

# Android 11.0 Device MD 组态编译系统使用指南

文档版本                      V1.1  
发布日期                      2020-11-20

**版权所有 © 紫光展锐（上海）科技有限公司。保留一切权利。**

本文件所含数据和信息都属于紫光展锐（上海）科技有限公司（以下简称紫光展锐）所有的机密信息，紫光展锐保留所有相关权利。本文件仅为信息参考之目的提供，不包含任何明示或默示的知识产权许可，也不表示有任何明示或默示的保证，包括但不限于满足任何特殊目的、不侵权或性能。当您接受这份文件时，即表示您同意本文件中内容和信息属于紫光展锐机密信息，且同意在未获得紫光展锐书面同意前，不使用或复制本文件的整体或部分，也不向任何其他方披露本文件内容。紫光展锐有权在未经事先通知的情况下，在任何时候对本文件做任何修改。紫光展锐对本文件所含数据和信息不做任何保证，在任何情况下，紫光展锐均不负任何与本文件相关的直接或间接的、任何伤害或损失。

请参照交付物中说明文档对紫光展锐交付物进行使用，任何人对紫光展锐交付物的修改、定制化或违反说明文档的指引对紫光展锐交付物进行使用造成的任何损失由其自行承担。紫光展锐交付物中的性能指标、测试结果和参数等，均为在紫光展锐内部研发和测试系统中获得的，仅供参考，若任何人需要对交付物进行商用或量产，需要结合自身的软硬件测试环境进行全面的测试和调试。

# 紫光展锐（上海）科技有限公司



# 前言

## 概述

本文档主要阐述 UNISOC Android 11.0 软件平台 device/sprd 目录下 board 及其 product 的配置。提供了操作规范和使用示例，用户可根据具体的使用场景，定制化配置。

## 读者对象

本文档主要适用于使用 UNISOC Android 11.0 软件平台的用户。用户必须具备以下经验和技能：




- 熟悉或了解 Android 平台的配置。
- 熟悉 Linux 环境。

## 缩略语

缩略语	英文全名	中文解释
MD	Modularize build System	模块化编译系统
SoC	System on Chip	片上系统

## 符号约定

在本文中可能出现下列标志，它所代表的含义如下。

符号	说明
 <b>说明</b>	用于突出重要/关键信息、补充信息和小窍门等。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害。
 <b>注意</b>	用于突出容易出错的操作。 “注意”不是安全警示信息，不涉及人身、设备及环境伤害。
 <b>警告</b>	用于可能无法恢复的失误操作。 “警告”不是危险警示信息，不涉及人身及环境伤害。

## 变更信息

文档版本	发布日期	修改说明
V1.0	2020-07-21	第一次正式发布
V1.1	2020-11-20	更新模板，优化内容

## 关键字

board、product、module。

# 目 录

1 简介.....	1
1.1 概述.....	1
1.2 device/sprd 目录结构.....	1
1.3 board 配置原则.....	3
2 配置指导.....	4
2.1 创建新 board.....	4
2.2 创建新 product.....	5
2.3 创建新模块.....	5
2.4 定制化配置模块.....	6
2.4.1 当前 product 使用的 module.....	6
2.4.2 添加模块.....	7
2.4.3 删除模块.....	7
2.4.4 修改模块.....	8
2.5 定制化配置模块 feature.....	9
2.5.1 添加 feature.....	9
2.5.2 禁用 feature.....	9
2.5.3 替换 feature.....	9
3 调试指导.....	11
3.1 查看完整 log.....	11
3.2 查看编译相关变量值.....	11
3.3 通过 log 检查异常.....	12
3.3.1 检查缺失文件.....	12
3.3.2 检查模块删除状态.....	12
3.4 查看最终配置.....	13

## 图目录

---

图 1-1 device/sprd 目录结构 .....	2
图 3-1 lunch 工程后输出完整 log 信息 .....	11
图 3-2 lunch 工程后输出变量 log 信息 .....	12
图 3-3 文件缺失 log 信息 .....	12
图 3-4 已删除模块和 feature log 信息 .....	13

