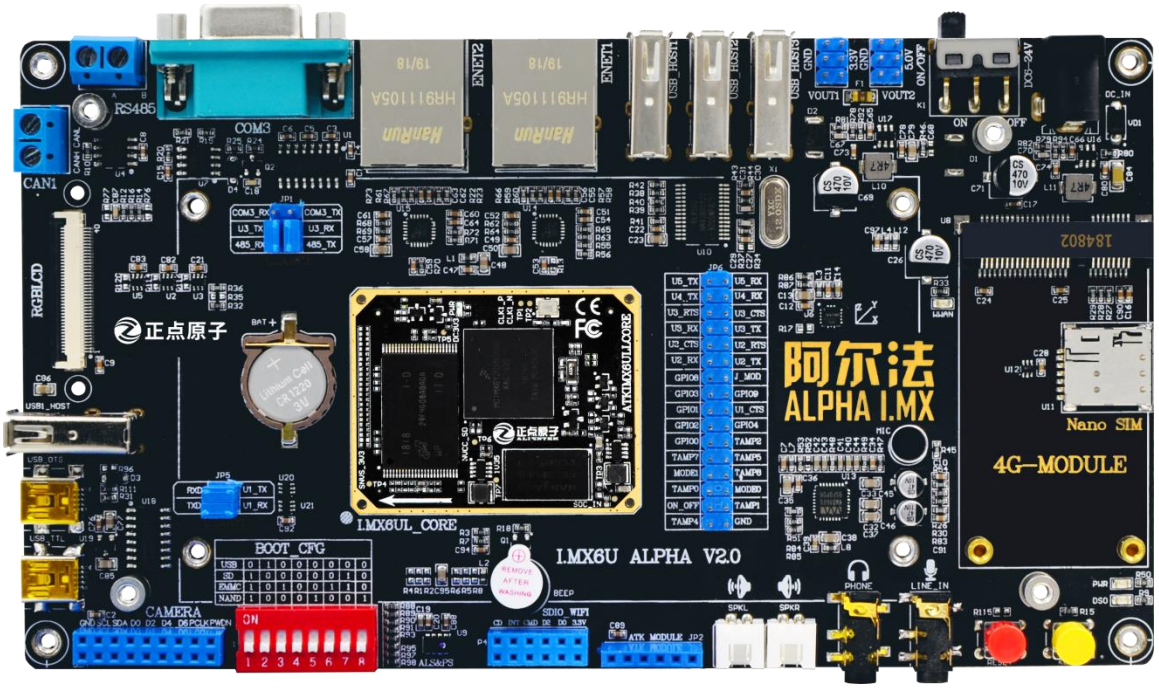
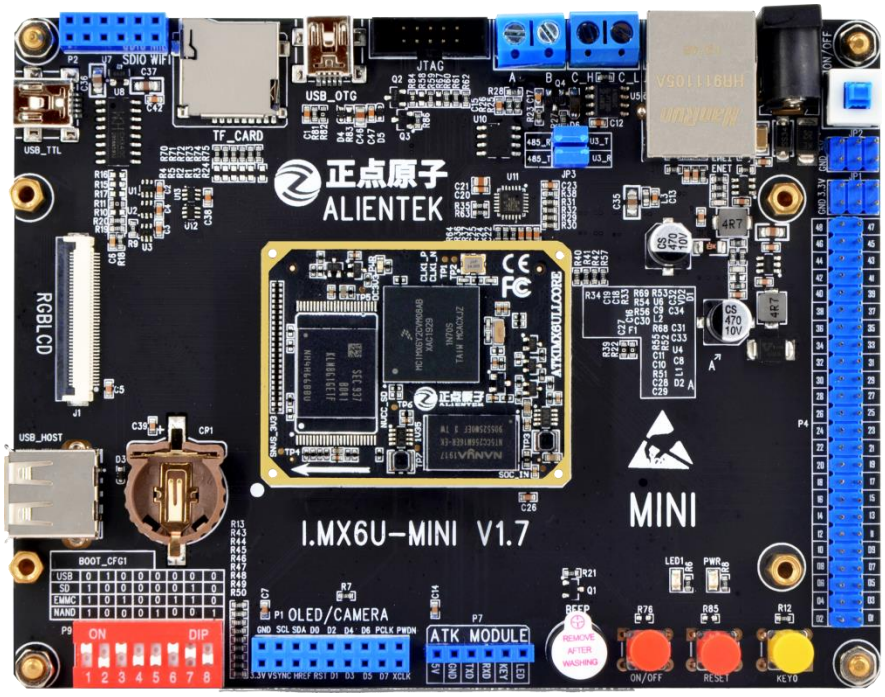
【正点原子】I.MX6U 移植 OpenCV V1.3





