实验 2 编译鸿蒙 LiteOS-a 内核与 APP

22920212204396 黄子安

一、实验目的

1、编译鸿蒙 LiteOS-a，生成对应的可执行文件并烧写到开发板
2、编译提供的 APP 小程序，并在开发板上进行运行

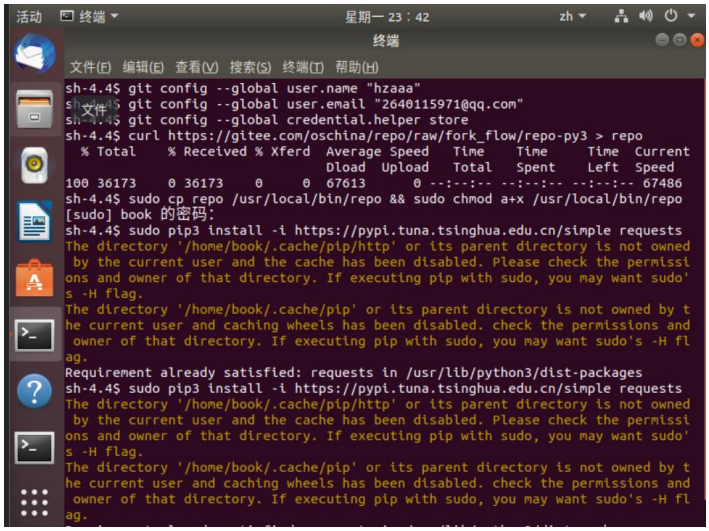
二、实验环境

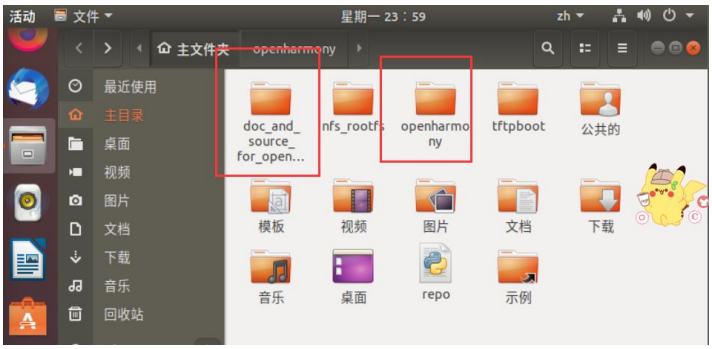
1.物理机：windows 操作系统
2.VMware 虚拟机：ubuntu 18.04.6
3.开发板：imx6ull Mini

三、实验内容

1.下载源码

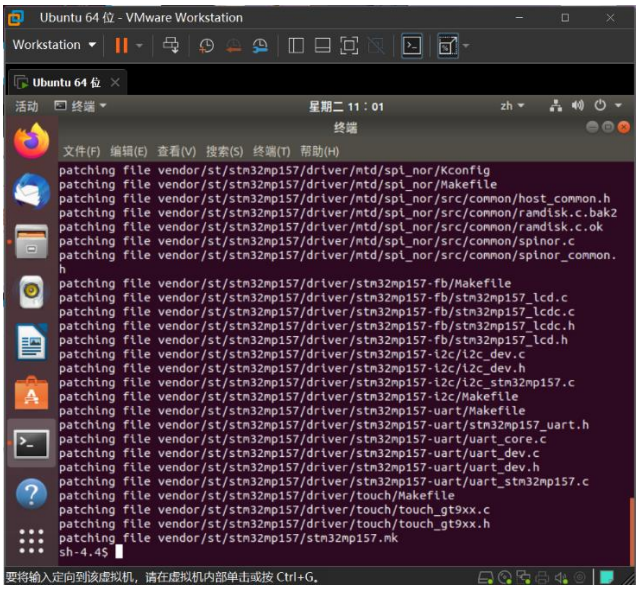
根据教程配置 git，之后新建一个文件夹 openharmony 作为本地仓库，使用命令 sync 同步下载鸿蒙的源码（所提供的实验文档中 repo sync -c -j8 后面多了个 y），另外将相关文档和工具也进行下载，下载得到的源码和文档如下图所示