# 1 简介

## 1.1 概述

Android 源码中，device/sprd 目录是 UNISOC 的板级开发包和产品配置信息。Android 11.0 上，为了简化用户工程的创建，采用了面向对象的思想对 device/sprd 目录下配置信息的存放方式和组合方式进行了重构，这种组合方式称之为 MD（Modularize build System）组态编译系统。

device/sprd 目录下，所有 board 的通用配置信息、product 的通用配置信息，以及 UNISOC 提供的用于板卡上的模块配置信息，都以模块化的方式存放在 mpool 目录下。用户基于 MD 组态编译系统提供的接口，创建自己的 board 和 product。

本文档主要介绍基于 MD 组态编译系统，进行 device/sprd 目录下 board 和 product 的创建，以及用户定制化信息的添加。

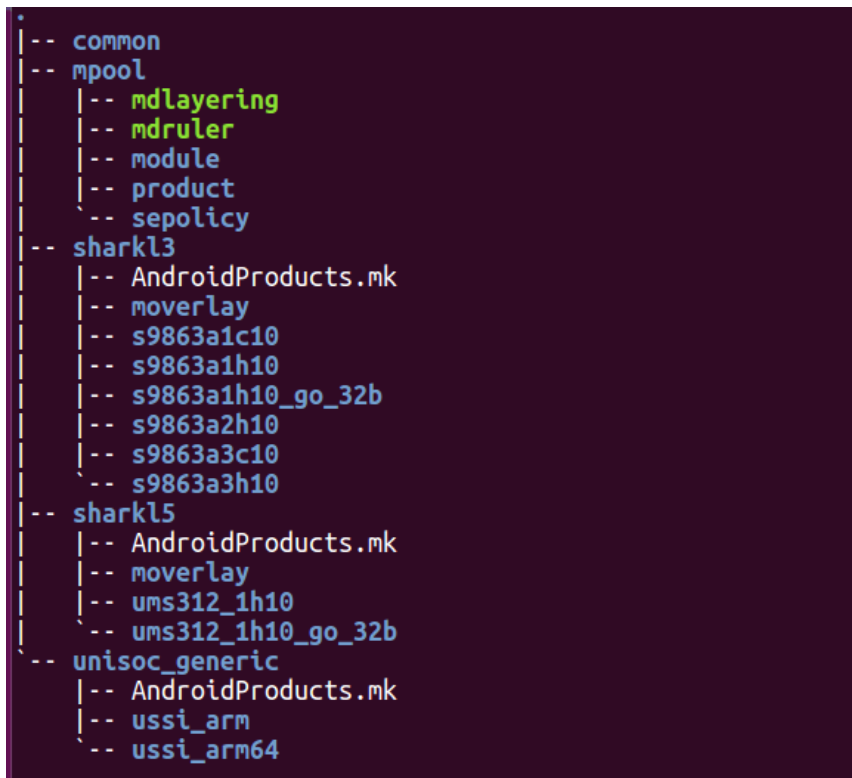
### 说明

- <xxx>：用于特指 xxx 类型中的某一个元素。例如，sharkl3/<board>，表示 sharkl3 目录下任意一款 board。
- board：board 级配置，描述一个物理板卡的特性、基本参数等，与硬件紧密相关的配置文件。
- product：product 级配置，描述基于一个物理板卡定义的产品特性、功能(如换壳、换主题)等。
- module：模块级配置信息，存放于具体平台、产品有关的模块的配置信息。相当于文中的“模块”。
- feature：模块功能级的配置，存放针对模块某一项功能的配置。

## 1.2 device/sprd 目录结构

device/sprd 目录结构如[图 1-1](#) 所示。

图1-1 device/sprd 目录结构



子目录说明：

- common 目录  
已经废弃，内容为空。
- mpool 目录  
UNISOC 提供的模块配置信息和工程母板，由 UNISOC 维护，用户不可修改。
- sharkl3、sharkl5 等目录  
目录名称为芯片的标识，存放同款芯片的 board。以 device/sprd/sharkl3 目录为例说明。
  - sharkl3 目录：存放 s9863a 芯片的所有 board。
  - sharkl3/s9863a1h10 目录：s9863a 芯片的一款 board s9863a1h10。
  - sharkl3/s9863a1h10/module 目录：存放 s9863a1h10 这款 board 上，用户新增的模块，或用户对 device/sprd/mpool/module 目录下模块的修改信息。该目录下的各个子目录分别代表不同的模块。
  - sharkl3/s9863a1h10/product 目录：s9863a1h10 这款 board 的所有 product 顶层目录。
  - sharkl3/s9863a1h10/product/s9863a1h10\_Natv 目录：s9863a1h10\_Natv 是 s9863a1h10 这款 board 的一个具体 product。该目录下存放 s9863a1h10\_Natv 这款 product 的配置信息。
- unisoc\_generic 目录  
UNISOC 的 SSI 配置信息，用户无需修改。