正点原子

正点原子公司名称 ：广州市星翼电子科技有限公司

原子哥在线教学平台：www.yuanzige.com

开源电子网 / 论坛 ：http://www.openedv.com/forum.php

正点原子淘宝店铺 ：https://openedv.taobao.com

正点原子官方网站 ：www.alientek.com

正点原子 B 站视频 ：https://space.bilibili.com/394620890

电话：020-38271790 传真：020-36773971

请关注正点原子公众号，资料发布更新我们会通知。

请下载原子哥 APP，数千讲视频免费学习，更快更流畅。

扫码关注正点原子公众号

扫码下载“原子哥”APP

文档更新说明

版本	版本更新说明	负责人	校审	发布日期
V1.0	初稿：	正点原子 linux 团队	正点原子 linux 团队	2020.2.21
V1.1	更新：1. 更新 Qt OpenCV 工程的百度云链接，避免用户下载漏了 QOpenCV 这级目录。导致后面编译不过去。 修改：1. 在第 3、小节添加红色字体的话语，减少用户出错的概率！	正点原子 linux 团队	正点原子 linux 团队	2020.7.30
V1.2	修改：1.重写移植 OpenCV，删除 QOpenCV 工程的编译。	正点原子 linux 团队	正点原子 linux 团队	2020.11.23
V1.3	修改：1.修改文档格式，其他小修改。 2.添加第三章在 Qt 项目导入 Qt 库。	正点原子 linux 团队	正点原子 linux 团队	2021.06.19

目录

前言

在 2020.11.21 日后正点原子的 I.MX6U 出厂文件系统里已经添加了 OpenCV 3.1 的库，也就是说可以直接跑 OpenCV 而不需要再移植。写这个文档是为了学习如何移植 OpenCV。读者应要有 OpenCV 基础，否则移植这个 OpenCV 是没有多大用处的。如有错漏，请到正点原子论坛指正，或者联系本文档编写作者 QQ1252699831 指正错误。

本文档所使用的环境：

- Windows 7 64bits，也适用于 Windows 8-10。不建议用 Windows 32 位来开发，Windows 32 位支持的内存大小有限，系统性能有限。
- Ubuntu16.04，Ubuntu 建议使用 16.04 否则安装及编译环境不一样导致出错的，请自行解决！
- 要求读者会使用 FileZilla、WinSCP 及 Windows Git 进行 Ubuntu 与 Windows 间互传文件的方法。
- OpenCV 版本，选择为 OpenCV 3.4.1（与出厂系统版本的 OpenCV 3.1 版本库不一样，不过都是 3.x 版本，在使用上没多大区别）。

第一章 下载安装通用交叉编译器

编译 ARM 平台的 OpenCV，那么我们需要安装交叉编译器。注不要使用出厂系统的交叉编译器 arm-poky-linux-gnueabi-gcc 编译器来编译 Opencv 源码。因为出厂系统的编译器是经过 NXP 封装过的，不适用于出厂系统外的源码编译。

1.1 下载通用交叉编译器

我们要为 I.MX6U 移植 OpenCV，需要使用的 ARM 平台交叉编译器，这里下载 Linaro 出品的交叉编译器，也就是正点原子 I.MX6U 嵌入式 Linux 驱动开发指南第 4.3 小节里推荐的 4.9 版本的编译器，这里重复写下载的方法。如果已经知道怎么安装或者已经安装可跳过第一章。下面是下载地址。https://releases.linaro.org/components/toolchain/binaries/4.9-2017.01/arm-linux-gnueabihf/。

图 1.1 1 选择对应系统版本的交叉编译器下载

请根据个人 Ubuntu 系统的位数，选择对应版本下载即可。也可以在我们正点原子 I.MX6U 开发板光盘 A-基础资料->5、开发工具->1、交叉编译器下找到下载好的交叉编译器。如下图。

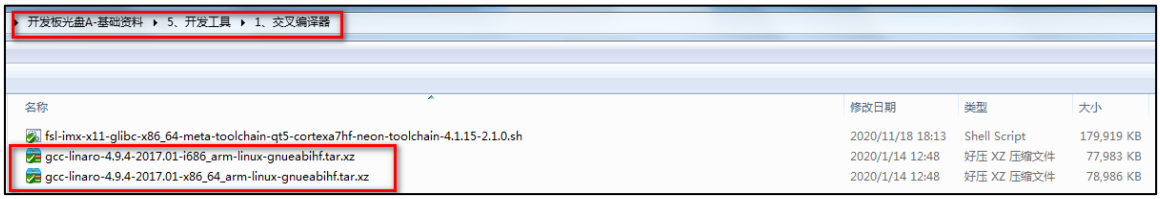


图 1.1 2 资料盘处的交叉编译器

1.2 安装通用交叉编译器

将上面 1.1 小节下载好的通用交叉编译器压缩包拷贝到 Ubuntu 虚拟机，解压进行安装。编者是 64 位的 Ubuntu。所以通用 FileZilla 或者 WinSCP 拷贝 gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf.tar.xz 到 Ubuntu 虚拟机。如下图，编者已经拷贝到“家”目录下。

图 1.2 1 拷贝交叉编译器到家目录

在 Ubuntu 目录下创建/usr/local/arm 文件夹，为下面安装到/usr/local/arm 这个文件夹做准备。

```
sudo mkdir /usr/local/arm
```

解压交叉编译器压缩包至/usr/local/arm 目录下，稍等片刻，解压完成如下。

```
sudo tar xf gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf.tar.xz -C /usr/local/arm/
```

```
alientek@ubuntu:~$ sudo mkdir /usr/local/arm
[sudo] alientek 的密码:
alientek@ubuntu:~$ sudo tar xf gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf.tar.xz -C /usr/local/arm/
alientek@ubuntu:~$
```

图 1.2 2 解压交叉编译器

使用 vi 指令编辑/etc/profile 这个文件。

```
sudo vi /etc/profile
```

打开/etc/profile 以后，在末尾添加如下所示内容。

```
export PATH=$PATH:/usr/local/arm/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin
```

