

3、拷贝 SSH 密钥文件到~/.ssh/git_key.pub，如无此目录则自行创建

4、修改 SSH 密钥文件权限

```
$ chmod 600 ~/.ssh/git_key.pub
```

5、为 Git SSH 连接创建配置文件~/.ssh/config，参考配置文件内容如下

```
host git
        user emdoor
        hostname 192.168.0.214
        identityfile ~/.ssh/git_key.pub
```

6、配置 Git 用户信息

```
$ git config --global user.name “你的姓名”
$ git config --global user.email “你的电子邮件地址”
```

二、使用 Git clone 命令获取代码

1、跟同事获得所需代码的 git 仓库路径，例如 MT8127 Android 5.0 当前 Git 仓库路径为 git:mt8127_lo_0205

2、克隆代码

```
$ git clone git:mt8127_lo_0205
```

三、更新代码

在修改代码或编译固件前建议将本地代码保持与远程仓库同步，具体操作为：

```
$ git pull origin master
```

上述命令表示将 master 分支与 origin 远程仓库同步，如果当前使用其他分支或远程仓库请自行修改命令参数，如果当前只有一个分支及远程仓库，可将上述命令简化为：

```
$ git pull
```

四、提交修改

我们可能修改、添加、删除代码文件或二进制文件，具体操作步骤有：

1、查看修改内容

```
$ git status
```

git status 命令可以查看当前目录修改的内容，此命令还可以查看指定目录的修改内容，如

```
$ git status packages/apps/Settings
```

用于查看 packages/apps/Settings 下所有修改的内容

2、添加修改到缓冲区

```
$ git add -u
```

git add packages/apps/Settings -u

3、提交修改

git commit -m "对所修改的问题的说明"

4、同步到远程仓库

git push origin master

此命令参数与 git pull 命令类似，如果当前仓库只包含一个分支和远程仓库，则可以简化为

git push

1.1 使用 Git 进行代码管理

一、首次使用 Git

在首次使用 Git 获取所需代码前需要进行以下操作以设置 Git 权限和用户信息

1、获取用于 Git 连接认证 SSH 密钥文件，此文件由负责 Git 远程仓库管理同事创建并提供

2、使用终端工具（建议使用 Xshell 或 putty）连接到编译服务器

3、拷贝 SSH 密钥文件到~/.ssh/git_key.pub，如无此目录则自行创建

4、修改 SSH 密钥文件权限

```
$ chmod 600 ~/.ssh/git_key.pub
```

5、为 Git SSH 连接创建配置文件~/.ssh/config，参考配置文件内容如下

```
host git
        user emdoor
        hostname 192.168.0.214
        identityfile ~/.ssh/git_key.pub
```

6、配置 Git 用户信息

```
$ git config --global user.name “你的姓名”
$ git config --global user.email “你的电子邮件地址”
```

二、使用 Git clone 命令获取代码

1、跟同事获得所需代码的 git 仓库路径，例如 MT8127 Android 5.0 当前 Git 仓库路径为 git:mt8127_lo_0205

2、克隆代码

```
$ git clone git:mt8127_lo_0205
```

三、更新代码

在修改代码或编译固件前建议将本地代码保持与远程仓库同步，具体操作为：

```
$ git pull origin master
```

上述命令表示将 master 分支与 origin 远程仓库同步，如果当前使用其他分支或远程仓库请自行修改命令参数，如果当前只有一个分支及远程仓库，可将上述命令简化为：

```
$ git pull
```

四、提交修改

我们可能修改、添加、删除代码文件或二进制文件，具体操作步骤有：

1、查看修改内容

```
$ git status
```

git status 命令可以查看当前目录修改的内容，此命令还可以查看指定目录的修改内容，如

```
$ git status packages/apps/Settings
```

用于查看 packages/apps/Settings 下所有修改的内容

2、添加修改到缓冲区

```
$ git add -u
```

git add packages/apps/Settings -u

3、提交修改

git commit -m "对所修改的问题的说明"

4、同步到远程仓库

git push origin master

此命令参数与 git pull 命令类似，如果当前仓库只包含一个分支和远程仓库，则可以简化为

git push

1.2 Git 常用命令参考

2. 编译环境配置及使用技巧

根据 Google Android 官方资料，Android Lollipop 建议编译环境为 Ubuntu 14.04 LTS 64bit，另外 12.04 LTS 64bit 也仍在支持范围。

