Window系统下配置QT6 + opencv

前言

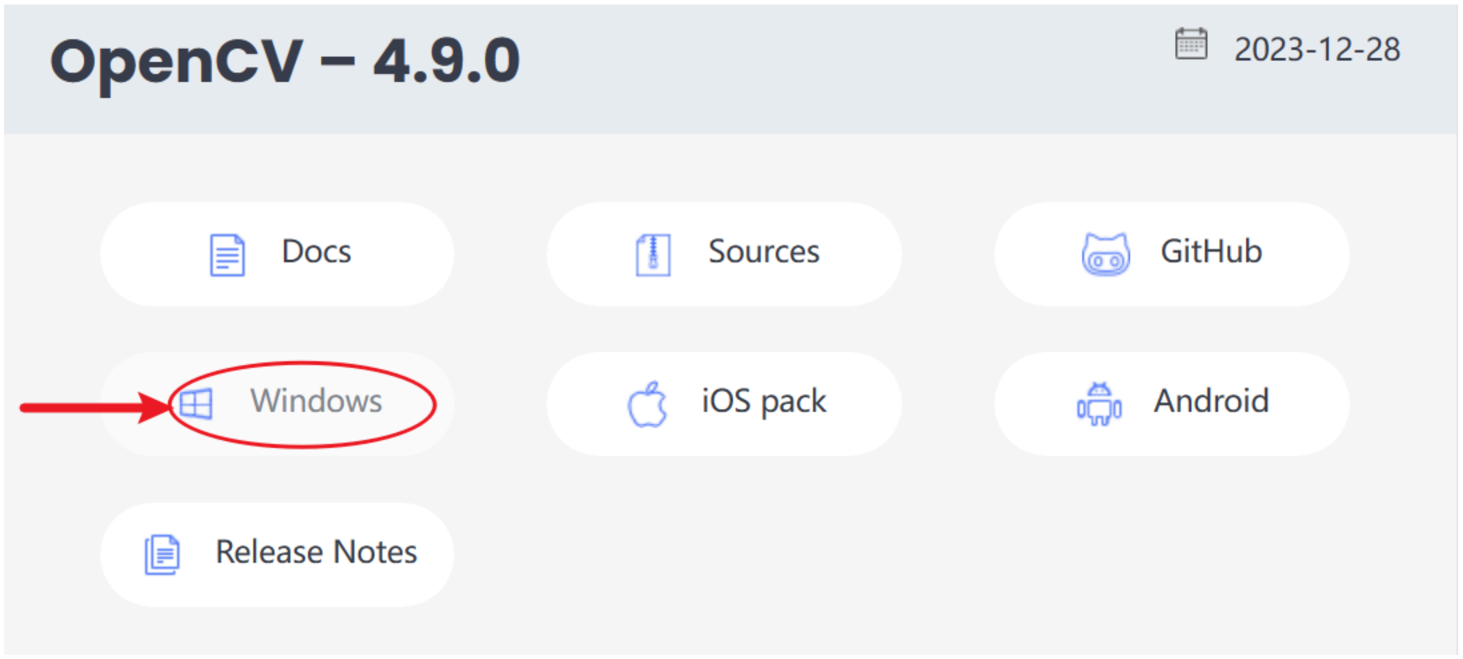
虽然网上有很多教程，但它们通常涉及过时的环境。作为开发者，我们习惯于使用最新的版本。经过一系列的尝试和探索，我们终于成功配置了环境。目前，我和我的一位朋友已经顺利完成了编译。由于我们的样本数量有限，我们不确定在过程中是否会遇到其他问题。如果您在编译过程中遇到问题，欢迎随时咨询我，我会尽力提供帮助。当然，在联系我之前，建议您先在网上搜索相关帖子，看看是否有现成的解决方案
邮箱地址：krivewyh@163.com

一、环境介绍

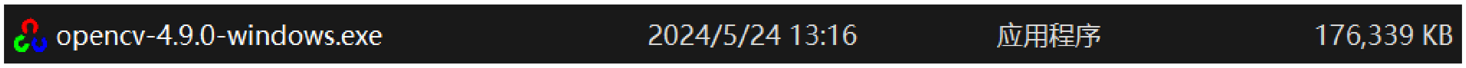
> 系统：Windows11
> QT的IDE版本：6.7.1
> OpenCV版本：4.9.0
> CMake版本：3.29.3

一、Opencv下载

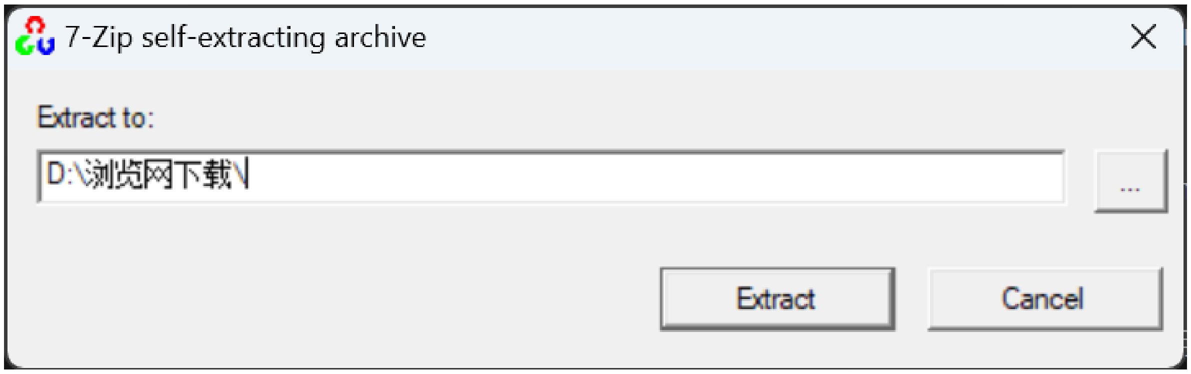
opencv下载



之后打开这个文件，进行opencv源码的解压


opencv-4.9.0-windows.exe 2024/5/24 13:16 应用程序 176,339 KB

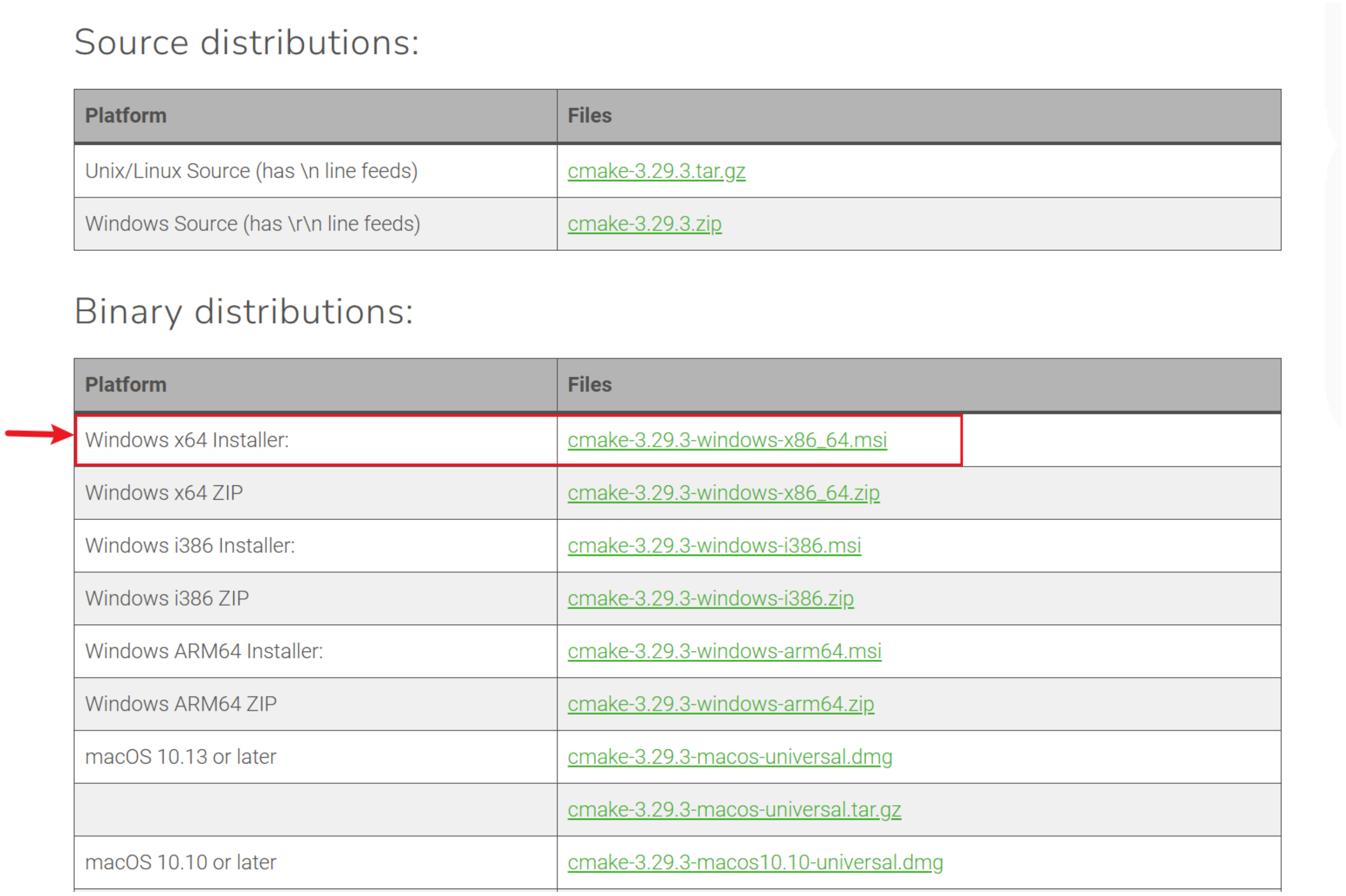
自己选择路径进行解压



二、CMake下载

cmake

选择下列版本



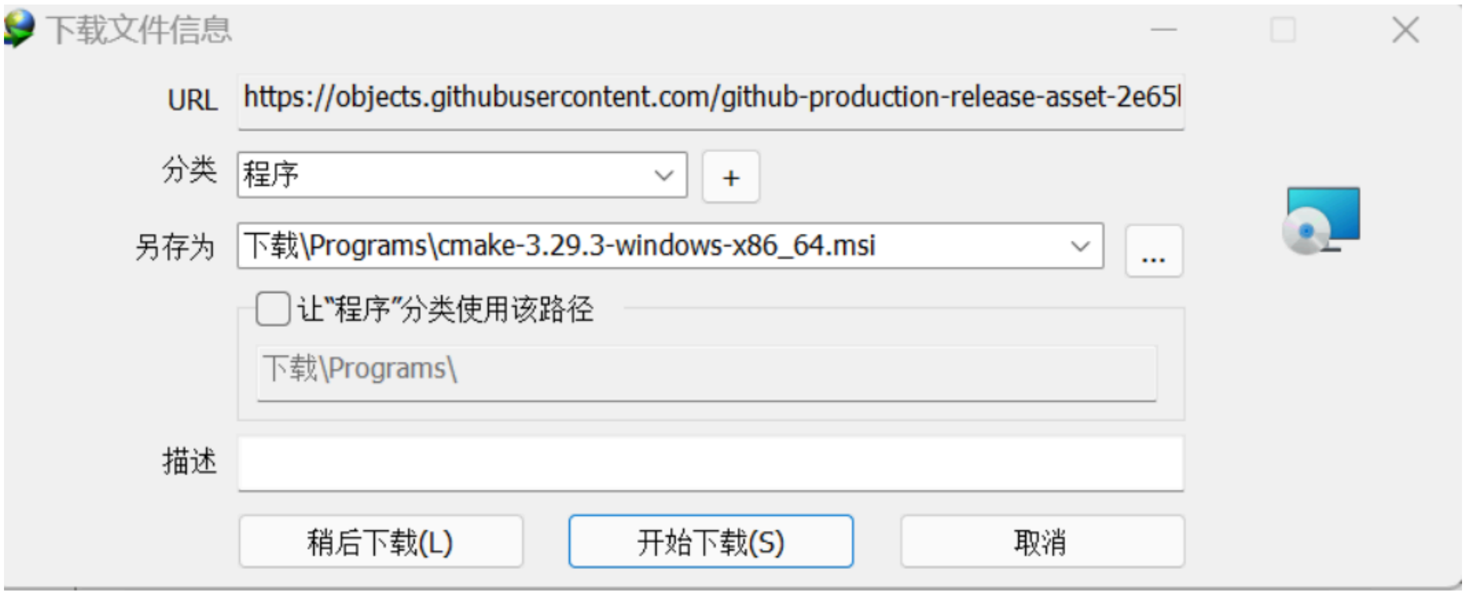
Source distributions:

Platform	Files
Unix/Linux Source (has \n line feeds)	cmake-3.29.3.tar.gz
Windows Source (has \r\n line feeds)	cmake-3.29.3.zip

Binary distributions:

Platform	Files
Windows x64 Installer:	cmake-3.29.3-windows-x86_64.msi
Windows x64 ZIP	cmake-3.29.3-windows-x86_64.zip
Windows i386 Installer:	cmake-3.29.3-windows-i386.msi
Windows i386 ZIP	cmake-3.29.3-windows-i386.zip
Windows ARM64 Installer:	cmake-3.29.3-windows-arm64.msi
Windows ARM64 ZIP	cmake-3.29.3-windows-arm64.zip
macOS 10.13 or later	cmake-3.29.3-macos-universal.dmg
	cmake-3.29.3-macos-universal.tar.gz
macOS 10.10 or later	cmake-3.29.3-macos10.10-universal.dmg

选择你的下载路径


下载文件信息

URL https://objects.githubusercontent.com/github-production-release-asset-2e65l

分类 程序

另存为 下载\Programs\cmake-3.29.3-windows-x86_64.msi

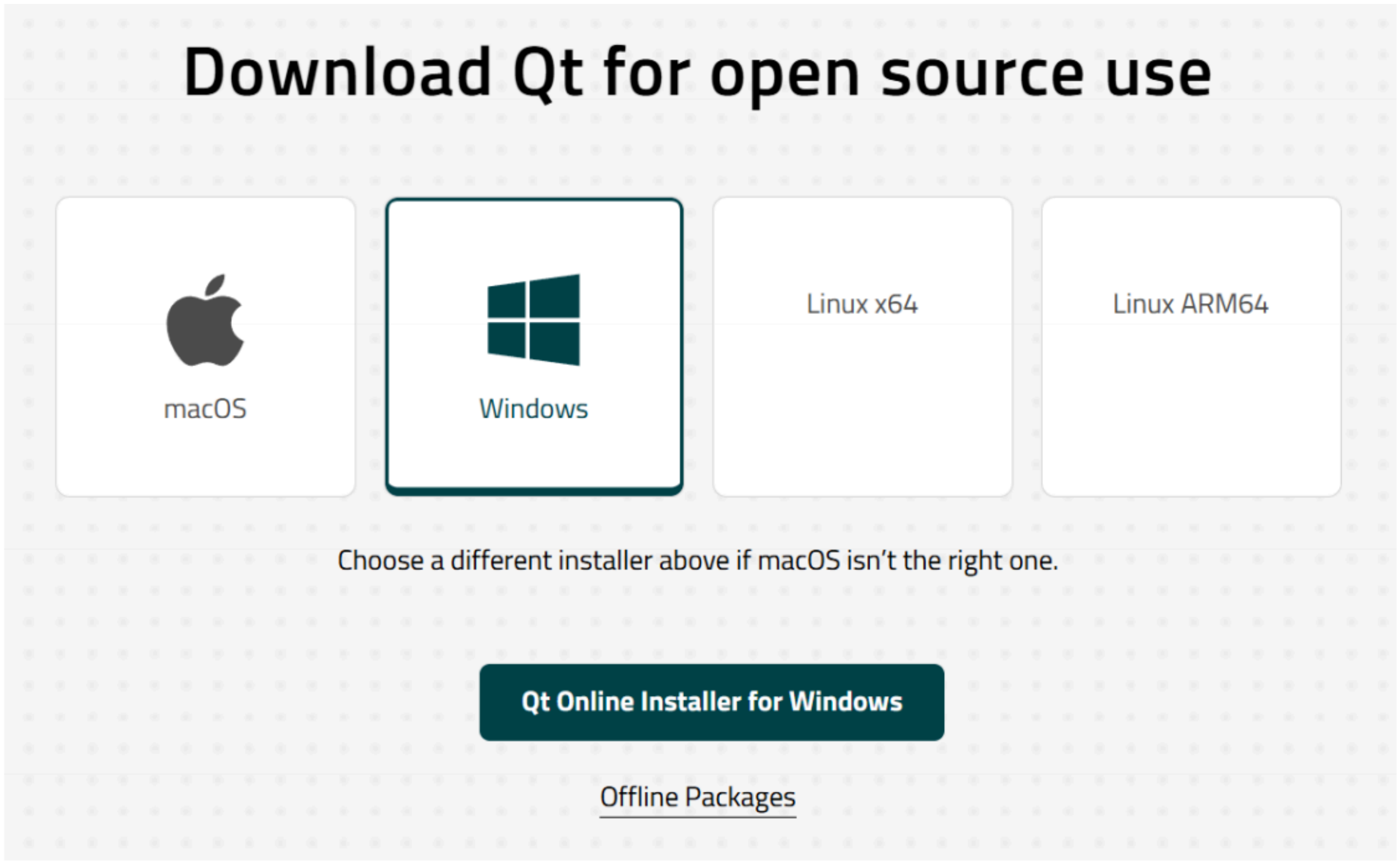
☐ 让"程序"分类使用该路径

下载\Programs\

描述

三、QT6下载

QT下载


Download Qt for open source use

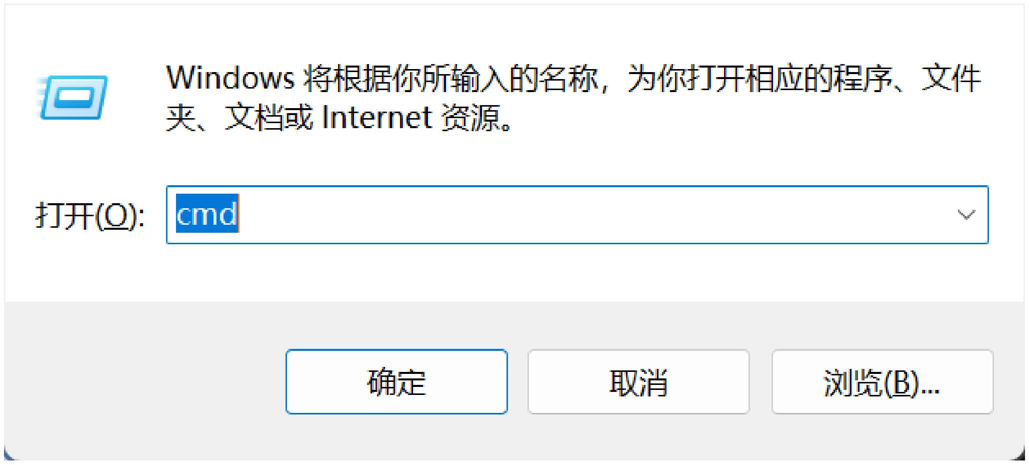
Choose a different installer above if macOS isn't the right one.

下载windows版本，点击 `Qt Online Install for Windows`

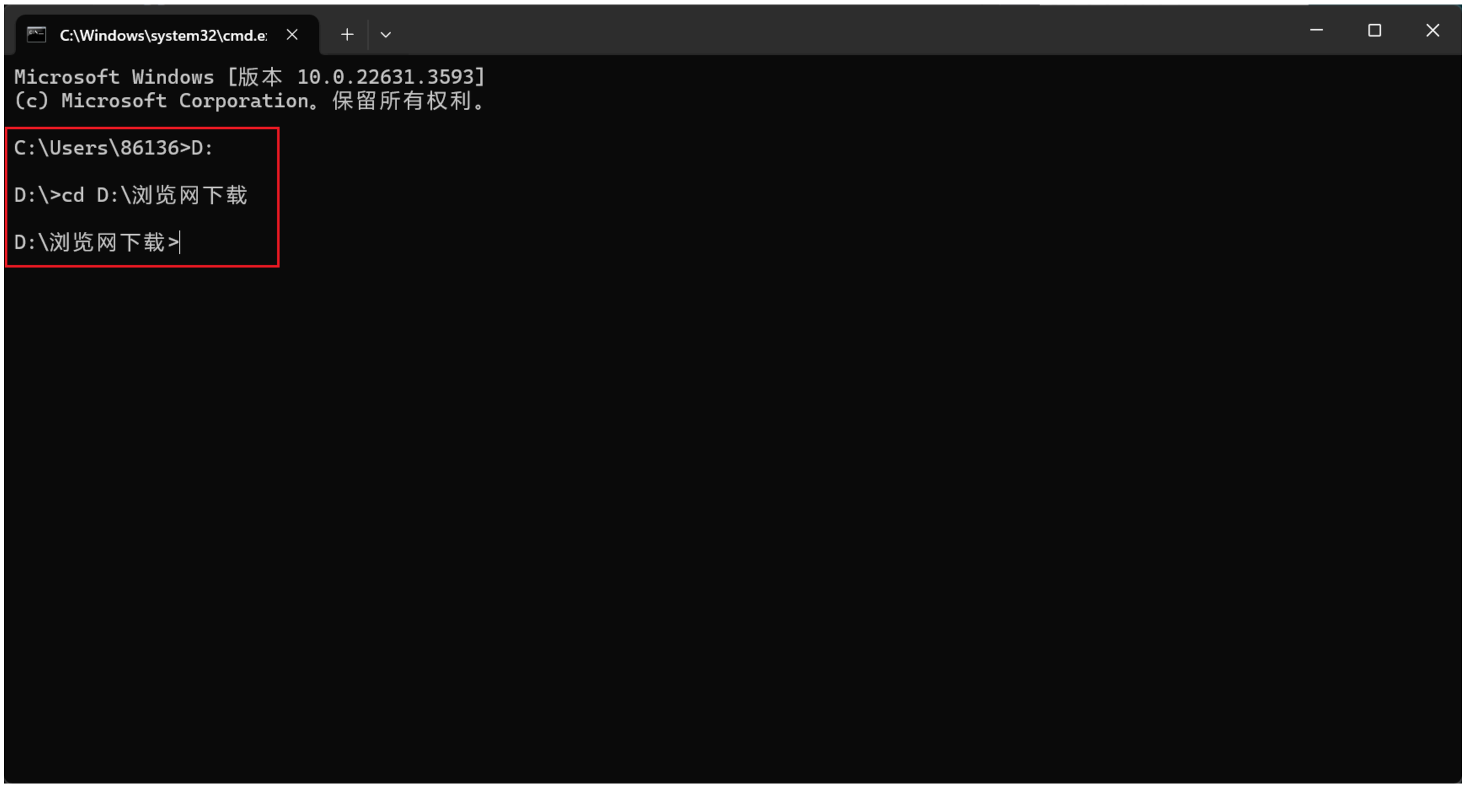
之后选择选择你的下载路径并且记住
此时不要打开下载安装，因为Qt目前使用的是国外的源，下载会很慢，有可能失败。

换源具体做法如下：

打开终端,使用 `win+R` ,输入 `cmd`


Windows 将根据你所输入的名称，为你打开相应的程序、文件夹、文档或 Internet 资源。
打开(O): cmd
确定
取消
浏览(B)...