添加完成如下图，保存退出，重启系统。

```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ "$PS1" ]; then
  if [ "$BASH" ] && [ "$BASH" != "/bin/sh" ]; then
    # The file bash.bashrc already sets the default PS1.
    # PS1='\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi
export PATH=$PATH:/usr/local/arm/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin
~
```

图 1.2 3 在/etc/profile 下添加全局环境变量

1.3 验证通用交叉编译器

要使用此编译器，还要在 Ubuntu 上安装一些库。

```
sudo apt-get install lsb-core lib32stdc++6
```

在 1.2 小节修改环境变量重启系统后，在终端输入 arm-linux-gnueabihf-gcc -v 来查看安装的交叉编译器版本号。看到如下结果，表明成功！

```
arm-linux-gnueabihf-gcc -v
```

```
alientek@ubuntu:~$ arm-linux-gnueabihf-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabihf-gcc
COLLECT_LTO_WRAPPER=/usr/local/arm/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin/../libexec/gcc/arm-linux-gnueabihf/4.9.4/lto-wrapper
Target: arm-linux-gnueabihf
Configured with: /home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/snapshots/gcc-linaro-4.9-2017.01/configure SHELL=/bin/bash --with-mpc=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu --with-mpfr=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu --with-gmp=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu --with-gnu-as --with-gnu-ld --disable-libmudflap --enable-lto --enable-objc-gc --enable-shared --without-included-gettext --enable-nls --disable-sjlj-exceptions --enable-gnu-unique-object --enable-linker-build-id --disable-libstdcxx-pch --enable-c99 --enable-clocale=gnu --enable-libstdcxx-debug --enable-long-long --with-cloog=no --with-ppl=no --with-isl=no --disable-multilib --with-float=hard --with-mode=thumb --with-tune=cortex-a9 --with-arch=armv7-a --with-fpu=vfpv3-d16 --enable-threads=posix --enable-multiarch --enable-libstdcxx-time=yes --with-build-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/sysroots/arm-linux-gnueabihf --with-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu/arm-linux-gnueabihf/libc --enable-checking=release --disable-bootstrap --enable-languages=c,c++,fortran,lto --build=x86_64-unknown-linux-gnu --host=x86_64-unknown-linux-gnu --target=arm-linux-gnueabihf --prefix=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu
Thread model: posix
gcc version 4.9.4 (Linaro GCC 4.9-2017.01)
alientek@ubuntu:~$
```

图 1.3 1 验证交叉编译器

第二章 搭建 OpenCV 3.4.1 的编译环境

安装完成交叉编译器后，才能开始搭建 OpenCV 的编译环境。

2.1 下载 OpenCV 3.4.1 源码

OpenCV 官方源码下载链接为 https://opencv.org/releases/ 可以看到有很多版本的 OpenCV 源码下载。本次选择的是 opencv-3.4.1 版本源码，其他版本请自行选择编译，但是不确保其他源码编译不出现错误。请自行点击下载，国外网站，下载常失败。所以我们把下载好源码放在我们开发板光盘 A-基础资料->1、例程源码->7、第三方库源码->opencv-3.4.1.tar.gz。

拷贝 opencv-3.4.1.tar.gz 到 Ubuntu 虚拟机，使用下面的指令直接解压。然后进入解压后的目录查看。

```
tar xf opencv-3.4.1.tar.gz
cd opencv-3.4.1/
ls
```

```
alientek@ubuntu:~$ tar xf opencv-3.4.1.tar.gz
alientek@ubuntu:~$ cd opencv-3.4.1/
alientek@ubuntu:~/opencv-3.4.1$ ls
3rdparty  apps  cmake  CMakeLists.txt  CONTRIBUTING.md  data  doc  include  LICENSE  modules  platforms  README.md  samples
alientek@ubuntu:~/opencv-3.4.1$
```

图 2.1 1 解压 OpenCV 源码

2.2 配置 OpenCV 环境

创建 build、install 文件夹。build 为构建文件夹，install 为安装文件夹。

```
mkdir build install
ls
```

```
alientek@ubuntu:~/opencv-3.4.1$ mkdir build install
alientek@ubuntu:~/opencv-3.4.1$ ls
3rdparty  apps  build  cmake  CMakeLists.txt  CONTRIBUTING.md  data  doc  include  install  LICENSE  modules  platforms  README.md  samples
alientek@ubuntu:~/opencv-3.4.1$
```

图 2.2 1 新建 build 和 install 目录

安装 cmake 和 cmake-gui 工具

```
sudo apt-get install cmake cmake-qt-gui  cmake-curses-gui
```

进入 build 目录下，执行指令 cmake-gui，开始配置 OpenCV 的环境。

```
cd build
cmake-gui
```

```
alientek@ubuntu:~/opencv-3.4.1$ cd build/
alientek@ubuntu:~/opencv-3.4.1/build$ cmake-gui
```

