实验8 鸿蒙 LiteOS-a 内核移植

——根文件系统内容与制作

《实用操作系统》实验报告　22920212204392 黄勖

1　实验环境

Windows10 21H2、Vmware Workstation Pro 16、Ubuntu 18.04

配置了相关的软件。

2　实验目的

学习并实现根文件系统的内容和制作

学习init进程

3　实验步骤与内容

3.1　根文件系统的内容

看看一个简单的程序：

```
#include <stdio.h>
int main(int argc, char **argv)
{
    printf("hello, world!\n");
    return 0;
}
```

可以编译出一个APP：hello。
有几个问题要考虑：

- printf不是我们实现的，它在哪？
- hello放在板子上后，怎么启动它？能否自动启动？

解决这几个问题后，就可以知道根文件系统的内容了：

- /lib：库，比如printf函数就是在库里的
- /bin：APP，hello这样的程序放在/bin或/usr/bin这些目录里
- 至少有这些APP：
  - init：内核启动的第一个APP，它会去启动其他APP，比如shell
  - shell：也是一个APP，可以让我们输入各类命令

  - 我们自己的APP：比如hello

- /etc：想自动启动APP怎么办？应该有配置文件，init进程根据配置文件去启动其他APP
  - 比如/etc/init.cfg
- /dev：设备节点，在Liteos-a中不需要我们自己创建

3.1.1 Makefile分析

在`kernnel/liteos_a`目录执行`make help`：

```
book@100ask:~/openharmony/kernel/liteos_a$ make help
------------------------------------------------------------
1.====make help:    get help infomation of make
2.====make:         make a debug version based the .config
3.====make debug:   make a debug version based the .config
4.====make release: make a release version for all platform
5.====make release PLATFORM=xxx:  make a release version only for platform xxx
6.====make rootfsdir: make a original rootfs dir
7.====make rootfs FSTYPE=***: make a original rootfs img
8.====make test: make the testsuits_app and put it into the rootfs dir
9.====make test_apps FSTYPE=***: make a rootfs img with the testsuits_app in it
xxx should be one of (hi3516cv300 hi3516ev200 hi3556av100/cortex-a53_aarch32
hi3559av100/cortex-a53_aarch64)
*** should be one of (jffs2)
------------------------------------------------------------
```

可以知道：执行`make rootfs`可以制作根文件系统。
分析Makefile确定它的制作过程。

3.1.1.1 ROOTFS目标：

```
$(ROOTFS): $(ROOTFSDIR)
    $(HIDE)$(shell $(LITEOSTOPDIR)/tools/scripts/make_rootfs/rootfsimg.sh
$(ROOTFS_DIR) $(FSTYPE)  ${ROOTFS_SIZE})
    $(HIDE)cd $(ROOTFS_DIR)/.. && zip -r $(ROOTFS_ZIP) $(ROOTFS)
ifneq ($(OUT), $(LITEOS_TARGET_DIR))
    $(HIDE)mv $(ROOTFS_DIR) $(LITEOS_TARGET_DIR)rootfs
endif

$(ROOTFSDIR): prepare $(APPS)
    $(HIDE)$(MAKE) clean -C apps
    $(HIDE)$(shell $(LITEOSTOPDIR)/tools/scripts/make_rootfs/rootfsdir.sh
$(OUT)/bin $(OUT)/musl $(ROOTFS_DIR))
ifneq ($(VERSION),)
    $(HIDE)$(shell $(LITEOSTOPDIR)/tools/scripts/make_rootfs/releaseinfo.sh
"$(VERSION)" $(ROOTFS_DIR))
endif
```

```
prepare:
    $(HIDE)mkdir -p $(OUT)/musl
ifeq ($(LOSCFG_COMPILER_CLANG_LLVM), y)
    $(HIDE)cp -f
$(LITEOSTOPDIR)/../../prebuilts/lite/sysroot/usr/lib/$(LLVM_TARGET)/a7_softfp_n
eon-vfpv4/libc.so $(OUT)/musl
    $(HIDE)cp -f $(LITEOS_COMPILER_PATH)/lib/$(LLVM_TARGET)/c++/a7_softfp_neon-
vfpv4/libc++.so $(OUT)/musl
else
    $(HIDE)cp -f $(LITEOS_COMPILER_PATH)/target/usr/lib/libc.so $(OUT)/musl
    $(HIDE)cp -f $(LITEOS_COMPILER_PATH)/arm-linux-musleabi/lib/libstdc++.so.6
$(OUT)/musl
    $(HIDE)cp -f $(LITEOS_COMPILER_PATH)/arm-linux-musleabi/lib/libgcc_s.so.1
$(OUT)/musl
    $(STRIP) $(OUT)/musl/*
endif

$(APPS): $(LITEOS_TARGET)
    $(HIDE)$(MAKE) -C apps all
```