进入你的目标文件夹，具体如下：



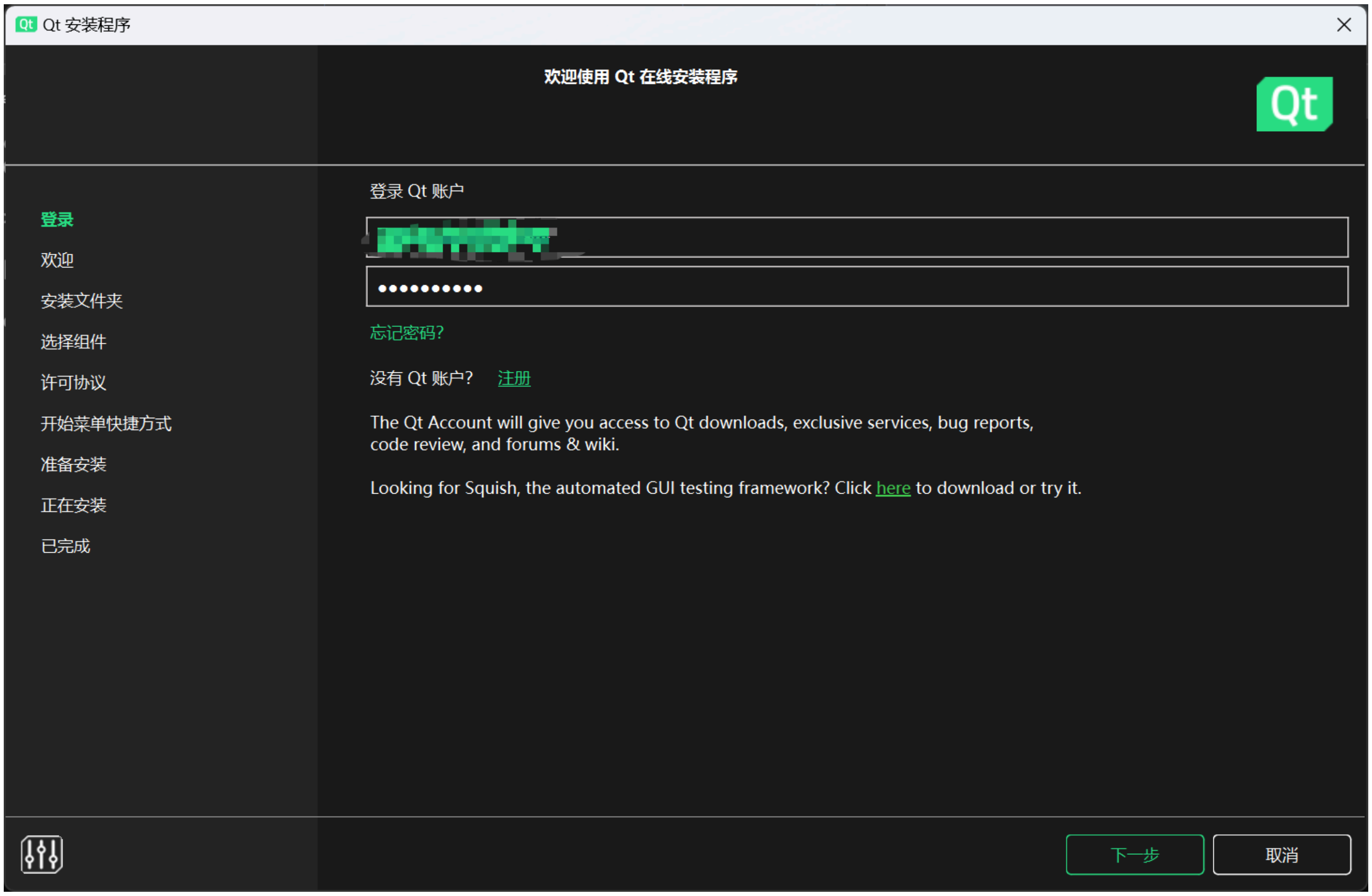
```
Microsoft Windows [版本 10.0.22631.3593]
(c) Microsoft Corporation。保留所有权利。

C:\Users\86136>D:

D:\>cd D:\浏览网下载

D:\浏览网下载>
```

输入输入 `qt-unified-windows-x64-X.X.X-online.exe --mirror https://mirror.nju.edu.cn/qt` ，注意将 `qt-unified-windows-x64-X.X.X-online.exe` ，替换成你下载好的Qt下载器的名称

这里我的是 `qt-online-installer-windows-x64-4.8.0.exe --mirror https://mirror.nju.edu.cn/qt`

之后安装程序会自动打开

需要登陆或注册一个qt账号


Qt 安装程序

欢迎使用 Qt 在线安装程序

登录
欢迎
安装文件夹
选择组件
许可协议
开始菜单快捷方式
准备安装
正在安装
已完成

登录 Qt 账户

[redacted]

••••••••••

忘记密码?

没有 Qt 账户? 注册

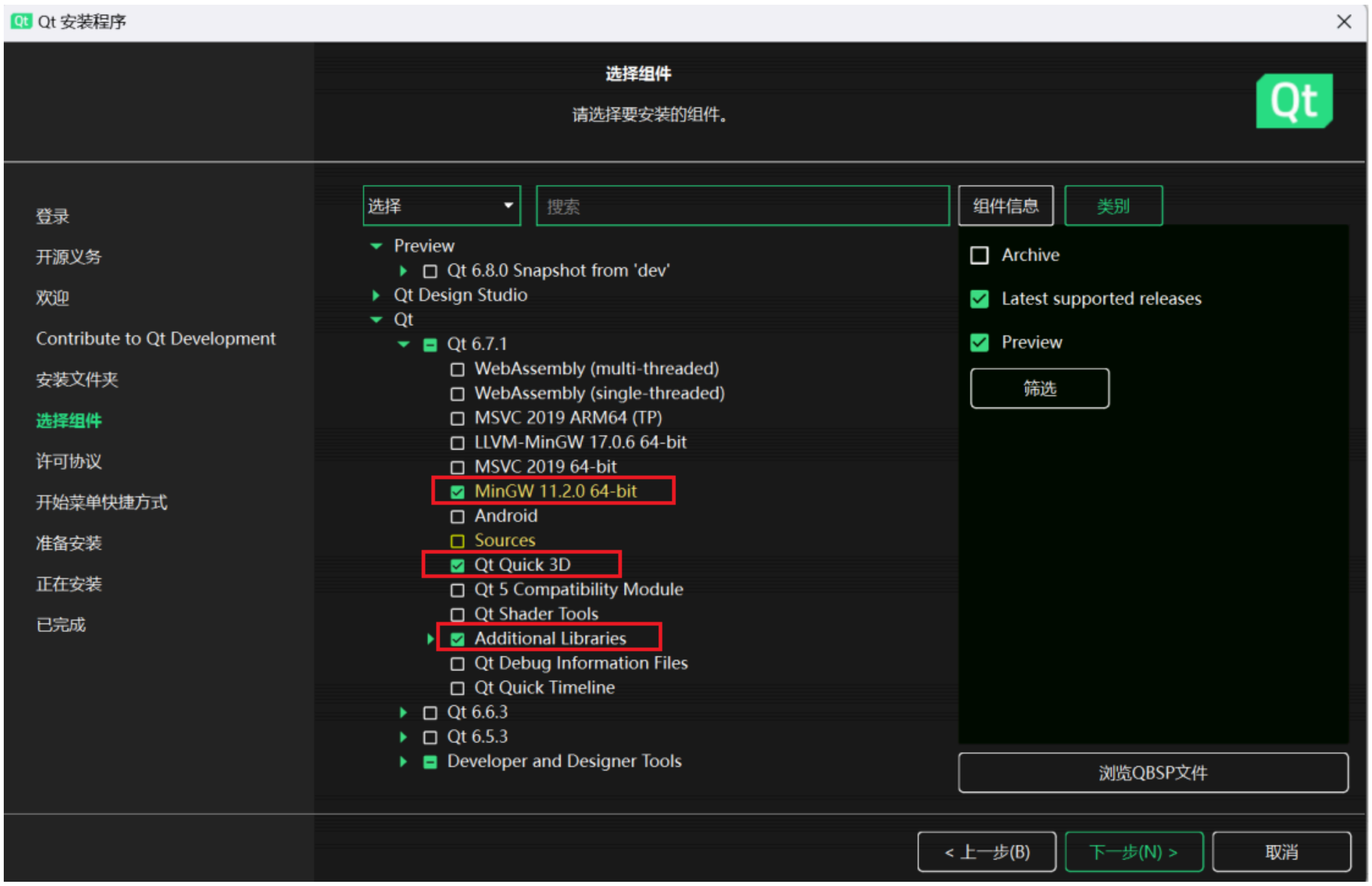
The Qt Account will give you access to Qt downloads, exclusive services, bug reports, code review, and forums & wiki.

Looking for Squish, the automated GUI testing framework? Click here to download or try it.

下一步 取消

之后就是常规的Qt下载安装了，除了需要下载的组件需要注意外，（组件的下载在后面需要的时候，如果没有下载的，可以再下载），下载安装没什么太多需要注意的。

这里我选择的是


Qt 安装程序

选择组件

请选择要安装的组件。

- 登录
- 开源义务
- 欢迎
- Contribute to Qt Development
- 安装文件夹
- 选择组件
- 许可协议
- 开始菜单快捷方式
- 准备安装
- 正在安装
- 已完成