2.添加补丁文件

使用 patch 命令为代码添加补丁文件（FTP 上所提供的 patch 文件好像存在一定的问题，后来使用了 git 上拉下来的文件后才编译成功）



3.编译内核

进入到相应的目录，之后复制配置文件，使用 make 命令进行编译

问题：无法使用 make -j 8 进行编译得到内核

原因：在配套资料中提到可以使用 python 检测环境是否存在问题，运行后提示没有找到 clang 编译器，检查下载的文件中发现存在 clang，但是不是在 book 用户当中，说明环境配置有问题，无法找到对应的编译器

▪ 1.2.4 编译官方版本

我们先为官方板子 hi3518ev300 编译 Liteos-a，确保环境没有问题。后面再打上补丁，为 IMX6ULL 编译鸿蒙。

进入源码根目录，执行以下命令：

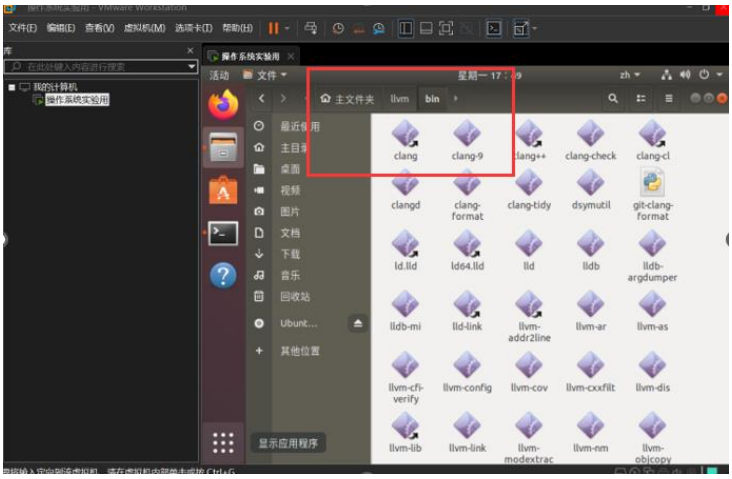
```
cd  /home/book/openharmony
python build.py ipcamera_hi3518ev300 -b debug
```

等待一会，结果如下：

```
make: Leaving directory '/home/book/openharmony/third_party/ltp/testcases/open_posix_testsuite/conformance/interfaces'
[1006/1016] STAMP obj/test/xts/acts/open_posix_testsuite/conformance/interfaces/prebuild_ActsOpenPosixTest.stamp
[1007/1016] ACTION //test/xts/acts/open_posix_testsuite/conformance/interfaces:ActsOpenPosixTest(//build/lite/toolchain:linux_x86_64_clang)
[1008/1016] STAMP obj/test/xts/acts/open_posix_testsuite/conformance/interfaces/ActsOpenPosixTest.stamp
[1009/1016] STAMP obj/test/xts/acts/acts_component.stamp
[1010/1016] COPY bin/query.bin suites/acts/resource/tools/query.bin
[1011/1016] STAMP obj/test/xts/acts/query_copy.stamp
[1012/1016] ACTION //test/xts/acts:acts(//build/lite/toolchain:linux_x86_64_clang)
[1013/1016] STAMP obj/test/xts/acts/acts.stamp
[1014/1016] STAMP obj/build/lite/ohos.stamp
[1015/1016] ACTION //build/lite:gen_rootfs(//build/lite/toolchain:linux_x86_64_clang)
[1016/1016] STAMP obj/build/lite/gen_rootfs.stamp
ohos ipcamera_hi3518ev300 build success!   编译成功，在这个目录下可以看到成果
book@100ask:~/openharmony$
book@100ask:~/openharmony$ ls out/ipcamera_hi3518ev300/
args.gn      build.ninja          dev_tools     liteos.bin      OHOS_Image.asm  rootfs.tar      userfs
bin          build.ninja.d        foundation.map  media_server.map  OHOS_Image.bin  suites          userfs.img
bm_tool.map  bundle_daemon_tool.map  gen           obj             OHOS_Image.map  test            vendor
build.log    data                 libs          OHOS_Image      rootfs.img      toolchain.ninja
book@100ask:~/openharmony$
```

