实验3 鸿蒙 LiteOS-a 内核移植——增加一个单板

《实用操作系统》实验报告　22920212204392 黄勖

1　实验环境

Windows10 21H2、Vmware Workstation Pro 16、Ubuntu18.04

配置了相关的软件。

2　实验目的

1. 通过修改源码，完成添加一个单板的任务

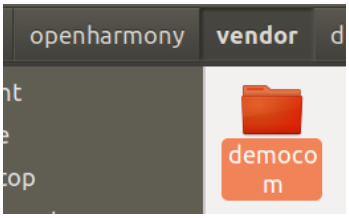
3　实验步骤与内容

新增单板信息为：

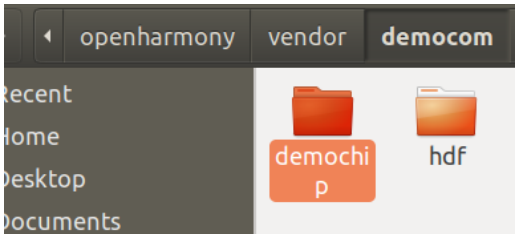
芯片公司：DemoCom
芯片名称：DemoChip

3.1　准备工作

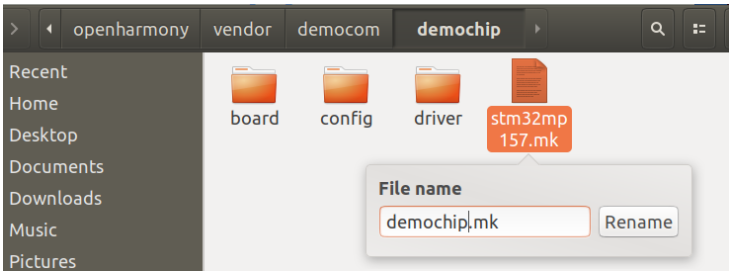
1. 进入openharmony/vendor目录下，复制一份st文档夹并改名为democom



2. 进入democom文档夹，并修改stm32mp157文档夹名为demochip



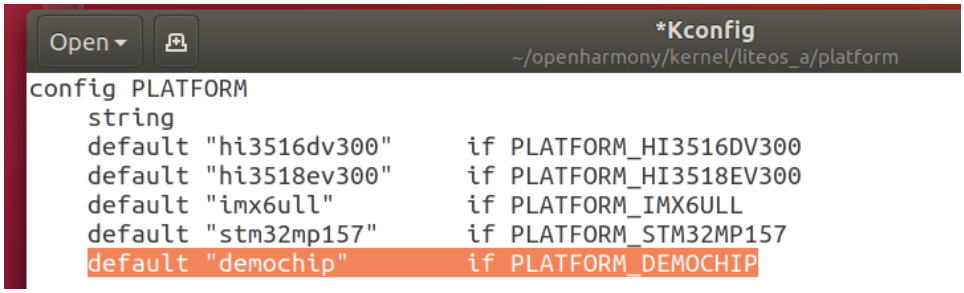
3. 进入demochip，并修改stm32mp157.mk文档名为demochip.mk



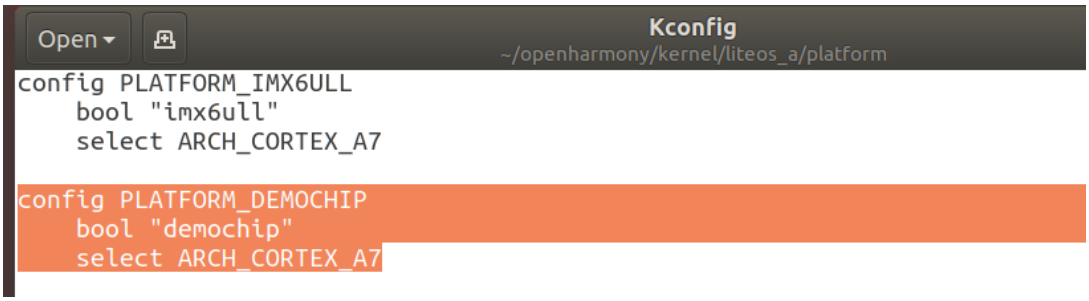
3.2 修改openharmony/kernel/liteos_a/platform/Kconfig文档

添加单板信息

```
default "demochip"          if PLATFORM_DEMOCHIP
```

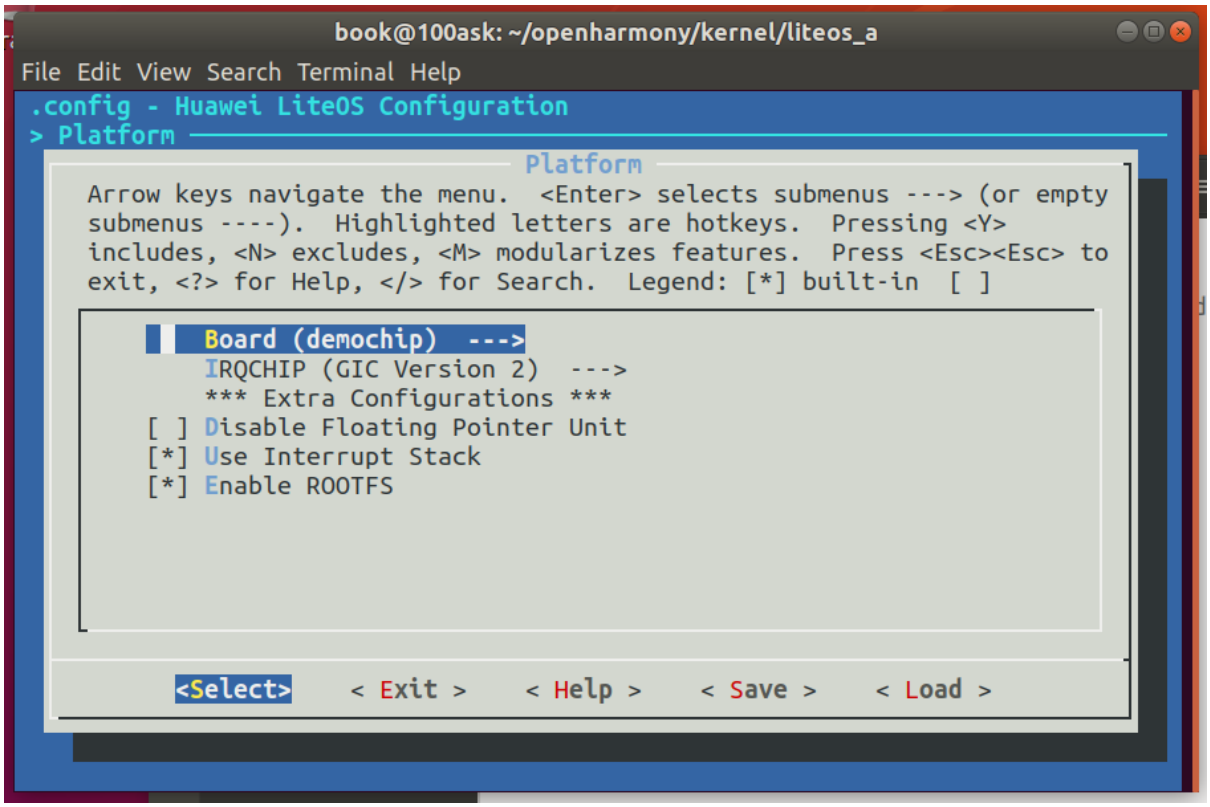



```
config PLATFORM_DEMOCHIP
    bool "demochip"
    select ARCH_CORTEX_A7
```