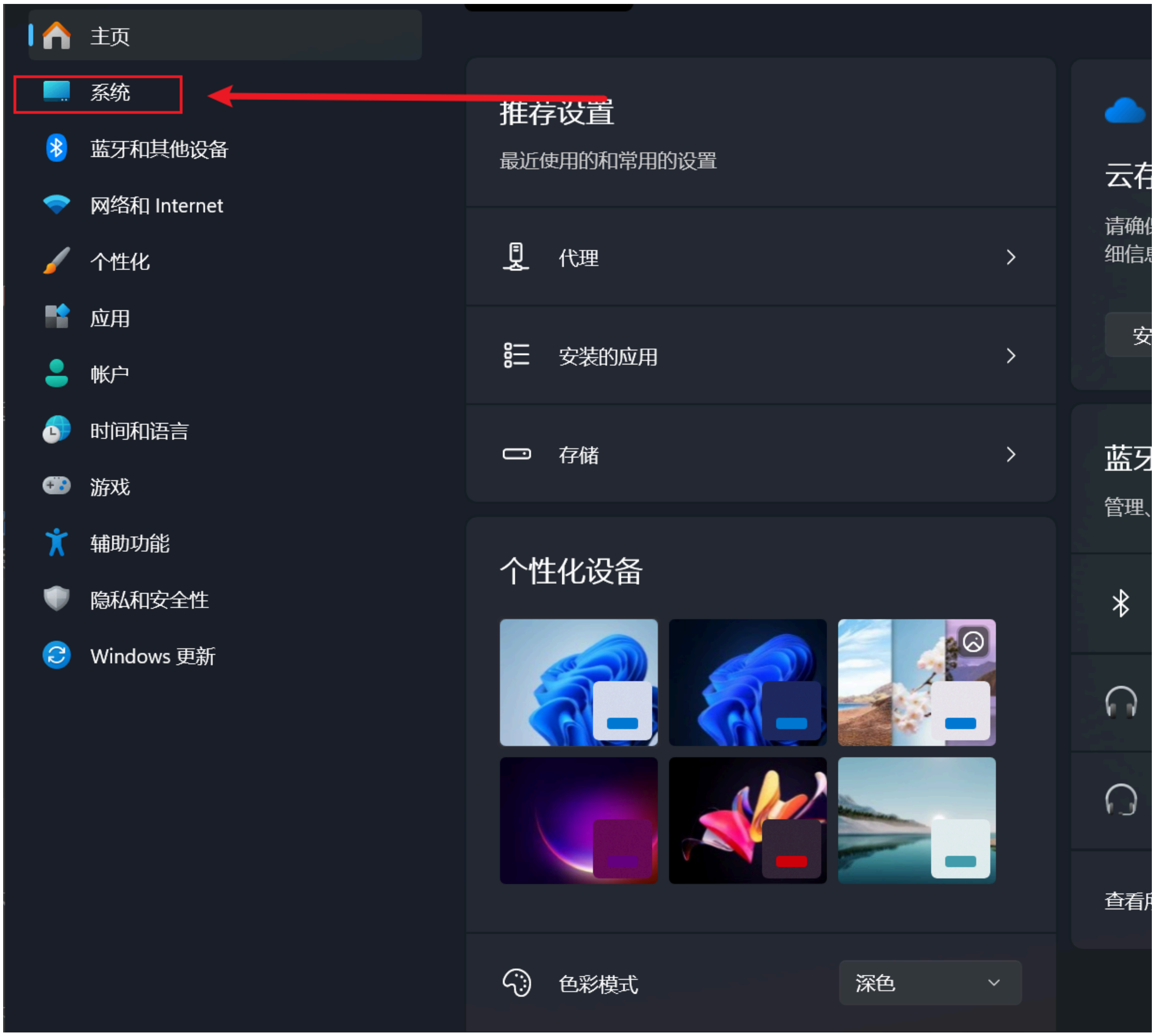
```
▼ Preview
    ▶ ☐ Qt 6.8.0 Snapshot from 'dev'
▶ Qt Design Studio
▼ Qt
    ▼ ▬ Qt 6.7.1
          ☐ WebAssembly (multi-threaded)
          ☐ WebAssembly (single-threaded)
          ☐ MSVC 2019 ARM64 (TP)
          ☐ LLVM-MinGW 17.0.6 64-bit
          ☐ MSVC 2019 64-bit
          ☑ MinGW 11.2.0 64-bit
          ☐ Android
          ☐ Sources
          ☑ Qt Quick 3D
          ☐ Qt 5 Compatibility Module
          ☐ Qt Shader Tools
        ▶ ☑ Additional Libraries
          ☐ Qt Debug Information Files
          ☐ Qt Quick Timeline
    ▶ ☐ Qt 6.6.3
    ▶ ☐ Qt 6.5.3
    ▶ ▬ Developer and Designer Tools
```

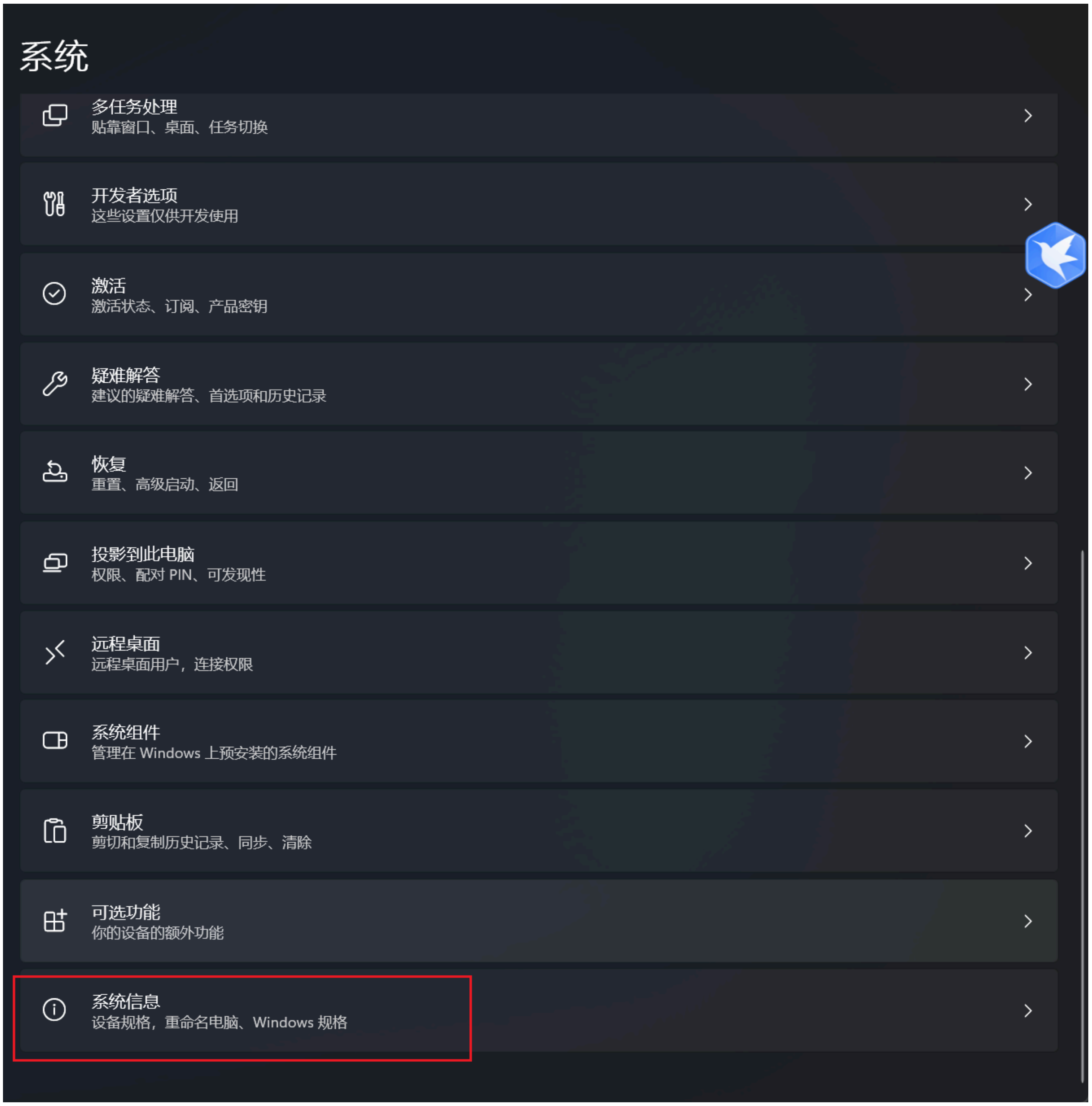
☐ Archive
☑ Latest supported releases
☑ Preview

最后不断点击下一步就可以了，

四、配置环境变量

打开系统设置


主页
系统
蓝牙和其他设备
网络和 Internet
个性化
应用
帐户
时间和语言
游戏
辅助功能
隐私和安全性
Windows 更新
推荐设置
最近使用的和常用的设置
代理
安装的应用
存储
个性化设备
色彩模式
深色
云存
请确
细信
安
蓝牙
管理、
查看



系统

多任务处理
贴靠窗口、桌面、任务切换

开发者选项
这些设置仅供开发使用

激活
激活状态、订阅、产品密钥

疑难解答
建议的疑难解答、首选项和历史记录

恢复
重置、高级启动、返回

投影到此电脑
权限、配对 PIN、可发现性

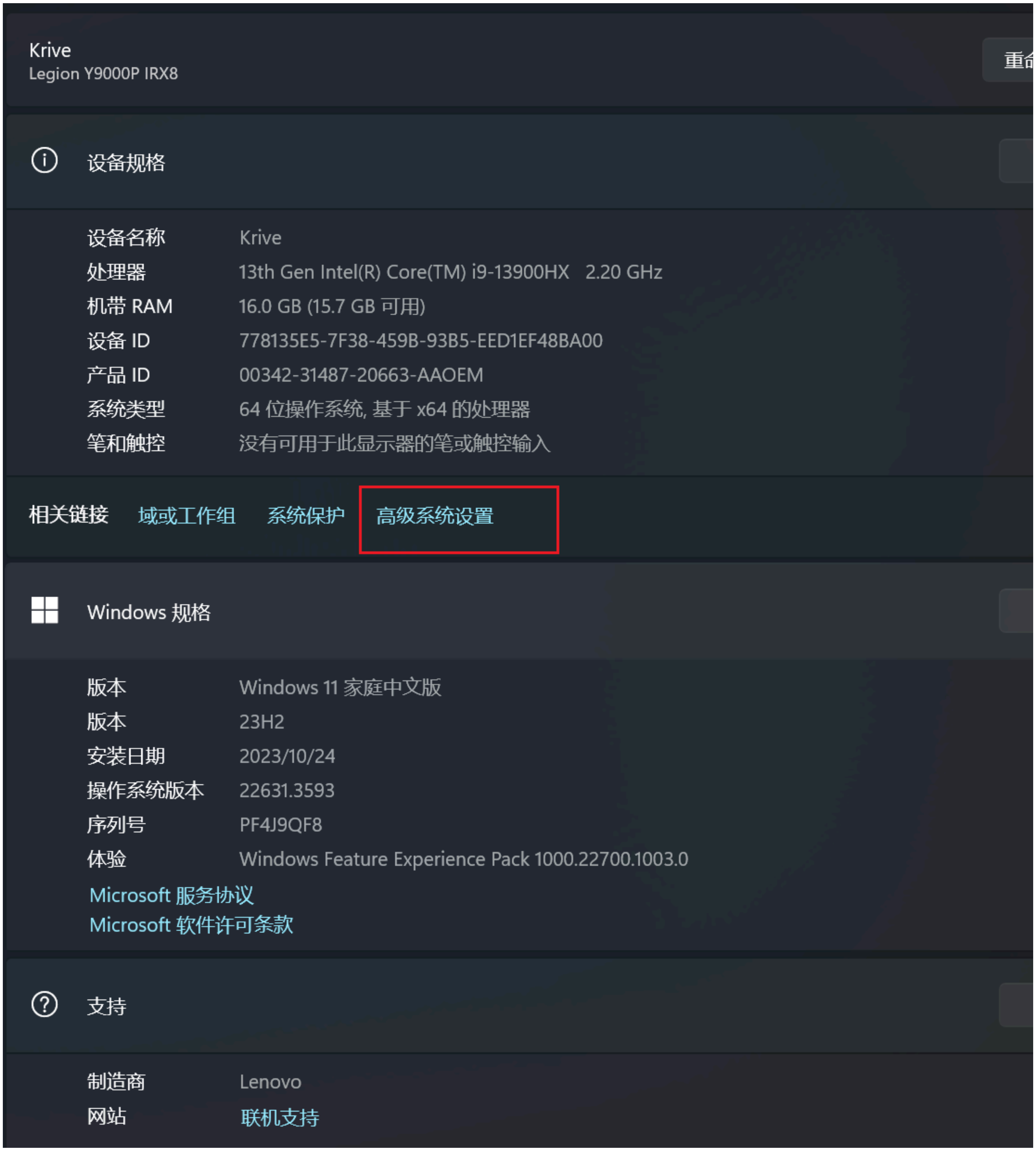
远程桌面
远程桌面用户，连接权限

系统组件
管理在 Windows 上预安装的系统组件

剪贴板
剪切和复制历史记录、同步、清除

可选功能
你的设备的额外功能

系统信息
设备规格，重命名电脑、Windows 规格



Krive
Legion Y9000P IRX8

重命

设备规格

设备名称	Krive
处理器	13th Gen Intel(R) Core(TM) i9-13900HX 2.20 GHz
机带 RAM	16.0 GB (15.7 GB 可用)
设备 ID	778135E5-7F38-459B-93B5-EED1EF48BA00
产品 ID	00342-31487-20663-AAOEM
系统类型	64 位操作系统, 基于 x64 的处理器
笔和触控	没有可用于此显示器的笔或触控输入

