

1. Neptune w800初学 Ubuntu下程序编译与烧写

1. 一、学习

1. 1.1一些基本概念

2. 1.2构建流程

3. 1.3目录说明

4. 1.4使用说明

2. 二、Neptune w800开发版实操

1. 2.1 准备工作

1. 2.1.1 示例工程下载

2. 2.1.2 编译

1. 1. 编译工具链配置

2. 2. WiFiOT环境配置

3. 3. 安装编译工具hb

4. 4.编译

2. 2.2 烧写镜像

1. 2.2.1 Ubuntu 烧写工具

2. 2.2.2 烧写过程

Neptune w800初学 Ubuntu下程序编译与烧写

一、学习

1.1一些基本概念

在开发编译构建前，应了解如下基本概念：

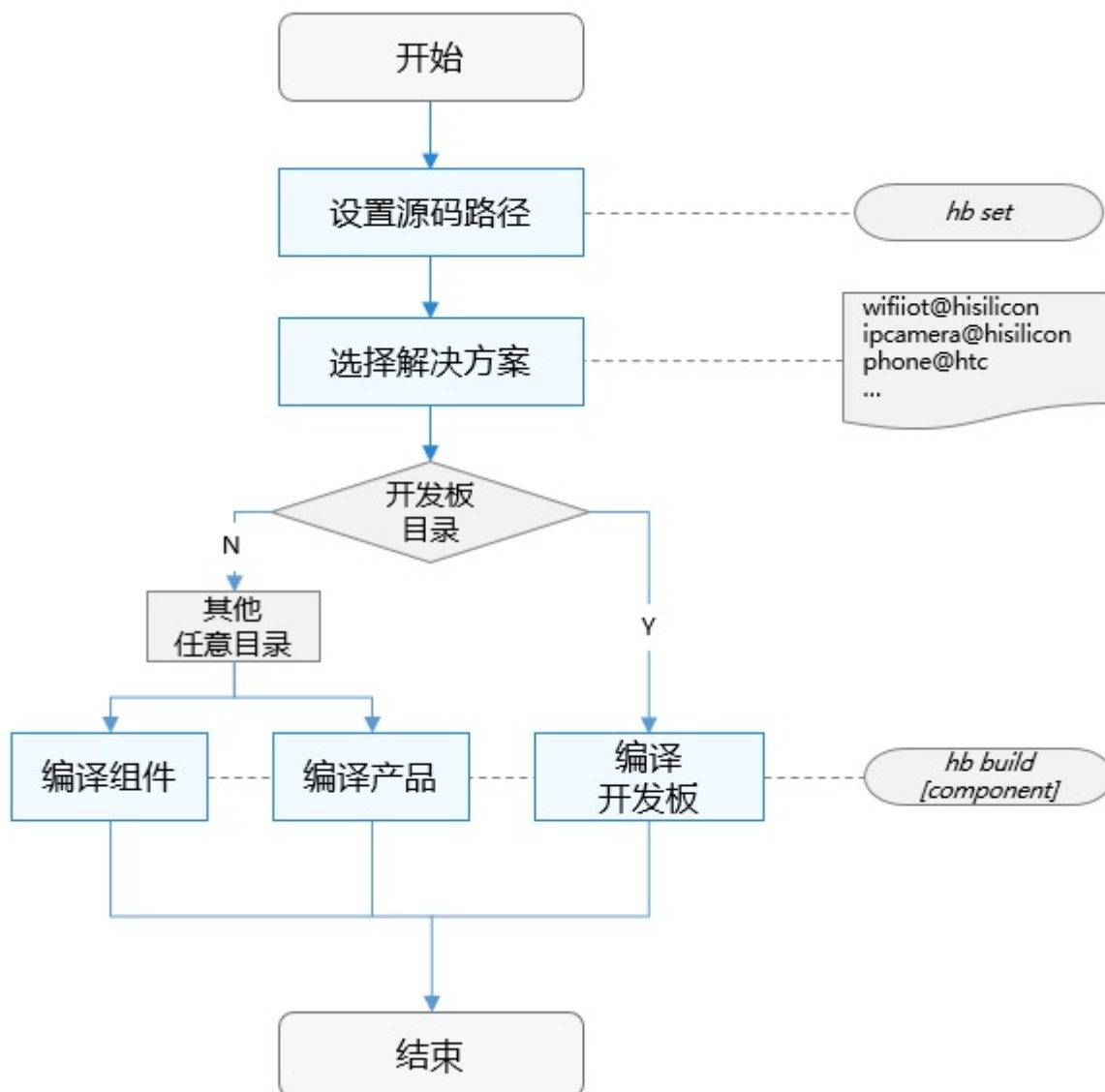
- **组件** 可复用的软件单元，它可包含源码、配置文件、资源文件和编译脚本等。
- **gn** Generate ninja的缩写，一种元构建系统，用于产生ninja文件。
- **ninja** ninja是一个专注于速度的小型构建系统。
- **hb工具**