图 2.2 2 进入 build 目录，执行 cmake-gui

执行完成后会出现图形化工具 cmake-gui。如下图

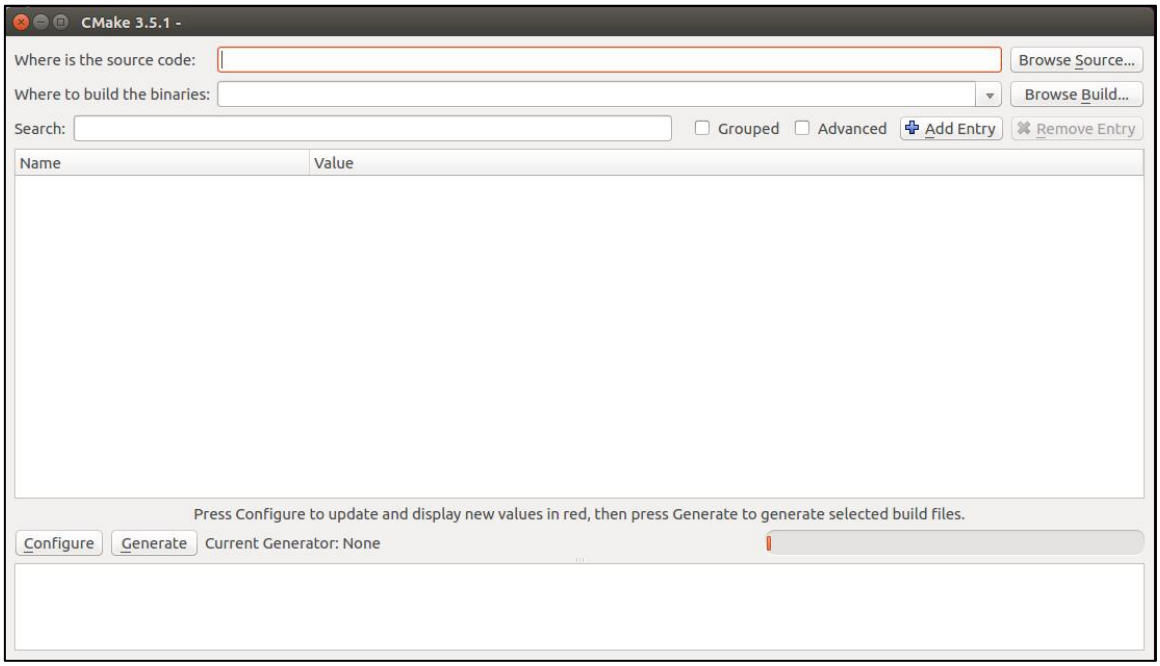


图 2.2 3 CMake 图形界面

指定我们源码的所在路径和构目录。按如下图设置，记得修改成个人的路径。再点击 Generate。

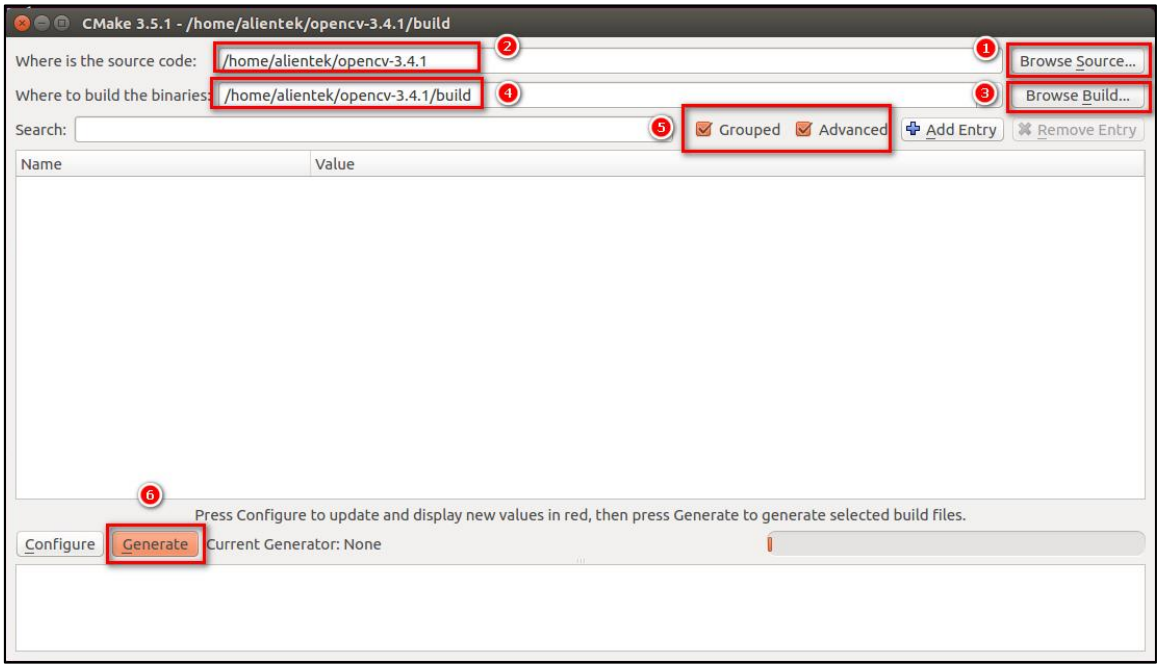


图 2.2 4 配置源码路径

选择 Unix Makefiles，然后选择 Specify options for cross-compiling，再点击 Next。