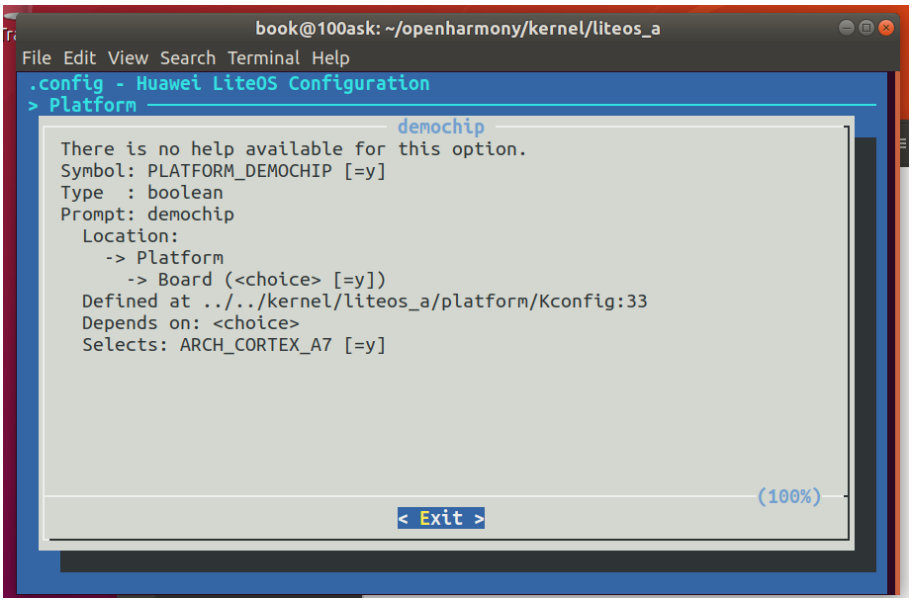


3.3 配置界面里添加单板

打开 menuconfig 查看新的单板配置 Select 新增板子



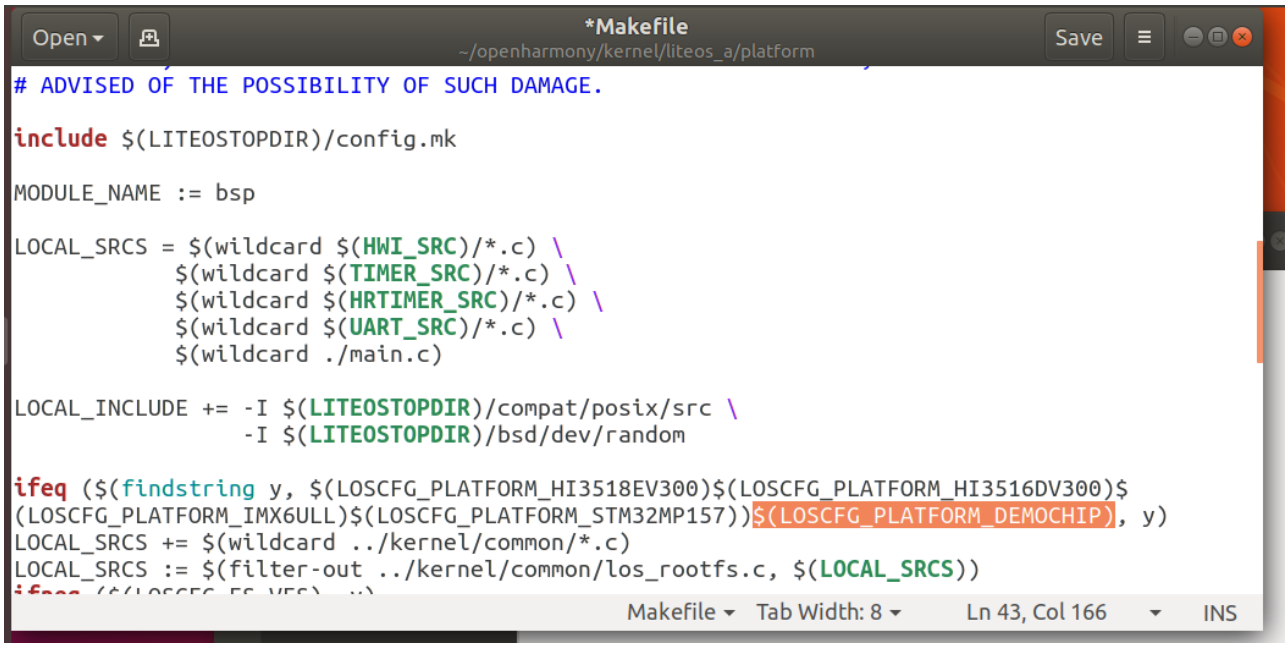
附：路径说明如下：



3.4 修改 openharmony/kernel/liteos_a/platform/Makefile 文件

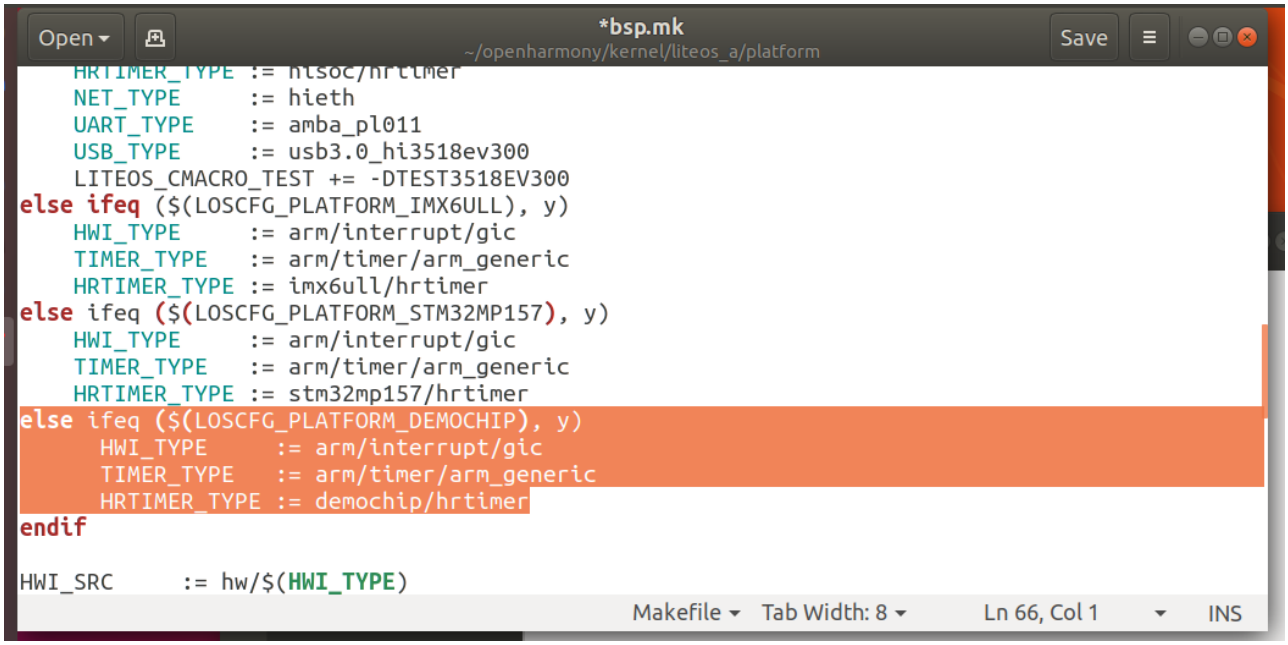
注：以下修改文件可由csdn教程类似`grep "LOSCFG_PLATFORM_STM32MP157" * -nr`命令搜索当前目录及其子目录中的所有文件，找到包含字符串的行和行号，以此来移植编译STM32MP157单板的信息。

```
$(LOSCFG_PLATFORM_WUFEIZHI)
```

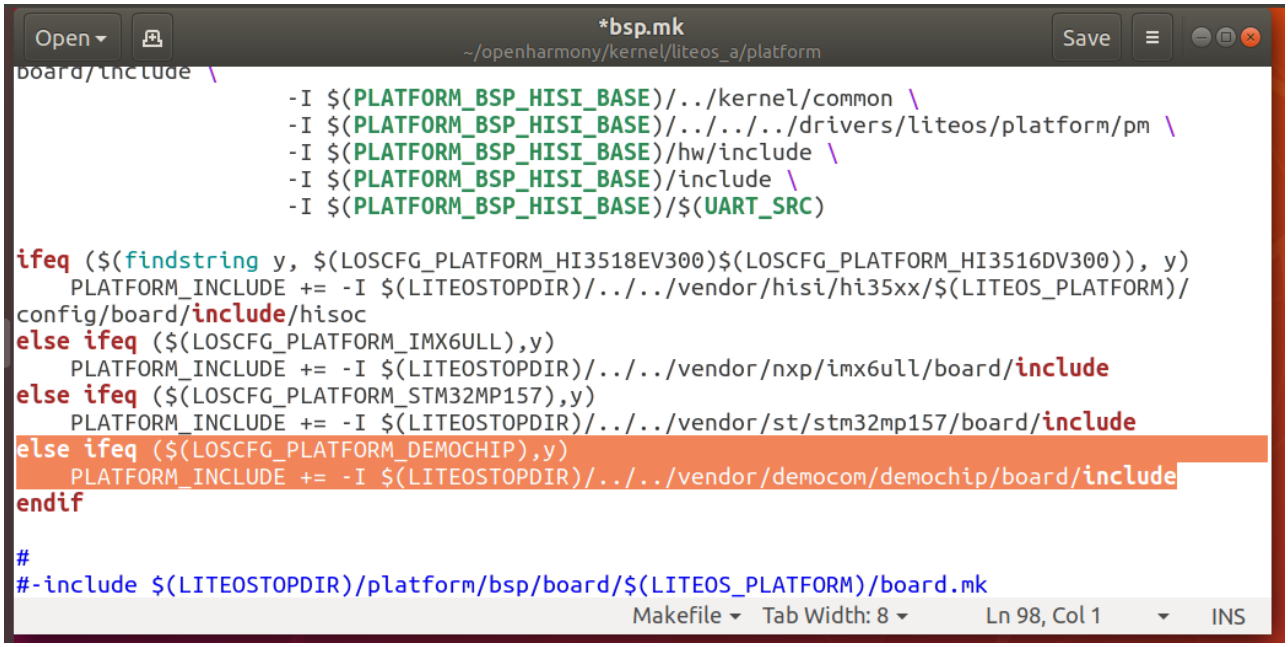


3.5 修改 openharmony/kernel/liteos_a/platform/bsp.mk 文件

```
else ifeq ($(LOSCFG_PLATFORM_DEMOCHIP), y)
    HWI_TYPE      := arm/interrupt/gic
    TIMER_TYPE    := arm/timer/arm_generic
    HRTIMER_TYPE := demochip/hrtimer
```

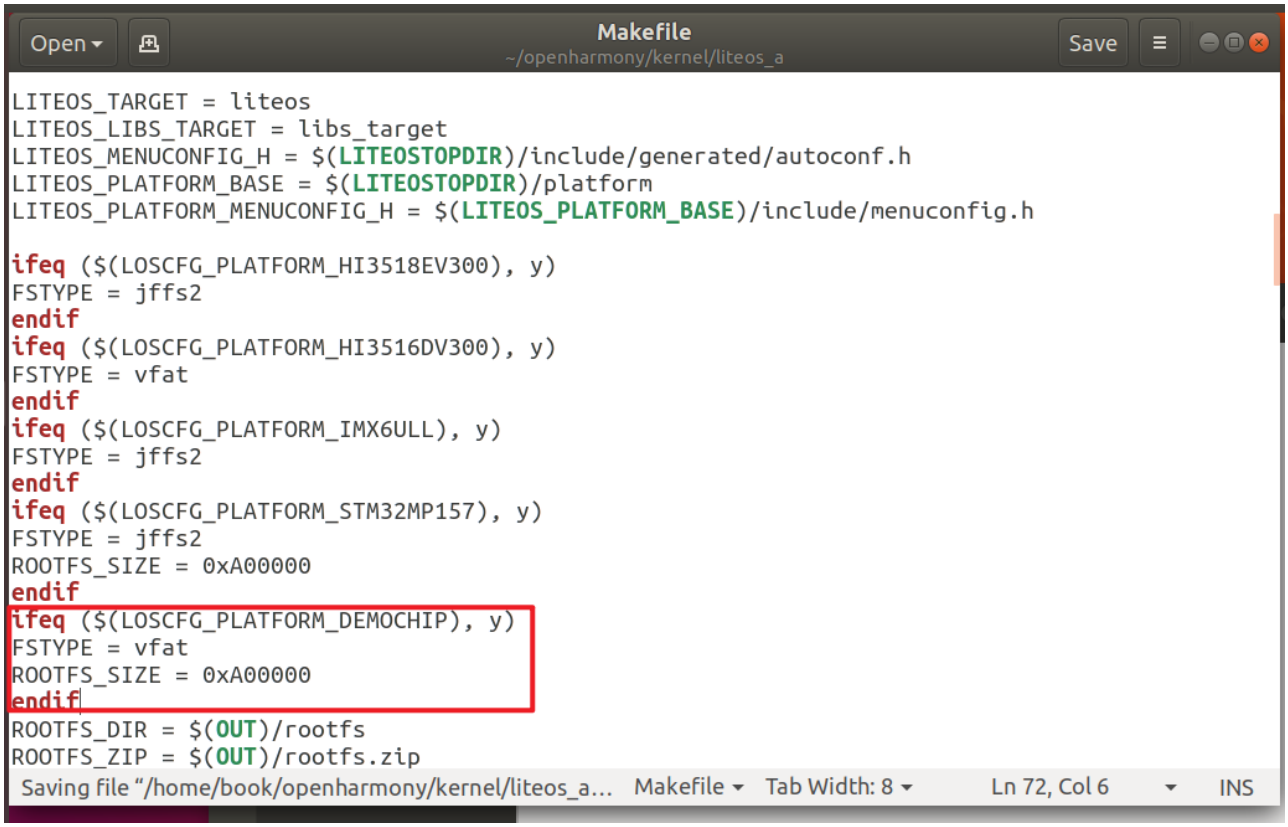



```
else ifeq ($(LOSCFG_PLATFORM_DEMOCHIP),y)
    PLATFORM_INCLUDE += -I
$(LITEOSTOPDIR)/../../vendor/democom/demochip/board/include
```

