

OpenCV 安装教程

机械是血肉，电控是大脑，视觉是灵魂。

一、安装环境

- Ubuntu 系统版本：20.04 LTS，链接: <https://pan.baidu.com/s/1ojBoCBSHbMVZHhD8HOHYyA> 提取码: 76wv (不建议下载，因为反而慢)
- OpenCV 版本：4.5.3，链接: <https://pan.baidu.com/s/1foen04ULGOwGpwLJUwvS2A> 提取码: vwmq
- OpenCV_contrib版本：需与 OpenCV 一致，链接: <https://pan.baidu.com/s/1wI7IgSBt3sSBjE374Gnksg> 提取码: 7mef

通过 Github 访问可能会非常缓慢，所以我们提供了百度网盘下载地址。

二、前言

OpenCV（开源的计算机视觉库）是基于 BSD 协议，因此它可免费用于学术和商业用途。其提供 C++、C、Python 和 Java 接口，支持 Windows、Linux、Mac OS、iOS 和 Android。OpenCV 致力于高效运算和即时应用开发。因其是用优化的 C/C++ 编写的，故其可以充分利用多核处理优势，并且还启用了 OpenSL，它可以利用底层异构计算平台的硬件加速。可以说，想要入门机器视觉，安装 OpenCV 是躲不掉的第一步。我在 RoboMaster 的组员一年后还在参考此教程，足见它的实用性了。

配置环境是做项目的第一步，也是非常重要的一步，如果你希望培养自己的这种能力，我非常鼓励你先自己查找资料探索整个安装步骤，并给出你可能需要查阅资料用到的关键词：Git、OpenCV、OpenCV_contrib、cmake/cmake-gui、以及可能出现的问题：权限不足、文件下载失败（TimeOut）等，这些问题一般只需要搜索关键字或将报错的句子复制到搜索引擎中，就能找到解决方法，其中大部分问题在 CSDN 中都有解答，少部分问题需要查阅 Overstackflow 查看英文解决方法，查阅官方资料（实际上这是最重要的一种方式）和论坛也是一种不错的方法。

三、OpenCV 安装详细步骤

准备阶段

一般来说我们的电脑都是 Windows 和 MacOS 为操作系统的，虽然在它们上面也能进行开发，但是使用体验不如 Linux 系统方便快捷。我这么说的原因是：

- 大部分开发环境对于 Linux 的支持更好，操作以终端命令为主，环境部署效率高，而 Windows 和 MacOS 则对图形化界面更友好，舒适度取决于软件开发商
- Linux 本身就是面向服务器而非个人电脑的，因此更适合于大型项目的开发，其系统运行效率更高
- 相比较而言 Linux 更加简洁，很适合喜欢简洁的程序员
- 当然，这都看个人喜好，只要你用得顺手，对个人开发而言其实无所谓

为了拥有一个我们自己的 Linux 系统，有三种方法：

- 安装双系统
- 配置虚拟机
- 再买一台电脑重装 Linux：和安装双系统差不多，不做介绍

如果您的主机性能良好，那么这两个方案都是可选的，如果它的性能不足以同时运行两个系统，那么安装双系统可能是一个比较合理的选择。如果这两个方案都是可选的，那么首先你需要明白它们的优劣：

- 双系统
 - 优点：硬件使用效率更高，所有电脑的硬件将只被一个系统使用；**能够使用 GPU**，如果你的开发需要使用 GPU，如深度学习，那么安装双系统将是你唯一的选

择。

- 缺点：无法同时使用日常的操作系统，切换系统需要重启。

- 虚拟机：

- 优点：可以快速进行操作系统间的切换，以及之间的文件传输；虚拟机环境损坏后重装更方便一些，且拥有**系统快照**功能，方便一键还原

- 缺点：同时运行两个系统，消耗系统资源；虚拟机支持软件需要付费（VM 对学生免费）

由于虚拟机安装更快且坑较少，所以本篇教程现以虚拟机安装为主，而对双系统教程将会另外出一篇教程讲述。一般配置的 Linux 虚拟机至少需要分配 1 核 2 GB 才能运行起来，如果要运行比较耗资源的项目的话，2 核 4 GB 是推荐的选择，当然，SSD 的速度能在一定程度上弥补内存的不足。在安装完虚拟机后，仍应留有一定的资源供主机使用。一种直观的感受是，如果你的虚拟机运行卡顿，那么首先看你装在哪里，如果是机械硬盘，那么情有可原，试着提高分配的内存，不行再提高分配的核心，如果是固态硬盘，那么就意味着分配的核心或内存太少了，根据情况提高核心或内存或同时提高。

如果你选择了双系统，那么请提前规划好你的硬盘容量，因为双系统会在安装时需要固定死系统容量，当然以后改的方法还是有点。如果你选择了虚拟机，那么你需要一个底层的支撑软件，比如 VMware、Parallel Desktop 等，前者作为交大的学生可以在网络信息服务中心申请免费使用。

安装完虚拟机之后，就可以正式开始安装 OpenCV 了。

首先，我们先确保系统安装了一些基本的依赖库，打开终端输入以下命令：

```
sudo apt install git gcc g++ ffmpeg cmake cmake-gui make python3-dev python3-numpy
python3-pip libavcodec-dev libavformat-dev libswscale-dev libgstreamer-plugins-base1.0-
dev libgstreamer1.0-dev libgtk-3-dev libpng-dev libjpeg-dev libopenexr-dev libtiff-dev
libwebp-dev libavresample-dev libtbb-dev
sudo add-apt-repository "deb http://security.ubuntu.com/ubuntu xenial-security main"
sudo apt update
sudo apt install libjasper1 libjasper-dev
```

（非必须，推荐）安装 aptitude：

```
sudo apt install aptitude
```

(非必须) 安装 Boost 和 Eigen3:

```
sudo apt install libboost-all-dev  
sudo apt-get install libeigen3-dev
```

(非必须) 安装 ceres:

首先打开终端, 添加源:

```
sudo gedit /etc/apt/sources.list
```

不然你可能会遇到以下问题:

```
(base) hhj@hhj-virtual-machine:~/Applications/ceres-solver/build$ sudo apt-get install liblapack-dev libsuitesparse-dev libcxsparse3.1.2 libgflags-dev libgoogle-glog-dev libgtest-dev  
[sudo] password for hhj:  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
E: Unable to locate package libcxsparse3.1.2  
E: Couldn't find any package by glob 'libcxsparse3.1.2'  
E: Couldn't find any package by regex 'libcxsparse3.1.2'
```

在文件中加入:

```
deb http://cz.archive.ubuntu.com/ubuntu trusty main universe
```

然后关闭, 在终端输入以下命令, 更新源并安装依赖库:

```
sudo apt update  
sudo apt-get install liblapack-dev libsuitesparse-dev libcxsparse3.1.2 libgflags-dev  
libgoogle-glog-dev libgtest-dev
```

然后 `git clone https://github.com/ceres-solver/ceres-solver.git` 从 GitHub 下载 ceres-solver 至 Applications 文件夹, 在项目目录下打开终端:

```
mkdir build
cd build
cmake ..
make -j$(nproc)
sudo make install
```

(非必须) 安装 qt5, 这样 OpenCV 将会和 Qt 一同编译:

```
sudo apt install qt5-default qtcreator
```