## 说明

s9863a 为芯片平台 SC9863A。



## 1.3 board 配置原则

- mpool 目录是 UNISOC 提供的配置信息，禁止用户进行源码级修改。
- 用户创建、配置 board 中，对 mpool 目录下的模块添加、删除、修改、覆盖只能在自己创建的 board 目录下使用 md 提供的接口或文件规范进行设置。
- 用户在工程配置文件中设定参数时，尽量避免采用 “:=”、“=”、“?=” 等方式直接对变量赋值，应采用 “md-set” 函数。

# 2

## 配置指导

用户操作主要在 device/sprd/<soc> 目录下进行。device/sprd 目录下用户配置信息的创建，主要包括：

- 创建新 board
- 创建新 product
- 创建新 module
- 定制化配置模块
- 定制化配置 feature

### 2.1 创建新 board

Android 11.0 创建一款新 board，除了在 device/sprd 目录下创建 board 配置外，还涉及到 bsp 和 vendor 目录下 board 级配置信息的创建，配置的目录和层级较多。Android 11.0 提供了自动化脚本，可以一次性完成 device、bsp、vendor 三个目录下 board 的创建。

芯片平台自带的 board 称之为 base board，用户创建新 board，需要基于同一芯片的 base board 进行克隆。新 board 创建完成后，用户可以手动在新 board 上进行定制化修改。

例如，在 sharkl3 目录下，创建一款新 board s9863a3h10。选取 board s9863a1h10 为 base board。

操作步骤：

步骤 1 采用如下命令进入脚本所在路径

```
cd vendor/sprd/tools/board_configure_tool/board-creates/
```

步骤 2 运行脚本创建 board，使用如下命令

```
./board-create.sh a11 sharkl3 s9863a arm64 s9863a1h10 s9863a3h10 4.14
```

#### 说明

board-create.sh 是 shell 脚本，在 Linux 环境下运行。该脚本有 7 个参数：

- 第一个参数，Android 版本，a11(Android 11.0)、a10(Android 10.0)。
- 第二个参数，存放 board 的目录名称，如 s9863a 芯片的 board 存放在 sharkl3 目录，ums312 芯片的 board 存放在 sharkl5 目录。
- 第三个参数，芯片名称。如 s9863a、ums512 等。
- 第四个参数，CPU ARCH，如 arm64 或 arm。
- 第五个参数，base board。例如基于 s9863a1h10 创建 s9863a3h10，第五个参数为 s9863a1h10。
- 第六个参数，要创建的新 board 名称。例如基于 s9863a1h10 创建 s9863a3h10，第六个参数为 s9863a3h10。
- 第七个参数，Linux 版本，5.4(kernel5.5)、4.14(kernel4.14)。用户工程一般采用 kernel4.14 版本。

---结束

## 2.2 创建新 product

创建新 product，指在 board 下创建特定 product 的目录及其配置文件。例如，在 s9863a3h10 board 下，创建一款新 product s9863a3h10\_New。

操作步骤：

步骤 1 在 device/sprd/sharkl3/s9863a3h10/product 目录下创建 s9863a3h10\_New 目录。

```
mkdir device/sprd/sharkl3/s9863a3h10/product/s9863a3h10_New
```

步骤 2 在 device/sprd/sharkl3/s9863a3h10/product/s9863a3h10\_New 目录下创建 main.mk 文件。

```
touch device/sprd/sharkl3/s9863a3h10/product/s9863a3h10_New/main.mk
```

步骤 3 在 device/sprd/sharkl3/s9863a3h10/product/s9863a3h10\_New/main.mk 文件中写如下内容：

- 对于 full 版本  
\$(call md-inherit-product, ,full,arm64,armv8-a)
- 对于 go 1G 版本和 go 2G 版本  
\$(call md-inherit-product, ,go,arm64,armv8-a)
- 对于 go 512M  
\$(call md-inherit-product, ,go512,arm64,armv8-a)