- ROOTFSDIR：看到 ROOTFS 依赖于 ROOTFSDTIR，ROOTFS 依赖于 prepare，ROOTFSDTIR 使用rootfsdir.sh创建一些目录
- prepare：创建输出目录，并且将 C 库和 C++库等复制到刚刚创建的目录 musl 里面，如复制libc.so、libc++.so
- APPS：编译应用程序，进入-C 这个目录，也就是进入apps目录执行`make all`

3.1.1.2 编译APP

有目录：`kernel/liteos_a/apps`，这个目录下有：

- module.mk：定义了

  ```
  APP_SUBDIRS += shell
  APP_SUBDIRS += init
  ```

- Makefile：

  ```
  $(APPS):
  ifneq ($(APP_SUBDIRS), )
          $(HIDE) for dir in $(APP_SUBDIRS); do $(MAKE) -C $$dir ; done
  endif
  ```

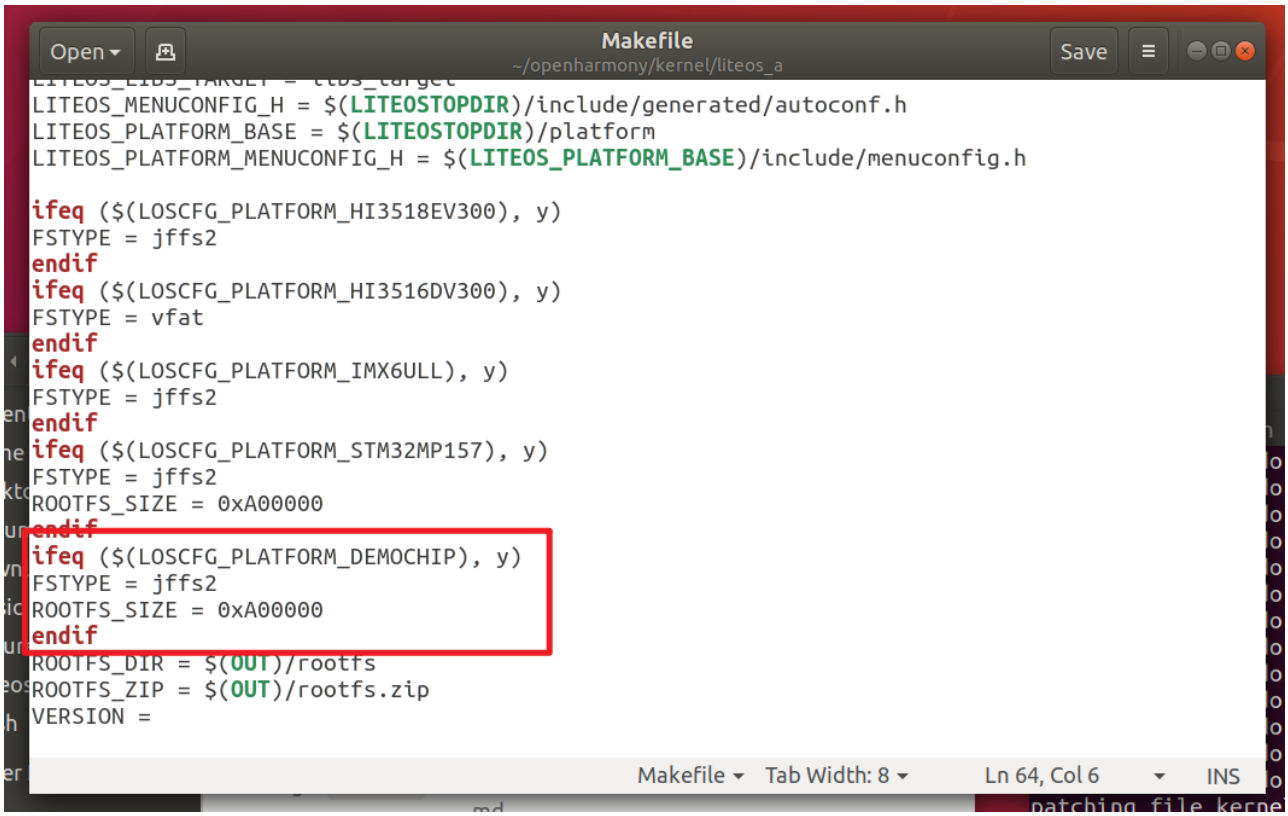
  就是再次进入shell、init目录，执行`make`命令，去变量shell程序、init程序。

