SDK 脚本编译手册

Draft v1.0

内容目录

1. SDK 说明

本文档是 NUSMART 系列芯片的新版 SDK 开发和使用手册。SDK 是针对客户的需求而制作的一体化编译工具，使用 SDK 开发，可以用简单的一行命令，就 可以选择合适的 xloader, 编译出 uboot, kernel, android 文件系统， 并且能够打包生成用户可以直接烧录的 ROM。

2. SDK 使用的前提

使用本文档的时候，请确保必要的软件已经安装完毕,如 Python, jdk, make, git 等。如何还没有安装上述的软件，请参考:《NS115 编译环境搭建》, 完成 SDK 基本的环境搭建和 SDK 的代码获取。

3. 快速开始 SDK 编译

- 编译

如果用户不想阅读文档，快速开始编译 ROM。可以进入源代码根目录，然后进入 bsp/tools 目录, 执行全体编译:

```
./build.sh -B
```

会弹出类似下面的菜单:

```
INFO: build all
You're building on Linux
Lunch menu... pick a combo:
    1. full_stingray-userdebug
    2. full_wingray-userdebug
    3. nusmart2_smp-user
    4. full_nusmart3_nubox-user
    5. full_nusmart3_pad-user
    6. full_nusmart3_pad_nw51-user
```

```
7. full_nusmart3_phone-user
8. full_nusmart3_prototype-user
9. full_nusmart3_stick-user
10. full_crespo-userdebug
11. full_crespo4g-userdebug
12. full_maguro-userdebug
13. full_toro-userdebug
14. full_tuna-userdebug
15. full_panda-user
Which would you like? [full_nusmart3_pad-user]
```

选择需要编译的目标类型，如:full_nusmart3_pad-user，可以输入 full_nusmart3_pad-user 或者 5，然后回车，就可以开始编译。编译顺利的话,您应该看到 out/{target} 里面生成 out.ROM，out.ROM 仅可使用 NUSMART 提供的烧写工具进行烧录。

如果想单独编译 xloader，uboot，kernel，android 或者 pack 打包操作。 可以输入子工程的首字母作为脚本参数，然后选择编译的目标类型即可。

如：编译 xloader：./build.sh -x
编译 uboot：./build.sh -u
编译 kernel：./build.sh -k
编译 android：./build.sh -a
执行打包操作：./build.sh -p

注意：单独执行打包时，需确保 xloader，uboot，kernel，android 都已经正确编译完毕，否则会因为缺少文件而打包失败。

编译完毕后，可在源代码的根目录下的 out 目录下，找到对应编译目标的文件夹，在文件夹下面就可以找到最终的二进制文件。

上面的命令可以组合使用， 例如想同时编译 xloader 和 uboot，可以输入：

```
./build.sh -xu
```

- 清理

  清理所有工程，可以简单执行： ./build.sh -Z

清理单个工程，可在构建脚本的参数前加上-c 参数，即可执行对应的清理操作。

如：单独清理 xloader ：./build.sh –cx

单独清理 uboot: ./build.sh -cu

单独清理 kernel: ./build.sh -ck

单独清理 android: ./build.sh -ca

单独清理打包项目: ./build.sh -cp

同样清理也可以组合使用，如：同时清理 xloader 和 uboot 可以使用如下命令。

```
./build.sh -cxu
```

4. SDK 工程的目录结构

代码下载完后，可以看到如下的代码目录

工程根目录:

```
├─── doc: 所有 SDK 和工具的文档
├─── user-config: 需要用户提供和修改的文件（非编译产生），用于编译打包
├─── user-tools: 提供给用户的工具
├─── android: android 源代码目录
├─── bsp
|     ├─── xloader: xloader 源代码
|     ├─── uboot: uboot 源代码
|     ├─── linux_kernel: 内核源代码
|     └─── tools: 大编译脚本 build.sh
|           └─── pack :打包工具
└─── out:所有工程的输出目录
```

注意：必须严格照此目录结构存放，否则会造成编译失败。

目录结构中的 out 目录是编译自动生成的目录，用于存放所有的编译输出。

5. SDK 工程的编译

首先进入源代码根目录的 bsp/tools 目录。

```
cd bsp/tools
```

在 tools 目录下有一个 build.sh 文件，这个 build.sh 文件就是 SDK 编译的入口文件。

5.1. build.sh 的参数

在 tools 目录下，执行./build.sh

输出如下:

```
$ ./build.sh
NAME
    build.sh - The release tools for xload, uboot, kernel and android.
SYNOPSIS
    ./build.sh [-h] [-x build xload] [-u build uboot] [-k build kernel] [-a build android]
OPTIONS
    -h                      Display help message
    -x                      Build xload, build use xload directory.
    -u                      Build uboot, build use uboot directory.
    -k                      Build kernel, build use kernel directory.
    -o                      Copy kernel to android.
    -a                      Build android, build use android directory.
    -t <target>             Build prodcut target.
    -p                      Package to ROM file.
    -c                      Clean project, clean android, e.g: ./build.sh -c -a.
    -B                      Build all.
    -Z                      Clean all.
    -X <xload version>      Build xload with version, e.g. 1.0: use xload1.0,...
    -U <uboot version>      Build uboot with version, e.g. 1.0: use uboot1.0,...
    -K <kernel version>     Build kernel with version, e.g. 2.6: use kernel2.6,...
    -A <android version>    Build android with version, e.g. 4.0.3,4.1, ...
    -C <android version>    Build lunch menu with android version, e.g. 4.0.3,4.1, ...
```