hb是HarmonyOS2.0里新增加的编译构建命令行工具。需要Python 3.7.4及以上版本的支持，建议安装3.8.x。源码在~/build\lite\hb这个目录下。

1.2构建流程

编译构建流程如图1所示，主要包括设置和编译两步：

图 1 编译构建流程



1. hb set: 设置OpenHarmony源码目录和要编译的产品。

2. hb build: 编译产品、开发板或者组件。解决方案编译实现如下：

- 读取开发板配置：主要包括开发板使用的编译工具链、编译链接命令和选项等。
- 调用gn: 调用gn gen命令，读取产品配置(主要包括开发板、内核、选择的组件等)生成解决方案out目录和ninja文件。
- 调用ninja: 调用ninja -C out/company/product启动编译。
- 系统镜像打包：将组件编译产物打包，制作文件系统镜像。

1.3目录说明

build/lite	# 编译构建主目录
├ components	# 组件描述文件。
├ hb	# hb pip安装包源码。
├ make_rootfs	# 文件系统制作脚本。
├ config	# 编译相关的配置项
├ └ component	# 组件相关的模板定义。包括：静态库、动态库、扩展组
件、模拟器库等	
├ └ kernel	# 内核的编译配置参数
├ └ subsystem	# 子系统模板
├ ndk	# Native API相关编译脚本与配置参数
├ product	# 产品全量配置表，包括：配置单元、子系统列表、编译器
等。	
└ toolchain	# 编译工具链相关，包括：编译器路径、编译选项、链接选项等。

1.4使用说明

1. 前提条件

- Linux服务器，Ubuntu16.04及以上64位系统版本。
- Python 3.7.4及以上。
- OpenHarmony源码build_lite仓下载成功。

2. 安装hb

- 在源码根目录下执行：

```
python3 -m pip install --user build/lite
```

- 执行hb -h有相关帮助信息，有打印信息即表示安装成功：

```
usage: hb

OHOS build system

positional arguments:
  {build,set,env,clean}
    build                Build source code
    set                  OHOS build settings
    env                  Show OHOS build env
    clean                Clean output

optional arguments:
  -h, --help            show this help message and exit
```

- 卸载方法：

```
python3 -m pip uninstall ohos-build
```

3. 编译命令

1. hb set

```
hb set -h
usage: hb set [-h] [-root [ROOT_PATH]] [-p]

optional arguments:
  -h, --help            Show this help message and exit.
  -root [ROOT_PATH], --root_path [ROOT_PATH]
                        Set OHOS root path.
  -p, --product          Set OHOS board and kernel.
```

- hb set 后无参数，进入默认设置流程
- hb set -root [ROOT_PATH] 直接设置代码根目录
- hb set -p --product 设置要编译的产品

2. hb env 查看当前设置信息

```
hb env
[OHOS INFO] root path: xxx
[OHOS INFO] board: hispark_taurus
[OHOS INFO] kernel: liteos
[OHOS INFO] product: ipcamera
[OHOS INFO] product path: xxx/vendor/hisilicon/ipcamera
[OHOS INFO] device path:
xxx/device/hisilicon/hispark_taurus/sdk_linux_4.19
```