3.1.2 根文件系统的制作

进入 openharmony/kernel/liteos_a/Makefile 修改

将 DEMOCHIP对应的 FSTYPE 改为 jffs2

```
@@ -67,7 +67,7 @@
 ROOTFS_SIZE = 0xA00000
 endif
 ifeq ($(LOSCFG_PLATFORM_DEMOCHIP), y)
-FSTYPE = vfat
+FSTYPE = jffs2
 ROOTFS_SIZE = 0xA00000
 endif
 ROOTFS_DIR = $(OUT)/rootfs
```

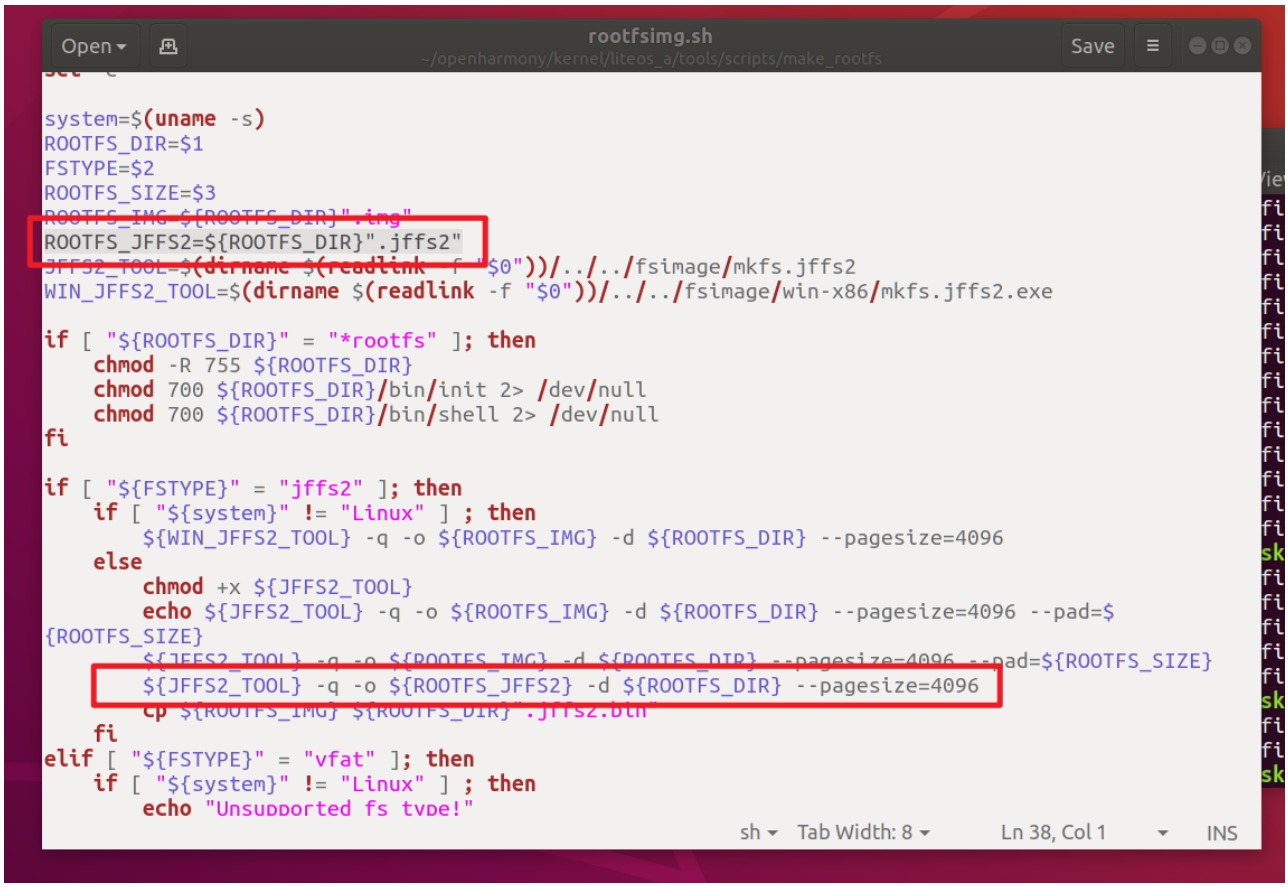


进入 openharmony/kernel/liteos_a/tools/scripts/make_rootfs/rootfsimg.sh 修改

新增 ROOTFS_JFFS2

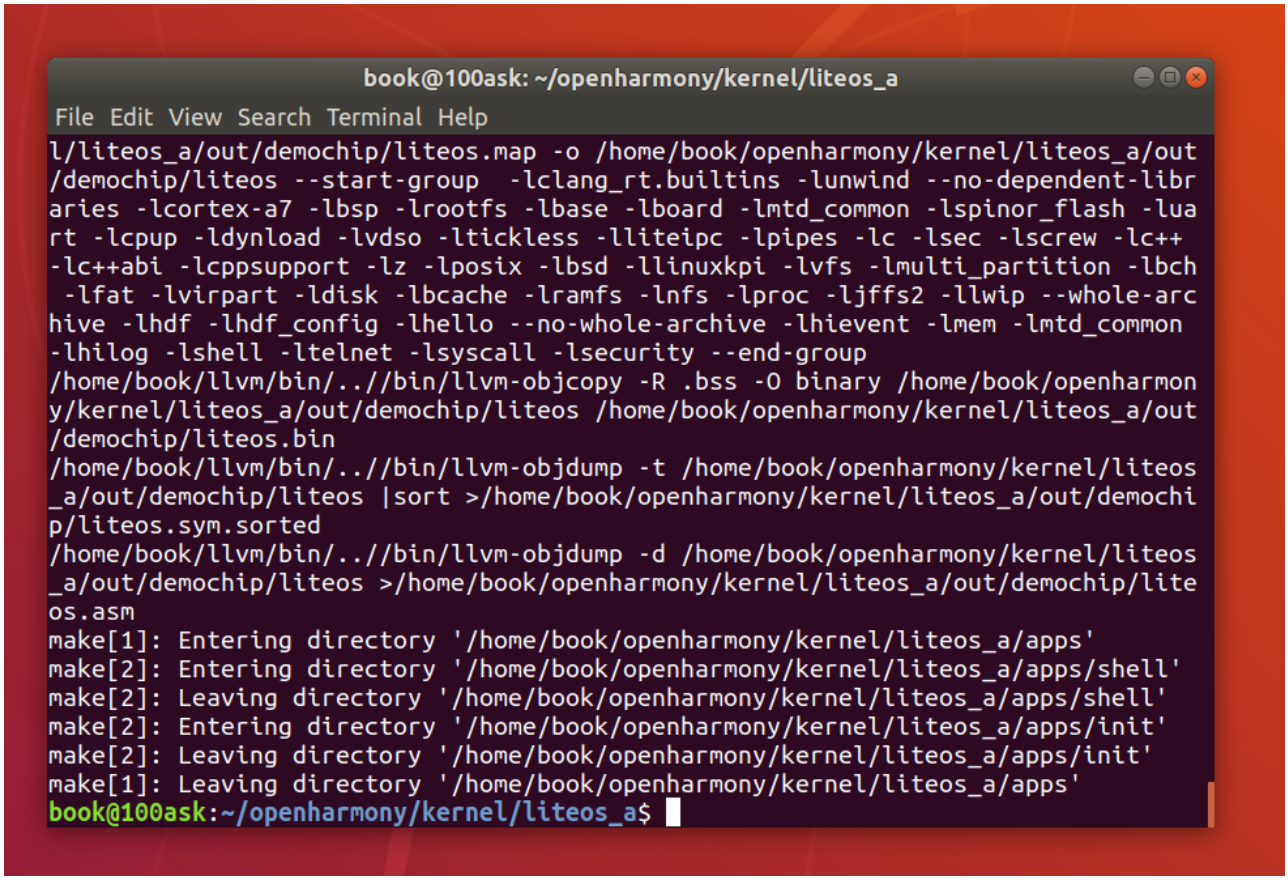
```
@@ -35,6 +35,7 @@
 FSTYPE=$2
 ROOTFS_SIZE=$3
 ROOTFS_IMG=${ROOTFS_DIR}".img"
+ROOTFS_JFFS2=${ROOTFS_DIR}".jffs2"
 JFFS2_TOOL=$(dirname $(readlink -f "$0"))/../../fsimage/mkfs.jffs2
 WIN_JFFS2_TOOL=$(dirname $(readlink -f "$0"))/../../fsimage/win-
x86/mkfs.jffs2.exe

@@ -51,7 +52,7 @@
         chmod +x ${JFFS2_TOOL}
         echo ${JFFS2_TOOL} -q -o ${ROOTFS_IMG} -d ${ROOTFS_DIR} --
pagesize=4096 --pad=${ROOTFS_SIZE}
         ${JFFS2_TOOL} -q -o ${ROOTFS_IMG} -d ${ROOTFS_DIR} --pagesize=4096 --
pad=${ROOTFS_SIZE}
```