### 说明

md-inherit-product 函数用于指定当前 product 使用的母板，通过如下 4 个参数选择母板：

- 第一个参数：product 母板路径，缺省为默认母板。
- 第二个参数：go、full 版本。Google 规定，内存低于等于 2GB 为 go 设备。
- 第三个参数：cpu 类型，arm 或 arm64。
- 第四个参数：cpu 版本，armv7、armv8、armv8-a 等。

---结束

## 2.3 创建新模块

创建新模块，指在 board 下创建特定模块的目录及其配置文件。例如，在 device/sprd/sharkl3 目录下，针对 s9863a3h10 board，创建一个新模块 security。

操作步骤：

步骤 1 在 board 所在路径的 module 目录下创建新模块的配置信息存放目录。在 device/sprd/sharkl3/s9863a3h10/module 目录下创建 security 目录。

```
mkdir -p device/sprd/sharkl3/s9863a3h10/module/security
```

步骤 2 在 device/sprd/sharkl3/s9863a3h10/module/security 目录下，创建 security 模块相关的文件和子目录。模块的配置内容较多，并且不同模块之间也有差异。配置内容按类别存储规范如下表所示，按需创建。

名称	功能
md.mk	模块的通用配置。

名称	功能
md.rc	模块的 rc 相关配置，可为空。
mfeature	模块不同 feature 的配置存放在 mfeature 下，可为空。
其他文件	属于该模块的其他文件。可为空。

- a 创建 md.mk 文件，将 security 模块的通用配置信息写到 md.mk 文件。

```
touch device/sprd/sharkl3/s9863a3h10/module/security/md.mk
```

- b 创建 md.rc 文件，把属于 security 模块的 rc 相关配置信息写到该文件。

```
touch device/sprd/sharkl3/s9863a3h10/module/security/md.rc
```

- c 模块 security 有 facied 和 tui 两个不同的 feature。在 mfeature 下创建 facied 和 tui 两个子目录，并将 facied 相关的配置信息存放在 facied 目录，tui 相关的配置信息存放在 tui 目录。

```
mkdir device/sprd/sharkl3/s9863a3h10/module/security/mfeature
```

```
mkdir device/sprd/sharkl3/s9863a3h10/module/security/mfeature/facied
```

```
mkdir device/sprd/sharkl3/s9863a3h10/module/security/mfeature/tui
```

在 facied 目录下创建 facied.mk 文件，写 facied 的配置信息。

```
touch device/sprd/sharkl3/s9863a3h10/module/security/mfeature/tui/tui.mk
```

在 tui 目录下创建 tui.mk 文件，写 tui 的配置信息。

```
touch device/sprd/sharkl3/s9863a3h10/module/security/mfeature/facied/facied.mk
```

## 说明

- md.mk 的模块会对基于此 board 的所有产品自动生效，无需在 product 上配置。
- 如果新添加的模块，只对此款 board 特定的 product 生效，需要将文件名称 md 修改为 main 并在 product 配置文件使用“md-add”函数添加模块(这种情况在产品端出现较少)。

---结束

## 2.4 定制化配置模块

### 2.4.1 当前 product 使用的 module

product 中使用的 module 主要来自于两处：

- 用户当前 board 中定义的 module。即 device/sprd/<soc>/<board>/module 目录下各个 module。该目录下的 module 无需在 product 中显式调用，MD 组态编译系统会自动查找并将 module 加入到 board 下的所有 product 中。
- mpool 仓库中定义的 module。即 device/sprd/mpool/module 目录下各个 module。该目录下的 module 都有一个 main.mk 文件，main.mk 文件被显式调用，模块才会生效。mpool 仓库下的模块调用被定义在工程默认母板中，无需用户逐个添加。

**注意**

device/sprd/mpool/module 下模块的 msoc 目录，存放的是模块在不同平台上的配置。例如，device/sprd/mpool/module/display 模块，msoc 目录下有 sharkl3、pike2 等子目录。msoc/sharkl3 目录下放的 display 模块针对 device/sprd/sharkl3 路径下用户工程的配置，msoc/pike2 目录下放的 display 模块针对 device/sprd/pike2 路径下用户工程的配置，其余目录以此类推。

用户可通过如下方式查看当前工程使用的模块：首先 lunch 工程，然后在生成的 out/md.stack 文件中搜索“md.mk”和“main.mk”。每个 device/sprd/<soc>/<board>/module/<module>/md.mk 或 device/sprd/mpool/module/<module>/main.mk 对应一个模块。详细内容参见 3.3 通过 log 检查异常。