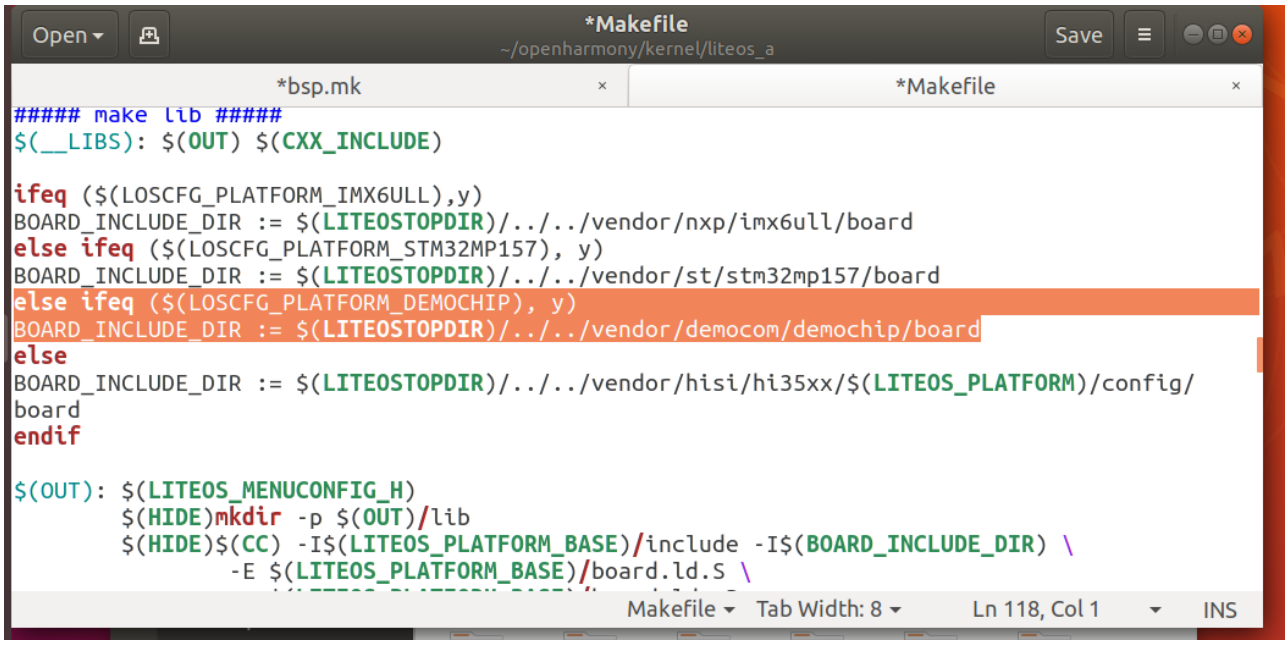


3.6 修改 openharmony/kernel/liteos_a/Makefile 文件

```
ifeq ($(LOSCFG_PLATFORM_DEMOCHIP), y)
FSTYPE = vfat
ROOTFS_SIZE = 0xA00000
endif
```

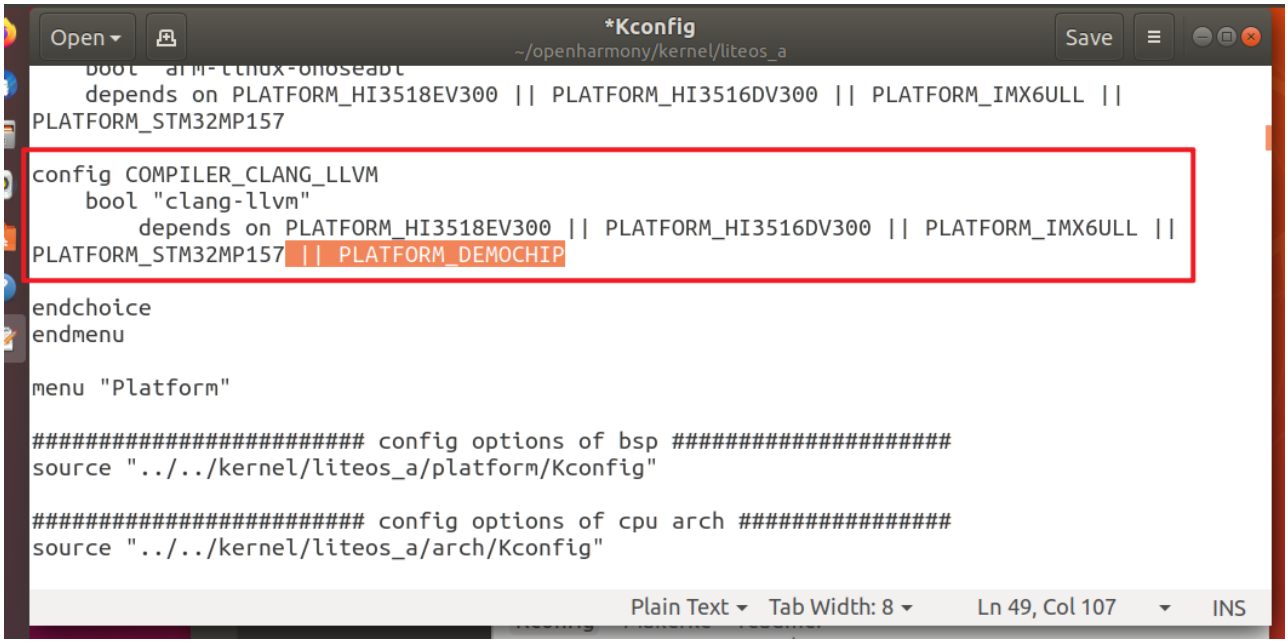



```
else ifeq ($(LOSCFG_PLATFORM_DEMOCHIP), y)
BOARD_INCLUDE_DIR := $(LITEOSTOPDIR)/../../vendor/democom/demochip/board
```

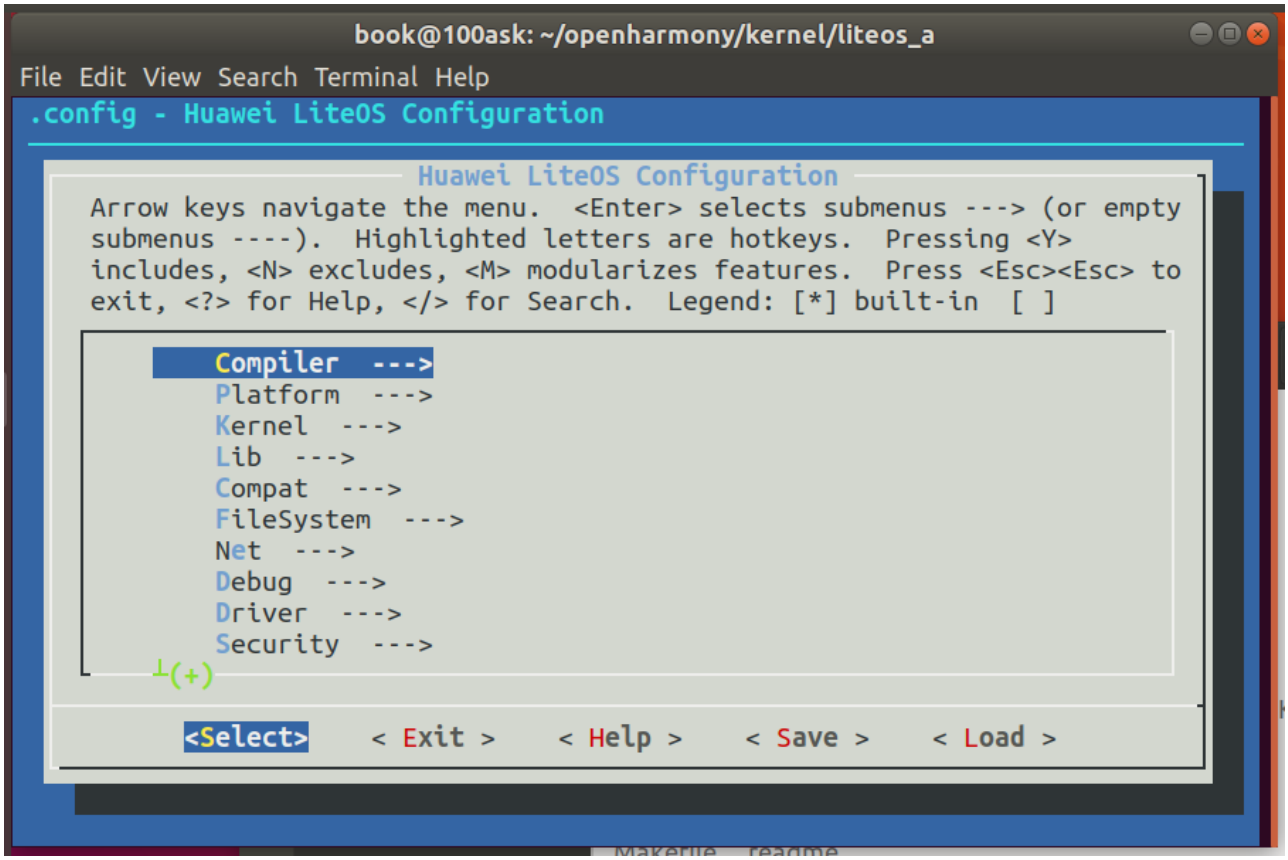


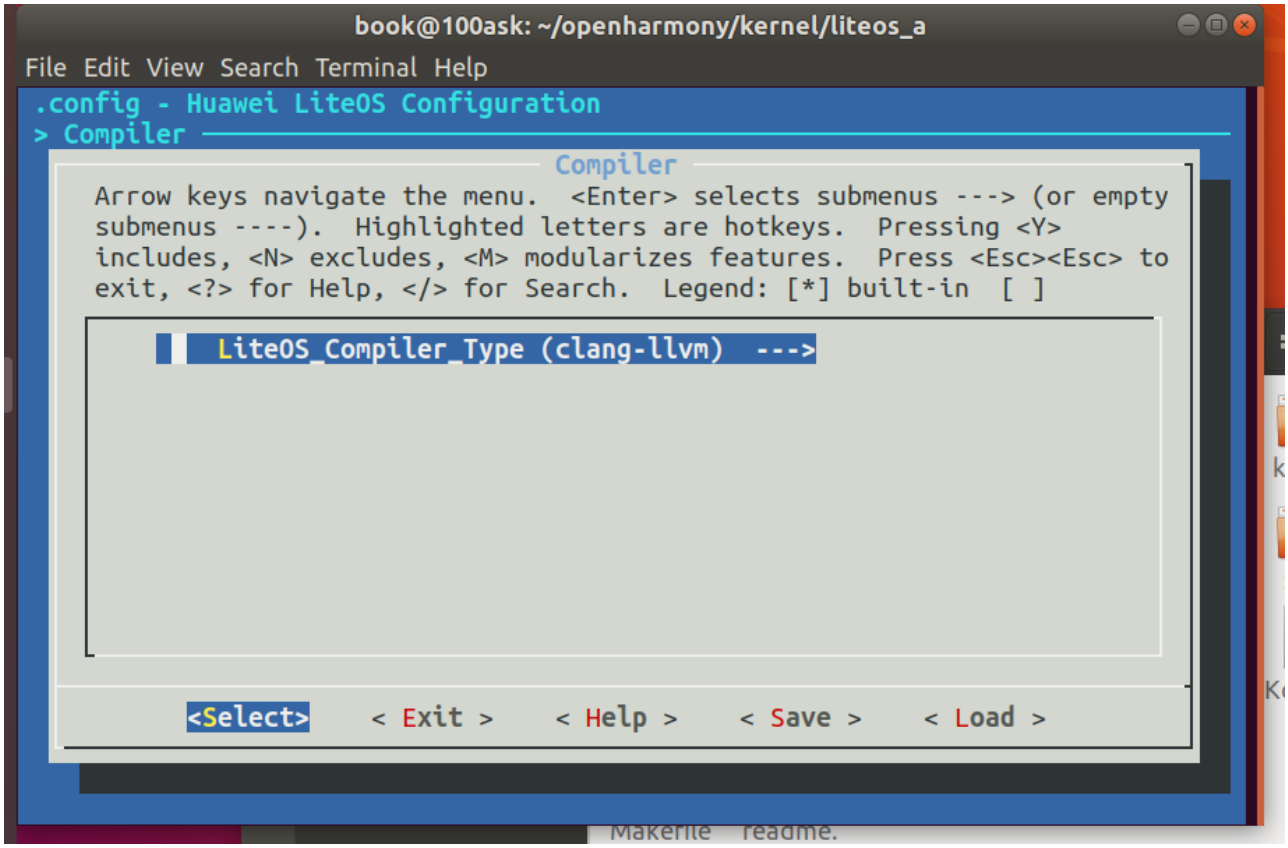
3.7 修改 openharmony/kernel/liteos_a/Kconfig 文件