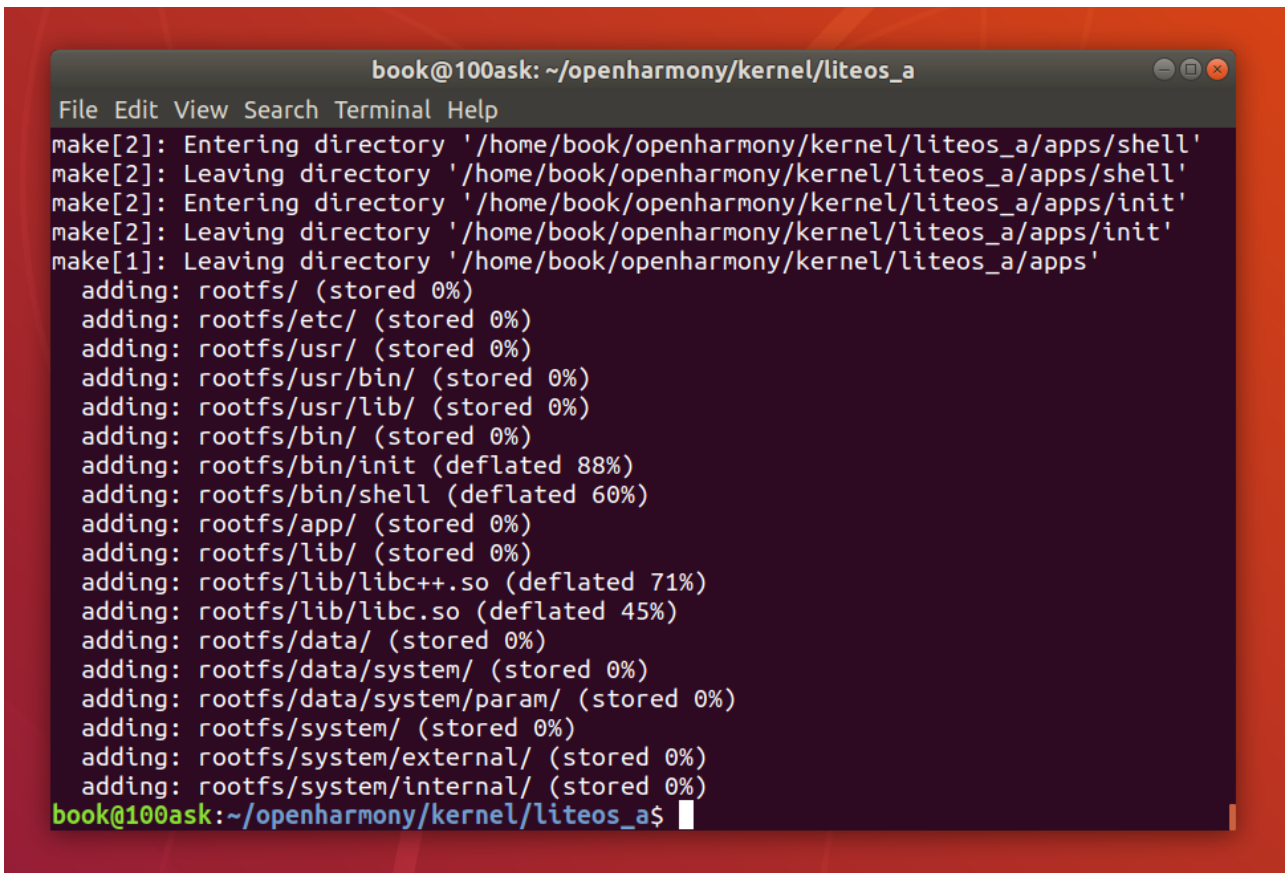
```
-    cp ${ROOTFS_IMG} ${ROOTFS_DIR}".jffs2"
+         ${JFFS2_TOOL} -q -o ${ROOTFS_JFFS2} -d ${ROOTFS_DIR} --pagesize=4096
     cp ${ROOTFS_IMG} ${ROOTFS_DIR}".jffs2.bin"
      fi
 elif [ "${FSTYPE}" = "vfat" ]; then
```