## 2.4.2 添加模块

添加模块，指将 device/sprd/mpool/module 中定义的模块添加到用户工程。例如，将 device/sprd/mpool 目录下 camera 模块加入到 s9863a3h10 board 下的 s9863a3h10\_New product。

操作步骤：

步骤 1 创建 device/sprd/sharkl3/s9863a3h10/product/s9863a3h10\_New/var.mk 文件。如果文件已存在，则无需再创建。

```
touch device/sprd/sharkl3/s9863a3h10/product/s9863a3h10_New/var.mk
```

步骤 2 在 device/sprd/sharkl3/s9863a3h10/product/s9863a3h10\_New/var.mk 文件中写如下内容：

```
$(call md-add, $(MPOOL)/module/camera/main.mk)
```

**说明**

\$(MPOOL)表示 device/sprd/mpool 目录。

---结束

## 2.4.3 删除模块

删除模块，是指将用户 board 下定义的模块，或 device/sprd/mpool/module 中定义的模块，从当前工程中删除。例如，s9863a3h10 board 上默认使用了 device/sprd/mpool 目录下的 light 模块。如果该 board 的 s9863a3h10\_New product 不需要 light 模块，则将 light 模块从 product 编译中删除。

操作步骤：

步骤 1 创建 device/sprd/sharkl3/s9863a3h10/product/s9863a3h10\_New/var.mk 文件。如果文件已存在，则无需再创建。

```
touch device/sprd/sharkl3/s9863a3h10/product/s9863a3h10_New/var.mk
```

步骤 2 在 device/sprd/sharkl3/s9863a3h10/product/s9863a3h10\_New/var.mk 文件中写如下内容：

```
$(call md-disable, $(MPOOL)/module/light/main.mk)
```

**说明**

\$(MPOOL)表示 device/sprd/mpool 目录。

---结束

## 2.4.4 修改模块

用户获取具体变量的配置可参考 [3 调试指导](#)。修改模块，是指对 product 中已经使用的模块进行参数修改。模块已经添加到 product 中使用，但部分配置信息需要修改，可分为以下两种情况：

- 情况一：修改模块的参数，对同一 board 下的所有 product 都生效。
- 情况二：修改模块的参数，只对 board 下一款 product 有效。

### 对所有 product 有效

在 board 所在路径下的 module 目录下，修改模块的参数，对同一 board 下的所有 product 都生效。例如，在 device/sprd/sharkl3/s9863a3h10/module/目录下，board s9863a3h10 中修改已添加的模块 display。

操作步骤：

步骤 1 在 device/sprd/sharkl3/s9863a3h10/module/目录下创建 display 目录，存放 display 的修改信息。

```
mkdir device/sprd/sharkl3/s9863a3h10/module/display
```

步骤 2 创建 md.mk 文件，将模块的修改信息写在 md.mk 文件

```
touch device/sprd/sharkl3/s9863a3h10/module/display/md.mk
```

----结束

### 对特定 product 有效

修改模块的参数，只对 board 下一个 product 有效，在特定 product 的 var.mk 文件中进行。例如，product s9863a3h10\_New 中修改已添加的模块 display。将 display 模块的 PRODUCT\_AAPT\_PREF\_CONFIG 参数修改为 xxhdpi。在 device/sprd/sharkl3/s9863a3h10/product/s9863a3h10\_New/var.mk 文件中直接添加修改信息。

操作步骤：

步骤 1 创建 device/sprd/sharkl3/s9863a3h10/product/s9863a3h10\_New/var.mk 文件。如果文件已存在，则无需再创建。

```
touch device/sprd/sharkl3/s9863a3h10/product/s9863a3h10_New/var.mk
```

步骤 2 将修改信息写在 var.mk 文件。通过 md-set 函数修改 PRODUCT\_AAPT\_PREF\_CONFIG 参数为 xxhdpi

```
$(call md-set, PRODUCT_AAPT_PREF_CONFIG, xxhdpi)
```

### 说明

md-set 函数用来修改变量的值。有 2 个参数：

- 第一个参数：变量名称。
- 第二个参数：变量的值。

----结束

## 2.5 定制化配置模块 feature

### 2.5.1 添加 feature

本文中，feature 指模块针对不同服务场景的不同配置。添加 feature，指将模块特定功能的配置添加到 product 编译。一般情况下，一个模块有多个功能，功能（feature）相关的配置信息存放在 mfeature 目录下。例如 device/sprd/mpool/module/目录下的 telephony 模块，有 carriers、cdma、simcount 等 7 个 feature，存放在 device/sprd/mpool/module/telephony/目录下。