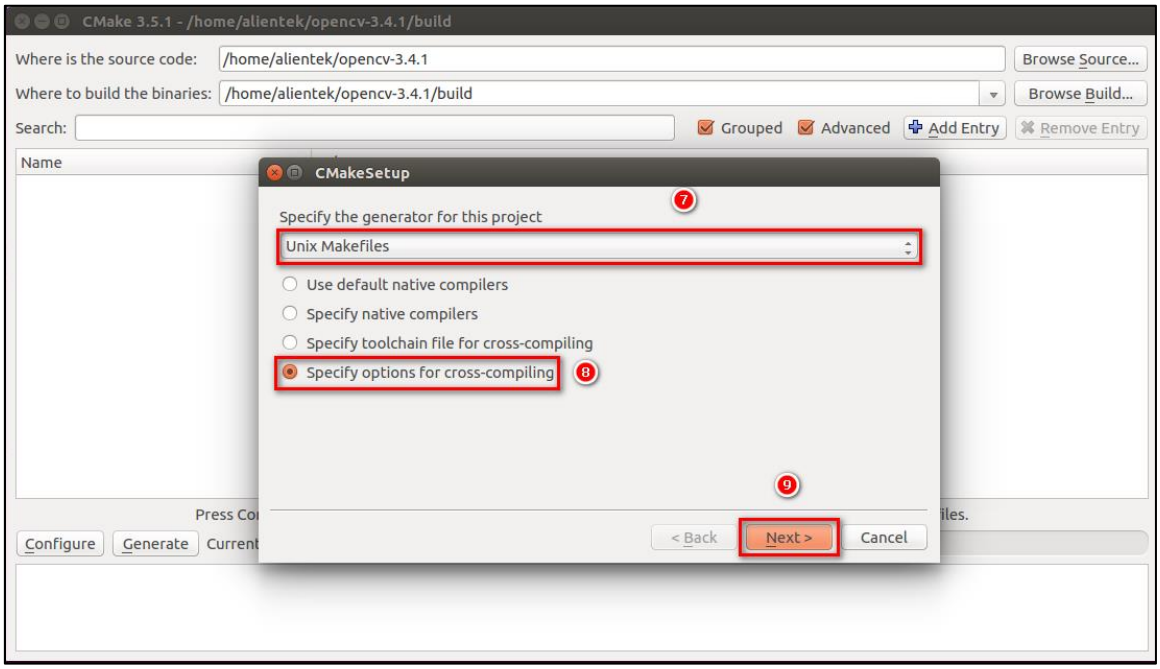


图 2.2 5 选择 Unix Makefiles

填写交叉编译器的路径，填写第一章安装的交叉编译器路径。

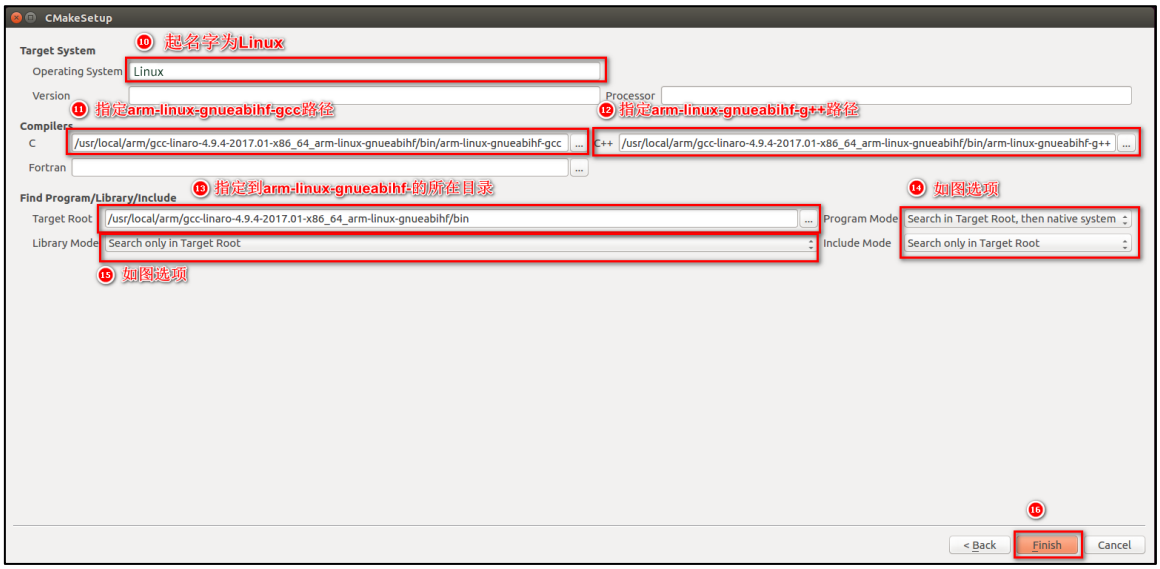


图 2.2 6 填写交叉编译器路径

下面就是您配置的信息，可以配置很多项，比如要编译哪些库等都可以在此选择编译或者不编译。比如常见的 V4L2 都需要自己打开（V4L2 是处理摄像头类用的）。

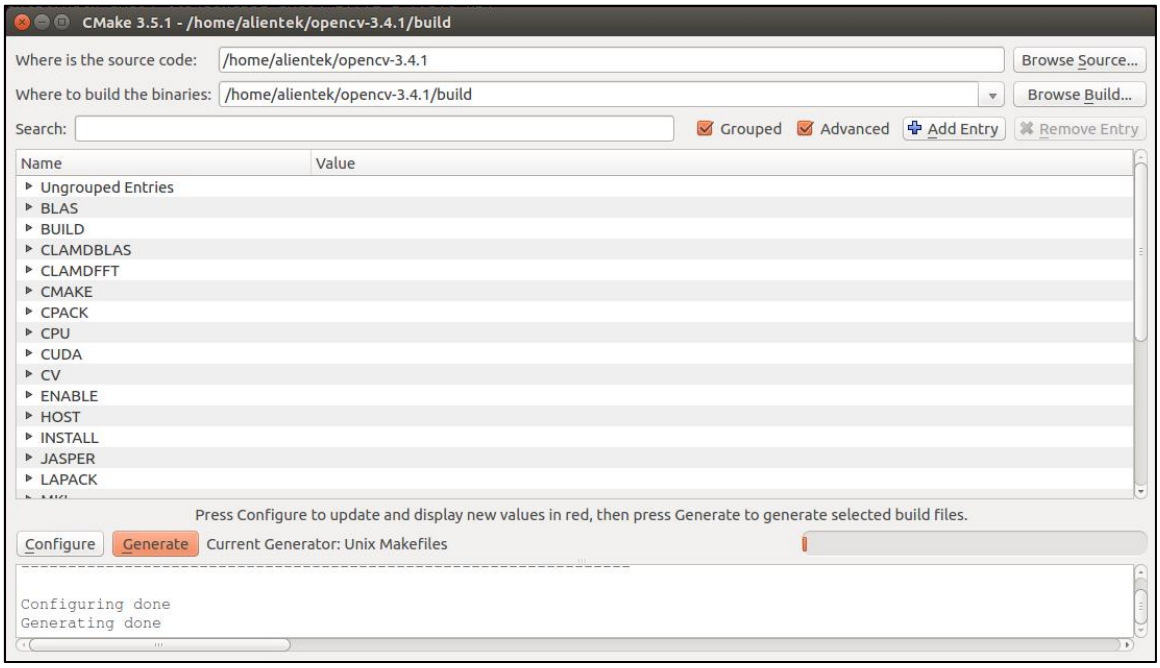


图 2.2 7 进入配置信息页面