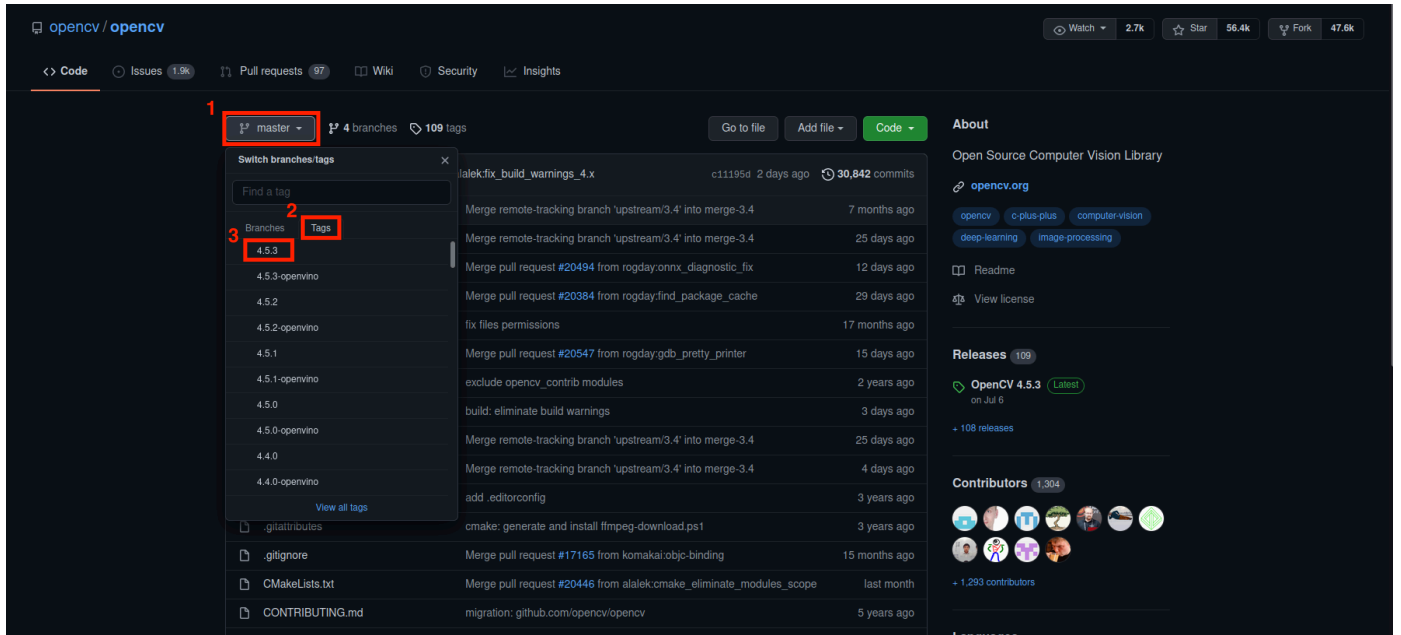
(虚拟机安装不了) CUDA: 教程将会另外发布。

当然还有一些其他的 Ubuntu 配置, 比如中文输入法等, 有需要的话自己查资料配置。这里提一下, 如果你以后希望使用 JetBrains 全家桶 (e.g. Clion、Pycharm) 的话, 那么请不要安装搜狗输入法, 会导致程序异常退出。另外, 搜狗输入法也不支持 Ubuntu 20.04, 请安装 fcitx 下的 Google 输入法。

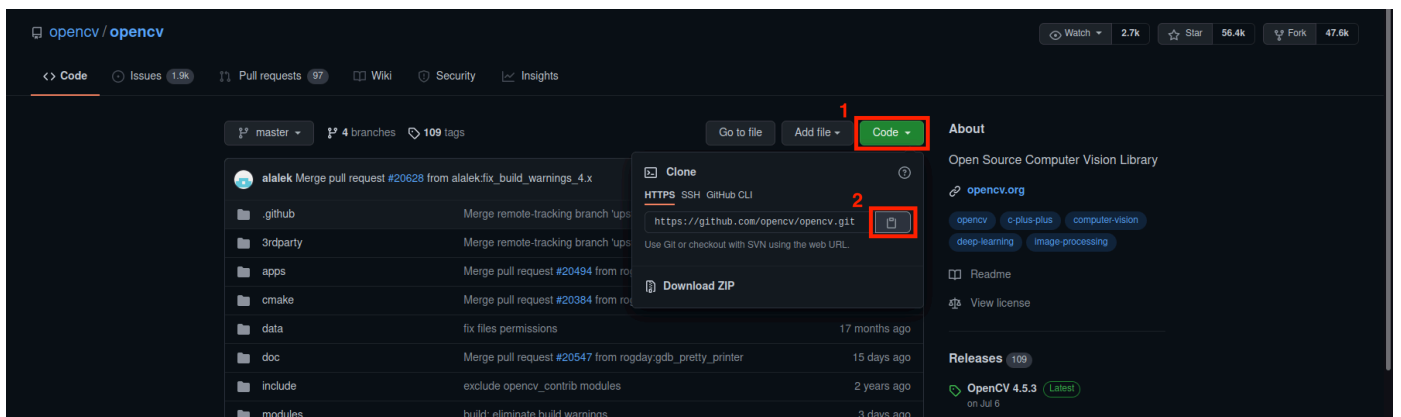
编译阶段

在安装了所需的依赖后, 我们终于可以编译 OpenCV 了。

先从 GitHub 下载最新版的 [OpenCV](#) 和 [OpenCV_contrib](#), 点击下图红框, 选择下拉按钮 master -> Tags -> 4.5.3。



然后点击右侧绿色按钮 Code，如果你打算选择最新的版本，直接复制地址，如果你选择了特定的版本，那么点击 Download ZIP，解压至之后的 Applications 文件夹下。