例如，carriers 这个 feature，针对不同的 product 有 cmcc、ctcc 两个版本。将 telephony 模块的 carriers feature 添加到 s9863a3h10\_New product，并且采用 cmcc 版本。

操作步骤：

步骤 1 创建 device/sprd/sharkl3/s9863a3h10/product/s9863a3h10\_New/var.mk 文件。如果文件已存在，则无需再创建。

```
touch device/sprd/sharkl3/s9863a3h10/product/s9863a3h10_New/var.mk
```

步骤 2 在 device/sprd/sharkl3/s9863a3h10/product/s9863a3h10\_New/var.mk 文件中写如下内容：

```
$(call md-mfeature-add, telephony/carriers/cmcc)
```

#### 说明

md-mfeature-add 函数用来添加一个 feature。只有 1 个参数，指 mfeature 目录下具体 feature 相对路径的前缀。

----结束

### 2.5.2 禁用 feature

禁用 feature，指在特定 product 中不使用针对模块的某一特定功能的配置信息。例如，s9863a3h10 board 的所有 product 中都默认使用了 security 模块的 facied feature。s9863a3h10\_New product 不支持 facied feature，在该 product 中禁用 facied feature 相关的配置。

操作步骤：

步骤 1 创建 device/sprd/sharkl3/s9863a3h10/product/s9863a3h10\_New/var.mk 文件。如果文件已存在，则无需再创建。

```
touch device/sprd/sharkl3/s9863a3h10/product/s9863a3h10_New/var.mk
```

步骤 2 在 device/sprd/sharkl3/s9863a3h10/product/s9863a3h10\_New/var.mk 文件中写如下内容：

```
$(call md-disable, security/facied)
```

----结束

### 2.5.3 替换 feature

替换 feature，指将一个 feature 的配置由一个版本替换为另一个版本。例如 device/sprd/mpool/module/目录下 telephony 模块的 carriers feature，针对 cmcc 和 ctcc 有两种不同的配置。s9863a3h10 这款 board 的所有 product 默认使用 carriers 的 ctcc 版本。但 s9863a3h10\_New product 使用 cmcc 版本，所以需要将 carriers 由 ctcc 替换为 cmcc。

操作步骤：

步骤 1 创建 device/sprd/sharkl3/s9863a3h10/product/s9863a3h10\_New/var.mk 文件。如果文件已存在，则无需再创建。

```
touch device/sprd/sharkl3/s9863a3h10/product/s9863a3h10_New/var.mk
```

步骤 2 通过 md-del 函数删除当前 product 的 cmcc，在 device/sprd/sharkl3/s9863a3h10/product/s9863a3h10\_New/var.mk 文件中写如下内容：

```
$(call md-disable, telephony/carriers/cmcc)
```

步骤 3 通过 md-mfeature-add 函数在当前 product 中添加 ctcc，在 device/sprd/sharkl3/s9863a3h10/product/s9863a3h10\_New/var.mk 文件中写如下内容：

```
$(call md-mfeature-add, telephony/carriers/ctcc)
```

---结束



# 3

## 调试指导

device/sprd 目录下的配置可以单独调试，有助于用户进行配置信息检查和异常定位。主要调试方式包括：

1. 通过设置环境变量 MDLOG=1，打开 lunch 时完成输出 log。
2. 通过 log 和 get\_build\_var 命令查看变量的值。
3. 通过 log 中的特殊颜色和标识查看定制化配置信息和异常信息。
4. 通过生成的中间文件检查 lunch 结果。