3. hb build

```
hb build -h
usage: hb build [-h] [-b BUILD_TYPE] [-c COMPILER] [-t [TEST [TEST
...]]]
                        [--dmverity] [-p PRODUCT] [-f] [-n]
                        [component [component ...]]
```

positional arguments:

component	Name of the component.
-----------	------------------------

optional arguments:

-h, --help	Show this help message and exit.
------------	----------------------------------

```

-b BUILD_TYPE, --build_type BUILD_TYPE
                        Release or debug version.
-c COMPILER, --compiler COMPILER
                        Specify compiler.
-t [TEST [TEST ...]], --test [TEST [TEST ...]]
                        Compile test suit.
--dmverity             Enable dmverity.
-p PRODUCT, --product PRODUCT
                        Build a specified product with
                        {product_name}@{company}, eg:
ipcamera@hisilcon.
-f, --full             Full code compilation.
-T [TARGET [TARGET ...]], --target [TARGET [TARGET ...]]
                        Compile single target

```

- **hb build**后无参数，会按照设置好的代码路径、产品进行编译，编译选项使用与之前保持一致。
- **hb build component**：基于设置好的产品对应的单板、内核，单独编译组件（e.g.: **hb build kv_store**）。
- **hb build -p PRODUCT**：免set编译产品，该命令可以跳过set步骤，直接编译产品。
- 在device/device_company/board下单独执行**hb build**会进入内核选择界面，选择完成后会根据当前路径的单板、选择的内核编译出仅包含内核、驱动的镜像。

4. **hb clean** 清除out目录对应产品的编译产物，仅剩下args.gn、build.log。清除指定路径可输入路径参数：**hb clean xxx/out/xxx**，否则将清除hb set的产品对应out路径

```

hb clean
usage: hb clean [-h] [out_path]

positional arguments:
  out_path      Clean a specified path.

optional arguments:
  -h, --help  Show this help message and exit.

```

二、Neptune w800开发版实操

环境：

Ubuntu 20.04 LTS

2.1 准备工作

2.1.1 示例工程下载

直接进入这个gitee的网址下载zip压缩包也可以（因为我用的是git，懒得再修改git配置文件）

```
git clone git@gitee.com:hihopeorg_group/neptune-harmony-os1.1-iot.git
```

2.1.2 编译

1. 编译工具链配置

工具包名：csky-elfabiv2-tools-x86_64-minilibc-xxxxx.tar.gz

下载地址：<https://occ.t-head.cn/community/download>

选择 工具 - 工具链-800 Series 中相应的版本

CPU

I系列

I805

C系列

S系列

E系列

SoC

工具

开发-软件开发指南

工具链-800系列

V3.10.29

工具链手册

工具链-900系列

资源名称	更新时间	资源大小	操作
ReleaseNotes.pdf	2021-02-26 14:09:35	201.47KB	下载
csky-elf-noneabiv2-tools-x86_64-newlib-20210423	2021-04-27 21:25:42	92.33MB	下载
csky-elf-noneabiv2-tools-i386-newlib-20210423	2021-04-27 21:25:36	92.95MB	下载
csky-elf-noneabiv2-tools-mingw-newlib-20210423	2021-04-27 21:25:30	99.94MB	下载
csky-elfabiv2-tools-x86_64-minilibc-20210423.tar	2021-04-27 21:25:24	76.72MB	下载
csky-elfabiv2-tools-i386-minilibc-20210423	2021-04-27 21:25:13	77.60MB	下载
csky-elfabiv2-tools-mingw-minilibc-20210423	2021-04-27 21:25:04	75.00MB	下载
csky-linux-uclibc-tools-x86_64-uclibc-linux-4.9.56-20210423	2021-04-27 21:24:40	65.85MB	下载
csky-linux-uclibc-tools-i386-uclibc-linux-4.9.56-20210423	2021-04-27 21:24:21	66.39MB	下载

下载完成后将工具包拷贝到相应的目录下，我的是：`~/HarmonyOS_Tool`

解压、配置环境变量

操作命令：(注意你的压缩包名字)

```
tar xzvf csky-elfabiv2-tools-x86_64-minilibc-20210423.tar.gz
```

在`~/.bashrc`文件添加（根据你的路径来）：

```
sudo gedit ~/.bashrc
```

```
export PATH=~/.HarmonyOS_Tool/csky-elfabiv2-tools-x86_64-minilibc-20210423/bin:$PATH
```