在 Ubuntu 中打开终端（Terminal），输入以下命令：

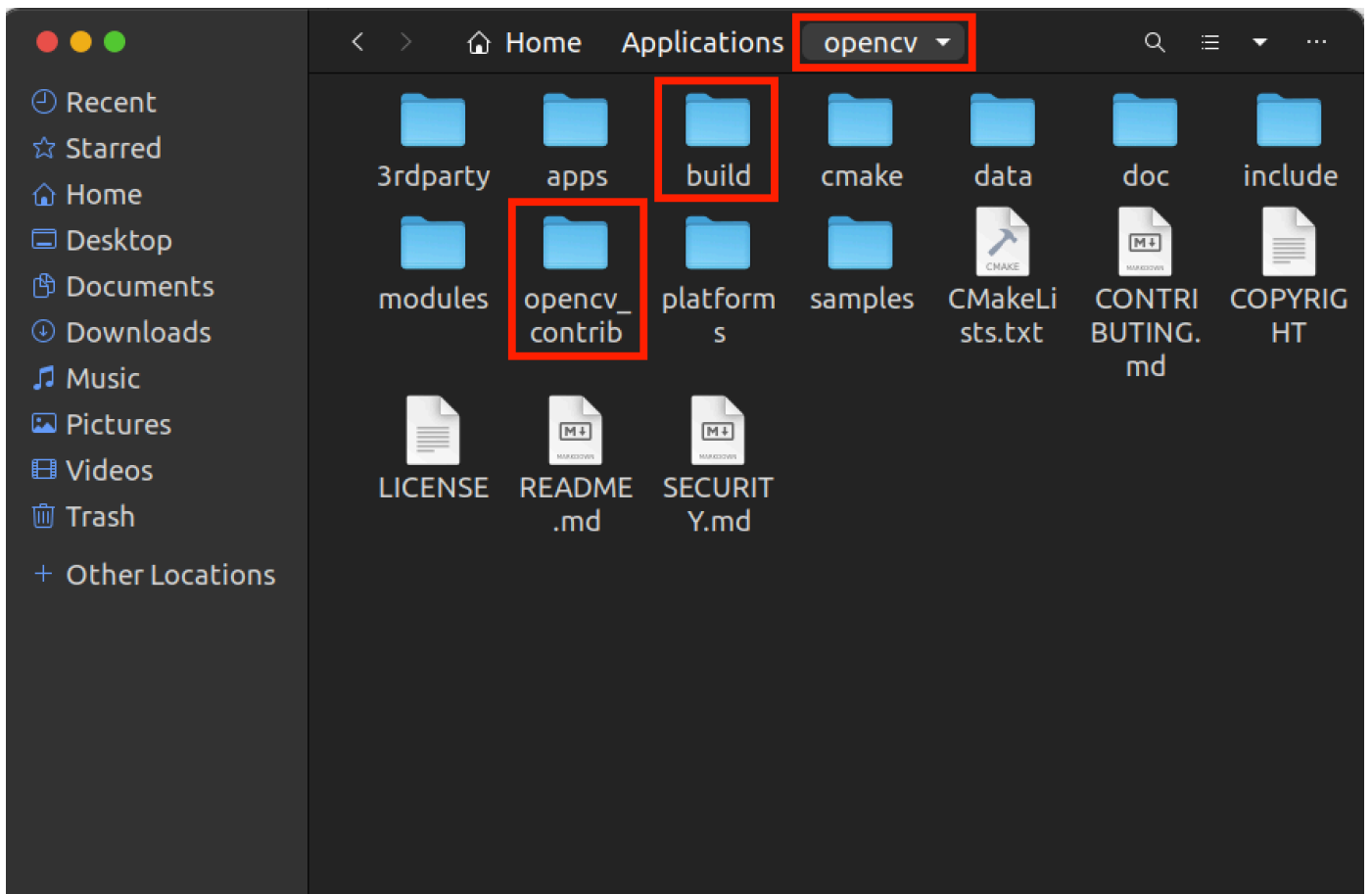
```
cd ~  
mkdir Applications  
cd Applications  
# 以下命令针对复制网址下载的  
git clone <path/of/opencv>  
git clone <path/of/opencv_contrib>
```

将 opencv_contrib 目录移动到 opencv 目录中，以下使用 <opencv> 和 <opencv_contrib> 分别代表着两个文件夹名，根据实际情况带入修改：

```
cd ~/Applications/<opencv>  
mkdir build
```

```
(base) hhj@hhj-virtual-machine:~$ cd Applications/
(base) hhj@hhj-virtual-machine:~/Applications$ git clone https://github.com/opencv/opencv.git
Cloning into 'opencv'...
remote: Enumerating objects: 295127, done.
remote: Total 295127 (delta 0), reused 0 (delta 0), pack-reused 295127
Receiving objects: 100% (295127/295127), 485.49 MiB | 973.00 KiB/s, done.
Resolving deltas: 100% (205154/205154), done.
Updating files: 100% (6880/6880), done.
(base) hhj@hhj-virtual-machine:~/Applications$ cd opencv/
(base) hhj@hhj-virtual-machine:~/Applications/opencv$ ls
3rdparty  CMakeLists.txt  data  LICENSE  README.md
apps      CONTRIBUTING.md  doc   modules  samples
cmake     COPYRIGHT        include  platforms  SECURITY.md
(base) hhj@hhj-virtual-machine:~/Applications/opencv$ git clone https://github.com/opencv/opencv_contrib.git
Cloning into 'opencv_contrib'...
remote: Enumerating objects: 35613, done.
remote: Counting objects: 100% (860/860), done.
remote: Compressing objects: 100% (564/564), done.
remote: Total 35613 (delta 382), reused 580 (delta 244), pack-reused 34753
Receiving objects: 100% (35613/35613), 131.32 MiB | 1.30 MiB/s, done.
Resolving deltas: 100% (21892/21892), done.
(base) hhj@hhj-virtual-machine:~/Applications/opencv$ mkdir build
```