一、Android Lollipop 编译环境配置（需要获取 sudo 权限）

1、安装 JDK

Android Lollipop 要求使用 Open JDK 7 为 Java 环境，安装方法:

```
$ sudo apt-get update
$ sudo apt-get install openjdk-7-jdk
```

2、选择 Java 版本

如果你的 Ubuntu 安装了其他版本的 Java 环境，需要选择 Open JDK 7 (1.7.x)为默认 Java 环境，使用命令

```
$ sudo update-alternatives --config java
$ sudo update-alternatives --config javac
```

选择 Java 版本后使用如下命令检查 Java 版本

```
$ java -version
java version "1.7.0_75"
OpenJDK Runtime Environment (IcedTea 2.5.4) (7u75-2.5.4-1~utopic1)
OpenJDK 64-Bit Server VM (build 24.75-b04, mixed mode)
$ javac -version
javac 1.7.0_75
```

3、安装必备组件

Ubuntu 14.04 LTS 64bit

```
$ sudo apt-get install bison g++-multilib git gperf libxml2-utils make zlib1g-dev:i386 zip
```

Ubuntu 12.04 LTS 64bit

```
$ sudo apt-get install git gnupg flex bison gperf build-essential zip curl libc6-dev libncurses5-dev:i386 x11proto-core-dev libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-dri:i386 libgl1-mesa-dev g++-multilib mingw32 tofrodos python-markdown libxml2-utils xsltproc zlib1g-dev:i386
$ sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1 /usr/lib/i386-linux-gnu/libGL.so
```

二、Android Kitkat 编译环境配置

安装 Oracle JDK,编译 Android 4.4 不能用 OpenJDK，必须用 Oracle JDK1.6，JDK 1.6 安装及配置请参考下一节《JDK 安装及环境变量配置方法》。

2.1 Android 系统编译环境配置方法

根据 Google Android 官方资料，Android Lollipop 建议编译环境为 Ubuntu 14.04 LTS 64bit，另外 12.04 LTS 64bit 也仍在支持范围。

一、Android Lollipop 编译环境配置（需要获取 sudo 权限）

1、安装 JDK

Android Lollipop 要求使用 Open JDK 7 为 Java 环境，安装方法:

```
$ sudo apt-get update
$ sudo apt-get install openjdk-7-jdk
```

2、选择 Java 版本

如果你的 Ubuntu 安装了其他版本的 Java 环境，需要选择 Open JDK 7 (1.7.x)为默认 Java 环境，使用命令

```
$ sudo update-alternatives --config java
$ sudo update-alternatives --config javac
```

选择 Java 版本后使用如下命令检查 Java 版本

```
$ java -version
java version "1.7.0_75"
OpenJDK Runtime Environment (IcedTea 2.5.4) (7u75-2.5.4-1~utopic1)
OpenJDK 64-Bit Server VM (build 24.75-b04, mixed mode)
$ javac -version
javac 1.7.0_75
```

3、安装必备组件

Ubuntu 14.04 LTS 64bit

```
$ sudo apt-get install bison g++-multilib git gperf libxml2-utils make zlib1g-dev:i386 zip
```

Ubuntu 12.04 LTS 64bit

```
$ sudo apt-get install git gnupg flex bison gperf build-essential zip curl libc6-dev libncurses5-dev:i386 x11proto-core-dev libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-dri:i386 libgl1-mesa-dev g++-multilib mingw32 tofrodos python-markdown libxml2-utils xsltproc zlib1g-dev:i386
$ sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1 /usr/lib/i386-linux-gnu/libGL.so
```

二、Android Kitkat 编译环境配置

安装 Oracle JDK,编译 Android 4.4 不能用 OpenJDK，必须用 Oracle JDK1.6，JDK 1.6 安装及配置请参考下一节《JDK 安装及环境变量配置方法》。

2.2 JDK 安装及版本切换方法

Android 不同版本推荐的 JDK 可能不一样，这样导致了在多个不同 Android 版本下工作会需要不同版本的 jdk

Android L 以及最新的 Google 推荐的是 openjdk-7-jdk，并且限制只能使用 jdk7 以上的版本，但是在 4.4 以及以下版本中，jdk7 编译失败，很多项目编译失败，除了修复编译失败的问题之外我们还可以再安装一个 jdk6。