加入 `|| PLATFORM_DEMOCHIP`



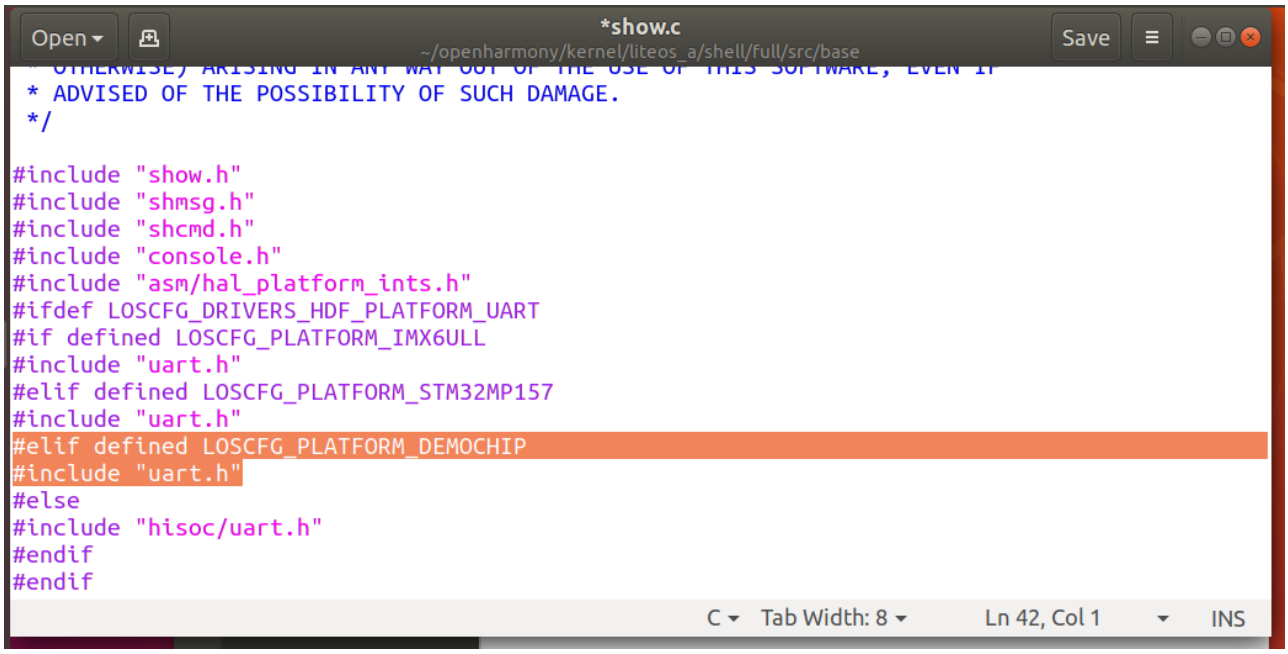
进入menuconfig，加入编译器





3.8 修改 openharmony/kernel/liteos_a/shell/full/src/base/show.c 文件

```
#elif defined LOSCFG_PLATFORM_DEMOCHIP
#include "uart.h"
```



3.9 修改 openharmony/kernel/liteos_a/shell/full/src/base/shcmd.c 文件

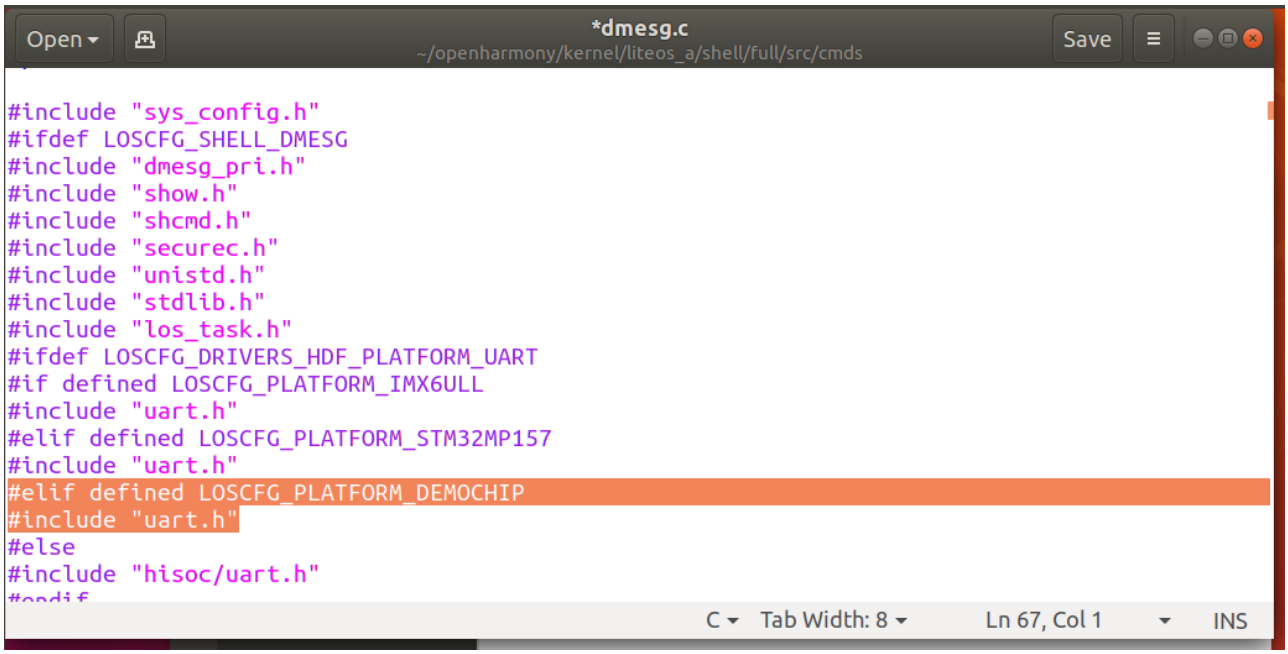
```
#elif defined LOSCFG_PLATFORM_DEMOCHIP
#include "uart.h"
```

3.10 修改 openharmony/kernel/liteos_a/shell/full/src/base/shmsg.c 文件

```
#elif defined LOSCFG_PLATFORM_DEMOCHIP
#include "uart.h"
```

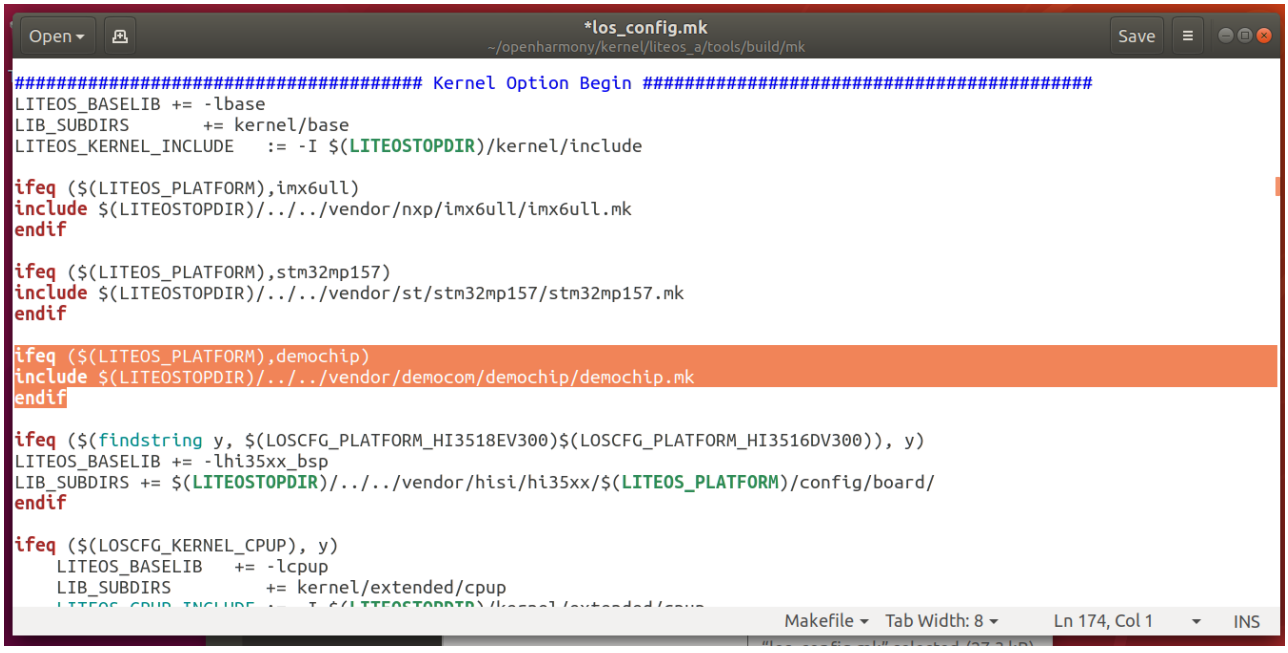
3.11 修改 openharmony/kernel/liteos_a/shell/full/src/cmds/dmesg.c 文件

```
#elif defined LOSCFG_PLATFORM_DEMOCHIP
#include "uart.h"
```



3.12 修改 openharmony/kernel/liteos_a/tools/build/mk/los_config.mk 文件

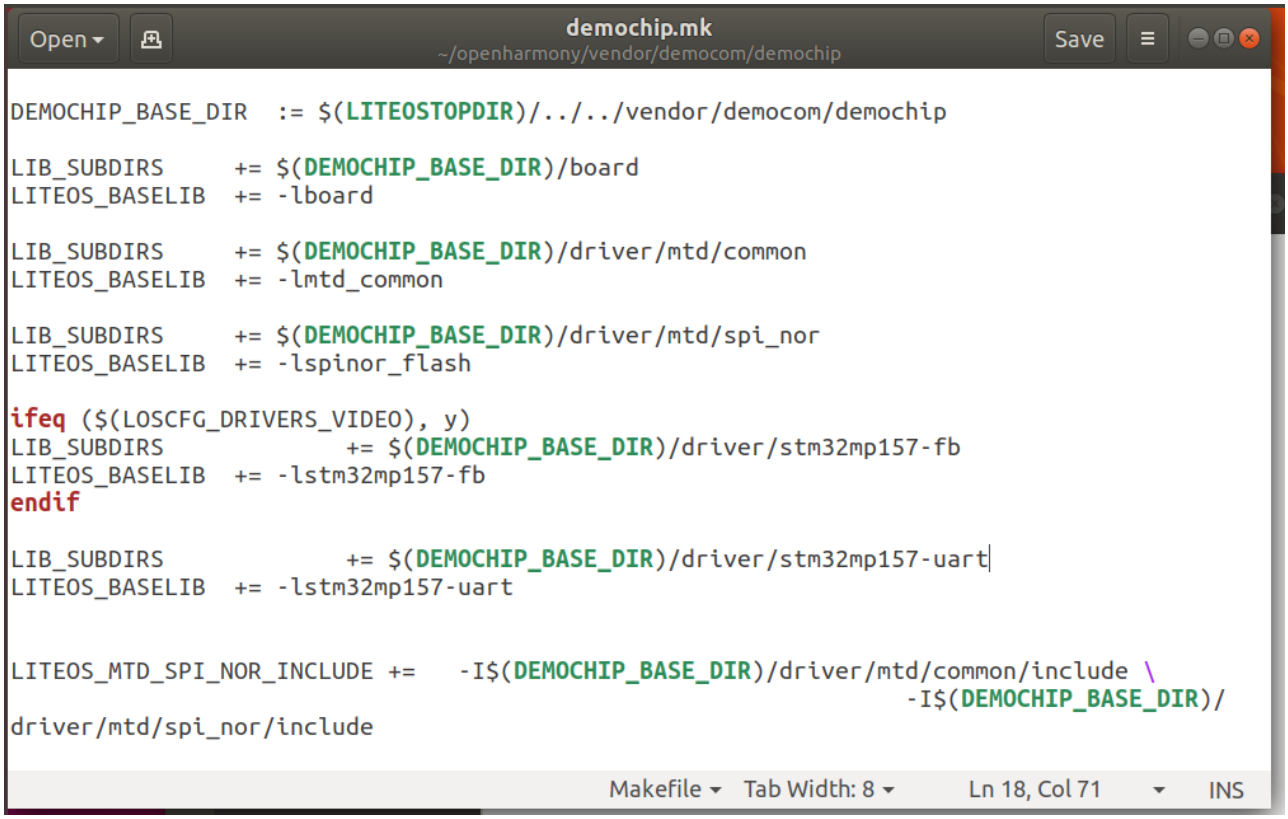
```
ifeq ($(LITEOS_PLATFORM),demochip)
include $(LITEOSTOPDIR)/../../vendor/democom/demochip/demochip.mk
endif
```



