# 实验二 编译鸿蒙LiteOS-a内核与APP

1. 实验目的
2. 实验内容

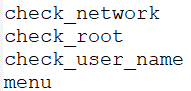
1. 下载源码并编译

1.1 在Ubuntu中安装开发环境

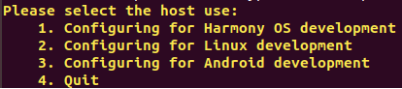
确保Ubuntu能上网之后，使用下面命令一键配置/初始化开发环境(其实就是安装tftp，nfs，vim等软件，此脚本只支持Ubuntu-16.04/Ubuntu-18.04)。 注意：为了方便大家复制，这些命令写成了多行，你要把它们复制到记事本，合并成一行，注意空格：

book@100ask: ~ $ wget --no-check-certificate -O Configuring\_ubuntu.sh https://weidongshan.coding.net/p/DevelopmentEnvConf/d/DevelopmentEnvConf/git/raw/master/Configuring\_ubuntu.sh && sudo chmod +x Configuring\_ubuntu.sh && sudo ./Configuring\_ubuntu.sh

首先，它会做一些检查，然后列出一个菜单供你选择：

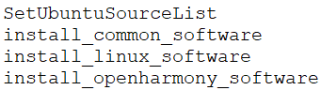


在这个菜单里，你可以选择这三项之一：



其实第一项包含了第二项，无论你是开发鸿蒙还是Linux，都可选择第一项。

假设你选择了第一项，该脚本会执行下面的函数：



从上图中各个函数的名字就可以知道，它会做这些事情：

设置Ubuntu软件源、安装通用的软件(比如GCC、manual手册)、安装Linux开发软件(比如NFS、SAMBA)、安装鸿蒙开发软件(比如llvm)。

1.2 下载、编译鸿蒙内核Liteos-a

Liteos-a内核变化比较大，华为提交得也比较频繁。无法再repo从官方GIT仓库下载本视频配套的源码。我们把本教程所使用的源码上传到了自己的GIT仓库里。

1.2.1 配置GIT与REPO

需要在ubuntu下执行如下命令来配置GIT客户端的用户信息，比如(下面命令的用户名、邮箱可以随便写)：

git config --global user.name "100ask"

git config --global user.email "weidongshan@qq.com"

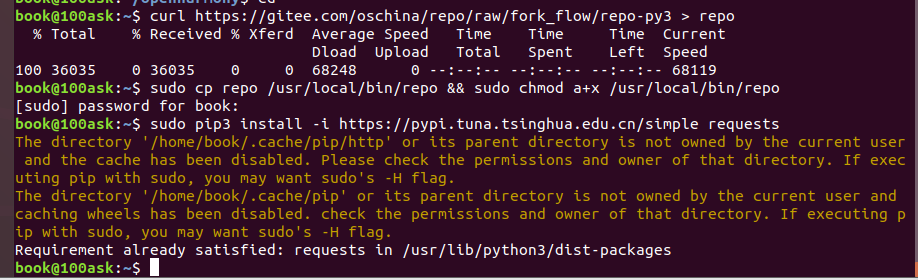
git config --global credential.helper store

curl https://gitee.com/oschina/repo/raw/fork\_flow/repo-py3 > repo

sudo cp repo /usr/local/bin/repo && sudo chmod a+x /usr/local/bin/repo

sudo pip3 install -i https://pypi.tuna.tsinghua.edu.cn/simple requests

在这里插入图片描述



1.2.2 下载源码

执行以下命令下载：

cd /home/book

git clone https://e.coding.net/weidongshan/openharmony/doc\_and\_source\_for\_openharmony.git

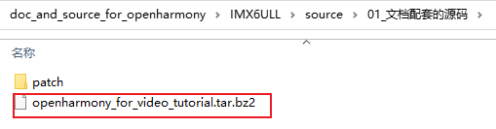
mkdir openharmony

cd openharmony/

repo init -u https://gitee.com/openharmony/manifest.git -b OpenHarmony-1.0

repo sync -c -j8y

可以得到如下文件，它就是源码(为了节省GIT仓库容量，STM32MP157所用的openharmony源码也放在这里)：



1.2.3 解压源码

执行如下命令：

tar xjf openharmony\_for\_video\_tutorial.tar.bz2 -C /home/book/

可以得到这目录：/home/book/openharmony

1.2.4 编译官方版本(不要试)

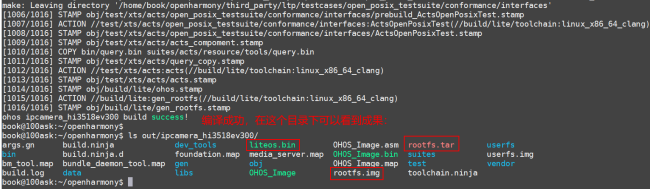
我们先为官方板子hi3518ev300编译Liteos-a，确保环境没有问题。后面再打上补丁，为IMX6ULL编译鸿蒙。