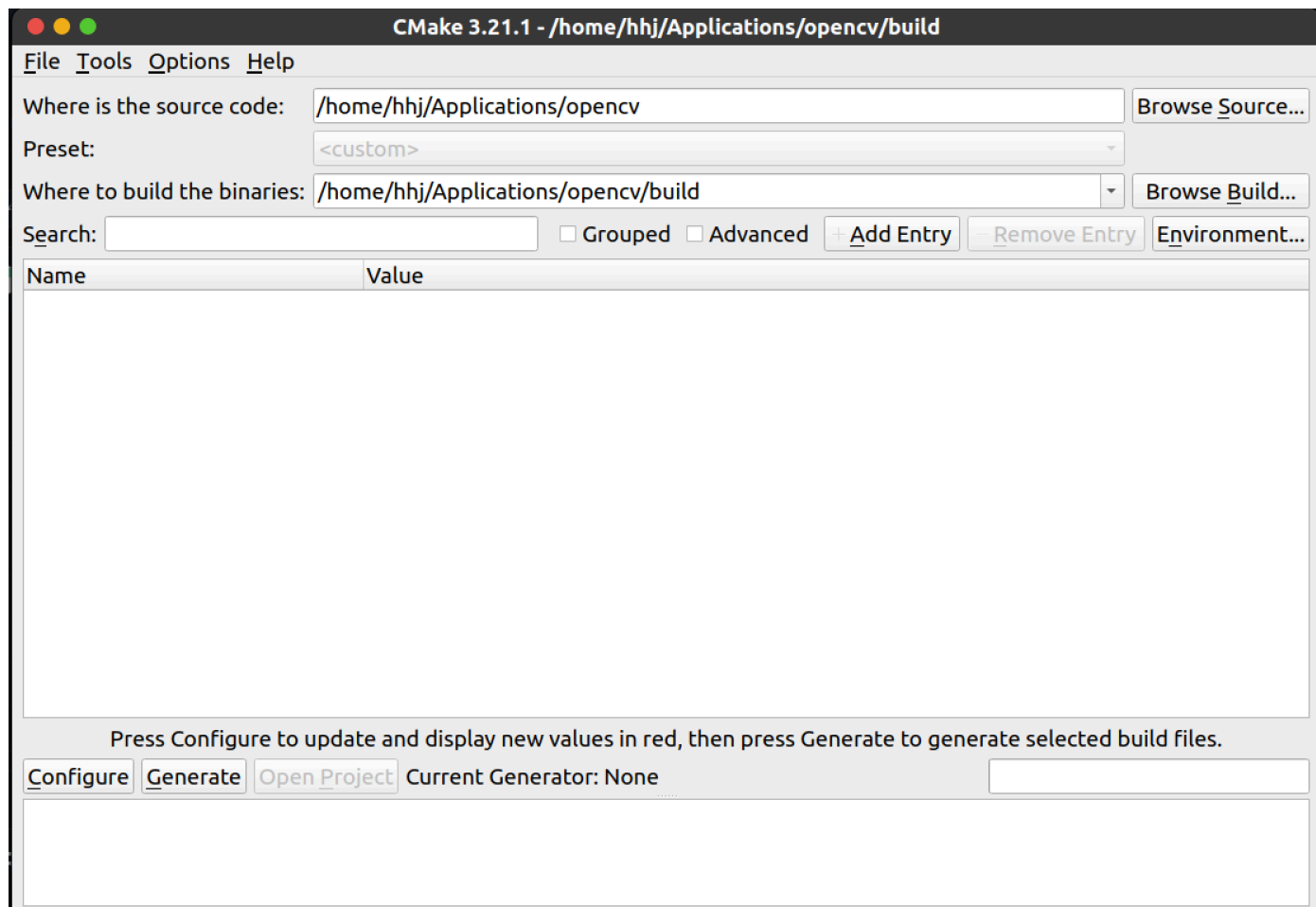
最终应该如下图：



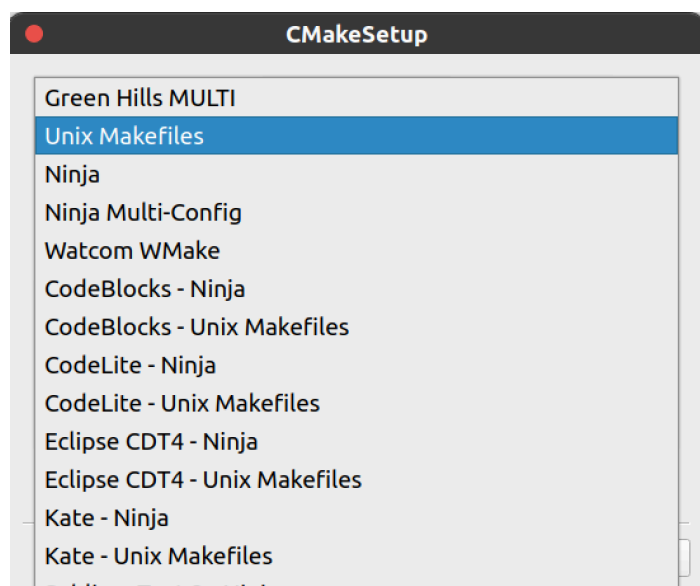
接下来借助 `cmake-gui` 编译，打开终端输入：

cmake-gui

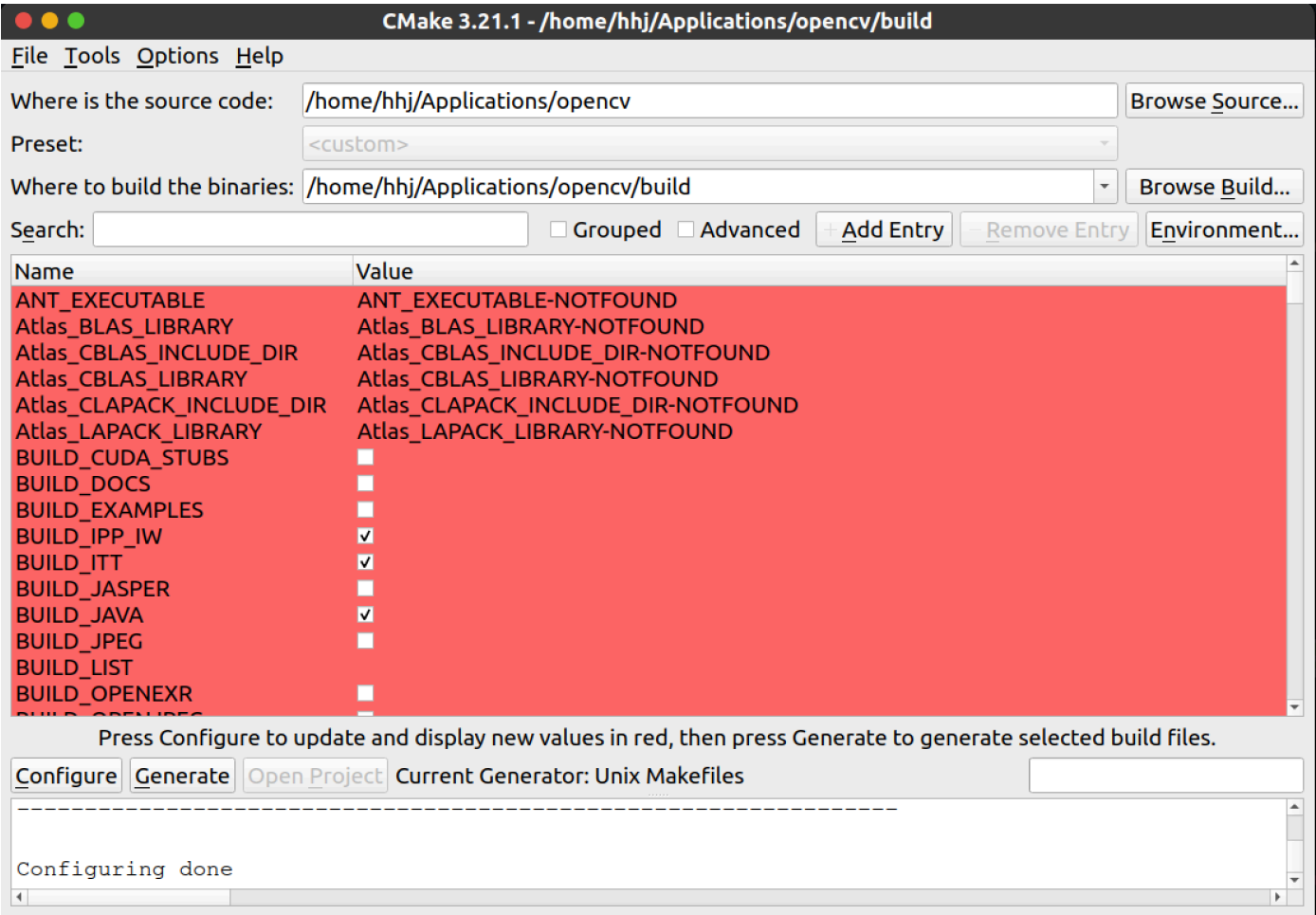
在出现的界面中，点击 Browse Source 选择源文件目录 opencv，点击 Browse Build 选择编译文件存放目录 build，然后点击 Configure。



会跳出一个弹窗，下拉框中选择 Unix Makefiles，然后点击 Finish。

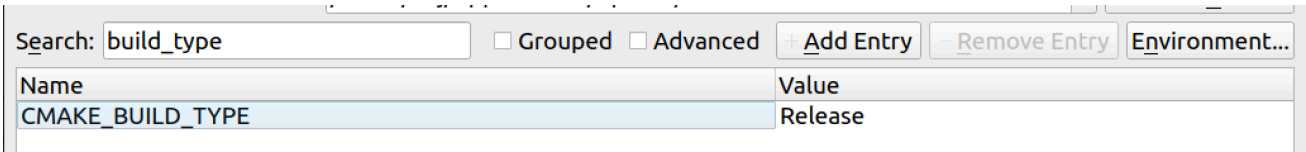


完成后界面如下：

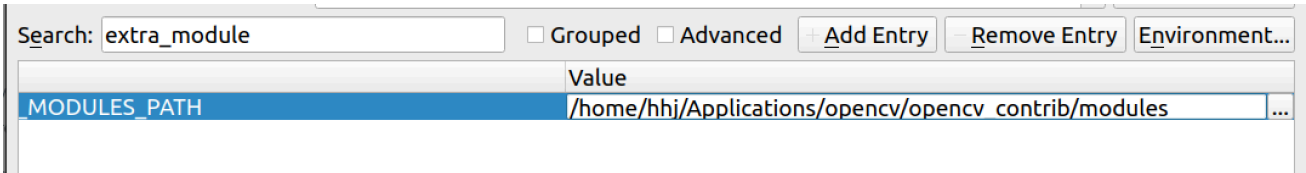


然后我们需要修改两个地方：

1. 通过 search 找到 CMAKE_BUILD_TYPE 处，输入 Release 。



2. 在 OPENCV_EXTRA_MODULES_PATH 处加入 opencv_contrib 模块路径。注意，不是选 opencv_contrib 文件夹，而是需要选中 opencv_contrib 里面的 modules 文件夹！