首先我们下载 sun-jdk-6 出来，安装略,我把文件释放到/opt/jdk 目录了

配置如下：

```
$ sudo vim /etc/profile
```

在文件的末尾加上如下内容，保存并关闭文件

```
# for java
export JAVA_HOME=/opt/jdk
export JRE_HOME=${JAVA_HOME}/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
export PATH=$PATH:${JAVA_HOME}/bin:${JRE_HOME}/bin
$ source /etc/profile
```

*如果你没有多版本的 jdk，最后一行可以这样写，那么会直接使用/opt/jdk，不需要切换

```
PATH=${JAVA_HOME}/bin:${JRE_HOME}/bin:$PATH
```

*如果你有多个 jdk 的话，可以一行可以这样写，那么会使用/usr/bin/java - > /etc/alternatives/java

```
export PATH=$PATH:${JAVA_HOME}/bin:${JRE_HOME}/bin
```

然后执行下面的

```
$ sudo update-alternatives --install /usr/bin/javah javah /opt/jdk/bin/javah 255
$ sudo update-alternatives --install /usr/bin/javac javac /opt/jdk/bin/javac 255
$ sudo update-alternatives --install /usr/bin/java java /opt/jdk/bin/java 255
$ sudo update-alternatives --install /usr/bin/jar jar /opt/jdk/bin/jar 255
```

安装完了 sun jdk 之后，我们来安装 openjdk （安装环境 Ubuntu 14.04）

```
$ sudo apt-get install openjdk-7-jdk
```

安装完毕之后,执行以下命令来切换你的 jdk

```
$ sudo update-alternatives --config java
$ sudo update-alternatives --config javac
$ sudo update-alternatives --config javah
$ sudo update-alternatives --config jar
```

2.3 亿道研发服务器使用指南

一、连接到 Windows 远程桌面
二、登录 Ubnutu 编译服务器
三、映射 Ubnutu 文件到 Windows
四、编辑修改代码
五、文件拷贝
六、发布编译结果
七、获取常用资料及工具

3. 固件编译

一、Android 标准编译系统初始化方法

1、执行编译初始化脚本

在 MTK 平台，Android Lollipop 版本中使用了标准 Android 编译流程，在通过终端连接到 Ubnutu 系统需要先进行编译初始化设置，具体操作为

1、进入 Android 系统代码根目录(具体目录名称请根据实际替换)

```
$ cd source_directory
```

2、执行初始化脚本

```
$ source build/envsetup.sh
或
$ . build/envsetup.sh
```

执行后如果终端打印类似以下内容并且无 error 提示为成功

```
including device/generic/mini-emulator-arm64/vendorsetup.sh
including device/generic/mini-emulator-x86_64/vendorsetup.sh
including device/generic/mini-emulator-x86/vendorsetup.sh
including device/generic/mini-emulator-mips/vendorsetup.sh
including device/generic/mini-emulator-armv7-a-neon/vendorsetup.sh
including device/pskyed/em_t8116_nand_lvds/vendorsetup.sh
including device/pskyed/em_t8512_nand/vendorsetup.sh
including device/pskyed/RCT6773W22B/vendorsetup.sh
including device/pskyed/em_t8611_nand_lvds/vendorsetup.sh
including device/pskyed/RCT6203W46_MLC/vendorsetup.sh
including device/pskyed/pskyed8127_tb_c_l/vendorsetup.sh
including device/pskyed/RCT6303W87DK/vendorsetup.sh
including device/pskyed/em_t8370_emmc/vendorsetup.sh
including device/pskyed/NS_P16AT10/vendorsetup.sh
including device/pskyed/em_t8511_emmc/vendorsetup.sh
including device/pskyed/em_t8992_nand/vendorsetup.sh
including device/pskyed/RCT6203W46L/vendorsetup.sh
including device/pskyed/em_t8778_emmc/vendorsetup.sh
including device/pskyed/EM8611_EMMC/vendorsetup.sh
including device/pskyed/em_t8370_nand/vendorsetup.sh
including device/pskyed/W109G/vendorsetup.sh
including device/pskyed/em_t8511a_nand/vendorsetup.sh
.....
```