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：



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。

2. 给IMX6ULL编译Liteos-a（直接贴这个补丁）

补丁文件openharmony\_100ask\_v1.2.patch既支持IMX6ULL，也支持STM32MP157，只需要打一次补丁。它位于如下目录，这两个目录里的文件是一样的：

doc\_and\_source\_for\_openharmony\IMX6ULL\source\01\_文档配套的源码\patch\

openharmony\_100ask\_v1.2.patch

doc\_and\_source\_for\_openharmony\STM32MP157\source\01\_文档配套的源码\A7\patch\

openharmony\_100ask\_v1.2.patch

2.1 下载补丁文件

如果已经在Windows里下载了GIT资料，可以把补丁文件上传到Ubuntu。 也可以在Ubuntu中执行以下命令再次下载：

cd /home/book

$ git clone https://e.coding.net/weidongshan/openharmony/doc\_and\_source\_for\_openharmony.git

2.1 使用补丁文件修改代码

假设在/home/book/openharmony目录下是鸿蒙的源码，补丁文件openharmony\_100ask\_v1.2.patch放在/home/book目录下。

执行以下命令打补丁：

cd /home/book/openharmony

patch -p1 < /home/book/openharmony\_100ask\_v1.2.patch

注意：鸿蒙的内核在不断更新，这个补丁是针对“OpenHarmony-1.0”版本的。

所以前面使用repo下载代码时必须指定“-b OpenHarmony-1.0”,参考《1.2.2 下载源码》。

2.3 编译

我的修改还不完善，还无法在根目录下编译。 需要进入kernel/liteos\_a目录下配置、编译：

cd /home/book/openharmony/kernel/liteos\_a

cp tools/build/config/debug/imx6ull\_clang.config .config // 配置

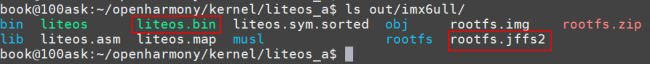
make clean // 先清除一下，否则会提示错误

make -j 8 // 编译内核，可以得到out/imx6ull/liteos.bin

make rootfs // 编译根文件系统，可以得到rootfs.img

cp out/imx6ull/rootfs.img out/imx6ull/rootfs.jffs2 // 改个名易辨认，烧写工具使用rootfs.jffs2

如果一切正常，结果如下：



把上面得到的liteos.bin、rootfs.jffs2，复制到烧写工具的files目录下，然后参考《第3篇

体验鸿蒙内核Liteos-a》，使用烧写工具下载运行它们。

3. 编译第一个APP：hello

3.1 Clang、LLVM与GCC介绍

我们在Ubuntu中给PC机编译程序时，执行这样的命令：

gcc -o hello hello.c

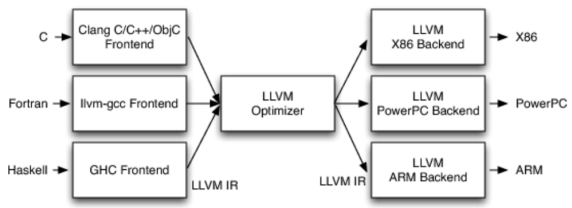
1

我们在Ubuntu给ARM板编译程序时，执行这样的命令(gcc前缀可能不一样)：

arm-linux-gcc -o hello hello.c

在Liteos-a中，使用LLVM来编译程序。LLVM的本意是“Low Level Virtual Machine”，一个底层的虚拟机。但是它现在已经发展成了一种编译器(compiler)的框架系统。简单地说，LLVM可以取代GCC，LLVM容易扩展，可以提供更好的性能。

LLVM的框架如下：



Clang是“Frontend”(前端)，用来执行词法分析、语法分析，简单地说就是：检查语法错误、包含头文件、扩展宏。

中间是优化器，把前端分析好的代码转换为LLVM IR(中间表示，intermediate representation)，IR与机器无关，可以很方便地添加一系列的优化。 优化后得到的IR，再由“Backend”(后端)转换为各种机器的机器码。

以上内容来自网络，我对Clang、LLVM理解不深，简单介绍一下而已。

3.2 编译hello程序

使用Clang时，我们要指定很多参数：为哪类CPU编译程序(ARM还是Risc-V)，CPU架构是什么(cortex A7还是cortex A15)，头文件/库文件在哪？ 在为Liteos-a编译程序时，以hello程序为例，可以执行以下命令：

cd /home/book/doc\_and\_source\_for\_openharmony/apps/hello

clang -target arm-liteos --sysroot=/home/book/openharmony/prebuilts/lite/sysroot/ \

-o hello hello.c

上述命令的sysroot参数中，指定有标准头文件、库文件的目录。

还可以像GCC一样使用“-I”、“-L”参数指定其他头文件的目录、库文件的目录，比如后面会编译这个程序：

cd /home/book/doc\_and\_source\_for\_openharmony/apps/freetype

clang -target arm-liteos --sysroot=/home/book/openharmony/prebuilts/lite/sysroot/ \