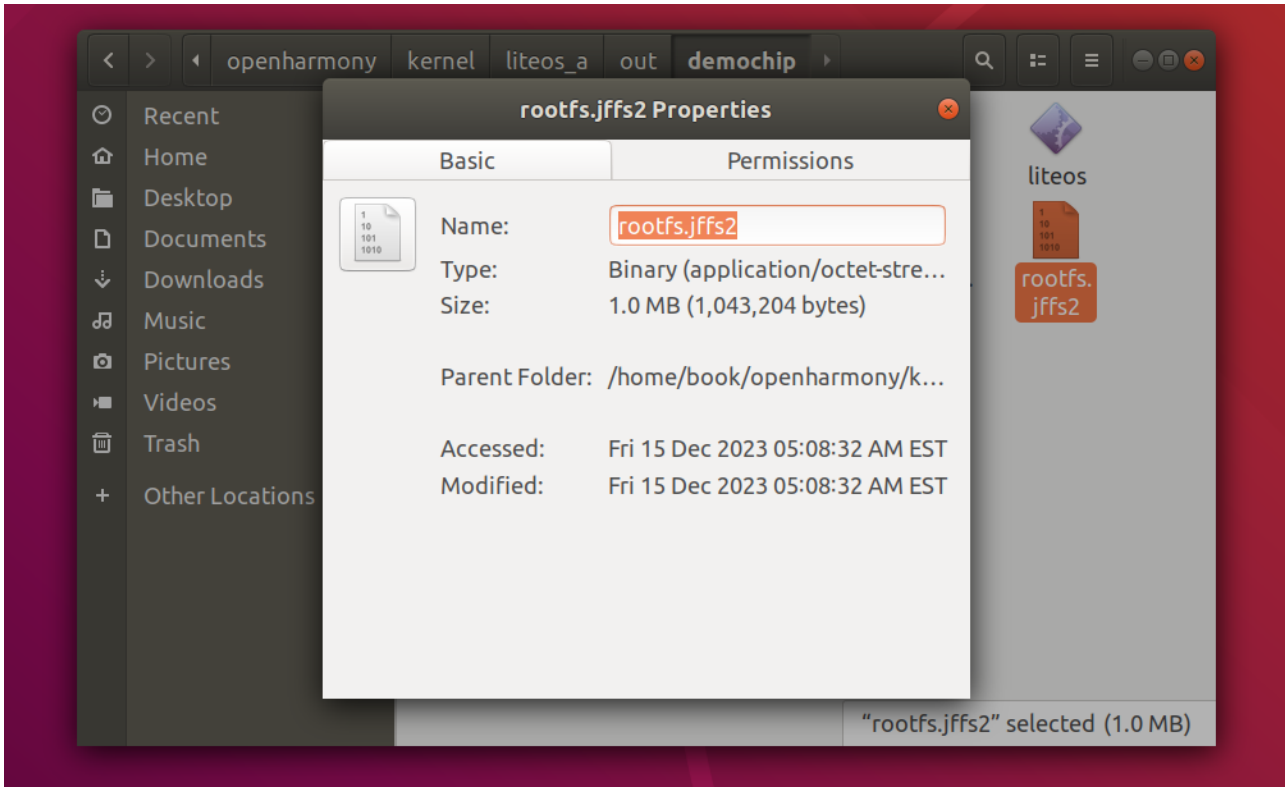
3.1.3 演示

```
cp tools/build/config/debug/demochip_clang.config .config
make clean
make
make rootfs
```