3.13 修改 openharmony/vendor/democom/demochip/demochip.mk 文件

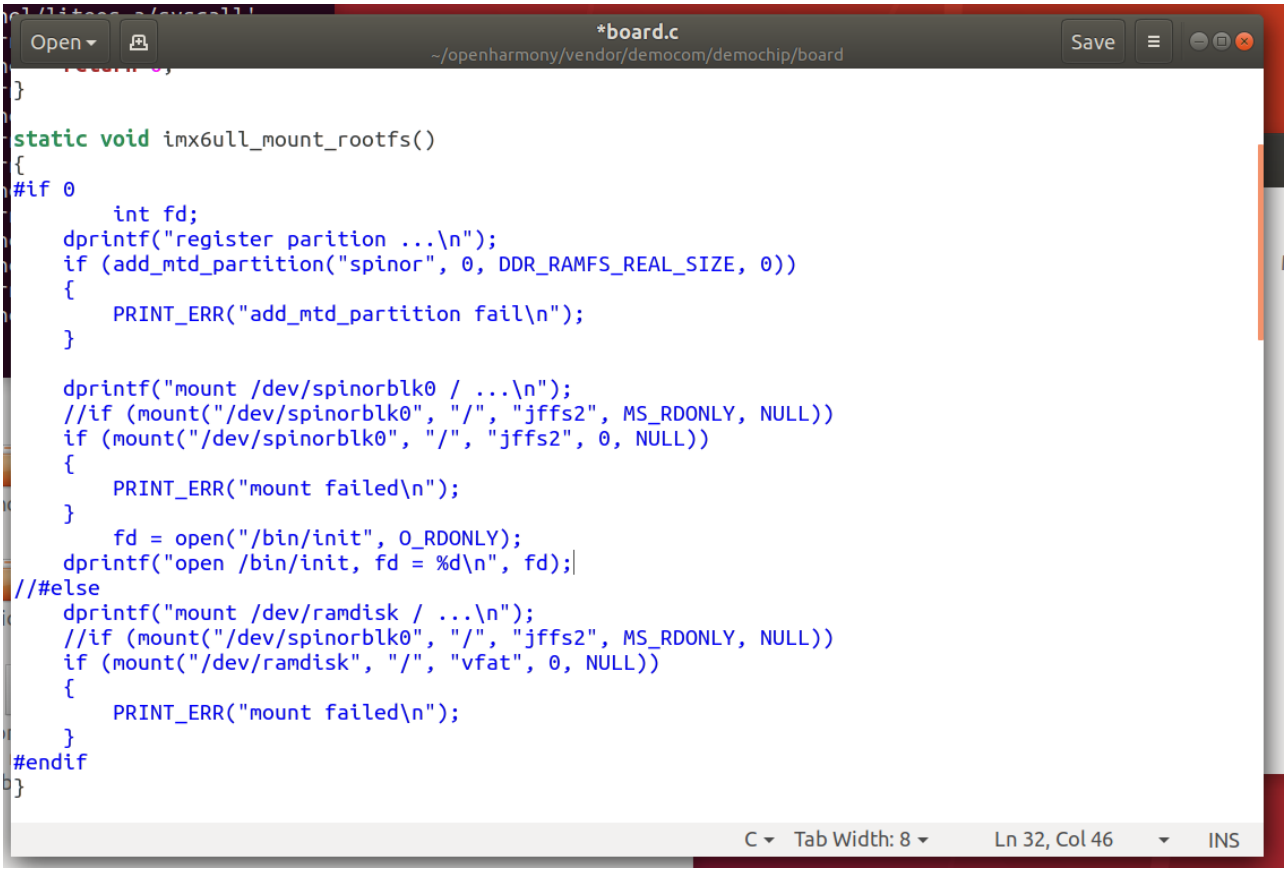
修改第一行路径为`DEMOCHIP_BASE_DIR := $(LITEOSTOPDIR)/../../vendor/democom/demochip`

修改所有`STM32MP157_BASE_DIR` 为 `DEMOCHIP_BASE_DIR`



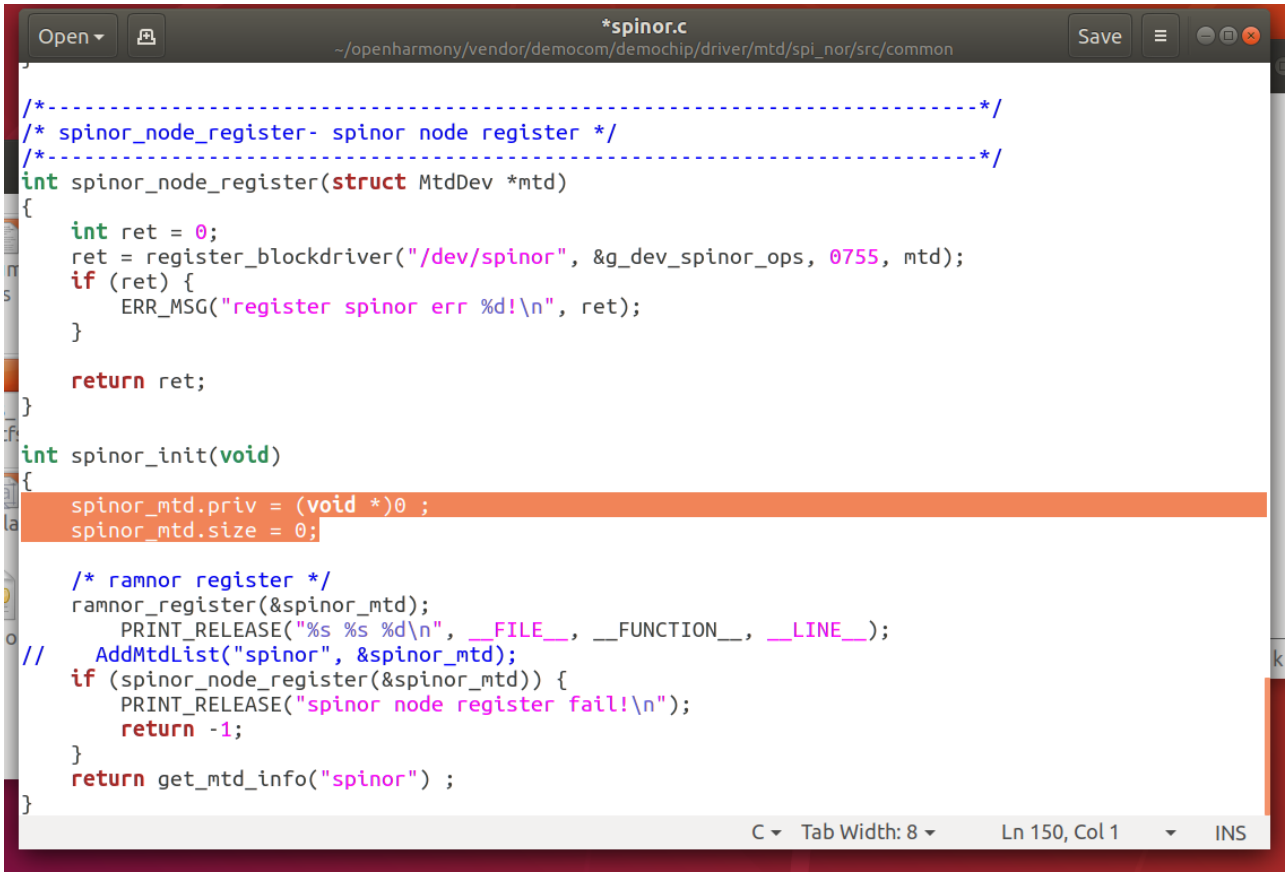
3.14 修改 openharmony/vendor/democom/demochip/board/board.c 文档

将第一个#if 1修改为#if 0，并将#if 与 #endif之间的else注释掉



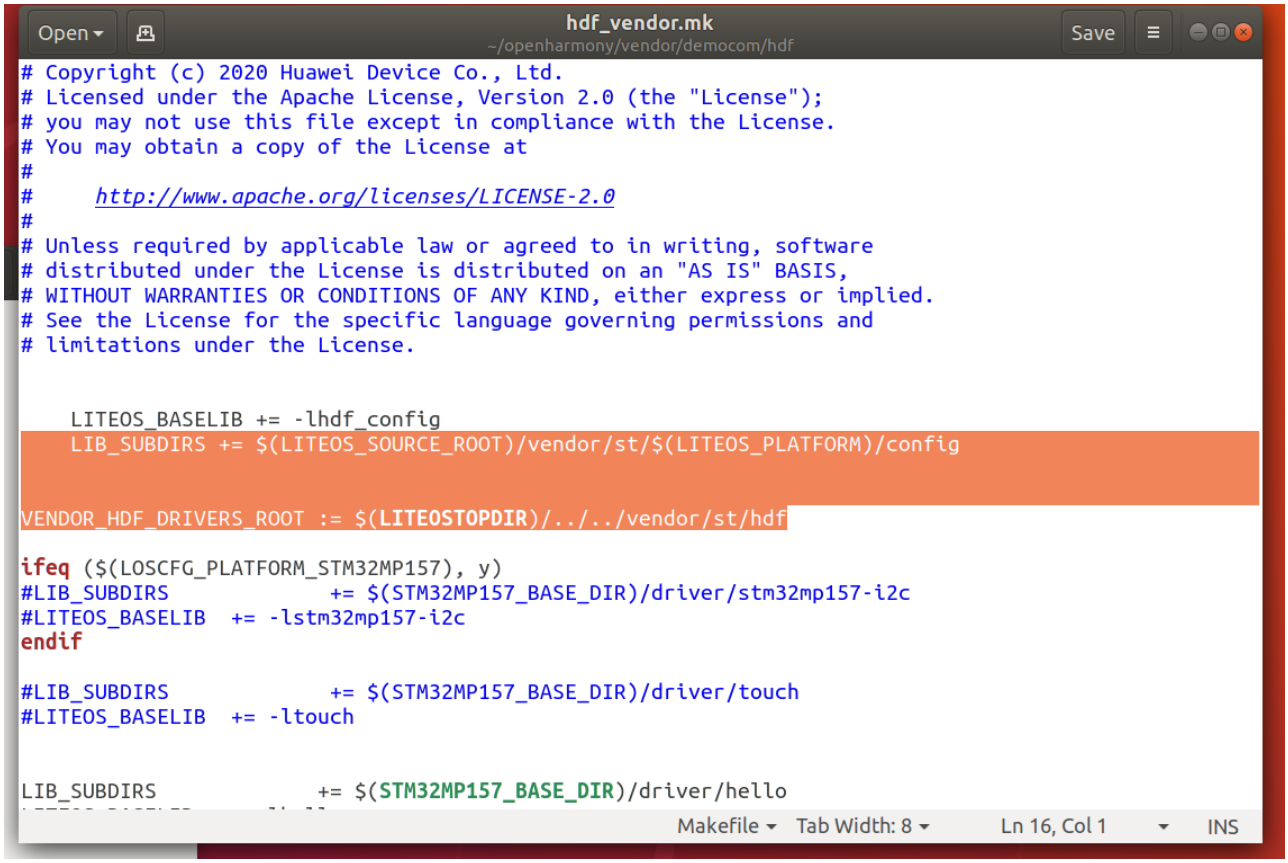
3.15 修改openharmony/vendor/democom/demochip/driver/mtd/spi_nor/src/common/spinor.c文档

```
spinor_mtd.priv = (void *)0 ;
spinor_mtd.size = 0;
```