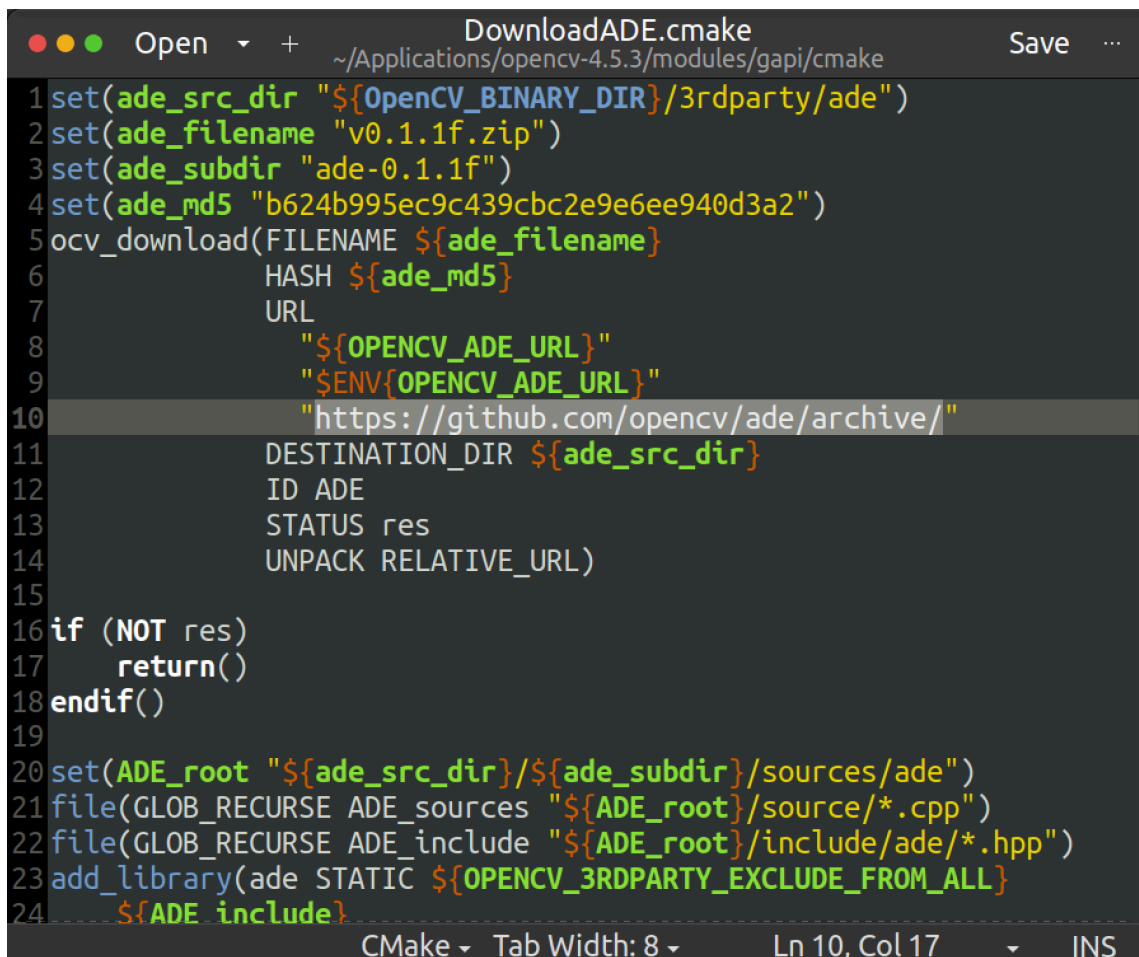


这里说明一下，CMAKE_INSTALL_PREFIX 为安装路径，系统默认为 /usr/local，如若对 Ubuntu 不熟悉，则不要更改，默认就好。

确认无误后，点击 Configure；先排除四个常见的问题：

1. ade* 无法下载：手动下载（链接：<https://pan.baidu.com/s/1oVUeBL6cbxeczRAd22-CHw> 提取码: j2v7），存放在 Downloads 文件夹里，打开

<opencv>/modules/gapi/cmake/DownloadADE.cmake，把第 10 行更换成自己下载的文件目录路径，例如 /home/<username>/Downloads/，其中 <username> 代表你的用户名。重新 Configure。



```
1 set(ade_src_dir "${OpenCV_BINARY_DIR}/3rdparty/ade")
2 set(ade_filename "v0.1.1f.zip")
3 set(ade_subdir "ade-0.1.1f")
4 set(ade_md5 "b624b995ec9c439cbc2e9e6ee940d3a2")
5 ocv_download(FILENAME ${ade_filename}
6               HASH ${ade_md5}
7               URL
8                 "${OPENCV_ADE_URL}"
9                 "${ENV{OPENCV_ADE_URL}}")
10                "https://github.com/opencv/ade/archive/"
11                DESTINATION_DIR ${ade_src_dir}
12                ID ADE
13                STATUS res
14                UNPACK RELATIVE_URL)
15
16 if (NOT res)
17     return()
18 endif()
19
20 set(ADE_root "${ade_src_dir}/${ade_subdir}/sources/ade")
21 file(GLOB_RECURSE ADE_sources "${ADE_root}/source/*.cpp")
22 file(GLOB_RECURSE ADE_include "${ADE_root}/include/ade/*.hpp")
23 add_library(ade STATIC ${OPENCV_3RDPARTY_EXCLUDE_FROM_ALL}
24               ${ADE_include})
```

2. ippicv_* 无法下载：手动下载（链接：<https://pan.baidu.com/s/1g6gZ8CrvdE9VWGmp8XmDyw> 提取码: hbor），存放在 Downloads 文件夹里，打开

<opencv>/3rdparty/ippicv/ippcv.cmake，把 42 行路径更换成自己的下载的文件目录路径，例如 /home/<username>/Downloads/，其中 <username> 代表你的用户名。重新 Configure。

```
Open + ippicv.cmake Save ...
~/Applications/opencv-4.5.3/3rdparty/ippicv
30 set(OPENCV_ICV_HASH "cd39bdf0c2e1cac9a61101dad7a2413e")
31 endif()
32 else()
33 return()
34 endif()
35
36 set(THE_ROOT "${OpenCV_BINARY_DIR}/3rdparty/ippicv")
37 ocv_download(FILENAME ${OPENCV_ICV_NAME}
38             HASH ${OPENCV_ICV_HASH}
39             URL
40               "${OPENCV_IPPICV_URL}"
41               "$ENV{OPENCV_IPPICV_URL}"
42               "https://raw.githubusercontent.com/opencv/-
opencv_3rdparty/${IPPICV_COMMIT}/ippicv/"
43             DESTINATION_DIR "${THE_ROOT}"
44             ID IPPICV
45             STATUS res
46             UNPACK RELATIVE_URL)
47
48 if(res)
49 set(${root_var} "${THE_ROOT}/${OPENCV_ICV_PACKAGE_SUBDIR}"
PARENT_SCOPE)
50 endif()
51 endfunction()
```