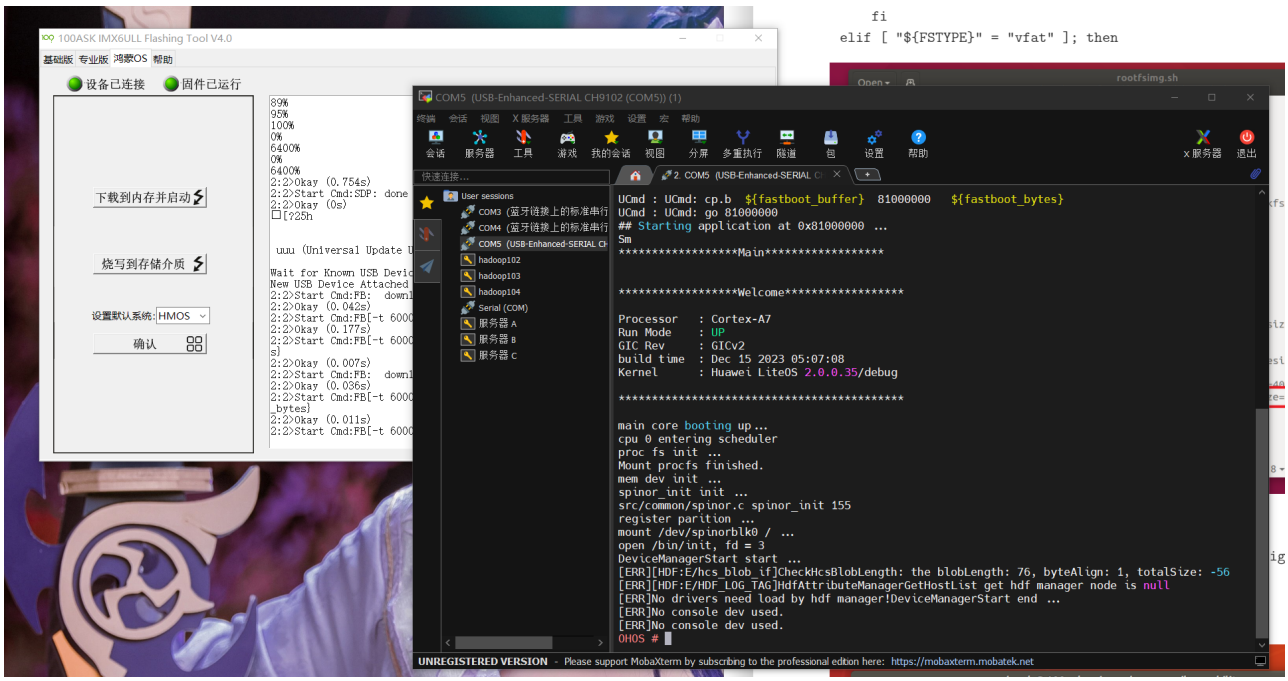




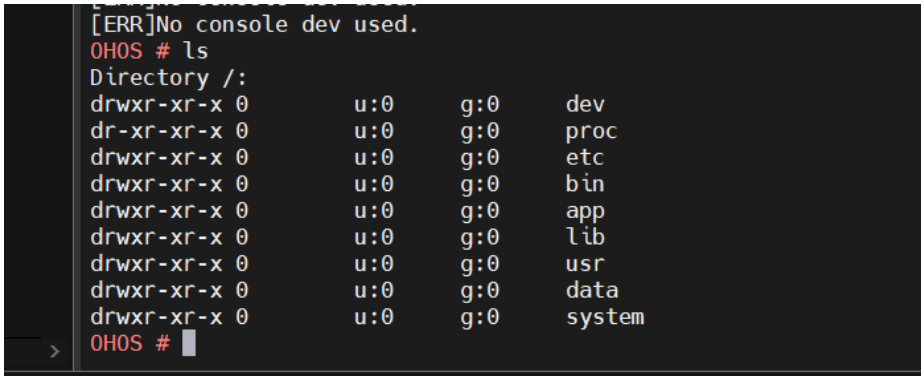
可以看到生成了一个根文件 rootfs.jffs2



可以成功烧录运行



ls 命令显示文件完整



3.2 init进程分析和改进

3.2.1 测试版本

源码：`kernel\liteos_a\apps\init\src\init.c`
我们在`kernel\liteos_a`目录下执行`make rootfs`时使用的就是测试版本，
它的功能很简单：只是启动`/bin/shell`程序，源码如下：

```
int main(int argc, char * const *argv)
{
    int ret;
    const char *shellPath = "/bin/shell";

    ret = fork();
    if (ret < 0) {
        printf("Failed to fork for shell\n");
    } else if (ret == 0) {
        (void)execve(shellPath, NULL, NULL);
        exit(0);
    }

    while (1) {
        ret = waitpid(-1, 0, WNOHANG);
        if (ret == 0) {
            sleep(1);
        }
    };
}
```

3.2.2 正式版本

源码：`base\startup\services\init_lite\src\main.c`

```c
int main(int argc, char * const argv[])
{
    // 1. print system info
    PrintSysInfo();

    // 2. signal register