3.16 修改openharmony/vendor/democom/hdf/hdf_vendor.mk文档

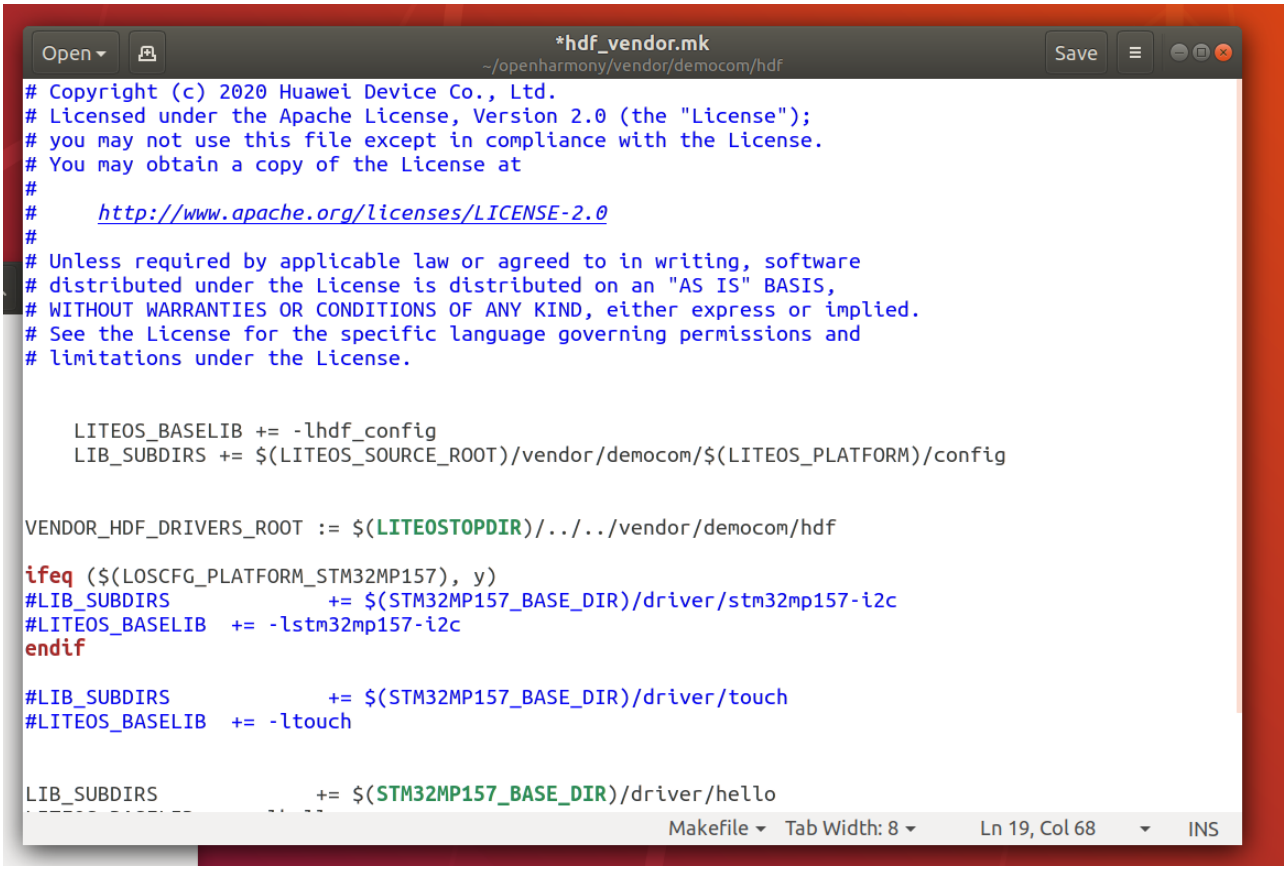
第二行和第三行中出现的路径



修改为

```
LIB_SUBDIRS += $(LITEOS_SOURCE_ROOT)/vendor/democom/$(LITEOS_PLATFORM)/config

VENDOR_HDF_DRIVERS_ROOT := $(LITEOSTOPDIR)/../../vendor/democom/hdf
```



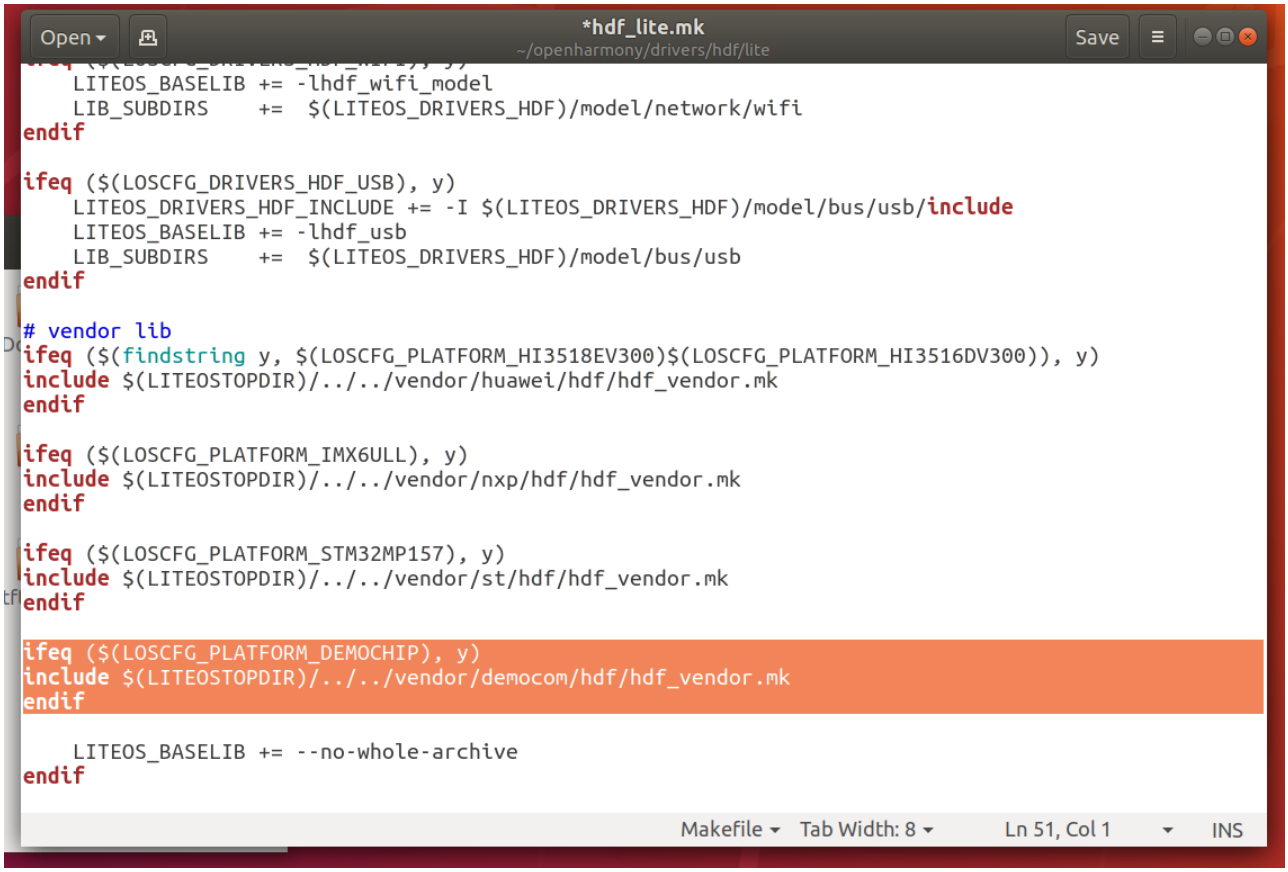
修改

```
LIB_SUBDIRS             += $(DEMOCHIP_BASE_DIR)/driver/hello
```

```
LIB_SUBDIRS             += $(DEMOCHIP_BASE_DIR)/driver/hello
LITEOS_BASELIB  += -lhello
```

3.17 修改openharmony/drvivers/hdf/lite/hdf_lite.mk文档

```
ifeq ($(LOSCFG_PLATFORM_DEMOCHIP), y)
include $(LITEOSTOPDIR)/../../vendor/democom/hdf/hdf_vendor.mk
endif
```

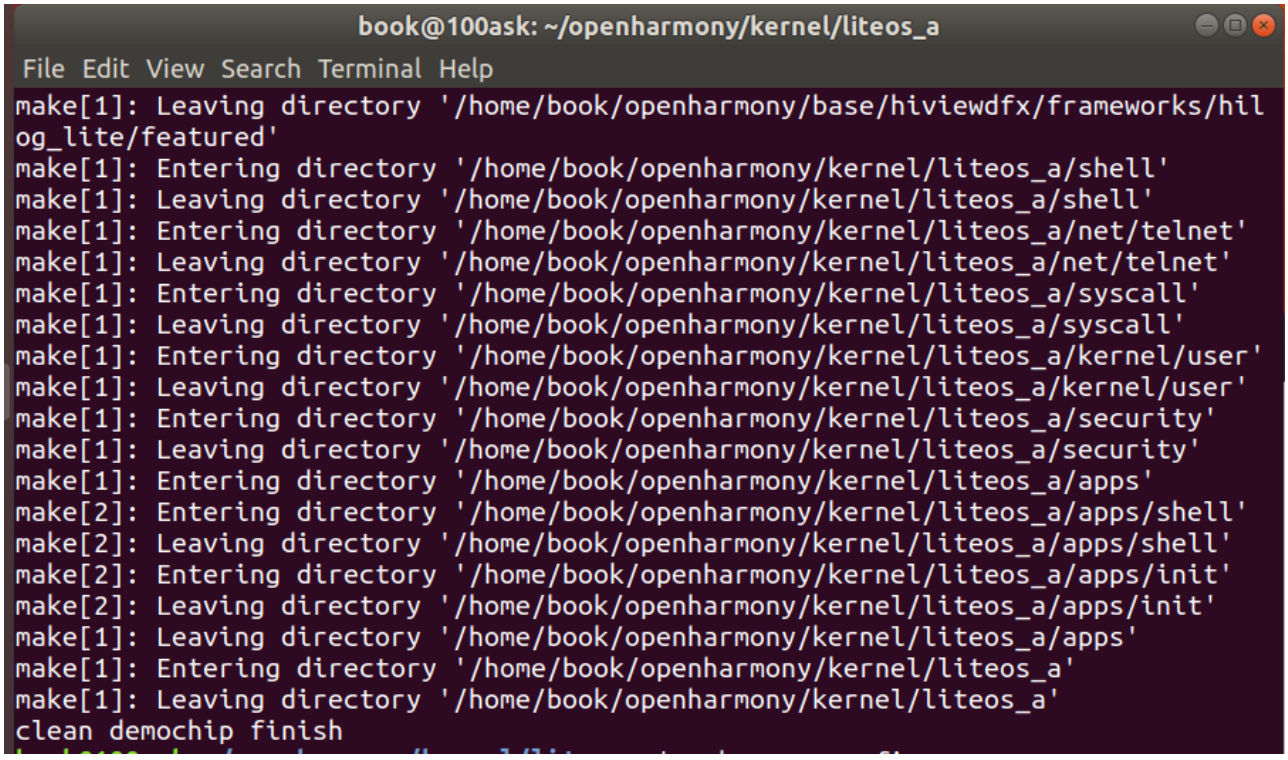


3.18 修改 openharmony/vendor/democom/demochip/driver/stm32mp157-fb/stm32mp157_lcd.c