我们不打算使用 hi3518ev300，上述命令只是为了确定开发环境没问题。

并且 rootfs.tar 中有很多库，比如 libfreetype.so，以后可以直接使用。



解决方法：

在 book 用户下在命令行中输入 gedit ~/.bashrc

之后在打开的文件最后添加以下四条环境变量

```
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
export PATH="$PATH:/home/hza/llvm/bin"
export PATH="$PATH:/home/hza/gn"
export PATH="$PATH:/home/hza/ninja"
export PATH="$PATH:/home/hza/hc-gen"
```

在 shell 使用 source ~./bashrc 激活配置文件，之后在可以 shell 中使用命令 clang、gn、ninja、hc-gen 查看是否配置成功，发现可以找到编译器，**也可以直接将四个文件夹拉到 book 用户中**，无需配置环境变量

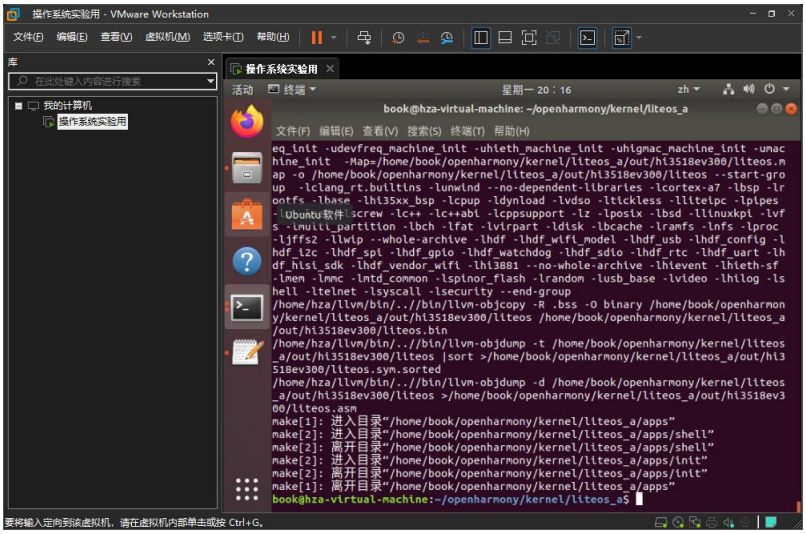
```
book@hza-virtual-machine:~$ gn -version
1523 (5bd8e26b)
```

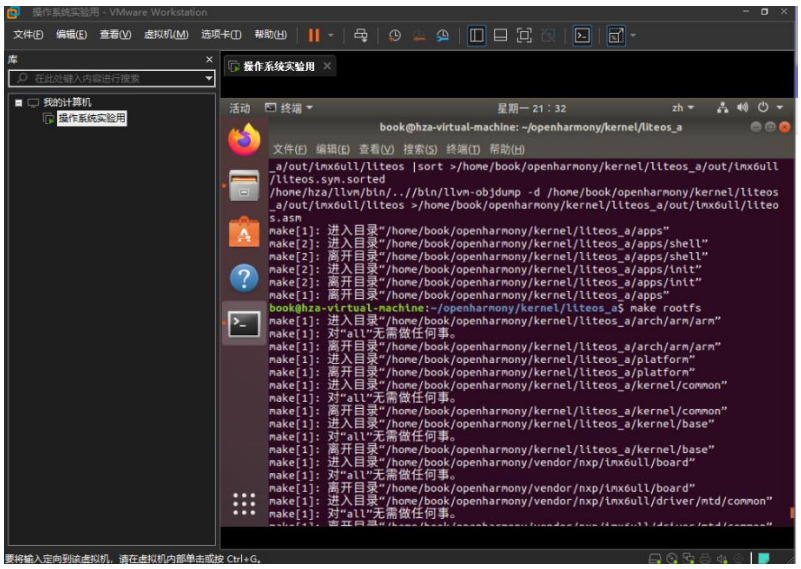
```
book@hza-virtual-machine:~$ hc-gen -v
Hcs compiler v0.65
Copyright (c) Huawei Technologies Co., Ltd. 2020
```

```
book@hza-virtual-machine:~$ clang -v
OHOS (34024) clang version 9.0.0 (llvm-project c20cd5feb33c9df88918ffe9a0df7649
9befaa46) (based on LLVM 9.0.0)
Target: x86_64-unknown-linux-gnu
Thread model: posix
InstalledDir: /home/hza/llvm/bin
Found candidate GCC installation: /usr/lib/gcc/x86_64-linux-gnu/7
Found candidate GCC installation: /usr/lib/gcc/x86_64-linux-gnu/7.5.0
Found candidate GCC installation: /usr/lib/gcc/x86_64-linux-gnu/8
Selected GCC installation: /usr/lib/gcc/x86_64-linux-gnu/7.5.0
Candidate multilib: .;@m64
Candidate multilib: 32;@m32
Candidate multilib: x32;@mx32
Selected multilib: .;@m64
```