相关链接　域或工作组　系统保护　高级系统设置

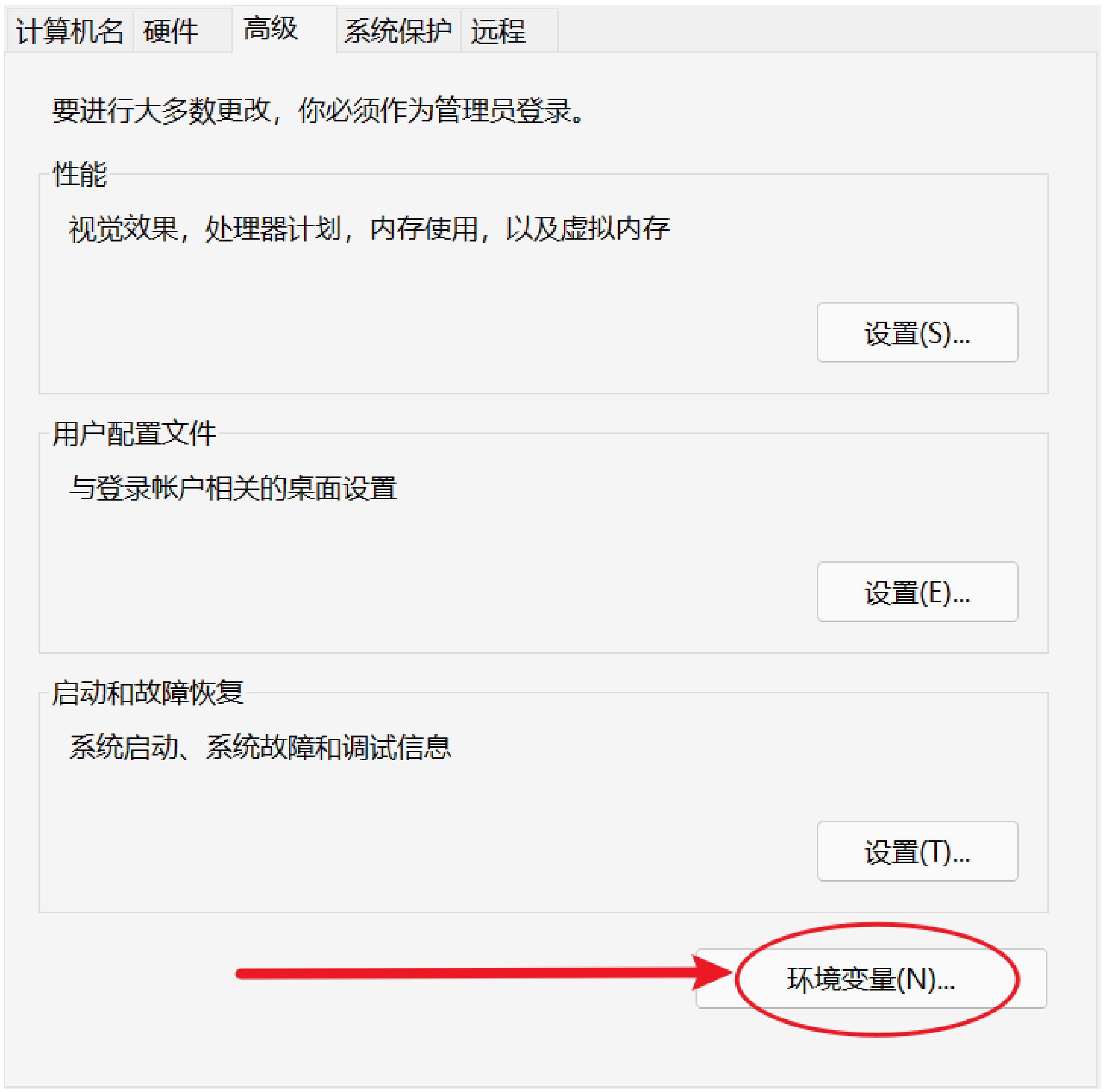
Windows 规格

版本	Windows 11 家庭中文版
版本	23H2
安装日期	2023/10/24
操作系统版本	22631.3593
序列号	PF4J9QF8
体验	Windows Feature Experience Pack 1000.22700.1003.0

Microsoft 服务协议
Microsoft 软件许可条款

支持

制造商	Lenovo
网站	联机支持



要进行大多数更改，你必须作为管理员登录。

性能

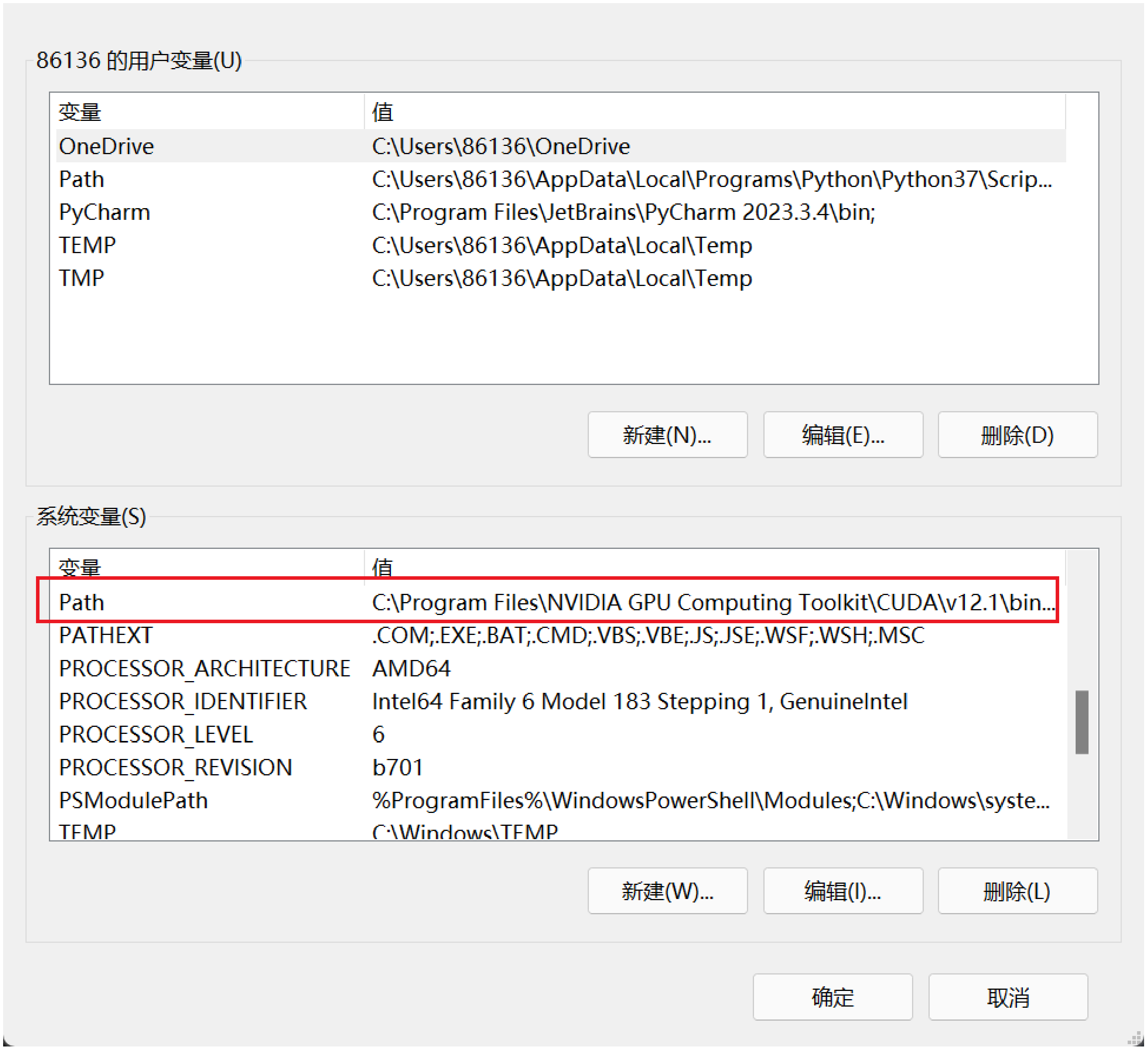
视觉效果，处理器计划，内存使用，以及虚拟内存

用户配置文件

与登录帐户相关的桌面设置

启动和故障恢复

系统启动、系统故障和调试信息



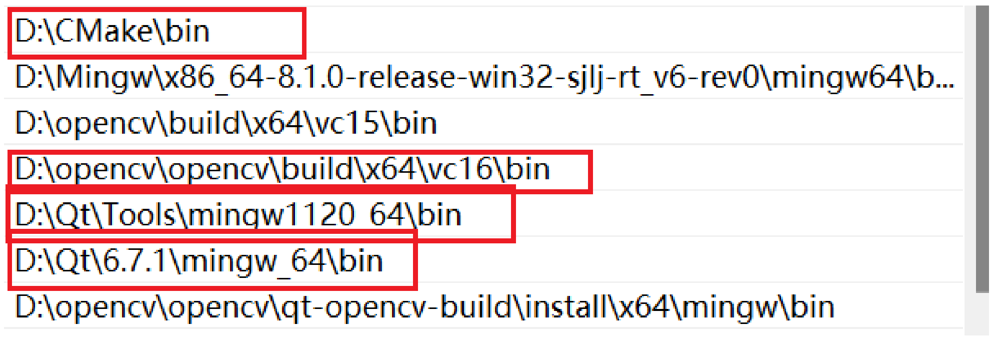
86136 的用户变量(U)

变量	值
OneDrive	C:\Users\86136\OneDrive
Path	C:\Users\86136\AppData\Local\Programs\Python\Python37\Scrip...
PyCharm	C:\Program Files\JetBrains\PyCharm 2023.3.4\bin;
TEMP	C:\Users\86136\AppData\Local\Temp
TMP	C:\Users\86136\AppData\Local\Temp

系统变量(S)

变量	值
Path	C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.1\bin...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 183 Stepping 1, GenuineIntel
PROCESSOR_LEVEL	6
PROCESSOR_REVISION	b701
PSModulePath	%ProgramFiles%\WindowsPowerShell\Modules;C:\Windows\syste...
TEMP	C:\Windows\TEMP