点击 CMAKE，在 CMAKE_EXE_LINKER_FLAGS 处添加上“-lpthread -lrt -ldl”添加这些是指定依赖库的链接参数。上面的“-lxx”其中“-l”是大写字母“L”的小写字母“l”，不是数字“1”。

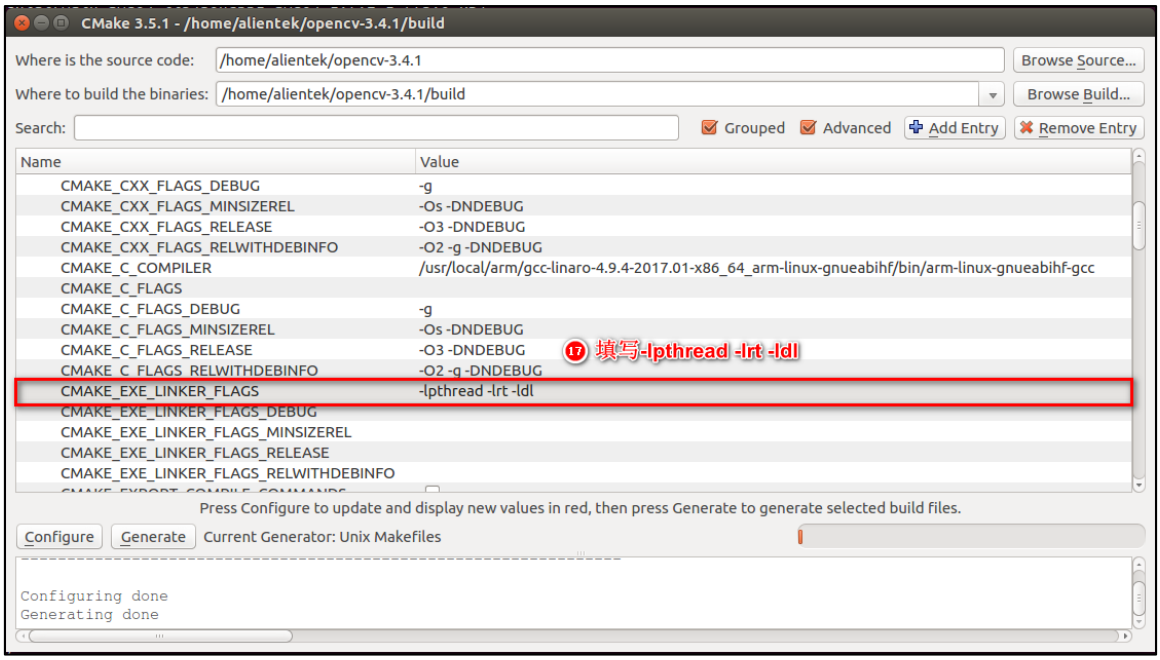


图 2.2 8 指定依赖库的链接参数

再在 CMAKE_INSTALL_PREFIX 处指定安装目录，我们在上面已经新建了 install 安装目录。如果不指定，它会默认安装到 Ubuntu 系统目录/usr/local 下。再点击关闭。到这里我们已经完成配置