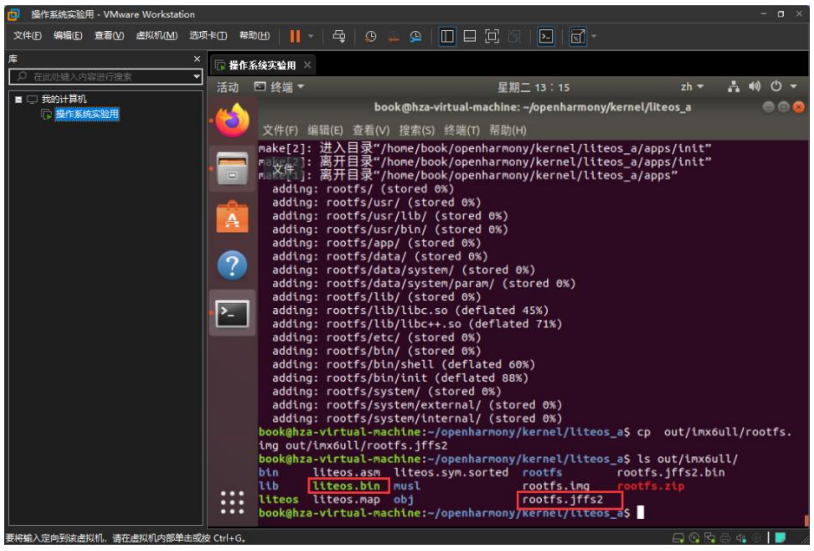
```
book@hza-virtual-machine:~$ ninja --version
1.9.0
```