3. face_landmark_model.dat 无法下载：手动下载（链接: <https://pan.baidu.com/s/1DKkQTfAY-F91r9vZ6qPUYw> 提取码: pe82），存放在 Downloads 文件夹里，打开相应的配置文件 <opencv>/<opencv_contrib>/modules/face/CMakeLists.txt，将 CMakeLists.txt 文件的第 19 行修改为本地路径，即将原来的网址修改为下载的文件保存的路径，"/home/<username>/Downloads/"，其中 <username> 代表你的用户名。重新 Configure。

```
Open + CMakeLists.txt Save ...
~/Applications/opencv-4.5.3/opencv_contrib-4.5.3/mod...

1 set(the_description "Face recognition etc")
2 ocv_define_module(face opencv_core
3   opencv_imgproc
4   opencv_objdetect
5   opencv_calib3d # estimateAffinePartial2D() (trainFacemark)
6   opencv_photo # seamlessClone() (face_swap sample)
7   WRAP python java objc
8 )
9 # NOTE: objdetect module is needed for one of the samples
10
11 set(__commit_hash "8afa57abc8229d611c4937165d20e2a2d9fc5a12")
12 set(__file_hash "7505c44ca4eb54b4ab1e4777cb96ac05")
13 ocv_download(
14   FILENAME face_landmark_model.dat
15   HASH ${__file_hash}
16   URL
17     "${OPENCV_FACE_ALIGNMENT_URL}"
18     "$ENV{OPENCV_FACE_ALIGNMENT_URL}"
19     "https://raw.githubusercontent.com/opencv/opencv_3rdparty/${__commit_hash}/"
20   DESTINATION_DIR "${CMAKE_BINARY_DIR}/${OPENCV_TEST_DATA_INSTALL_PATH}/cv/face/"
21   ID "data"
22   RELATIVE_URI
```

4. boostdesc_*.i、vgg_generated_*.i：手动下载（链接: <https://pan.baidu.com/s/1VCdMUUm2ipu-fbL2189lFg> 提取码: 88eo），存放在
<opencv>/<opencv_contrib>/modules/xfeatures2d/src/ 路径下即可，无需修改文件。

显示 Configuring done 后（注意上翻看看有没有红色的报错，因为即使报错也是会显示 Configuring done 的，当然以上这些报错你不解决其实不会影响多少），点击 Generate 生成文件；这时资源文件就出现在 build 文件夹中，我们可以关闭 cmake 界面了。

在 <opencv>/build 中打开终端，输入：

```
make -j$(nproc)
```

此步骤将花费较长时间，尤其是开启 CUDA 选项时。

完成后，输入：

```
sudo make install
```

至此 OpenCV 已经完成安装了，你可以删除 <opencv> 整个文件夹。

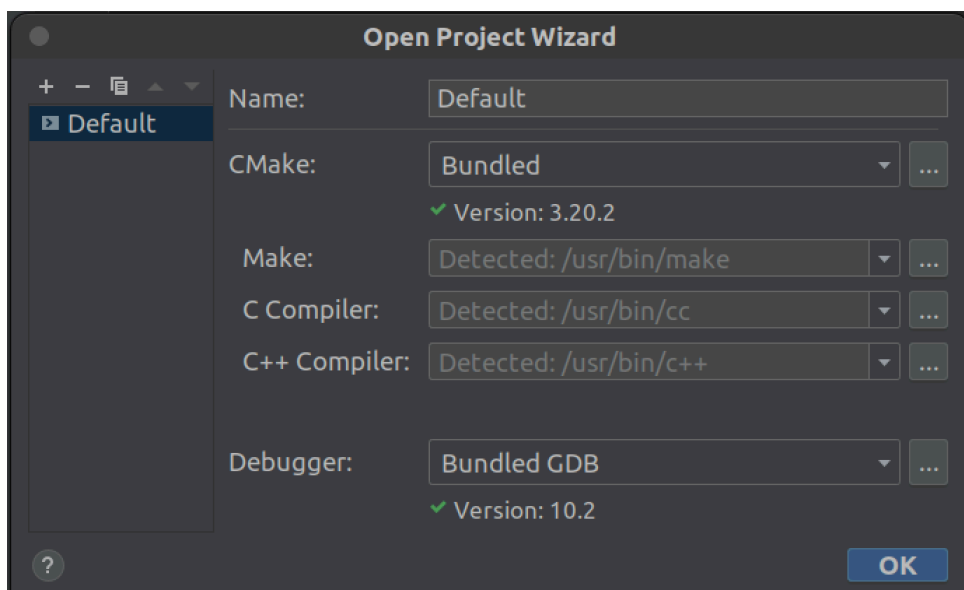
四、IDE 安装

从事项目开发，一款得心应手的 IDE 是必不可缺的。笔者目前使用下来有两款 IDE 比较好用：

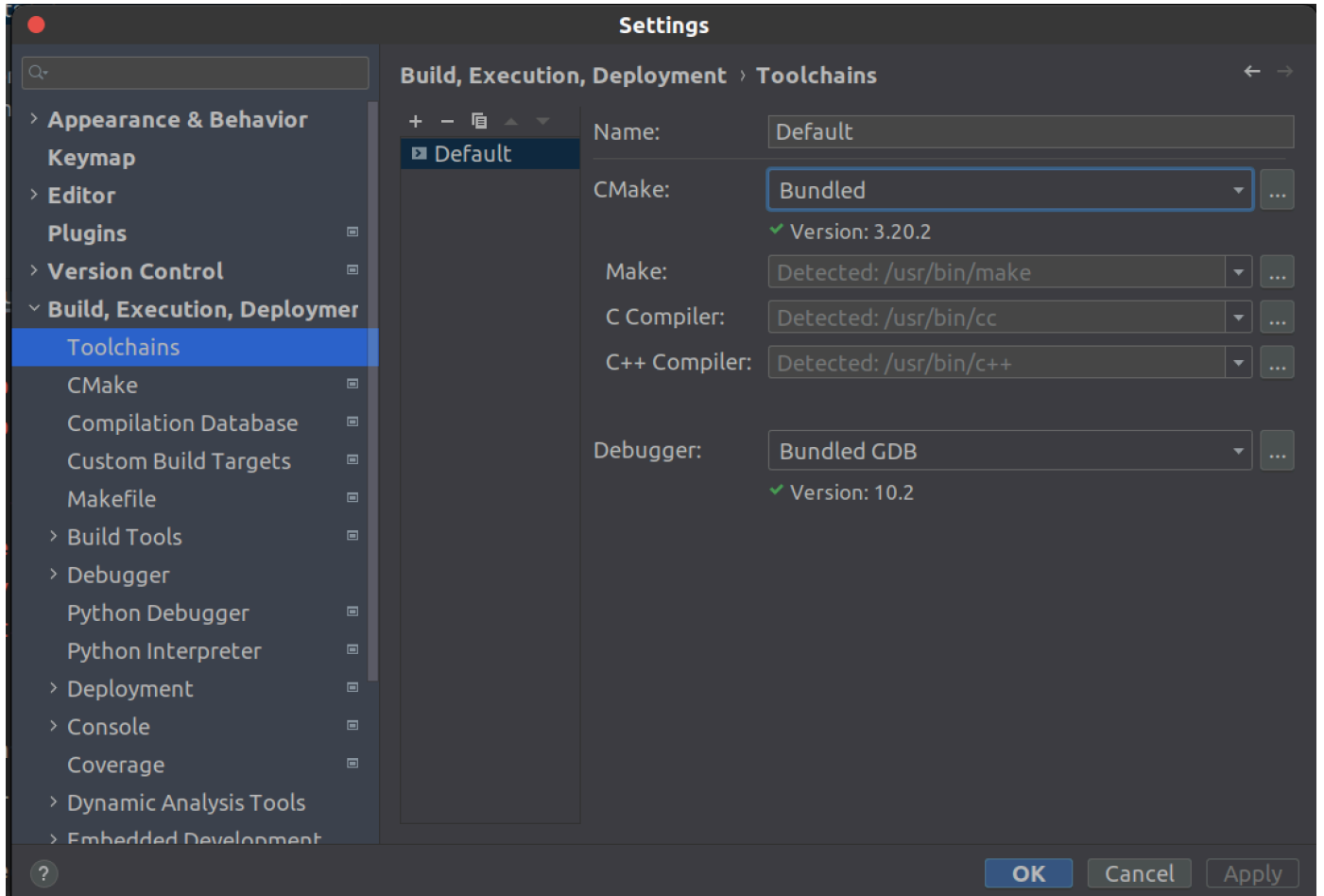
- Clion
 - 胜在集成功能全面，配置方便，几乎不用配置任何调试环境就能使用
 - 安装方式：Jetbrain 全家桶对于高校学生免费开放，你可以进入[官网](#)申请。选择 For students and teachers 下的 learn more，用自己的学校邮箱申请，然后打开邮箱内的确认邮件。然后创建自己的 JetBrains Account，在软件安装完之后的 activate 过程中输入账号密码就可以使用了。
- VScode
 - 胜在插件多，支持自定义，适合编程熟练的老手或追求自定义效果的用户，但需另外为项目配置 C++ 调试环境。
 - 安装方式：进入[官网](#)下载，