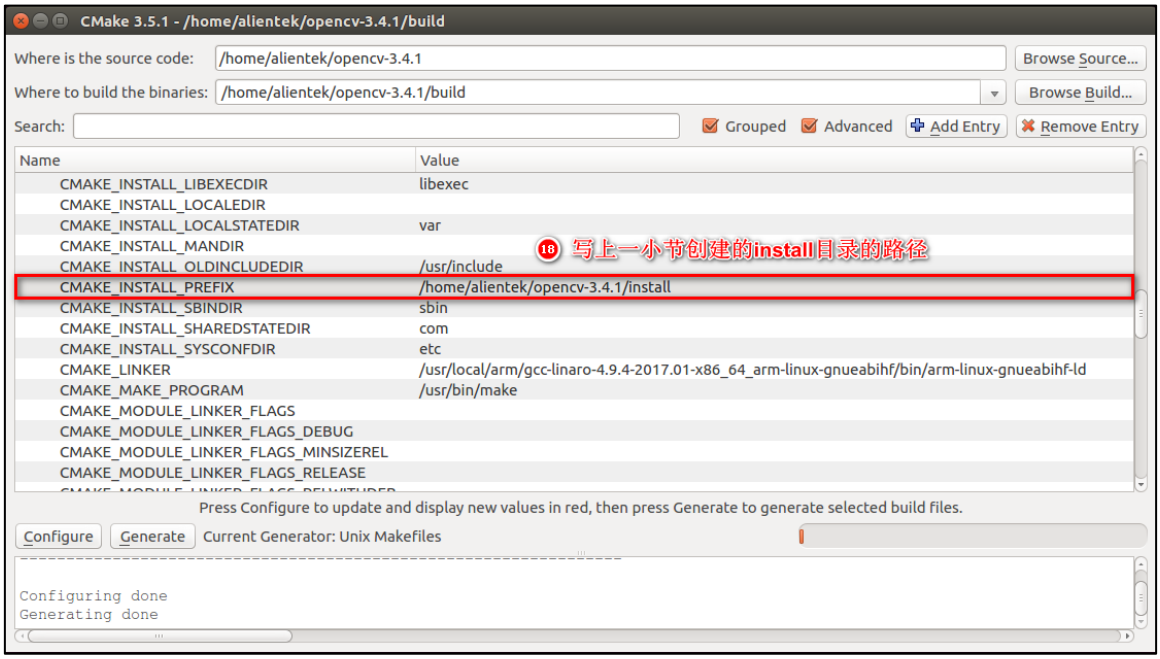


图 2.2 9 指定安装目录路径

至此我们配置完，现在我们需要生成 Makefile 等文件，我们先点击 Configure，再点击 Generate 就可以生成了。

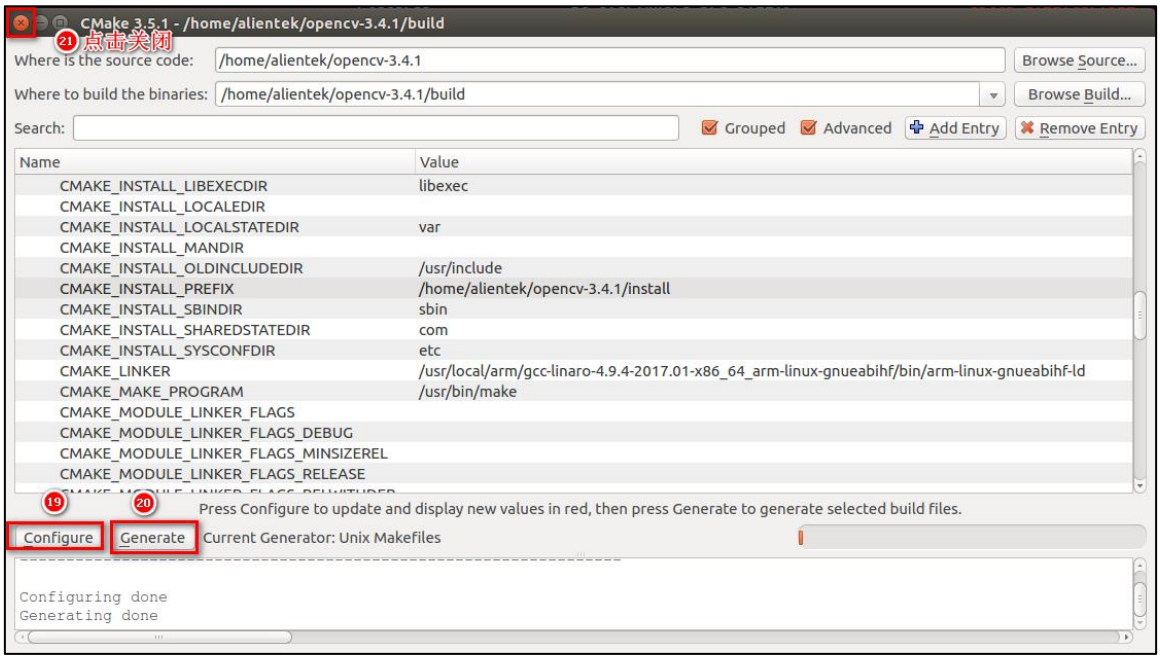


图 2.2 10 点击生成配置

2.3 编译 OpenCV 源码

先不要急着输入 make。首先在源码目录 3rdparty/protobuf/src/google/protobuf/stubs/common.cc 这个文件下第 33 行添加#define HAVE_PTHREAD 宏定义才可以编译的过。具体原因是 HAVE_PTHREAD 宏定义了 pthread 库。在如下位置添加即可。

```
cd ..                                          // 返回 opencv 源码顶层目录
vi 3rdparty/protobuf/src/google/protobuf/stubs/common.cc
```

```
1  // Protocol Buffers - Google's data interchange format
2  // Copyright 2008 Google Inc.  All rights reserved.
3  // https://developers.google.com/protocol-buffers/
4  //
5  // Redistribution and use in source and binary forms, with or without
6  // modification, are permitted provided that the following conditions are
7  // met:
8  //
9  //     * Redistributions of source code must retain the above copyright
10 // notice, this list of conditions and the following disclaimer.
11 //     * Redistributions in binary form must reproduce the above
12 // copyright notice, this list of conditions and the following disclaimer
13 // in the documentation and/or other materials provided with the
14 // distribution.
15 //     * Neither the name of Google Inc. nor the names of its
16 // contributors may be used to endorse or promote products derived from
17 // this software without specific prior written permission.
18 //
19 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
20 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
21 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
22 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
23 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
24 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
25 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
26 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
27 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
28 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
29 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
30
31 // Author: kenton@google.com (Kenton Varda)
32
33 #define HAVE_PTHREAD
34 #include <google/protobuf/message_lite.h>  // TODO(gerbens) ideally remove this.
35 #include <google/protobuf/stubs/common.h>
36 #include <google/protobuf/stubs/once.h>
37 #include <google/protobuf/stubs/status.h>
38 #include <google/protobuf/stubs/stringpiece.h>
39 #include <google/protobuf/stubs/strutil.h>