3、选择编译目标

上述初始化脚本执行成功后，使用 lunch 命令可以列出可以选择编译的版本类型，然后选择需要编译的版本的对应编号后按回车即可。

```
$ lunch
You're building on Linux
Lunch menu... pick a combo:
1. aosp_arm-eng
2. aosp_arm64-eng
```

```
3. aosp_mips-eng
4. aosp_mips64-eng
5. aosp_x86-eng
6. aosp_x86_64-eng
7. mini_emulator_arm64-userdebug
8. mini_emulator_x86_64-userdebug
9. mini_emulator_x86-userdebug
10. mini_emulator_mips-userdebug
11. mini_emulator_arm-userdebug
12. full_em_t8116_nand_lvds-eng
13. full_em_t8116_nand_lvds-userdebug
14. full_em_t8116_nand_lvds-user
15. em_t8512_nand-eng
16. em_t8512_nand-userdebug
17. em_t8512_nand-user
18. RCT6773W22B-eng
19. RCT6773W22B-userdebug
20. RCT6773W22B-user
.....

Which would you like? [aosp_arm-eng] 18
```

终端输出以下信息表示 lunch 命令执行成功

```
============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=5.0
TARGET_PRODUCT=RCT6773W22B
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a-neon
TARGET_CPU_VARIANT=cortex-a7
TARGET_2ND_ARCH=
TARGET_2ND_ARCH_VARIANT=
TARGET_2ND_CPU_VARIANT=
HOST_ARCH=x86_64
HOST_OS=linux
HOST_OS_EXTRA=Linux-3.16.0-23-generic-x86_64-with-Ubuntu-14.10-utopic
HOST_BUILD_TYPE=release
BUILD_ID=LRX21M
OUT_DIR=out
============================================
```

二、MTK Android 4.4（Kitkat）版本编译系统区别

在 Android 4.4 中，MTK 对 Android 编译系统进行了大量的修改，修改后的编译操作与 Android 原生设计有较大区别，其中与本节内容相关的区别就是省去了上述编译初始化和 lunch 的过程，针对 MTK Android 4.4 编译系统的详细使用方法后面内容会有介绍。

3.1 编译初始化设置

一、Android 标准编译系统初始化方法

1、执行编译初始化脚本

在 MTK 平台，Android Lollipop 版本中使用了标准 Android 编译流程，在通过终端连接到 Ubnutu 系统需要先进行编译初始化设置，具体操作为

1、进入 Android 系统代码根目录(具体目录名称请根据实际替换)

```
$ cd source_directory
```

2、执行初始化脚本

```
$ source build/envsetup.sh
```

或

```
$ . build/envsetup.sh
```

执行后如果终端打印类似以下内容并且无 error 提示为成功

```
including device/generic/mini-emulator-arm64/vendorsetup.sh
including device/generic/mini-emulator-x86_64/vendorsetup.sh
including device/generic/mini-emulator-x86/vendorsetup.sh
including device/generic/mini-emulator-mips/vendorsetup.sh
including device/generic/mini-emulator-armv7-a-neon/vendorsetup.sh
including device/pskyed/em_t8116_nand_lvds/vendorsetup.sh
including device/pskyed/em_t8512_nand/vendorsetup.sh
including device/pskyed/RCT6773W22B/vendorsetup.sh
including device/pskyed/em_t8611_nand_lvds/vendorsetup.sh
including device/pskyed/RCT6203W46_MLC/vendorsetup.sh
including device/pskyed/pskyed8127_tb_c_l/vendorsetup.sh
including device/pskyed/RCT6303W87DK/vendorsetup.sh
including device/pskyed/em_t8370_emmc/vendorsetup.sh
including device/pskyed/NS_P16AT10/vendorsetup.sh
including device/pskyed/em_t8511_emmc/vendorsetup.sh
including device/pskyed/em_t8992_nand/vendorsetup.sh
including device/pskyed/RCT6203W46L/vendorsetup.sh
including device/pskyed/em_t8778_emmc/vendorsetup.sh
including device/pskyed/EM8611_EMMC/vendorsetup.sh
including device/pskyed/em_t8370_nand/vendorsetup.sh
including device/pskyed/W109G/vendorsetup.sh
including device/pskyed/em_t8511a_nand/vendorsetup.sh
.....
```