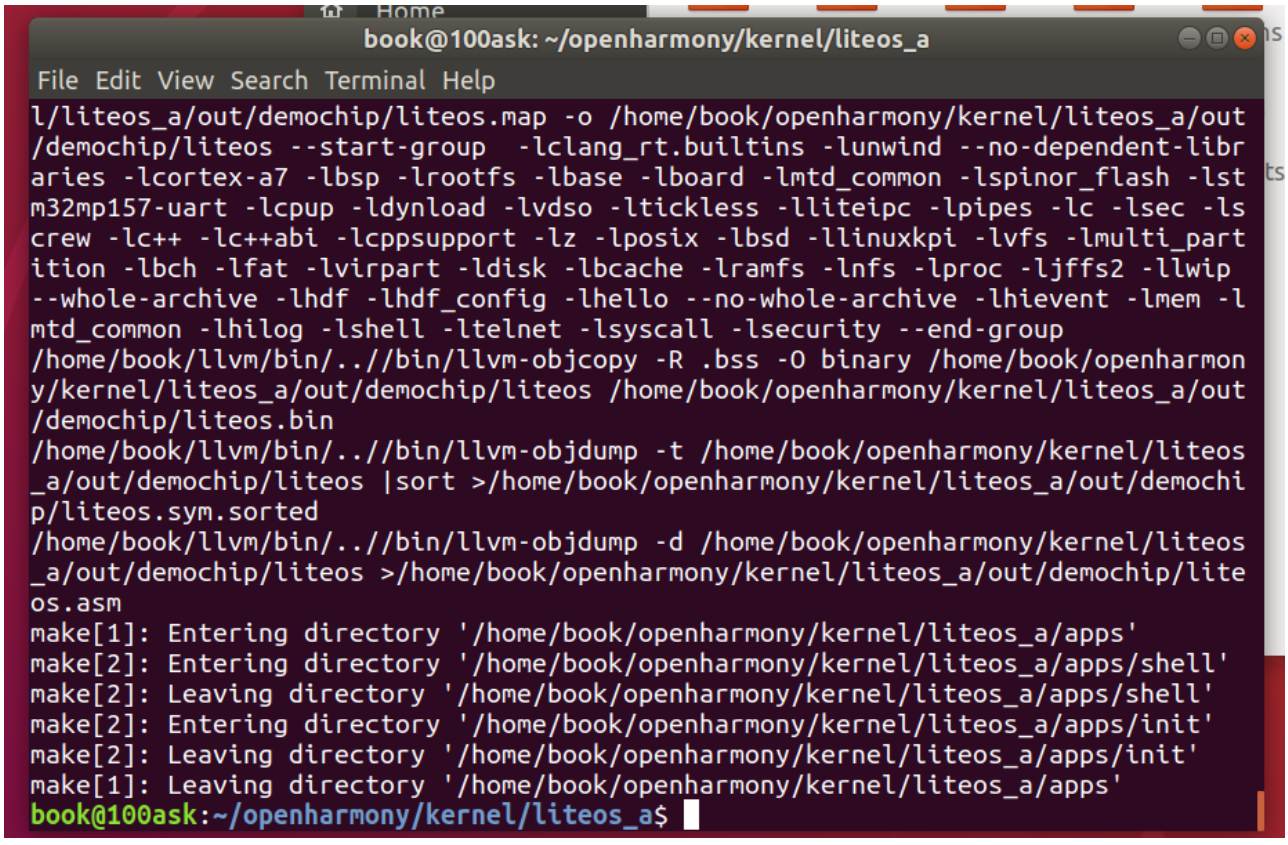
注释

3.19 编译单板

make clean



make -j 8

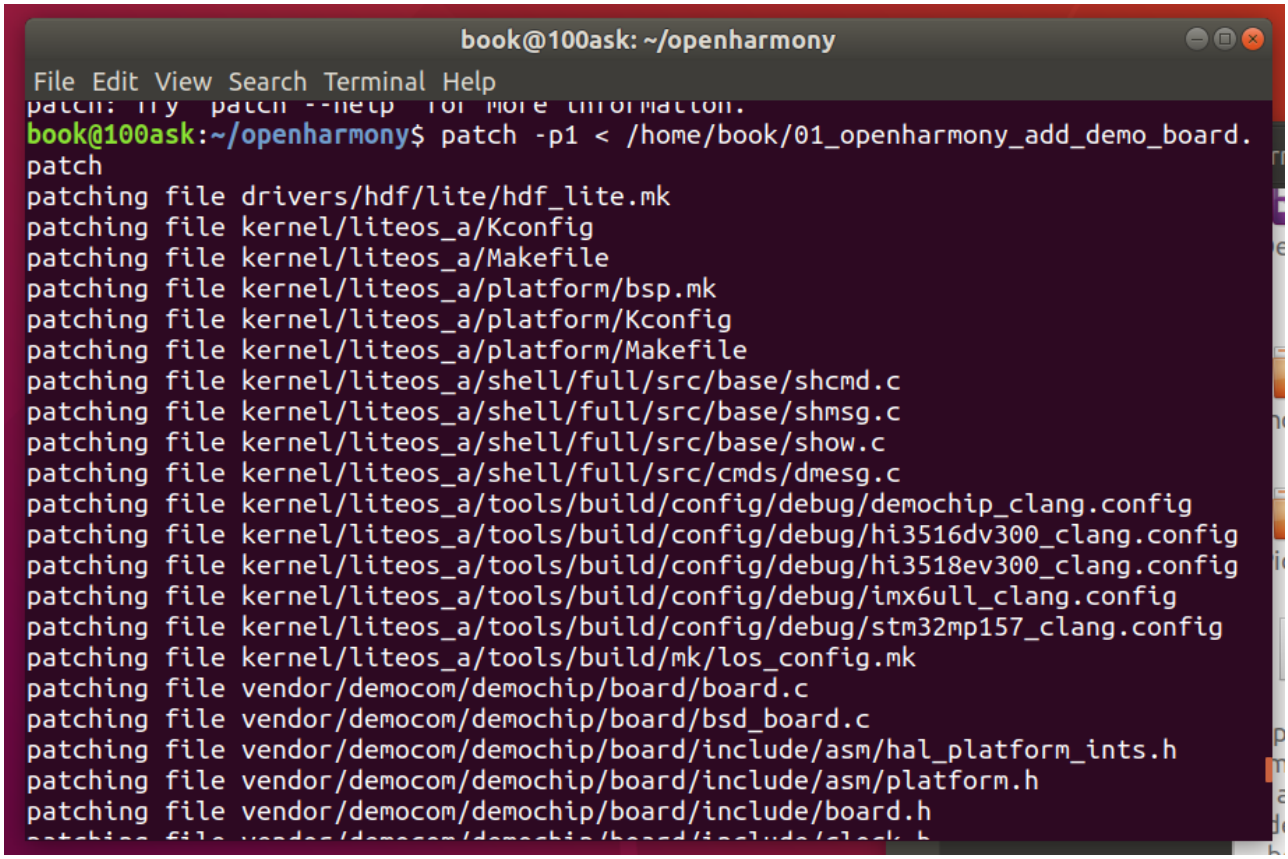


至此实验完成。

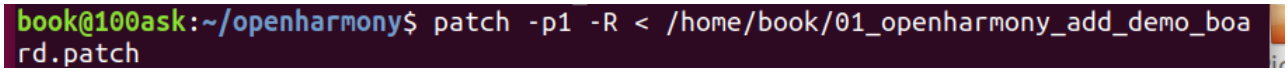
4 问题和解决方法

4.1 先直接打了补丁包完成了操作

由于v1.2.patch已经打过，只需打补丁：`01_openharmony_add_demo_board.patch` 即可完成实验

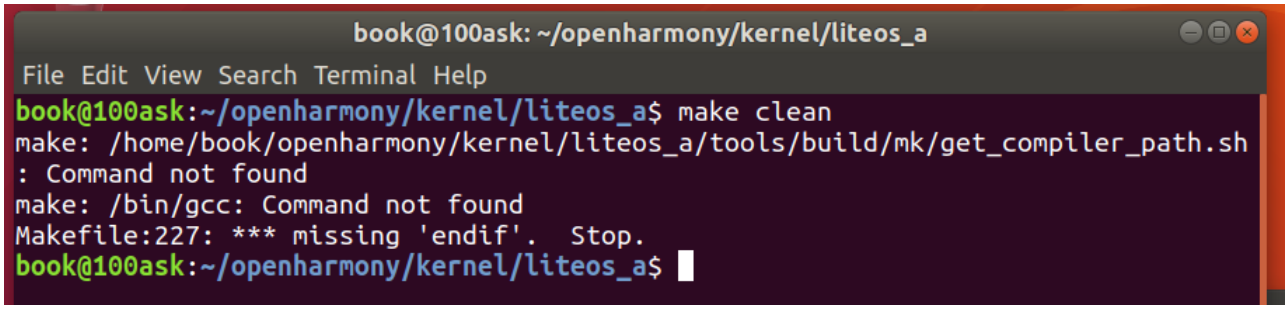


这时需要卸载补丁包：patch -p(n) -R < [补丁包路径] patch_name

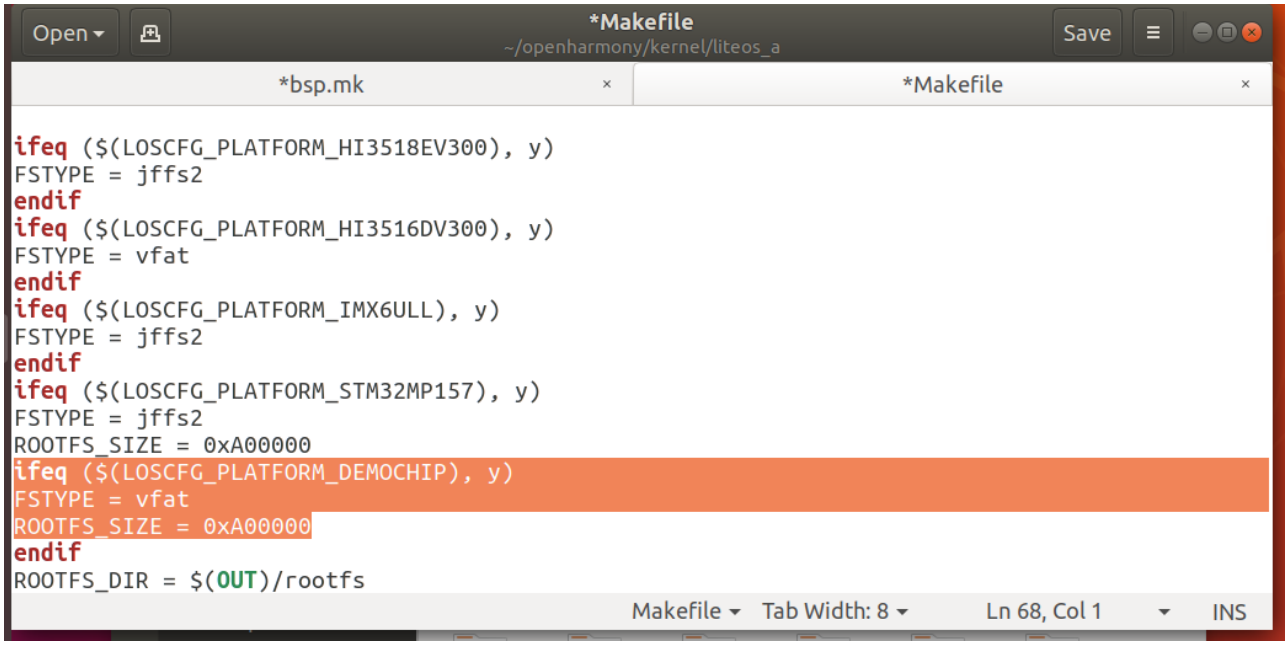


把修改还原，然后自己操作一遍

4.2 出现问题



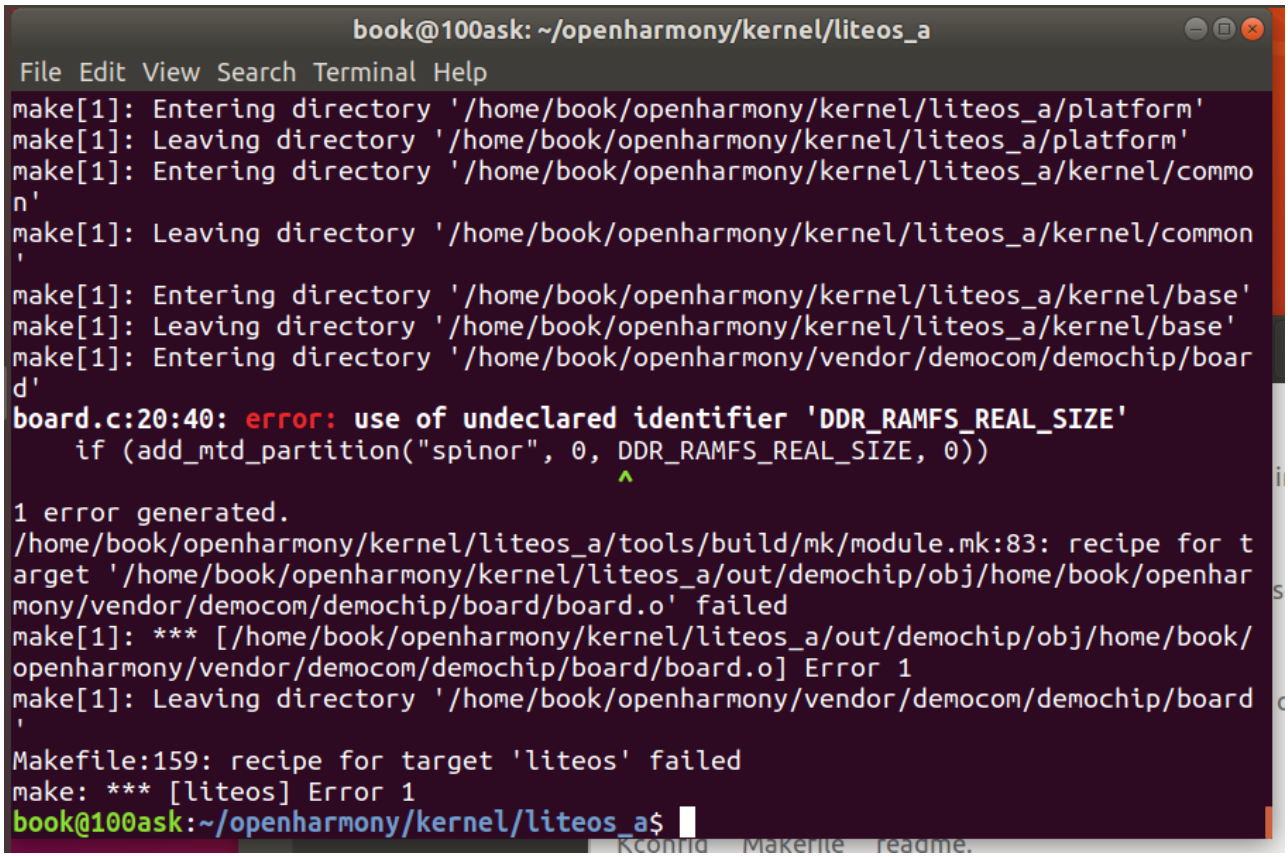
原因：步骤3.6漏了endif



3.6 修改 openharmony/kernel/liteos_a/Makefile 文件

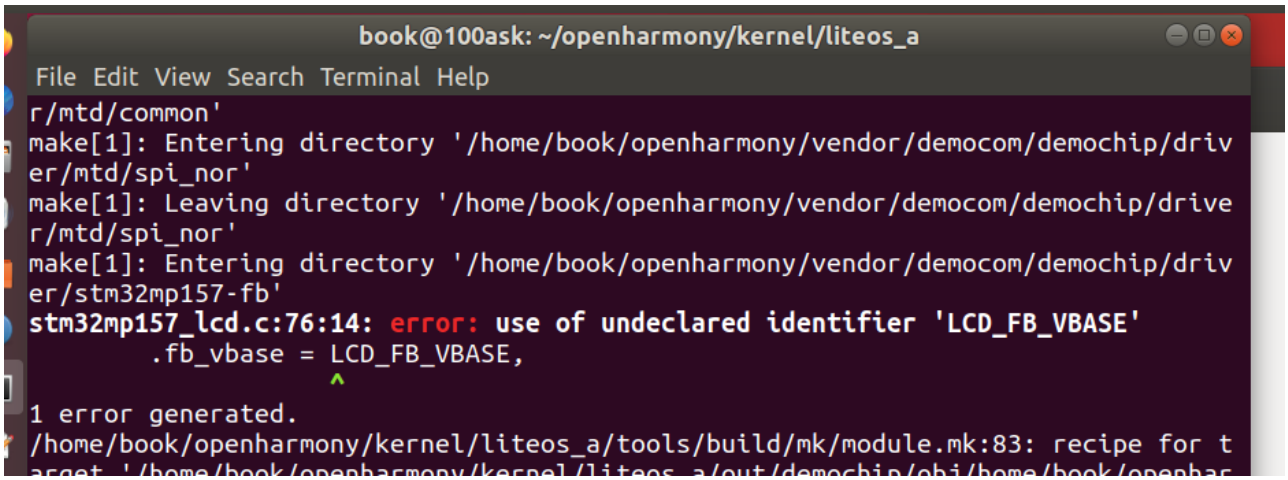
```