40 #include <google/protobuf/stubs/int128.h>
```

图 2.3 1 添加宏定义

修改完成后，返回于是 build 目录下，可以看到 bulid 目录下已经准备了构建文件。我们直接输入 make 构建即可。

```
cd build/
make -j 16
```

```
alientek@ubuntu:~/opencv-3.4.1$ cd build/
alientek@ubuntu:~/opencv-3.4.1/build$ make -j 16
Scanning dependencies of target gen-pkgconfig
Scanning dependencies of target libjpeg
Scanning dependencies of target libjasper
Scanning dependencies of target zlib
Scanning dependencies of target libwebp
Scanning dependencies of target libprotobuf
[  0%] Generate opencv.pc
[  0%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/compress.c.obj
[  0%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/adler32.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/deflate.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/crc32.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/gzlib.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/inflate.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/infback.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/gzwrite.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/inftrees.c.obj
[  1%] Building C object 3rdparty/libjpeg/CMakeFiles/libjpeg.dir/jutils.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/gzread.c.obj
[  1%] Building C object 3rdparty/zlib/CMakeFiles/zlib.dir/gzclose.c.obj
[  1%] Building C object 3rdparty/libjasper/CMakeFiles/libjasper.dir/jpc_t1enc.c.obj
```

图 2.3 2 执行 make 编译

编译完成如下图。如果有出错，先删除 build 目录下的所有文件，按第 2.2 小节重新再来一次！cmake-gui 尽量一次配置成功。

图 2.3 3 编译完成截图

输入 make install，把库安装在 2.2 小节我们创建的 install 目录下。然后使用 ls 指令查看安装目录 install。看到有如下图一样的文件，表明安装成功。

```
make install
ls ../install
```

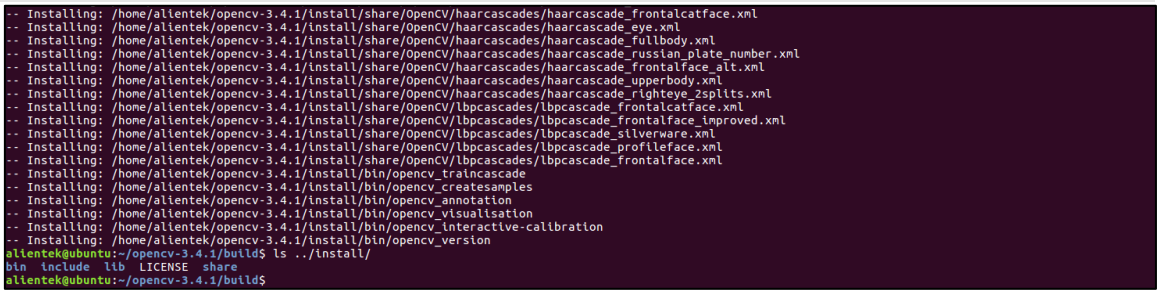


图 2.3 4 安装成功和查看安装

至此我们已经编译完成 OpenCV 了。将当前目录下的 lib 文件夹下的内容拷贝到正点原子 I.MX6U 驱动指南里的 busybox 文件系统/usr/lib 下即可。切记不要拷贝到开发板出厂系统里，因为出厂系统已经有 OpenCV 库，再拷贝过去就会与本地的冲突。

第三章 Qt 项目中加入 OpenCV

本章将介绍如何在 Qt 项目中加入 OpenCV。

3.1 在 Qt 项目中加入 OpenCV

打开 Qt 项目的 pro 文件，在里面添加如下内容。

```
INCLUDEPATH += /home/alientek/opencv/install/include
LIBS += ../../lib/libopencv_core.so \
        ../../lib/libopencv_highgui.so \
        ../../lib/libopencv_imgproc.so \
        ../../lib/libopencv_videoio.so \
        ../../lib/libopencv_imgcodecs.so
```

h 头文件加入以下内容。注根据所需要添加相应的头文件。

```
#include <opencv2/imgproc/imgproc_c.h>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv/cv.h>
```

其中/home/alientek/opencv/install/include 需要改为自己的 OpenCV 路径。Qt 的 pro 文件需要在上一章节 OpenCV 安装目录下的 lib 的前两级目录下才能正常使用！Qt + OpenCV 相关应用程序由读者自由开发。

附录-A