3、选择编译目标

上述初始化脚本执行成功后，使用 lunch 命令可以列出可以选择编译的版本类型，然后选择需要编译的版本的对应编号后按回车即可。

```
$ lunch
You're building on Linux
Lunch menu... pick a combo:
1. aosp_arm-eng
2. aosp_arm64-eng
```

```
3. aosp_mips-eng
4. aosp_mips64-eng
5. aosp_x86-eng
6. aosp_x86_64-eng
7. mini_emulator_arm64-userdebug
8. mini_emulator_x86_64-userdebug
9. mini_emulator_x86-userdebug
10. mini_emulator_mips-userdebug
11. mini_emulator_arm-userdebug
12. full_em_t8116_nand_lvds-eng
13. full_em_t8116_nand_lvds-userdebug
14. full_em_t8116_nand_lvds-user
15. em_t8512_nand-eng
16. em_t8512_nand-userdebug
17. em_t8512_nand-user
18. RCT6773W22B-eng
19. RCT6773W22B-userdebug
20. RCT6773W22B-user
.....

Which would you like? [aosp_arm-eng] 18
```

终端输出以下信息表示 lunch 命令执行成功

```
============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=5.0
TARGET_PRODUCT=RCT6773W22B
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a-neon
TARGET_CPU_VARIANT=cortex-a7
TARGET_2ND_ARCH=
TARGET_2ND_ARCH_VARIANT=
TARGET_2ND_CPU_VARIANT=
HOST_ARCH=x86_64
HOST_OS=linux
HOST_OS_EXTRA=Linux-3.16.0-23-generic-x86_64-with-Ubuntu-14.10-utopic
HOST_BUILD_TYPE=release
BUILD_ID=LRX21M
OUT_DIR=out
============================================
```

二、MTK Android 4.4（Kitkat）版本编译系统区别

在 Android 4.4 中，MTK 对 Android 编译系统进行了大量的修改，修改后的编译操作与 Android 原生设计有较大区别，其中与本节内容相关的区别就是省去了上述编译初始化和 lunch 的过程，针对 MTK Android 4.4 编译系统的详细使用方法后面内容会有介绍。

3.2 编译升级镜像

一、MTK Android 5.0/5.1 Lollipop 版本镜像编译方法

在 MTK 平台最新的 Android 系统源码环境中，编译方式与 Android 原生设计并无太大差异

Build the Code

Build everything with make. GNU make can handle parallel tasks with a -jN argument, and it's common to use a number of tasks N that's between 1 and 2 times the number of hardware threads on the computer being used for the build. For example, on a dual-E5520 machine (2 CPUs, 4 cores per CPU, 2 threads per core), the fastest builds are made with commands between make -j16 and make -j32.

常用编译命令参考：

```
$ make -j16 (完整编译系统并生成全部镜像)
```

```
$ make pl -j16 (编译 preloader 并生成 preloader_xxx.bin)
```

```
$ make lk   -j16 (编译 lk 并生成 lk.bin 及 logo.bin)
$ make bootimage -j16 (编译 kernel 并生成 boot.img)
$ make recoveryimage -j16 (编译 kernel 并生成 recovery.img)
$ make systemimage -j16 (编译 Android 系统并生成 system..img)
$ make otapackage -j16 (生成 target file 及 OTA 升级包)
$ make clean (清除所有的编译结果)
$ make update-api (添加、删除或修改系统 API 后重新生成 current.xml)
```

二、MTK Android 4.4 Kitkat 版本镜像编译方法

全新编译 RCT6773W22 机型全部镜像 user 版本

$./mk-o=TARGET_BUILD_VARIANT=user RCT6773W22 n

继续编译 RCT6773W22 机型全部镜像，此命令在上次编译中断或已经编译完成后继续修改代码再次编译时可以更快完成编译

$./mk-o=TARGET_BUILD_VARIANT=user RCT6773W22 r

编译 eng 版本所有镜像

$ ./mk RCT6773W22 n

编译时添加-t 参数可以控制编译系统输出打印信息

$ ./mk -t RCT6773W22 n

编译 preloader

$ ./mk -t RCT6773W22 n pl

编译 lk

```
$ ./mk   -t RCT6773W22 n lk
编译 bootimage
编译 system.img
打包 system.img
编译 OTA 升级包及 target file
```