之后编译内核，系统没有输出错误信息，说明运行顺利，内核编译成功



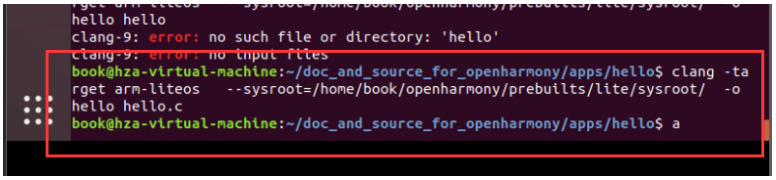


可以看到生成了 LiteOS 的 bin 文件和对应的 rootfs.jffs2 文件

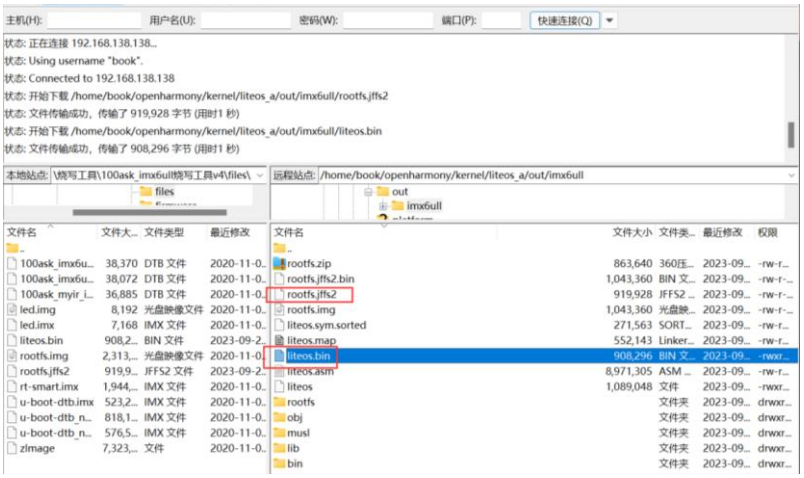


4.编译 APP

先编译 hello.c 文件，进入 hello.c 所在的文件目录，之后使用 clang 进行编译生成可执行文件

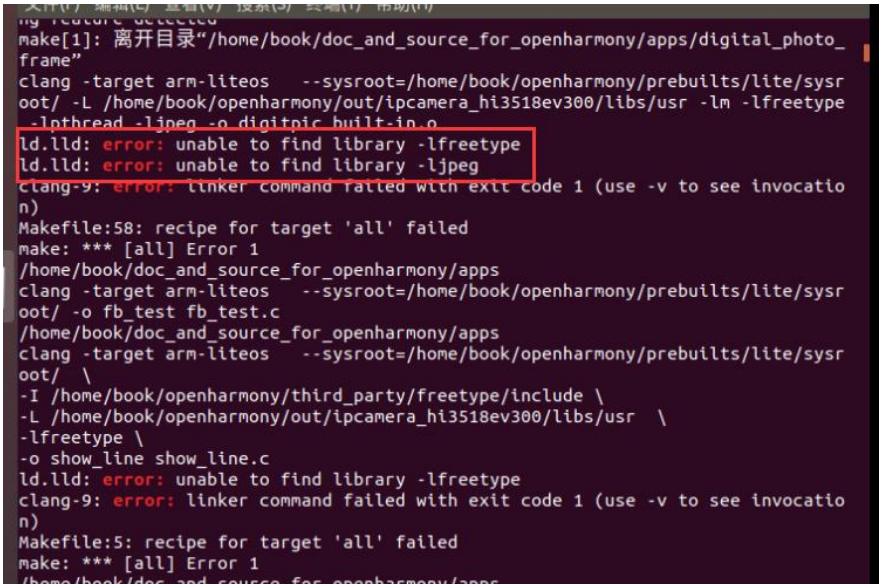


将 hello 程序放入到 rootfs 目录下的 bin 文件夹中，之后重新制作 rootfs.jffs2 文件，最后将生成的文件连同刚刚编译生成的内核文件通过 filezilla 传输到物理机当中

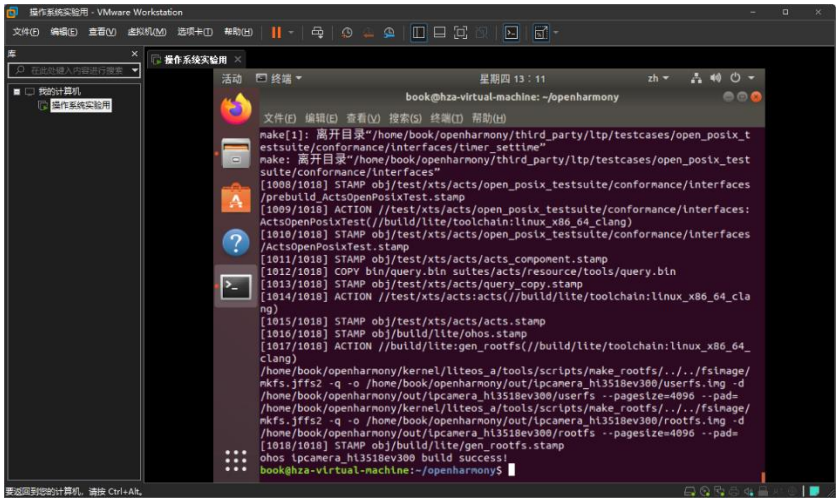


再使用烧写工具将内核与 rootfs.jffs2 烧写到开发板的内存中，启动系统并使用 Xterm 打开对应的串口，输入./bin/hello，可以发现成功运行对应并输出“Hello，Harmony”，说明上述过程没有出错