添加以下地址到环境变量中，这些地址需要你们自己去寻找自己文件夹地址进行替换

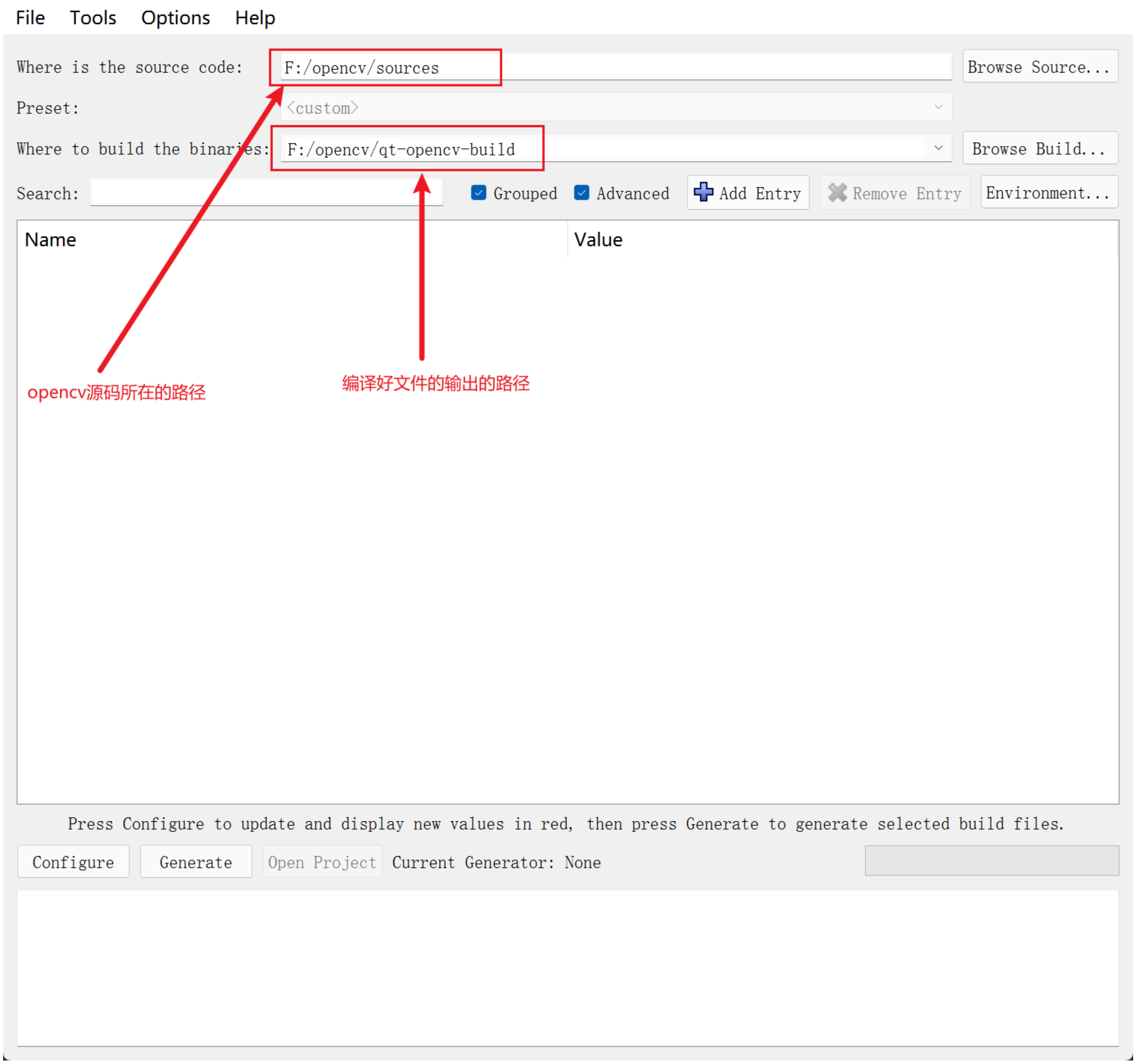


```
D:\CMake\bin
D:\Mingw\x86_64-8.1.0-release-win32-sjlj-rt_v6-rev0\mingw64\b...
D:\opencv\build\x64\vc15\bin
D:\opencv\opencv\build\x64\vc16\bin
D:\Qt\Tools\mingw1120_64\bin
D:\Qt\6.7.1\mingw_64\bin
D:\opencv\opencv\qt-opencv-build\install\x64\mingw\bin
```

五、CMake编译opencv库并生成

这一步是最关键的

Step1. 初步配置



Where is the source code: F:/opencv/sources

Preset: <custom>

Where to build the binaries: F:/opencv/qt-opencv-build

Search:

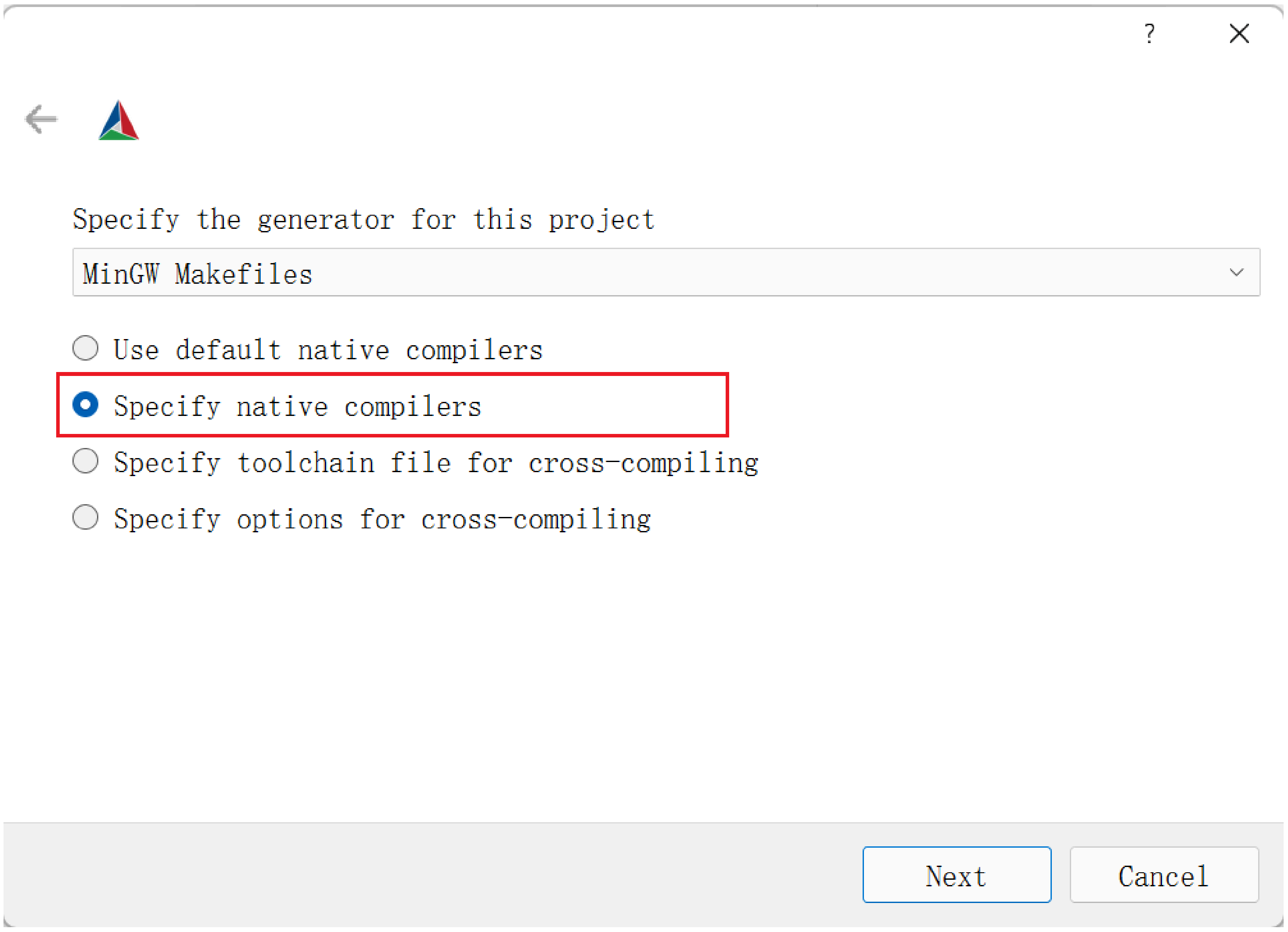
opencv源码所在的路径

编译好文件的输出的路径

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: None

设置好后点击 `Configure`


Specify the generator for this project

MinGW Makefiles

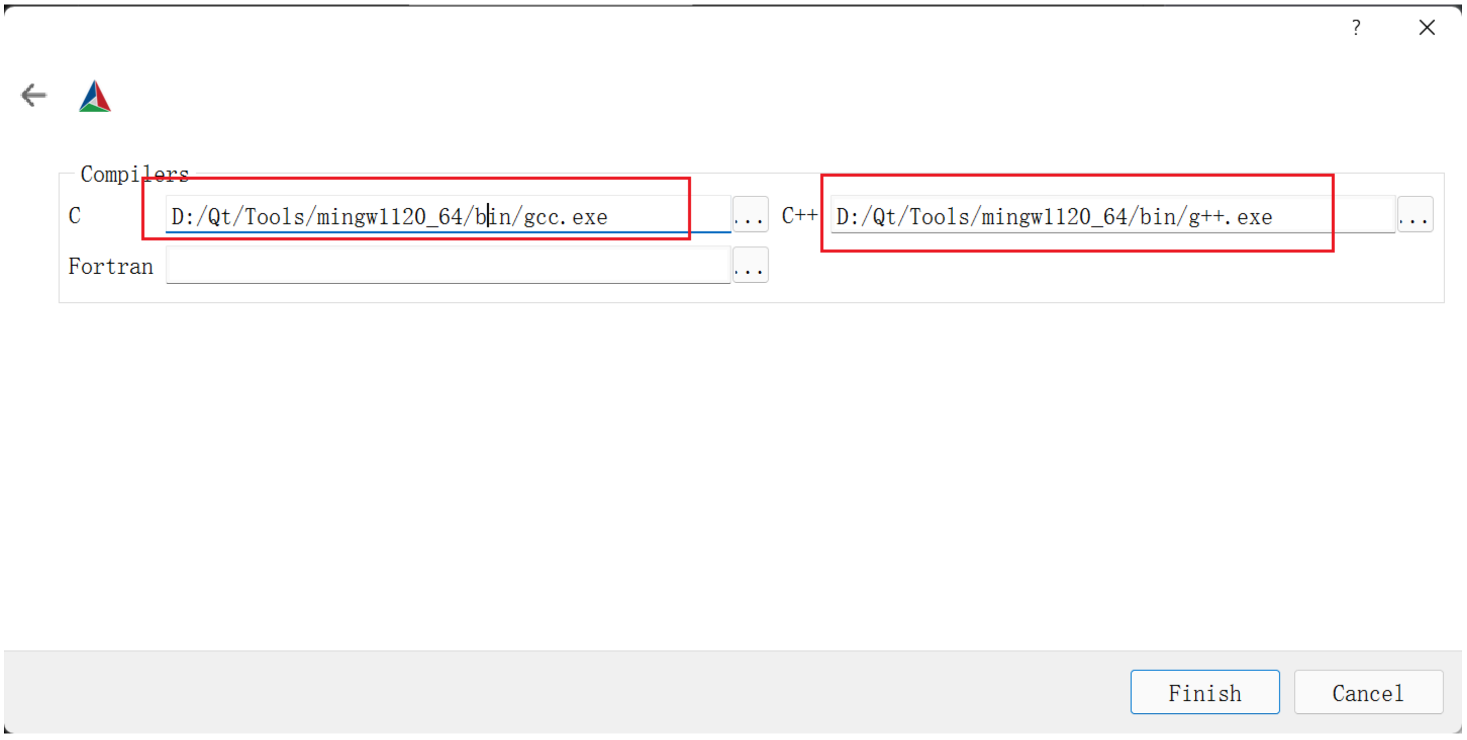
Use default native compilers

[✓ Specify native compilers]

Specify toolchain file for cross-compiling

Specify options for cross-compiling

选择qt自带的编译器MinGW的 gcc 和 g++


Compilers

C: D:/Qt/Tools/mingw1120_64/bin/gcc.exe

C++: D:/Qt/Tools/mingw1120_64/bin/g++.exe

Fortran:

路径范例为

```
D:\Qt\Tools\mingw1120_64\bin\gcc.exe
D:\Qt\Tools\mingw1120_64\bin\g++.exe
```