Clion 环境配置

初次进入 Clion，需要配置 Toolchains，创建一个新项目会出现以下弹窗，一般系统会自动检测 cmake 路径，如果没有那么手动选择以下，其余的会自动检测的。

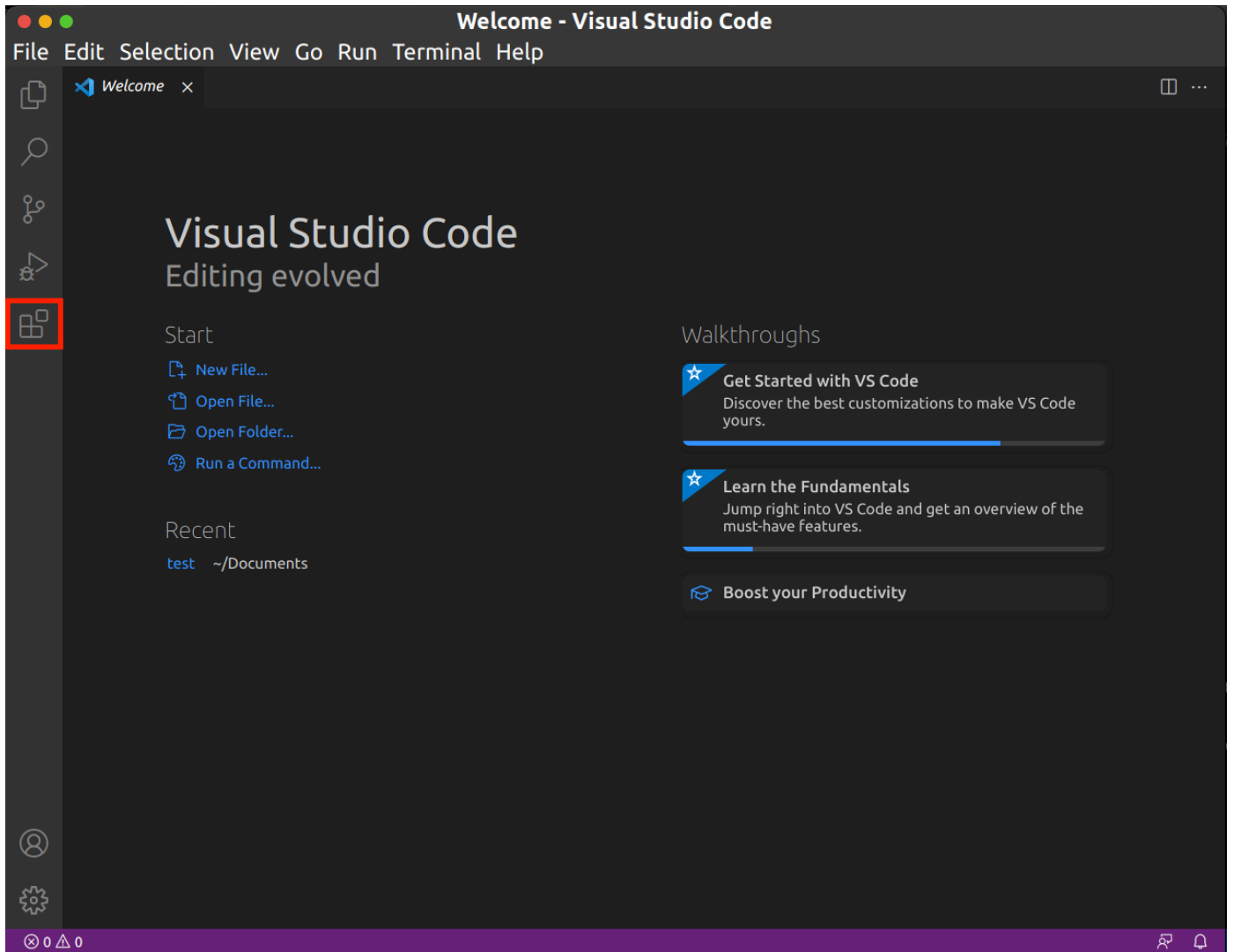


如果没有跳出以上弹窗，那么点击左上角 CLion -> Settings 或 Preferences -> Build, Execution, Deployment -> Toolchains，配置 Default。