然后来编译运行剩下的四个程序，使用脚本命令进行编译，但是**系统出现了报错，提示找不到对应的库文件**

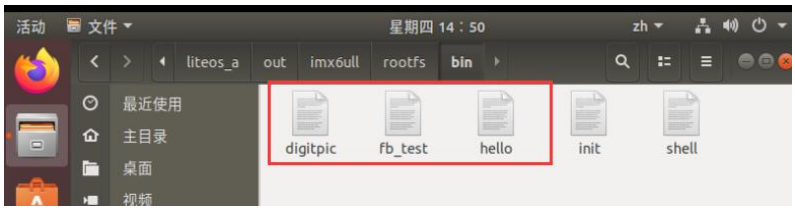


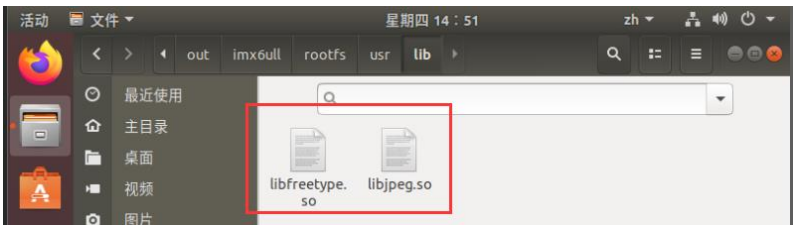
根据提供的实验文档可以发现这两个动态库需要通过 python build.py ipcamera_hi3518ev300 -b debug 生成并进行引用，按照实验文档前面的步骤重新执行这条命令

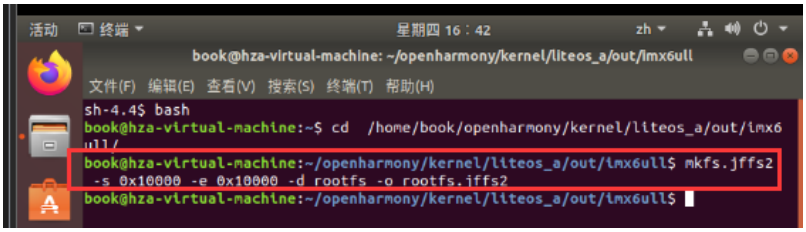


但是运行完这个程序之后发现刚刚的/home/book/openharmony/kernel/liteos_a/out/文件夹不见了，重新进入/home/book/openharmony/kernel/liteos_a 运行 make clean 和 make -j 8 命令，运行顺利重新获得该文件夹和里面的内容，再使用文档中提供的命令将程序与对应的链接库复制到 rootfs 文件夹中，最后使用命令 mkfs.jff2 制作对应的 rootfs.jffs2 文件并烧写到开发板当中

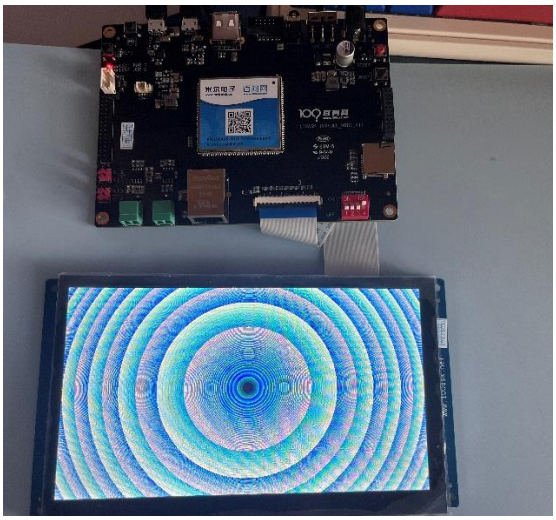
```
cd  /home/book/doc_and_source_for_openharmony/apps
# 拷贝应用程序
cp hello/hello  /home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/bin
cp fb_test/fb_test  /home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/bin
cp freetype/show_line /home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/bin
cp digital_photo_frame/digitpic /home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/bin
# 拷贝字体文件
cp freetype/simsun.ttc  /home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/
# 拷贝数码相框的图标
cp digital_photo_frame/rootfs /home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/  -rf
# 拷贝库文件
cp                    /home/book/openharmony/out/ipcamera_hi3518ev300/libs/usr/libfreetype.so
/home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/usr/lib/
cp                          /home/book/openharmony/out/ipcamera_hi3518ev300/libs/usr/libjpeg.so
/home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/usr/lib/
```