点击 Finish
经过漫长的等待后，应该是能够编译成功的。

Step2. 排查错误

以下是我的保错（仅供参考）

```
CMake Warning (dev) at cmake/OpenCVUtils.cmake:144 (find_package):
  Policy CMP0148 is not set: The FindPythonInterp and FindPythonLibs modules
  are removed.  Run "cmake --help-policy CMP0148" for policy details.  Use
  the cmake_policy command to set the policy and suppress this warning.

Call Stack (most recent call first):
  cmake/OpenCVDetectPython.cmake:64 (find_host_package)
  cmake/OpenCVDetectPython.cmake:286 (find_python)
  CMakeLists.txt:660 (include)
This warning is for project developers.  Use -Wno-dev to suppress it.

Found PythonInterp: D:/Anaconda/python.exe (found suitable version "3.11.5", minimum required
CMake Warning (dev) at cmake/OpenCVDetectPython.cmake:140 (find_package):
  Policy CMP0148 is not set: The FindPythonInterp and FindPythonLibs modules
  are removed.  Run "cmake --help-policy CMP0148" for policy details.  Use
  the cmake_policy command to set the policy and suppress this warning.

Call Stack (most recent call first):
  cmake/OpenCVDetectPython.cmake:286 (find_python)
  CMakeLists.txt:660 (include)
This warning is for project developers.  Use -Wno-dev to suppress it.
```

```
CMake Warning at cmake/OpenCVDownload.cmake:248 (message):
  ADE: Download failed: 7;"Couldn't connect to server"

  For details please refer to the download log file:

  F:/opencv/qt-opencv-build/CMakeDownloadLog.txt

Call Stack (most recent call first):
  modules/gapi/cmake/DownloadADE.cmake:5 (ocv_download)
  modules/gapi/cmake/init.cmake:20 (include)
  cmake/OpenCVModule.cmake:298 (include)
  cmake/OpenCVModule.cmake:361 (_add_modules_1)
  cmake/OpenCVModule.cmake:408 (ocv_glob_modules)
  CMakeLists.txt:1032 (ocv_register_modules)
```

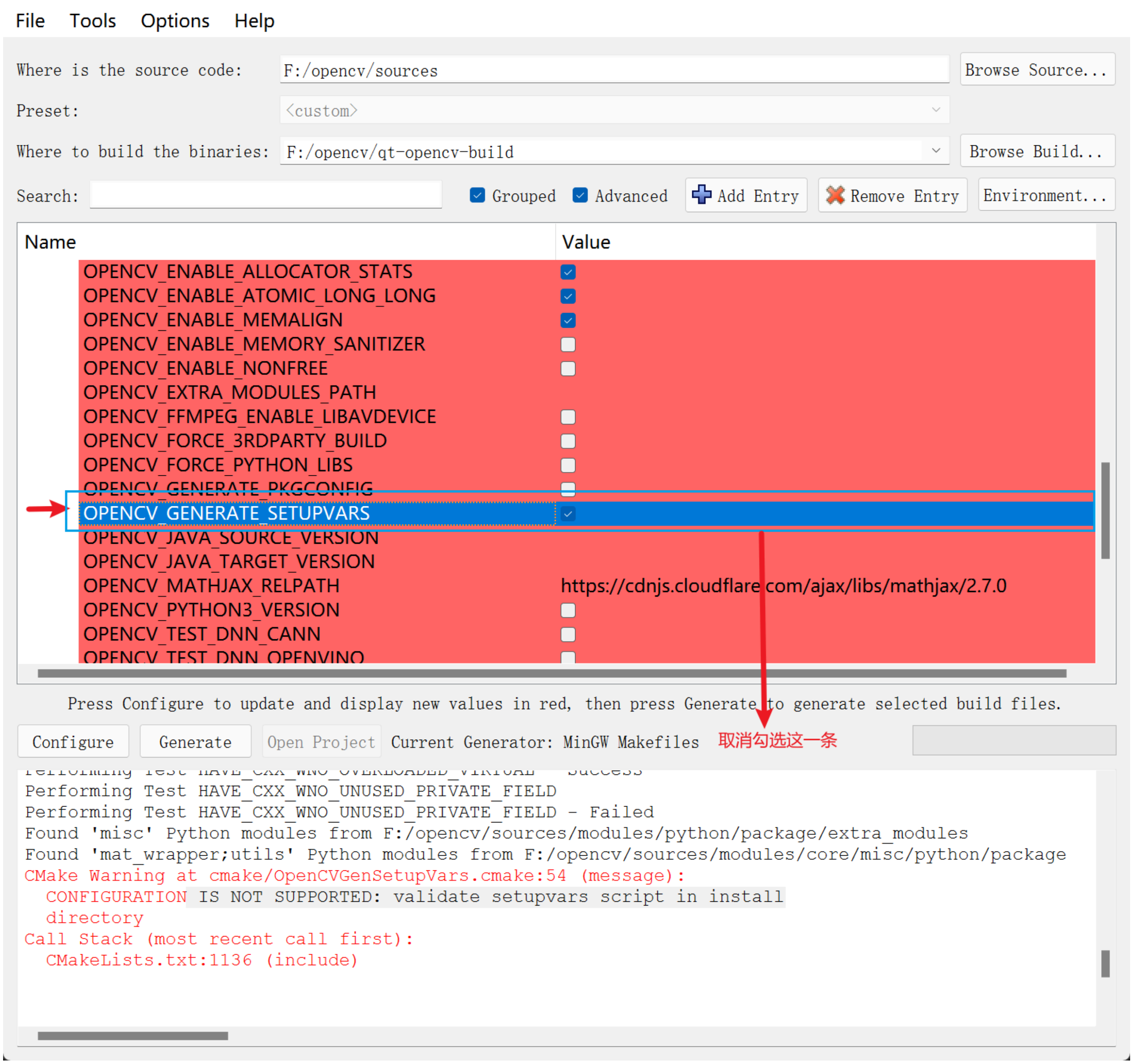
```
CMake Warning at cmake/OpenCVDownload.cmake:248 (message):
  FFMPEG: Download failed: 7;"Couldn't connect to server"

  For details please refer to the download log file:

  F:/opencv/qt-opencv-build/CMakeDownloadLog.txt

Call Stack (most recent call first):
  3rdparty/ffmpeg/ffmpeg.cmake:20 (ocv_download)
  modules/videoio/cmake/detect_ffmpeg.cmake:17 (download_win_ffmpeg)
  modules/videoio/cmake/init.cmake:7 (include)
  modules/videoio/cmake/init.cmake:11 (add_backend)
  cmake/OpenCVModule.cmake:298 (include)
  cmake/OpenCVModule.cmake:361 (_add_modules_1)
  cmake/OpenCVModule.cmake:408 (ocv_glob_modules)
  CMakeLists.txt:1032 (ocv_register_modules)
```

```
CMake Warning at cmake/OpenCVGenSetupVars.cmake:54 (message):
  CONFIGURATION IS NOT SUPPORTED: validate setupvars script in install
  directory
Call Stack (most recent call first):
  CMakeLists.txt:1136 (include)
```

以下报错分别为248，144，54

首先解决第54，144的信息的报错

取消勾选下面这一条，这个是路径错误的警告，取消勾选不会影响软件的运行



Where is the source code: F:/opencv/sources

Preset: <custom>

Where to build the binaries: F:/opencv/qt-opencv-build

Search: ✓ Grouped ✓ Advanced

Name	Value
OPENCV_ENABLE_ALLOCATOR_STATS	✓
OPENCV_ENABLE_ATOMIC_LONG_LONG	✓
OPENCV_ENABLE_MEMALIGN	✓
OPENCV_ENABLE_MEMORY_SANITIZER	☐
OPENCV_ENABLE_NONFREE	☐
OPENCV_EXTRA_MODULES_PATH	
OPENCV_FFMPEG_ENABLE_LIBAVDEVICE	☐
OPENCV_FORCE_3RDPARTY_BUILD	☐
OPENCV_FORCE_PYTHON_LIBS	☐
OPENCV_GENERATE_PKGCONFIG	☐
OPENCV_GENERATE_SETUPVARS	✓
OPENCV_JAVA_SOURCE_VERSION	
OPENCV_JAVA_TARGET_VERSION	
OPENCV_MATHJAX_RELPATH	https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.0
OPENCV_PYTHON3_VERSION	☐
OPENCV_TEST_DNN_CANN	☐
OPENCV_TEST_DNN_OPENVINO	☐

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: MinGW Makefiles

```
Performing Test HAVE_CXX_WNO_UNUSED_PRIVATE_FIELD
Performing Test HAVE_CXX_WNO_UNUSED_PRIVATE_FIELD - Failed
Found 'misc' Python modules from F:/opencv/sources/modules/python/package/extra_modules
Found 'mat_wrapper;utils' Python modules from F:/opencv/sources/modules/core/misc/python/package
CMake Warning at cmake/OpenCVGenSetupVars.cmake:54 (message):
  CONFIGURATION IS NOT SUPPORTED: validate setupvars script in install
  directory
Call Stack (most recent call first):
  CMakeLists.txt:1136 (include)
```

接下来解决248的报错

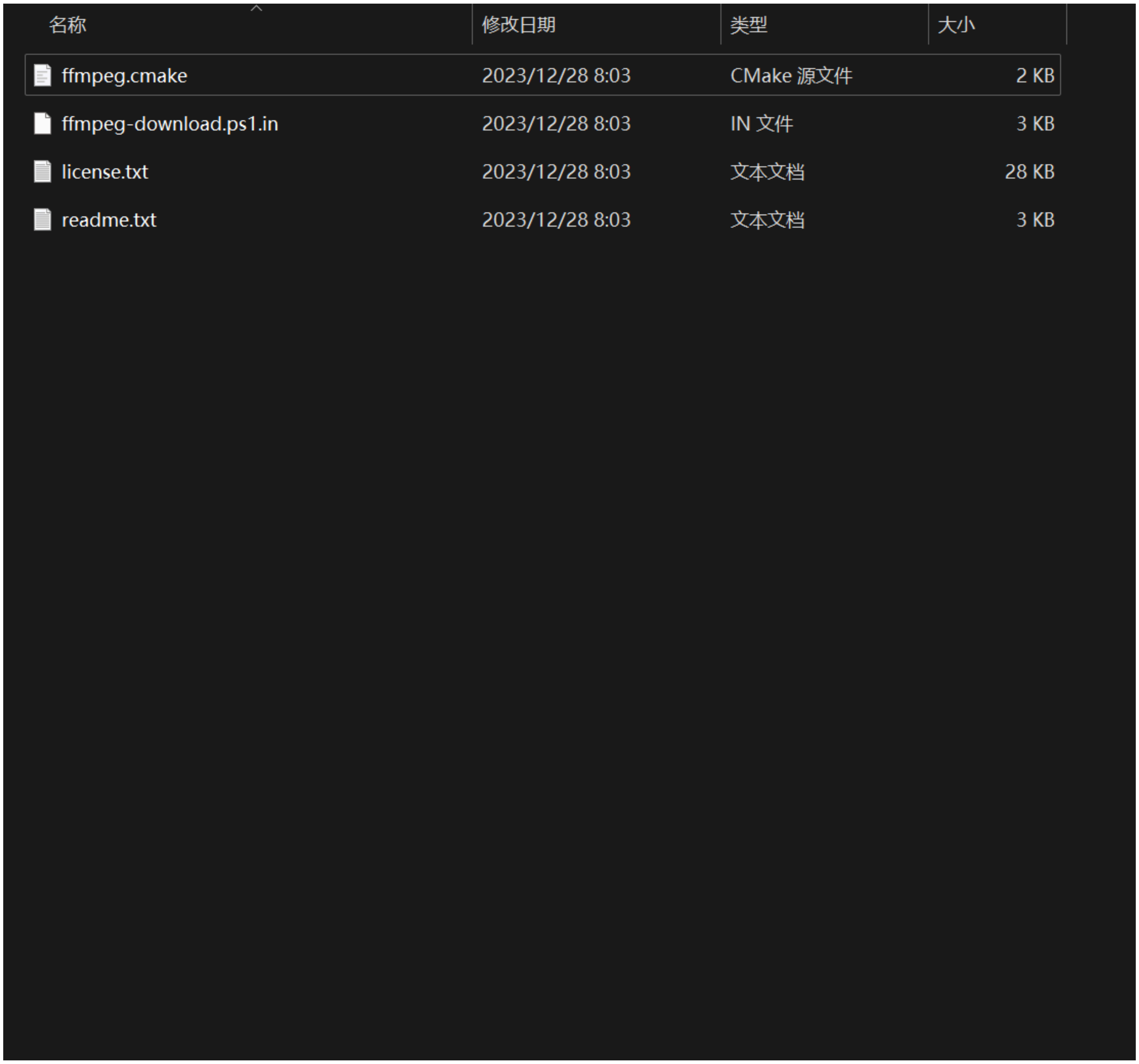
`CMake` 编译 `opencv` 时无法连接服务器，导致下载 `ffmpeg.dll` 、 `ippicv` 等 发生失败报错