更多编译参数请查看 mk 命令的帮助内容：

```
ptions:
   -t, -tee           : Print log information on the standard-out.
   -o, -opt=bypass_argument_to_make
                              : Pass extra arguments to make.
   -h, -help         : Print this message and exit.
Projects:
   one of available projects.
Actions:
   listp, listproject
                              : List all available projects.
   check-env         : Check if build environment is ready.
   check-dep         : Check feature dependency.
   n, new               : Clean and perform a full build.
   c, clean           : Clean the immediate files(such as, objects, libraries
etc.).
   r, remake          : Rebuild(target will be updated if any dependency
updats).
   mrproper           : Remove all generated files + config + various backup
files in Kbuild process.
   bm_new               : "new" + GNU make's "-k"(keep going when encounter
error) feature.
   bm_remake         : "remake" + GNU make's "-k"(keep going when encounter
error) feature.
   mm                       : Build module through Android native command
"mm"
   mma                      : Build module through Android native command
"mma"
   emigen               : Generate EMI setting source code.
   nandgen             : Generate supported NAND flash device list.
   codegen             : Generate trace DB(for META/Cather etc. tools used).
   drvgen               : Generate driver customization source.
   custgen             : Generate customization source.
   javaoptgen       : Generate the global java options.
   ptgen                  : Generate partition setting header & scatter file.
   bindergen         : Generate binder related information
   sign-image       : Sign all the image generated.
   encrypt-image : Encrypt all the image generated.
   update-api       : Android default build action
                                    (be executed if system setting or
anything removed from API).
```

```
   check-modem     : Check modem image consistency.
   upadte-modem   : Update modem image located in system.img.
   modem-info       : Show modem version
   gen-relkey       : Generate releasekey for application signing.
   check-appres   : Check unused application resource.
   sdk                    : Build sdk package.
   win_sdk             : Build sdk package with a few Windows tools.
   banyan_addon   : Build MTK sdk addon.
   banyan_addon_x86 :Build MTK sdk x86 addon.
   cts                    : Build cts package.
   bootimage         : Build boot image(boot.img).
   cacheimage       : Build cache image(cache.img).
   systemimage     : Build system image(system.img).
   snod                  : Build system image without dependency.
                              (that is, ONLY pack the system image, NOT
checking its dependencies.)
   recoveryimage : Build recovery image(recovery.img).
   secroimage       : Build secro image(secro.img).
   factoryimage   : Build factory image(factory.img).
   userdataimage : Build userdata image(userdata.img).
   userdataimage-nodeps
                            : Build userdata image without dependency.
                              (that is, ONLY pack the userdata image,
NOT checking its dependencies.)
              dump-products       :       Dump       products       related
configuration(PRODUCT_PACKAGE,PRODUCT_NAME ect.)
   target-files-package
                            : Build the target files package.
                              (A zip of the directories that map to the
target filesystem.
                                This zip can be used to create an OTA
package or filesystem image
                                as a post-build step.)
   updatepackage : Build the update package.
   dist                  : Build distribution package.
Modules:
   pl, preloader : Specify to build preloader.
   lk                     : Specify to build little kernel.
   md32                  : Specify to build DSP md32.
   tz, trustzone : Specify to build trusted execution environment.
   k,   kernel       : Specify to build kernel.
   dr, android     : Specify to build android.
   NULL                   : Specify to build all components/modules in
default.
   k <module path>
                             : Specify to build kernel component/module
with the source path.
   dr <module name>
                            : Specify to build android component/module
with module name.
Other tools:
   prebuilts/misc/linux-x86/ccache/ccache -M 10G
                            : Set CCACHE pool size to 10GB
Example:
   ./mk -t e1k emigen
```

: Generate EMIsettingsourcecode.
 ./mk-o=TARGET_BUILD_VARIANT=user e1k n
: Start a user mode full build.
./mk listp : List all available projects.
./mk e1k bootimage
: Build bootimage for e1k project.
./mk e1k bm_new k
: Build kernel for e1k project.
./makeMtk e1k c,bm_remake pl k
: Clean & Build preloader and kernel for e1k project.
./makeMtk e1k n k kernel/xxx/xxx
: Build(full build) kernel component/module under the path "kernel/xxx/xxx" for e1k project.
./makeMtk e1k r dr Gallery
: Rebuild android module named Gallery for e1k project.
./makeMtk e1k mm packages/apps/Settings
: Change Directory to packages/apps/Settings and execute "mm"
./makeMtk e1k mma packages/apps/Settings
: Change Directory to packages/apps/Settings and execute "mma"