刷新环境变量：

```
source ~/.bashrc
```

2. WiFIOT环境配置

安装python3.7以上，我这里使用ubuntu自带的python3.8

下载gn/ninja包

```
URL_PREFIX=https://repo.huaweicloud.com/harmonyos/compiler
DOWNLOAD_DIR=~/.Downloads          # 下载目录，可自行修改
TOOLCHAIN_DIR=~/.HarmonyOS_Tool/toolchain  # 工具链存放目录，可自行修改

[ -e $DOWNLOAD_DIR ] || mkdir $DOWNLOAD_DIR
[ -e $TOOLCHAIN_DIR ] || mkdir -p $TOOLCHAIN_DIR
wget -P $DOWNLOAD_DIR $URL_PREFIX/gn/1523/linux/gn.1523.tar
wget -P $DOWNLOAD_DIR $URL_PREFIX/ninja/1.9.0/linux/ninja.1.9.0.tar
```

解压gn/ninja/包：

```
tar -C $TOOLCHAIN_DIR/ -xvf $DOWNLOAD_DIR/gn.1523.tar
tar -C $TOOLCHAIN_DIR/ -xvf $DOWNLOAD_DIR/ninja.1.9.0.tar
```

在`~/.bashrc`文件添加添加环境变量（根据你的路径填写）：

```
export PATH=~/.HarmonyOS_Tool/toolchain/gn:$PATH
export PATH=~/.HarmonyOS_Tool/toolchain/ninja:$PATH
```

刷新环境变量：

```
source ~/.bashrc
```

我的~/.HarmonyOS_Tool目录结构：

```
jaychou@jaychou-TM1705:~/HarmonyOS_Tool$ tree -L 2
.
├── csky-elfabiv2-tools-x86_64-minilibc-20210423
│   ├── bin
│   ├── csky-elfabiv2
│   ├── include
│   ├── lib
│   ├── lib64
│   ├── libexec
│   ├── share
│   └── x86_64-linux
└── toolchain
    ├── gn
    └── ninja
```

3. 安装编译工具hb

下载源码后进入终端，进入源码根目录，执行：

```
python3 -m pip install --user build/lite
```

在~/.bashrc文件添加添加环境变量：

```
export PATH=~/.local/bin:$PATH
```

刷新：

```
source ~/.bashrc # 生效环境变量
```


至此，我添加的环境变量为：

```
# Harmony
export PATH=~/.HarmonyOS_Tool/csky-elfabiv2-tools-x86_64-minilibc-20210423/bin:$PATH
export PATH=~/.HarmonyOS_Tool/toolchain/gn:$PATH
export PATH=~/.HarmonyOS_Tool/toolchain/ninja:$PATH
export PATH=~/.local/bin:$PATH
# Harmony
```

4.编译

安装完毕后，执行**hb -h**，确认安装成功



```
jaychou@jaychou-TM1705:~/HarmonyOS_Tool$ hb -h
usage: hb
positional arguments:
  build      Build source code
  set        OHOS build settings
  env        Show OHOS build env
  clean      Clean output
  deps       OHOS components deps
optional arguments:
  -h, --help show this help message and exit
```

在源码根目录，执行：

```
hb set
```

- 1、设置源码目录，因我们在源码根目录下执行命令，所以直接输入“.”，表示当前目录
- 2、选择“wifiot_neptune@winnermicro”