    SignalInitModule();

    // 3. read configuration file and do jobs
    InitReadCfg();

    // 4. keep process alive
    printf("[Init] main, entering wait.\n");
    while (1) {
        // pause only returns when a signal was caught and the signal-catching function returned.
        // pause only returns -1, no need to process the return value.
        (void)pause();
    }
    return 0;
} « end main »
```

根据配置文件/etc/init.cfg
启动程序(do jobs)

可以使用这样的命令去编译：`python build.py ipcamera_hi3518ev300 -b debug`
可以得到rootfs目录，里面有`/bin/init, /etc/init.cfg`等文件。

3.2.3 配置文件分析

配置文件中内容分为两部分：

- services：定义了多个服务，它对应某些APP
- jobs：可以定义一些APP，也可去启动服务
  - pre-init：预先执行的初始化
  - init：初始化
  - post-init：最后的初始化

4 问题和解决方法

本次实验没有遇到什么较大的问题。

5 实验体会

本次实验我实现了鸿蒙 Liteos-a根文件系统制作。了解了在移植好内核之后，内核要去启动应用程序，而应用程序保存在 flash 上面，flash 上面的内容就是根文件系统，根文件系统有库和应用程序，应用程序一般保存在 bin 目录下面，包括用户写的和系统提供的（shell，init）。而Init根据配置文件启动应用程序。