ifeq ($(LOSCFG_PLATFORM_DEMOCHIP), y)
FSTYPE = vfat
ROOTFS_SIZE = 0xA00000
endif
```

4.3 出现问题



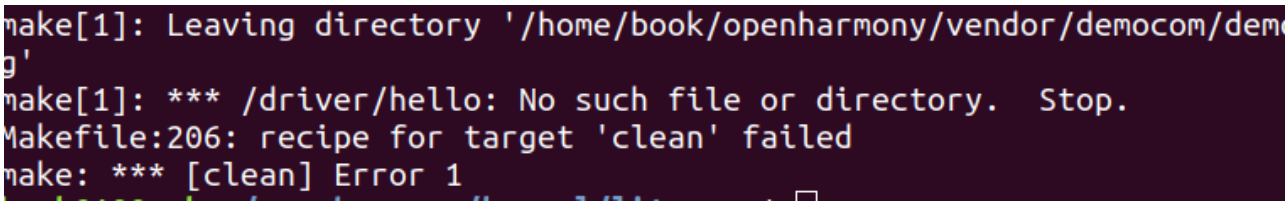
没有接着修改board.c文件，正确修改后解决

4.4 出现问题

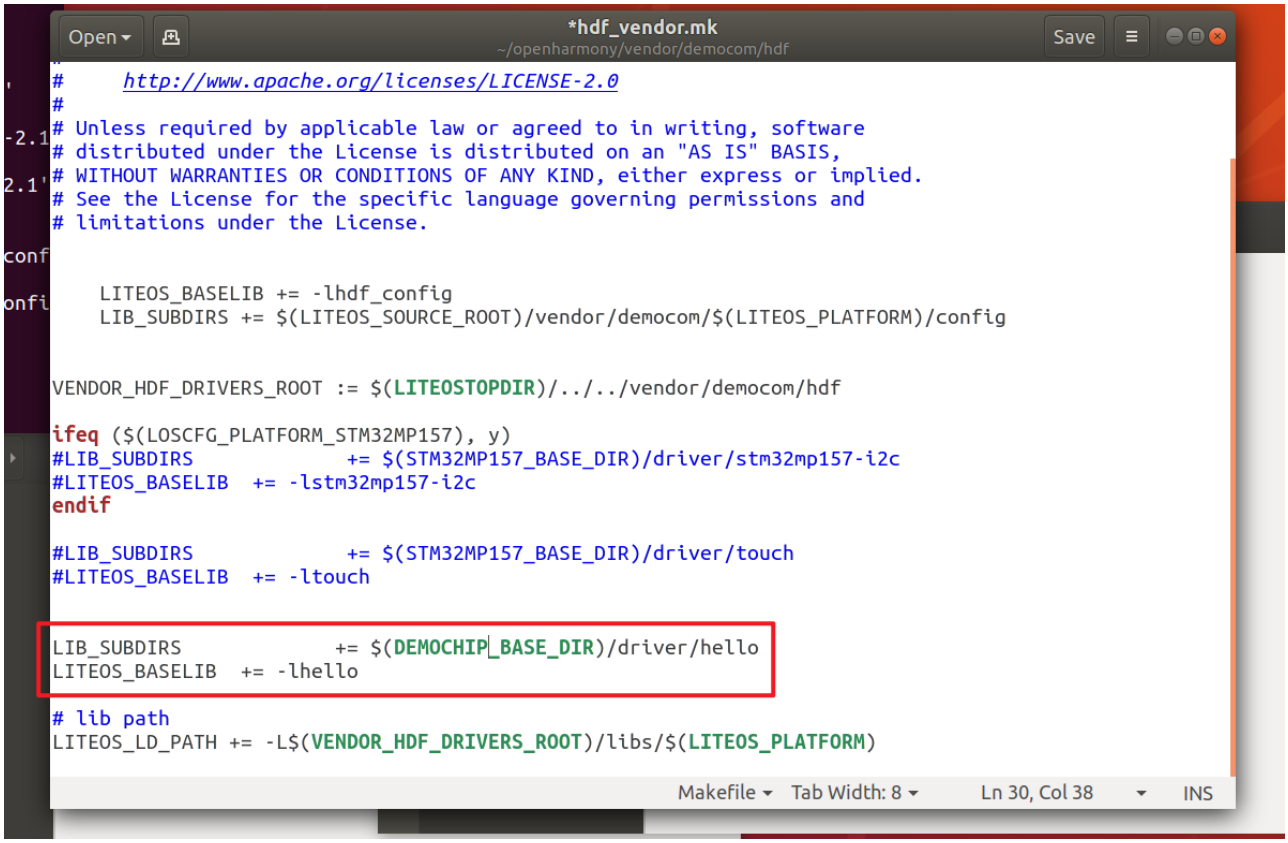


原因：没有注释完毕

4.5 出现问题



修改路径



5 实验体会

在本次实验中，我通过打补丁和在原有文件中添加新代码，再进行编译后，完成了增加一个新单板的任务。

内核启动流程可以分为4步骤:

- 启动
  - 使用汇编代码编写，涉及非常底层的设置，比如CPU设置、代码重定位等等
  - 地址映射也在这里实现
  - 它最终会调用main函数
- main函数
  - 以后的代码，基本都是使用C语言编写了
  - 主要工作是：调用OsMain进行各类初始化、最终会启动用户程序
- OsMain函数
  - 进行操作系统层面的初始化，比如异常初始化、任务初始化、IPC初始化
  - 调用SystemInit
- SystemInit
  - 偏向于应用程序的初始化
  - 挂载根文档系统
  - 启动第一个用户进程

6 参考文献

[1] HarmonyOS内核移植——添加单板