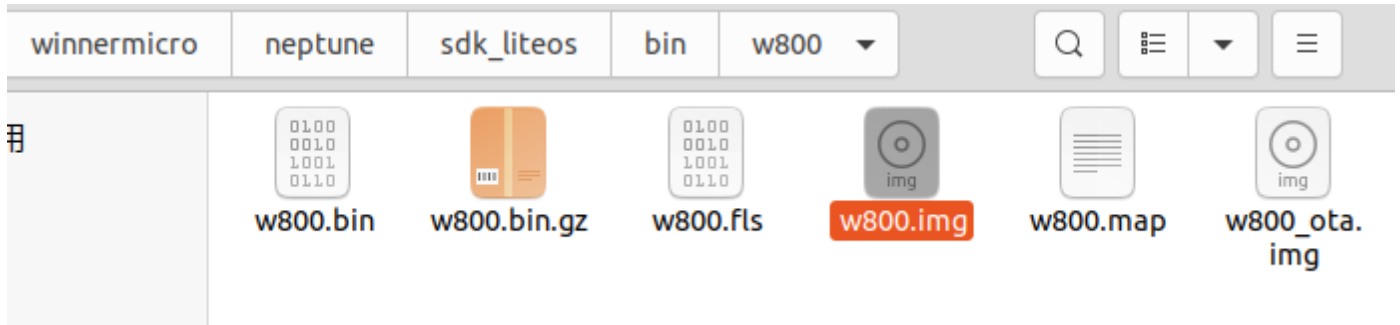
部分展示：

```
jaychou@jaychou-TM1705:~/neptune-harmony-os1.1-iot-master$ hb set
[OHOS INFO] Input code path: .
OHOS Which product do you need? wifiot_neptune@winnermicro
jaychou@jaychou-TM1705:~/neptune-harmony-os1.1-iot-master$ hb build -f -c gcc
[OHOS INFO] Done. Made 76 targets from 61 files in 92ms
-----省略-----
```

```
[OHOS INFO] wifiiot_neptune build success
jaychou@jaychou-TM1705:~/neptune-harmony-os1.1-iot-master$
```

使用**hb**命令指定**gcc**编译（默认是**clang**，我没有配置**clang**，所以使用**gcc**），执行 **hb build -f -c gcc**，等待编译直到出现“**wifiiot_neptune build success**”表示编译成功，否则根据提示处理。

编译完毕后会生成**w800.img**文件，路径：**~/neptune-harmony-os1.1-iot/device/winnermicro/neptune/sdk_liteos/bin/w800**



2.2 烧写镜像

2.2.1 Ubuntu 烧写工具

串口通信工具：minicom

```
sudo apt-get install minicom
```

安装Xmodem协议：（串口通信中广泛使用到的异步文件传输协议）

```
sudo apt-get install lrzsz
```

工具使用：

安装后输入：（-s是**setup**即建立连接）

```
sudo minicom -s
```

```

+-----[configuration]-----+
| Filenames and paths          |
| File transfer protocols      |
| Serial port setup           |
| Modem and dialing            |
| Screen and keyboard          |
| Save setup as dfl             |
| Save setup as..              |
| Exit                         |
| Exit from Minicom            |
+-----+

```

使用方向键 选择 Serial port setup，按Enter键，进入设置环境，如下图

```

+-----+
| A -   Serial Device           : /dev/ttyUSB0 |
| B - Lockfile Location         : /var/lock    |
| C -   Callin Program          :              |
| D -   Callout Program         :              |
| E -   Bps/Par/Bits             : 115200 8N1  |
| F - Hardware Flow Control     : No          |
| G - Software Flow Control     : No          |
|                                |
| Change which setting?      |
+-----+
| Screen and keyboard          |
| Save setup as dfl             |
| Save setup as..              |
| Exit                         |
| Exit from Minicom            |
+-----+

```

1. 输入a或者A，设置串口设备，我这里是/dev/ttyUSB0（输入a，光标就移动到A那一行了，然后键盘输入修改就行了，修改完成后回车，光标就回到了最下面，载输入E就可以编辑对应的选项了）；
2. 按图配置波特率；
3. Hardware Flow Control一定要关闭（No），不然看不到升级进度；
4. 都配置完后，按下Enter键返回上一界面，选择 **save setup as dfl**（即将其保存位默认配置）；
5. 选择Exit，关闭minicom。
6. 再次输入命令 **sudo minicom**，虚拟机上断开重新连接串口，即可看到开发板串口有输出。