对照上面的输出简单解释下./build.sh 参数的意义。

-h: 执行./build.sh -h 会显示帮助信息，和不输入参数输出内容一样。

-x: 执行./build.sh -x 编译 xloader

-u: 执行./build.sh -u 编译 uboot

-k: 执行./build.sh -k 编译 kernel

-o: 执行./build.sh -o 把 uImage 复制到 android 的代码空间

-a: 执行./build.sh -a 编译 android

-t <target>: 指定具体的编译目标平台,如 full_nusmart3_pad-user，如果不设置，则会提示进行选择

-p: 打包成 ROM 文件

-c: 清理单个工程，清理操作进行完后不会再进行构建

-B: 构建全部工程

-Z: 清理全部工程，清理操作进行完后不会再进行构建

其中 xukapt 这些参数都是可以组合使用的如: `./build.sh -xk`，即编译 `xloader` 和 `kernel`。如果一次完成全部工程的编译打包，需要使用命令`./build.sh -xukoap` 或者`./build.sh -B`。

同时指定多个参数，编译的顺序是固定的，总是先编译 `xloader`,然后是 `uboot`，其次是 `kernel`，再次是 `android`，最后是打包操作。

5.2. 使用 build.sh 构建 xloader

单独编译 `xloader`，执行`./build.sh -x`

```
You're building on Linux
Lunch menu... pick a combo:
    1. full_stingray-userdebug
    2. full_wingray-userdebug
    3. nusmart2_smp-user
    4. full_nusmart3_nubox-user
    5. full_nusmart3_pad_nw51-user
    6. full_nusmart3_pad-user
    7. full_nusmart3_phone-user
    8. full_nusmart3_prototype-user
    9. full_nusmart3_stick-user
    10. full_crespo4g-userdebug
    11. full_crespo-userdebug
    12. full_maguro-userdebug
```

```
13. full_toro-userdebug

14. full_tuna-userdebug

15. full_panda-user

Which would you like? [full_nusmart3_pad-user]
```

如果编译过 android 的源代码，会对上面的提示很熟悉。上面提示选择一个产品类型，这些产品类型都是从 android 的目录下获取的，和 lunch 的显示是一致的。 这个菜单的选择方式和 android 也是一样的，如果选择 full_nusmart3_pad-user 作为编译目标，可以直接输入 6，然后回车，也可以输入 full_nusmart3_pad-user,然后回车。 直接输入回车也是可以的，因为默认的编译目标就是 full_nusmart3_pad-user。如果编译最终的 eng 版，和 android 的编译相同，只能通过输入 full_nusmart3_pad-eng。

设置好目标板后，`xloader` 的构建脚本就会编译出 `xloader`，并复制到 out/{target}目录。

5.3. 使用 build.sh 构建 uboot

单独编译 `uboot`，执行`./build.sh -u`

如果没有指定-t 参数，还是会弹出一个菜单选择合适的编译目标。 选择后开始编译，如果没有编译错误，最后目标文件会生成到 out/{target}目录。

5.4. 使用 build.sh 构建 kernel

内核部分发布给客户的是部分源码，部分*.ko 文件不提供对应的源代码，编译方式执行: `./build.sh -k`

如果没有指定-t 参数，会弹出一个菜单选择合适的编译目标，最后目标文件 uImage 会生成到 out/{target}目录。

如果编译后直接想复制到 android 的代码空间可以执行`./build.sh -ko`，这样编译完后就直接更新 android 的内核了

5.5. 使用 build.sh 构建 android

`android` 编译和 `kernel` 类似。需要注意的是 `android` 的工程会依赖于 `uImage`，编译脚本执行的时候，如果内核有更新需要执行`./build.sh -o`， 把内核镜像复制到

android 的代码空间，然后编译 android 这样编译出的文件系统，就会使用新的内核镜像，如果不执行这个复制动作，默认使用的是旧内核。如果没有有效的 uImage，复制操作会失败。

编译方式执行：./build.sh -a

同样如果没有指定-t 参数，会弹出一个菜单选择合适的编译目标，最后目标文件 boot.img.ext4,system.img.ext4,recovery.img.ext4 会生成到 out/{target}目录。

5.6.使用 build.sh 打包成 ROM 文件

打包成 ROM 之前需要确保 out/{target}/目录下面，已经生成了所有的二进制文件，如:uImage、android 的三个镜像文件等。如果文件不全，单独执行打包操作，会出现文件找不到的错误。

注意：在打包的时候，有些步骤是需要 sudo 权限的， 如果第一次执行，必须有 sudo 权限才能执行打包操作，这个部分也是唯一一个需要 sudo 权限的地方，打包过程中会提示输入密码。

ROM 编译方式执行：./build.sh -p

如果没有错误出现，会在 out/{target}/目录下面生成 out.ROM 文件。

注意：out.ROM 请使用 NUSMART 提供的烧写工具进行烧录。

5.6.SDK 编译的 log 输出

编译过程中的控制台输出，都会定向到 out/${target}.log 文件，log 是以追加的方式写入，如果编译过程中有任何问题，可以分析 log 文件。

多次编译这个文件会越来越大，如果 log 文件太大，会影响编译速度，需手动清理。