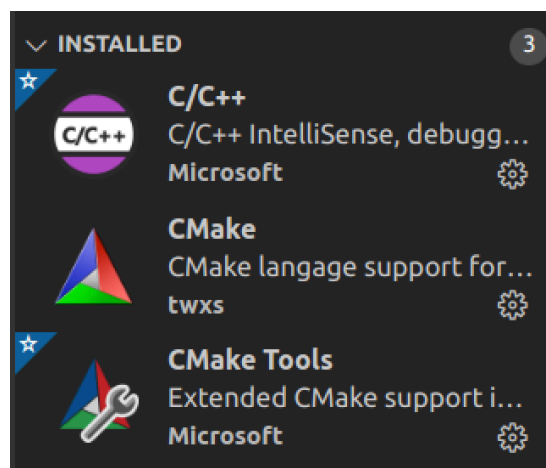


VScode 环境配置

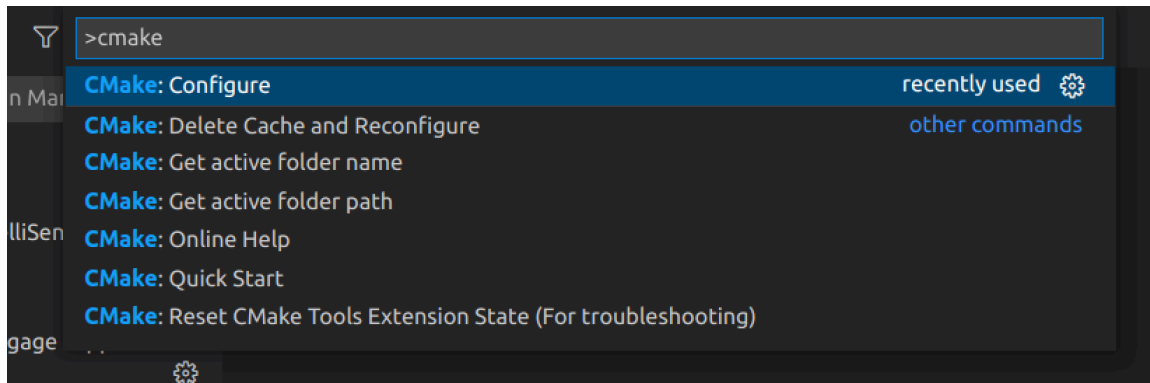
前往 Ubuntu Software 下载 Visual Studio Code ， 然后打开界面如下：



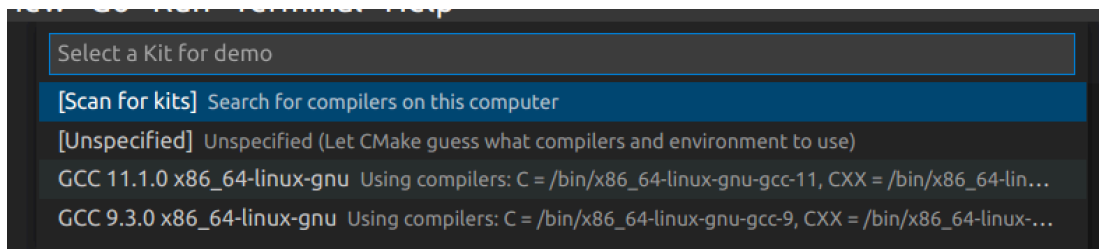
点击图中红色区域的按，进入 Extensions ，分别搜索安装以下三个插件：



安装完毕后，在任意位置创建一个空目录。VScode 不支持单文件编译，必须具有项目目录。创建完毕后，用 VScode 打开项目目录，点击 Shift+Ctrl+P （Ubuntu 版本快捷键），搜索 `cmake:configure`：

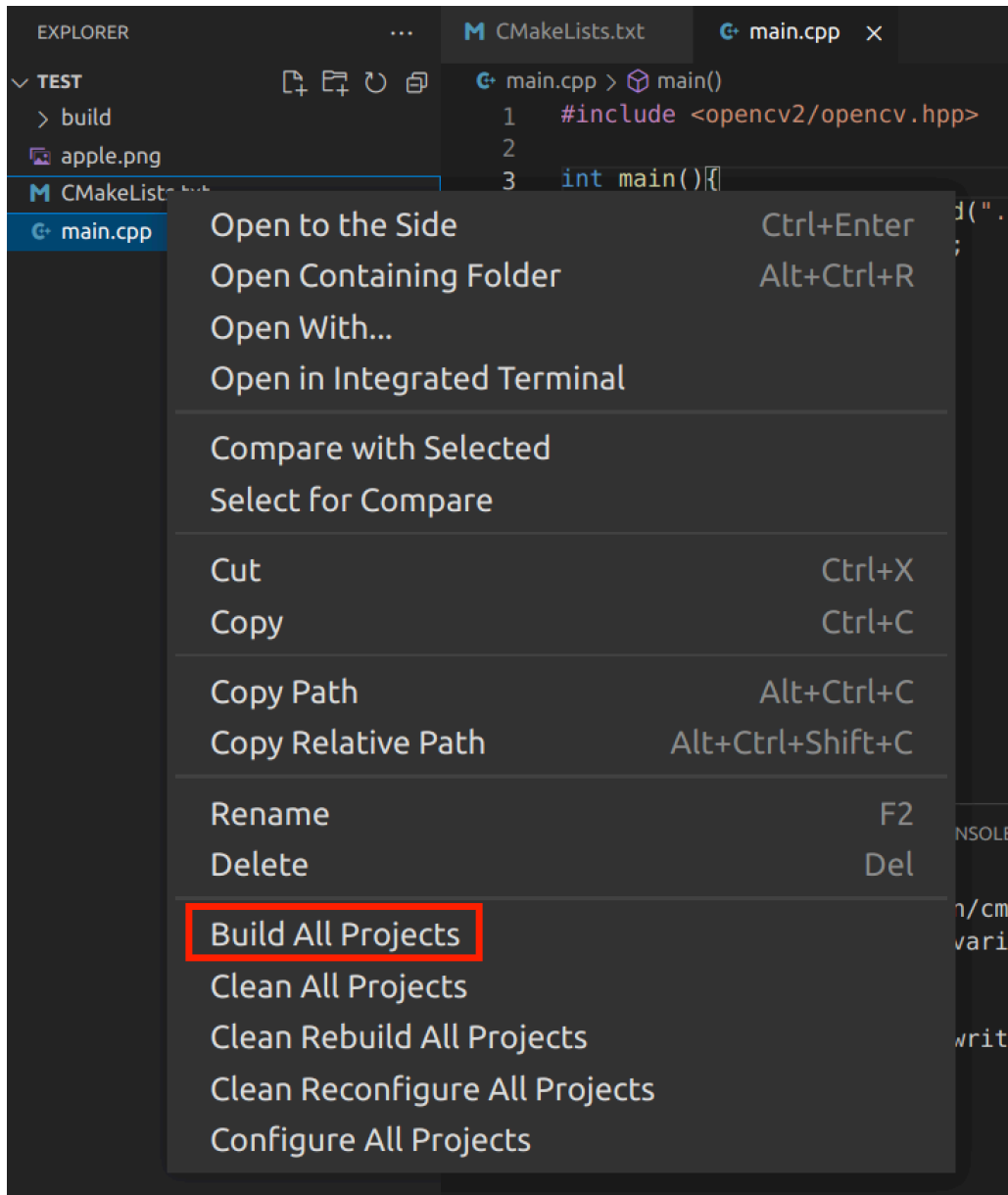


对每个项目首次进入时，会提示选择工具包，这里我们选择 GCC：



在左侧可以先建文件，使用以下示例代码测试。

以后每次编译项目，只需要简单地右击 CMakeLists.txt，选择 Build All Projects：



编译完成后，点击下方 Terminal ，输入以下命令运行程序：

```
cd build
./<name/of/binary_file> # 一般与项目同名
```

五、示例代码

点击下载，链接: <https://pan.baidu.com/s/1nCnebXdNe1XW4e1xzbrXOA> 提取码: 80dc。

最终运行程序应该出现一张图片显示.apple.png 的窗口。

恭喜你安装成功!

六、参考教程

1. [OpenCV学习笔记（一）OpenCV简介及安装](#)
2. [VSCode + CMake搭建C/C++开发环境（MacOS篇）](#)

作者：Harry-hhj, github主页：[传送门](#)