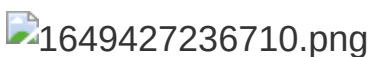
至此，开发板和ubuntu串口连通成功。

2.2.2 烧写过程

1. 在ubuntu 终端按住ESC，主板按下RST键（要多按住一会儿），工具界面输出cccccc时，则进入了刷机模式，按 ctrl+a，再按z

```
jaychou@jaychou-TM1705: ~
pdate the parameter in sram using partition - 0,2036,2036.
ead +-----+
atit|                                     Minicom Command Summary                                     |
alcu|                                     Commands can be called by CTRL-A <key>                                     |
ead|                                     +-----+                                     |
urre|                                     |                                     |
x ga|                                     |                                     |
9 69|                                     |                                     |
ac a| Dialing directory..D run script (Go)....G | Clear Screen.....C |
8 6D| Send files.....S Receive files.....R | cOnfigure Minicom..O |
ocke| comm Parameters....P Add linefeed.....A | Suspend minicom....J |
tart| Capture on/off.....L Hangup.....H | eXit and reset.....X | log l.
0 00| send break.....F initialize Modem...M | Quit with no reset.Q |
0 00| Terminal settings..T run Kermit.....K | Cursor key mode....I |
0 00| lineWrap on/off....W local Echo on/off..E | Help screen.....Z |
0 00| Paste file.....Y Timestamp toggle...N | scroll Back.....B |
0 00| Add Carriage Ret...U                                     |
0 00|                                     |
0 00|                                     Select function or press Enter for none. |
0 00+-----+
0 00:00:00 0 140 I 1/SAMGR: Bootstrap system and application services(count:0).
0 00:00:00 0 140 I 1/SAMGR: Initialized all system and application services!
0 00:00:00 0 140 I 1/SAMGR: Bootstrap dynamic registered services(count:0).
TRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | tyUSB0
```

2. 按下s，选择Send files
3. 选择 xmodem，选择路径，用上下箭头键选择目录，双击空格键进入目录，找到w800.img文件



4. 空格选中，左右方向键选择 [Okay] 按下回车即可，我这里编译出来的w800.img是581k，等待烧写完毕即可。

```
-----[xmodem upload - Press CTRL-C to quit]-----
Sending w800.img, 4539 blocks: Give your local XMODEM receive
command now.
Xmodem sectors/kbytes sent: 824/103k|
ev/ttyS0 (按网上设置的ttyUSB0怎么都不行，也许是虚拟机和真机区别) 按
按下Enter键返回上一界面，选择save setup as dfi (即将其保存位默认配置)
```

5. 烧写成功后，工具界面依然出现cccccc，这时按下RST键，系统启动。

```
update the parameter in sram using partition - 0,2036,2036.
read parameter partition - 1.
partition 1 magic - 0x4947414d, crc -0xc80562f5 .
calculate crc -0xc80562f5 .
read parameter partition modify count - 1.
current parameter partition modify count - 2.
tx gain dump length : 27
69 69 69 69 76 76 76 76 76 76 76 6A 76 76 76 76 - 76 76 6A 79 76 76 76 76 76 76
mac addr dump length : 6
28 6D CD 84 63 1A
socket_cfg.proto = 0, socket_cfg.client = 0, socket_cfg.port = 60000
start test task .....hiview init success.00 00:00:00 0 108 D 0/HIVIEW: log l
00 00:00:00 0 108 I 1/SAMGR: Bootstrap core services(count:3).
00 00:00:00 0 108 I 1/SAMGR: Init service:0x8156b8b TaskPool:0x20033f90
00 00:00:00 0 108 I 1/SAMGR: Init service:0x8156d73 TaskPool:0x20033fa8
00 00:00:00 0 108 I 1/SAMGR: Init service:0x8156d7d TaskPool:0x20033fc0
00 00:00:00 0 228 I 1/SAMGR: Init service 0x8156d7d <time: 2ms> success!
00 00:00:00 0 140 I 1/SAMGR: Init service 0x8156d73 <time: 2ms> success!
00 00:00:00 0 52 I 1/SAMGR: Init service 0x8156b8b <time: 4ms> success!
00 00:00:00 0 52 I 1/SAMGR: Initialized all core system services!
00 00:00:00 0 140 I 1/SAMGR: Bootstrap system and application services(count:0).
00 00:00:00 0 140 I 1/SAMGR: Initialized all system and application services!
00 00:00:00 0 140 I 1/SAMGR: Bootstrap dynamic registered services(count:0).
```