3.3 编译 OTA 升级包

3.4 OTA 功能测试

3.5 固件发布

3.6 调试过程中的编译方法

3.7 关于 eng、userdebug、user 版本的区别

一个 device 项目可能包括 eng、userdebug、user 三种版本类型，例如名称为 RCT6773W22B 的机型包括

18. RCT6773W22B-eng
19. RCT6773W22B-userdebug
20. RCT6773W22B-user

Google 官方描述：USER/USERDEBUG/ENG 版本的差异，参考 alps/build/core/build-system.html 的详细说明
eng This is the default flavor. A plain make is the same as make eng.
* Installs modules tagged with: eng, debug, user, and/or development.
* Installs non-APK modules that have no tags specified.
* Installs APKs according to the product definition files, in addition to tagged APKs.
* ro.secure=0
* ro.debuggable=1
* ro.kernel.android.checkjni=1
* adb is enabled by default.
* Setupwizard is optional
user make user
This is the flavor intended to be the final release bits.
* Installs modules tagged with user.
* Installs non-APK modules that have no tags specified.
* Installs APKs according to the product definition files; tags are ignored for APK modules.
* ro.secure=1
* ro.debuggable=0
* adb is disabled by default.
* Enable dex pre-optimization for all TARGET projects in default to speed up device first boot-up
userdebug make userdebug
The same as user, except:
* Also installs modules tagged with debug.
* ro.debuggable=1
* adb is enabled by default.

MTK 补充说明差异:
(1) Debug/LOG 方面，原则上 user 版本只能抓到有限的资讯，eng 可以抓到更多的资讯，Debug 能力更强，推崇使用 eng 版本开发测试
* 因 user/eng 版本设置 ro.secure 不同，导致 user 版本 adb 只拥有 shell 权限，而 eng 版本具有 root 权限
* MTK System LOG 在 ICS 以后，在 user 版本默认关闭全部 LOG， 在 eng 版本中默认打开，以便抓到完整的资讯
* 在 eng 版本上，LOG 量 >= user 版本的 log 量，一些地方会直接 check eng/user 版本来确认是否打印 LOG
* user 版本默认关闭 uart, eng 版本默认开启 uart
* 在 eng 版本上，开启 ANR 的 predump, 会抓取 ftrace，可以得到更多 ANR 的资讯
* 在 eng 版本上，可用 rtt 抓取 backtrace，可开启 kdb 进行 kernel debug, 可用 ftrace 抓取 cpu 执行场景
* MTK aee 在 ENG 版本抓取更多的异常资讯,比如 native exception 会抓取 core dump 信息

(2) 性能方面，原则上进行性能测试请使用 user 版本测试
* user 版本为提高第一次开机速度，使用了 DVM 的预优化，将 dex 文件分解成可直接 load 运行的 odex 文件，ENG 版本不会开启这项优化
* 更少的 LOG 打印，uart 的关闭，原则上 user 版本的性能要优于 eng 版本

(3) 如何确认 user/eng 版本
* Java 层，check android.os.Build 类中的 TYPE 值
* native 层，property_get("ro.build.type", char* value, "eng");然后 check value 值
* Debug 时， adb shell getprop ro.build.type 返回值如果是 user 即 user 版本，eng 即 eng 版本
* Log 确认， mobile log/Aplog_xxxxx/versions 中查看 ro.build.type 属性

4. 客制化修改

Write here...

4.1 创建项目

4.2 系统及界面客制化

4.2.1 修改开机 Logo

4.2.2 修改开机动画

4.2.3 修改机型名称及品牌信息

4.2.4 修改系统默认设置

4.2.5 添加 GMS 服务包

4.2.6 主界面客制化

4.2.7 添加壁纸

4.2.8 添加第三方应用

4.2.9 添加删除系统功能

4.2.10 添加 OTA 功能

4.2.11 修改分区容量

4.3 内核驱动客制化

4.3.1 DDR 内存型号

4.3.2 电池曲线

4.3.3 摄像头

4.3.4 触摸屏

4.3.5 重力传感器

5. 参考资料

Write here...

5.1 Android 系统源码结构介绍

5.2 MTK 官方 FAQ 列表