-I /home/book/openharmony/third\_party/freetype/include \

-L /home/book/openharmony/out/ipcamera\_hi3518ev300/libs/usr \

-lfreetype \

-o show\_line show\_line.c

3.3 把hello程序放入rootfs

编译出hello程序后，可以把它放入rootfs目录下的bin字母，比如：

cd /home/book/doc\_and\_source\_for\_openharmony/apps/hello

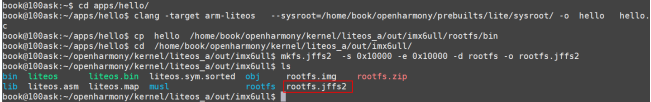
cp hello /home/book/openharmony/kernel/liteos\_a/out/imx6ull/rootfs/bin

然后重新制作rootfs.jffs2，执行以下命令：

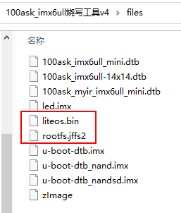
cd /home/book/openharmony/kernel/liteos\_a/out/imx6ull/

mkfs.jffs2 -s 0x10000 -e 0x10000 -d rootfs -o rootfs.jffs2

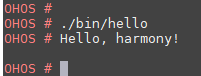
假设hello程序位于/home/book/apps/hello目录下，操作截图如下:



把得到的rootfs.jffs2放到烧写工具的files目录，就可以使用烧写工具启动了：



进入命令行后，就可以执行hello程序了，如下图所示：

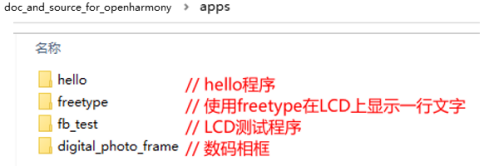


注意：必须用“./bin/hello”，不能用绝对路径“/bin/hello”

4. 编译其他APP

4.1 获取源码

我们提供了这几个程序：



放在GIT上，地址为：

https://e.coding.net/weidongshan/openharmony/doc\_and\_source\_for\_openharmony.git

为了方便，可以在Ubuntu中执行以下命令直接下载：

cd /home/book

git clone https://e.coding.net/weidongshan/openharmony/doc\_and\_source\_for\_openharmony.git

这将会得到这个目录：/home/book/doc\_and\_source\_for\_openharmony

4.2 编译程序

假设已经下载得到/home/book/doc\_and\_source\_for\_openharmony目录，执行以下命令即可编译：

cd /home/book/doc\_and\_source\_for\_openharmony/apps

for i in $(ls);

do cd $i;

make;

cd -;

done

解释以下上面的命令。 $(ls)里列出了这几个目录：digital\_photo\_frame fb\_test freetype hello。 对于其中的每一项“$i”，执行(do)这些命令：

cd $i;

make;

cd -

1

2

3

就是进入该目录，执行make命令，退到上一级目录。 你当然可以自己进入那些子目录，手工执行make命令。

注意：

(1)freetype、digital\_photo\_frame这2个程序，需要使用到前面编译出来的hi3518ev300的库文件。

(2)所以需要先执行“python build.py ipcamera\_hi3518ev300 -b debug”

(3)如果鸿蒙源码的路径不是/home/book/openharmony，那么需要修改Makefile，如下修改路径：

在这里插入图片描述

4.3 复制程序到rootfs

这2个程序不依赖其他动态库：hello、fb\_test；

但是freetype程序依赖libfreetype.so、digital\_photo\_frame程序依赖libfreetype.so、libjpeg.so。

这几个动态库在哪？我们取个巧，之前在《1.2.4 编译官方版本》执行过“python build.py ipcamera\_hi3518ev300 -b debug”，它为hi3518ev300开发板生成了这写库文件。hi3518ev300与imx6ull同属cortex A7架构，所以imx6ull可以使用这些库。以后完善imx6ull的代码后，就不需要这样取巧了。 执行以下命令复制程序：

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/

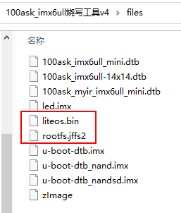
4.4 制作rootfs.jffs2

执行以下命令：

cd /home/book/openharmony/kernel/liteos\_a/out/imx6ull/

mkfs.jffs2 -s 0x10000 -e 0x10000 -d rootfs -o rootfs.jffs2

把得到的rootfs.jffs2放到烧写工具的files目录，就可以使用烧写工具启动了：



4.5 执行APP

这些APP，可以直接执行，如果需要参数的话，它会提示用法。

注意：必须用相对路径，比如“./bin/hello”；不能用绝对路径，比如“/bin/hello”

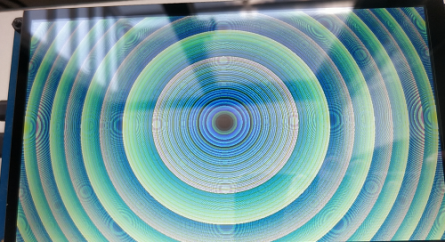
4.5.1 fb\_test

执行以下命令：

./bin/fb\_test /dev/fb0

1

效果如下：



4.5.2

digitpic

执行以下命令：

./bin/digitpic /simsun.ttc

1

可以点击屏幕上的图标，效果如下：