### 3.1 查看完整 log

lunch 工程时，log 信息不会完全显示到输出窗口。如需完整 Log 信息，在命令行窗口输入如下命令：

```
export MDLOG=1
```

然后 lunch 工程。命令行窗口会输出完整 log 信息，如图 3-1 所示。

图3-1 lunch 工程后输出完整 log 信息

```
luther.qe@jand01:~/sprdroid/trunk$ time lunch s9863alh0_Natv3-userdebug-native
[W] (2019-12-31T18:51:49+0800)[1] bool caps:inits(nsjconf_t *)():215 prctl(PR_CAP_AMBIENT, PR_CAP_AMBIENT_CLEAR_ALL): Invalid argument
build/make/core/product_config.mk:228: warning: MD: mode device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/main.mk
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/main.mk:1: warning: md-inherit-product device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/main.mk
/bin/bash: -c: line 0: syntax error near unexpected token `newline'
/bin/bash: -c: line 0: `mkdir -p `dirname ` `>'
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/main.mk:1: warning: MD: including device/sprd/sharkl3/s9863alh0/product/base/var.mk
device/sprd/sharkl3/s9863alh0/product/base/var.mk:1: warning: MD: including device/sprd/sharkl3/class/product/arm64/arm64.mk
device/sprd/sharkl3/s9863alh0/product/base/var.mk:3: warning: KERNEL_PATH: export has been deprecated. It is a global setting. See https://android.googlesource.com/platform/build/+master/
changes.md#export-keyword.
device/sprd/sharkl3/s9863alh0/product/base/var.mk:20: warning: MD: including device/sprd/sharkl3/class/base/board/var.mk
device/sprd/sharkl3/class/base/board/var.mk:10: warning: MD: including device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/var.mk
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/var.mk:3: warning: Auto Expand md.mk in device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/var.mk:3: warning: MD: including device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: aaaaaaaaaaaaaaaaaaaaa native 3
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: Done Auto Expand md.mk in device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: CHIP NAME sharkl3
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: TARGET_BOARD s9863alh0
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: PLATDIR device/sprd/sharkl3
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: PLATCOM device/sprd/sharkl3/common
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: BOARDDIR device/sprd/sharkl3/s9863alh0
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: ROOTDIR device/sprd/sharkl3/s9863alh0/rootdir
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: ROOTCOM device/sprd/sharkl3/common/rootdir
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: MBOARD device/sprd/sharkl3
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: MBOARD device/sprd/sharkl3/s9863alh0/class
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: MPRODUCT device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: MDMK out/md.mk
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: MDBC out/md.rc
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: MDRE out/md.rc
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: MDUEVENTD out/md.ueventd.rc
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: LOCAL_PRODUCT_ROOT device/sprd/sharkl3/s9863alh0/product
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: LOCAL_PRODUCT_ROOT device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv3/class/md.mk:1: warning: MD: including device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/main.mk
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/main.mk:1: warning: md-inherit-product device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv/main.mk
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/main.mk:1: warning: MD: including device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/var.mk
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/var.mk:3: warning: Auto Expand md.mk in device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/var.mk:3: warning: MD: including device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class/md.mk
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class/md.mk:1: warning: aaaaaaaaaaaaaaaaaaaaa native 2
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class/md.mk:1: warning: Done Auto Expand md.mk in device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class/md.mk:1: warning: CHIP NAME sharkl3
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class/md.mk:1: warning: TARGET_BOARD s9863alh0
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class/md.mk:1: warning: PLATDIR device/sprd/sharkl3
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class/md.mk:1: warning: PLATCOM device/sprd/sharkl3/common
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class/md.mk:1: warning: BOARDDIR device/sprd/sharkl3/s9863alh0
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class/md.mk:1: warning: ROOTDIR device/sprd/sharkl3/s9863alh0/rootdir
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class/md.mk:1: warning: ROOTCOM device/sprd/sharkl3/common/rootdir
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class/md.mk:1: warning: MBOARD device/sprd/sharkl3
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class/md.mk:1: warning: MBOARD device/sprd/sharkl3/s9863alh0/class
device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class/md.mk:1: warning: MPRODUCT device/sprd/sharkl3/s9863alh0/product/s9863alh0_Natv2/class
```

### 3.2 查看编译相关变量值

用户在调试过程中需要查看变量的值，可以通过 2 种方式：

- lunch 工程时，打印出的 log  
例如，lunch 了 s9863a1h10\_Natv-userdebug-native，输出的 log 信息如图 3-2 所示。

图3-2 lunch 工程后输出变量 log 信息

```
device/sprd/mpool/product/var.mk:34: warning: MDOS=androidq
device/sprd/mpool/product/var.mk:34: warning: MDPRODUCT_NAME= s9863a1h10_Natv
device/sprd/mpool/product/var.mk:34: warning: TARGET_BOARD =s9863a1h10
device/sprd/mpool/product/var.mk:34: warning: TARGET_BOARD_PLATFORM =sp9863a
....
device/sprd/mpool/product/var.mk:34: warning: MDRC= out/md.rc
device/sprd/mpool/product/var.mk:34: warning: MDUEVENTD =out/md.ueventd.rc
device/sprd/mpool/product/var.mk:34: warning: MDSTACK=out/md stack
```