之后在 xterm 中依次运行这些 APP，运行 fb_test,可以发现在显示屏上输出了对应的图片



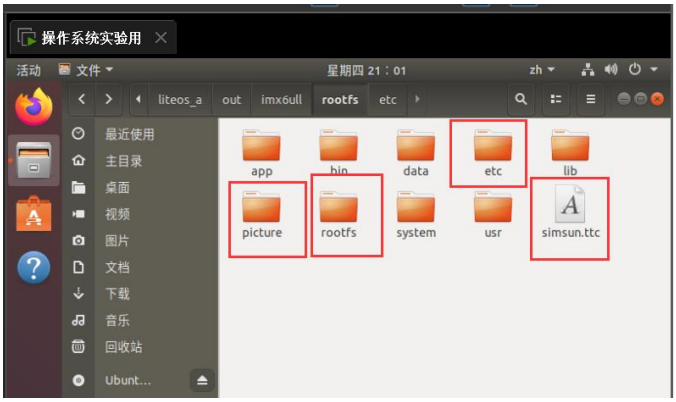
再使用 show_line 使其输出对应的文字，可以看到也成功运行

但是在运行 digitpic 的时候出现了报错，在 xterm 中**显示无法找到某些文件**，通过在形成 rootfs 文件之前虚拟机文件路径可以发现存在一定的问题

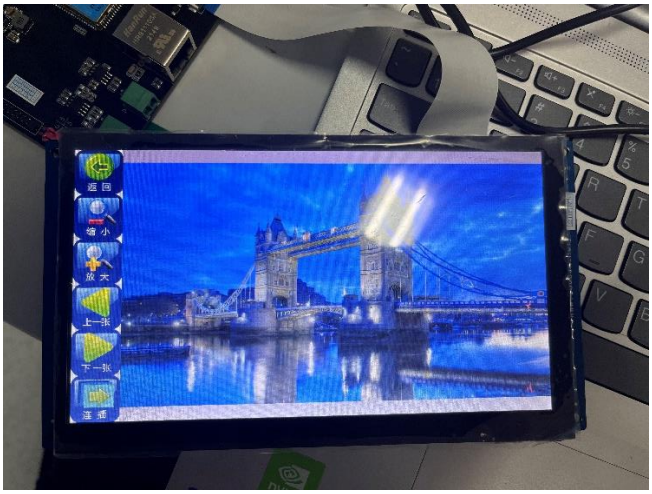
可以**看到在 rootfs 目录下又出现了一个 rootfs**，并且在 etc 文件夹里面没有任何内容，而在 rootfs 里面的 etc 文件夹中存在报错信息中需要的内容，因此可以推测出这里出现错误，应该将里层 rootfs 里内容搬到外层的 rootfs 当中

继续探究可以发现所提供的命令存在错误，该条命令会导致上面的情况出现，重新更改文件路径，得到的文件结构如下图所示

```
cd  /home/book/doc_and_source_for_openharmony/apps
# 拷贝应用程序
cp hello/hello  /home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/bin
cp fb_test/fb_test  /home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/bin
cp freetype/show_line /home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/bin
cp digital_photo_frame/digitpic /home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/bin
# 拷贝字体文件
cp freetype/simsun.ttc  /home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/
# 拷贝数码相框的图标
cp digital_photo_frame/rootfs /home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/  -rf
# 拷贝库文件
cp                    /home/book/openharmony/out/ipcamera_hi3518ev300/libs/usr/libfreetype.so
/home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/usr/lib/
cp                      /home/book/openharmony/out/ipcamera_hi3518ev300/libs/usr/libjpeg.so
/home/book/openharmony/kernel/liteos_a/out/imx6ull/rootfs/usr/lib/
```



最后进行重新烧写，运行程序可以看到出现了预期的效果，至此内核编译与 APP 运行成功



四、实验心得

本次实验磕磕绊绊其实做了非常久，虽然实验报告看起来比较顺利，但是中间实验遇到的很多问题都花了很长时间才解决掉，还出现虚拟机因为硬盘容量不够无法启动的问题，但是经过这么折腾也让对实验的步骤、文件结构等内容熟悉了很多，同时也学会了看报错的重要性，通过互相交流和阅读报错信息，能解决相当多的一部分问题，也让自己对虚拟机、开发板、物理机之间的互相传递消息也更加熟悉。