- get\_build\_var 命令获得指定变量的值  
例如，lunch 完工程后，获取变量 MDPRODUCT\_NAME 的值，输入如下命令：

```
get_build_var MDPRODUCT_NAME
```

log 打印完成后，最终输出变量值：

```
s9863a1h10_Natv
```

## 3.3 通过 log 检查异常

### 3.3.1 检查缺失文件

在工程配置过程中，如果使用了不存在的资源文件，或者资源文件缺失。这种异常在 lunch 输出的 log 中，以红色和“MIS”标识。如图 3-3 所示。

图3-3 文件缺失 log 信息

```
build/make/core/board_config.mk:143: warning: MD: MIS: 025 device/sprd/sharkl3/s9863a3h10/module/audio/system/etc/audio_params/codec_pga.xml
build/make/core/board_config.mk:143: warning: MD: MIS: 026 device/sprd/sharkl3/s9863a3h10/module/audio/system/etc/audio_params/audio_hw.xml
build/make/core/board_config.mk:143: warning: MD: MIS: 027 device/sprd/sharkl3/s9863a3h10/module/audio/system/etc/audio_params/audio_arm.xml
build/make/core/board_config.mk:143: warning: MD: MIS: 028 device/sprd/sharkl3/s9863a3h10/module/audio/system/etc/audio_params/audio_dsp.xml
build/make/core/board_config.mk:143: warning: MD: MIS: 029 device/sprd/sharkl3/s9863a3h10/module/audio/system/etc/audio_params/audio_dsp_ex.xml
build/make/core/board_config.mk:143: warning: MD: MIS: 030 device/sprd/sharkl3/s9863a3h10/module/audio/system/etc/audio_params/audio_dsp_ex_smat.xml
build/make/core/board_config.mk:143: warning: MD: MIS: 031 device/sprd/sharkl3/s9863a3h10/module/audio/system/etc/audio_params/cvs_params.xml
build/make/core/board_config.mk:143: warning: MD: MIS: 032 device/sprd/sharkl3/s9863a3h10/module/audio/system/etc/audio_params/dsp_codec_config.xml
```

### 3.3.2 检查模块删除状态

在工程配置中，有时用户通过 md-disable 函数将模块和 feature 从编译中删除。可以在 lunch 的输出 log 中查看模块或 feature 是否被删除，以红色和“disabled module”标识。如图 3-4 所示。

图3-4 已删除模块和 feature log 信息

```
build/make/core/board_config.mk:143: warning: MD: disabled module: 001 device/sprd/mpool/module/security/mfeature/security/faceid
build/make/core/board_config.mk:143: warning: MD: disabled module: 002 device/sprd/mpool/module/cts/main.mk
build/make/core/board_config.mk:143: warning: MD: disabled module: 003 device/sprd/mpool/module/ai/main.mk
build/make/core/board_config.mk:143: warning: MD: disabled module: 004 device/sprd/mpool/module/apr/main.mk
build/make/core/board_config.mk:143: warning: MD: disabled module: 005 device/sprd/mpool/module/tota/main.mk
```

## 3.4 查看最终配置

用户在 lunch 完工程后，device/sprd 目录下的配置信息最终会以一定的方式组合。最终在 out 目录下生成中间文件，便于检查自定义配置。例如 out 目录下生成如下文件：

- md.log 文件  
lunch 输出的 log 会保存在 out/md.log 文件，便于用户静态打开查看 log。
- md.stack 文件  
device/sprd 下的 mk 文件的调用顺序会保存在 out/md.stack 文件。文件引用的先后顺序会加以序号标识，便于用户调试。
- md.mk 文件  
device/sprd 下的 mk 文件按 MD 组态编译系统定义的逻辑组合在一起，按序展开，最终会形成一个文件 md.mk。lunch 工程后，用户可以查看该文件检查自己的配置是否符合预期。
- md.rc  
device/sprd/mpool 和用户 board 下，所有模块内部的 rc 文件按 MD 组态编译系统定义的逻辑组合在一起，按序展开，最终会形成一个 md.rc 文件。lunch 工程后，用户可以查看该文件检查自己的配置是否符合预期。