248的报错是来源于，GitHub是国外的网站，前期试了多种方法，比如梯子和更改电脑上网ip都没有解决，最后找到现在这种，修改下载的网站利用代理下载才成功解决

代理网站

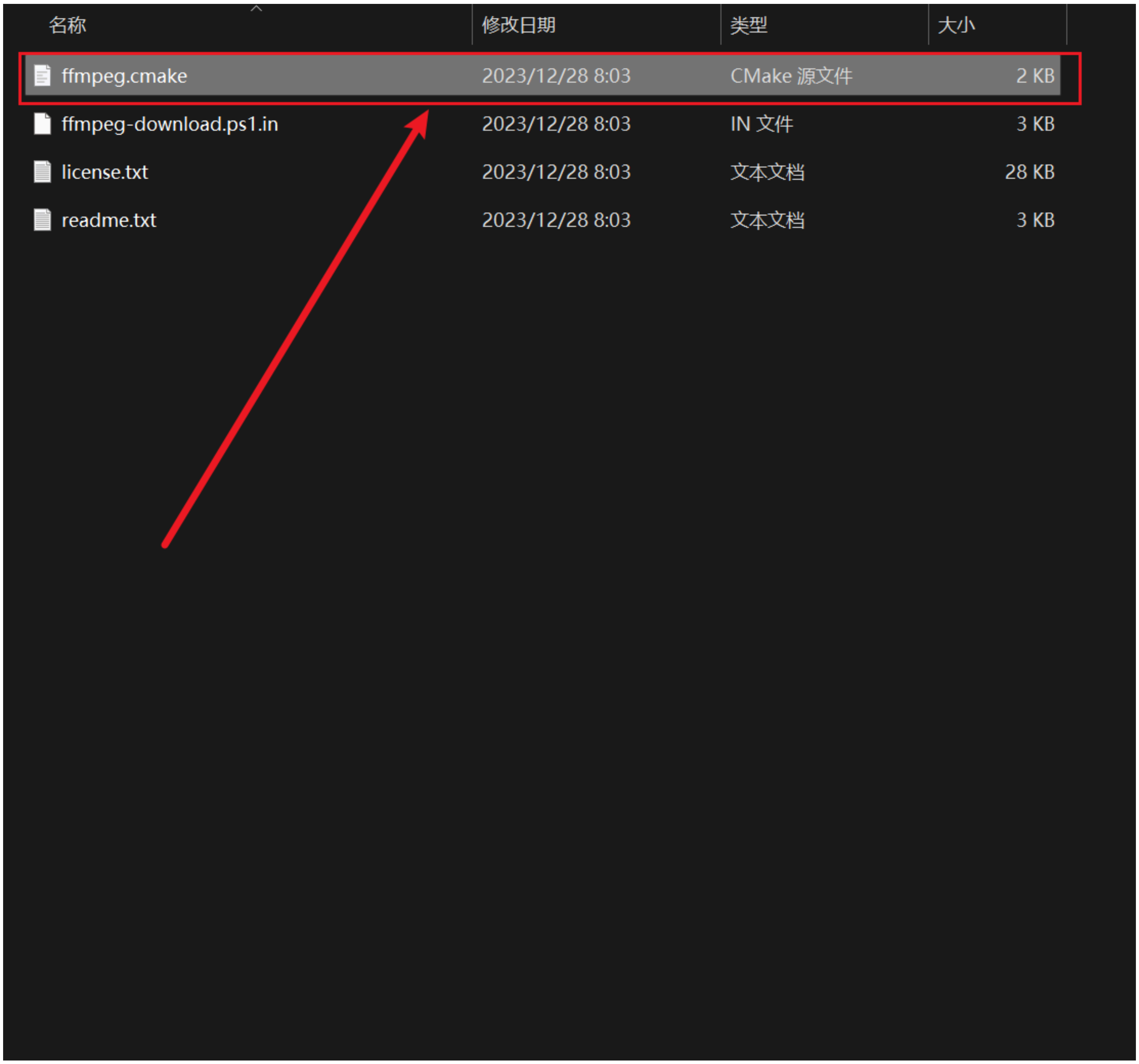
解决方案

1. 找到目标文件夹 `F:\opencv\sources\3rdparty\ffmpeg`



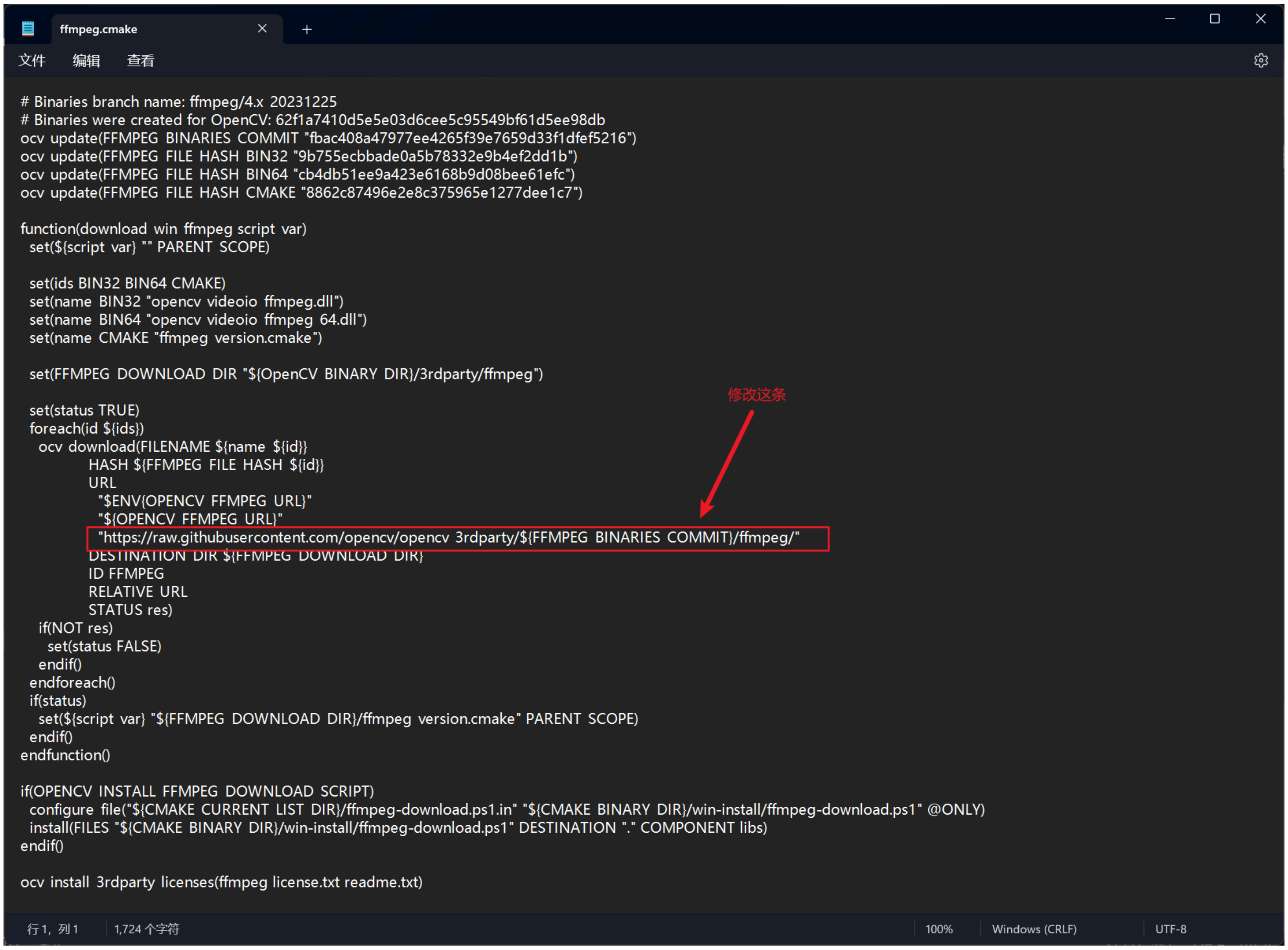
名称	修改日期	类型	大小
ffmpeg.cmake	2023/12/28 8:03	CMake 源文件	2 KB
ffmpeg-download.ps1.in	2023/12/28 8:03	IN 文件	3 KB
license.txt	2023/12/28 8:03	文本文档	28 KB
readme.txt	2023/12/28 8:03	文本文档	3 KB

修改下载路径的代码



名称	修改日期	类型	大小
ffmpeg.cmake	2023/12/28 8:03	CMake 源文件	2 KB
ffmpeg-download.ps1.in	2023/12/28 8:03	IN 文件	3 KB
license.txt	2023/12/28 8:03	文本文档	28 KB
readme.txt	2023/12/28 8:03	文本文档	3 KB

用记事本打开



```
# Binaries branch name: ffmpeg/4.x 20231225
# Binaries were created for OpenCV: 62f1a7410d5e5e03d6cee5c95549bf61d5ee98db
ocv update(FFMPEG BINARIES COMMIT "fbac408a47977ee4265f39e7659d33f1dfef5216")
ocv update(FFMPEG FILE HASH BIN32 "9b755ecbbade0a5b78332e9b4ef2dd1b")
ocv update(FFMPEG FILE HASH BIN64 "cb4db51ee9a423e6168b9d08bee61efc")
ocv update(FFMPEG FILE HASH CMAKE "8862c87496e2e8c375965e1277dee1c7")

function(download win ffmpeg script var)
  set(${script var} "" PARENT SCOPE)

  set(ids BIN32 BIN64 CMAKE)
  set(name BIN32 "opencv videoio ffmpeg.dll")
  set(name BIN64 "opencv videoio ffmpeg 64.dll")
  set(name CMAKE "ffmpeg version.cmake")

  set(FFMPEG DOWNLOAD DIR "${OpenCV BINARY DIR}/3rdparty/ffmpeg")

  set(status TRUE)
  foreach(id ${ids})
    ocv download(FILENAME ${name ${id}}
               HASH ${FFMPEG FILE HASH ${id}}
               URL
                 "$ENV{OPENCV FFMPEG URL}"
                 "${OPENCV FFMPEG URL}"
                 "https://raw.githubusercontent.com/opencv/opencv 3rdparty/${FFMPEG BINARIES COMMIT}/ffmpeg/"
               DESTINATION DIR ${FFMPEG DOWNLOAD DIR}
               ID FFMPEG
               RELATIVE URL
               STATUS res)
    if(NOT res)
      set(status FALSE)
    endif()
  endforeach()
  if(status)
    set(${script var} "${FFMPEG DOWNLOAD DIR}/ffmpeg version.cmake" PARENT SCOPE)
  endif()
endfunction()

if(OPENCV INSTALL FFMPEG DOWNLOAD SCRIPT)
  configure file("${CMAKE CURRENT LIST DIR}/ffmpeg-download.ps1.in" "${CMAKE BINARY DIR}/win-install/ffmpeg-download.ps1" @ONLY)
  install(FILES "${CMAKE BINARY DIR}/win-install/ffmpeg-download.ps1" DESTINATION "." COMPONENT libs)
endif()

ocv install 3rdparty licenses(ffmpeg license.txt readme.txt)
```

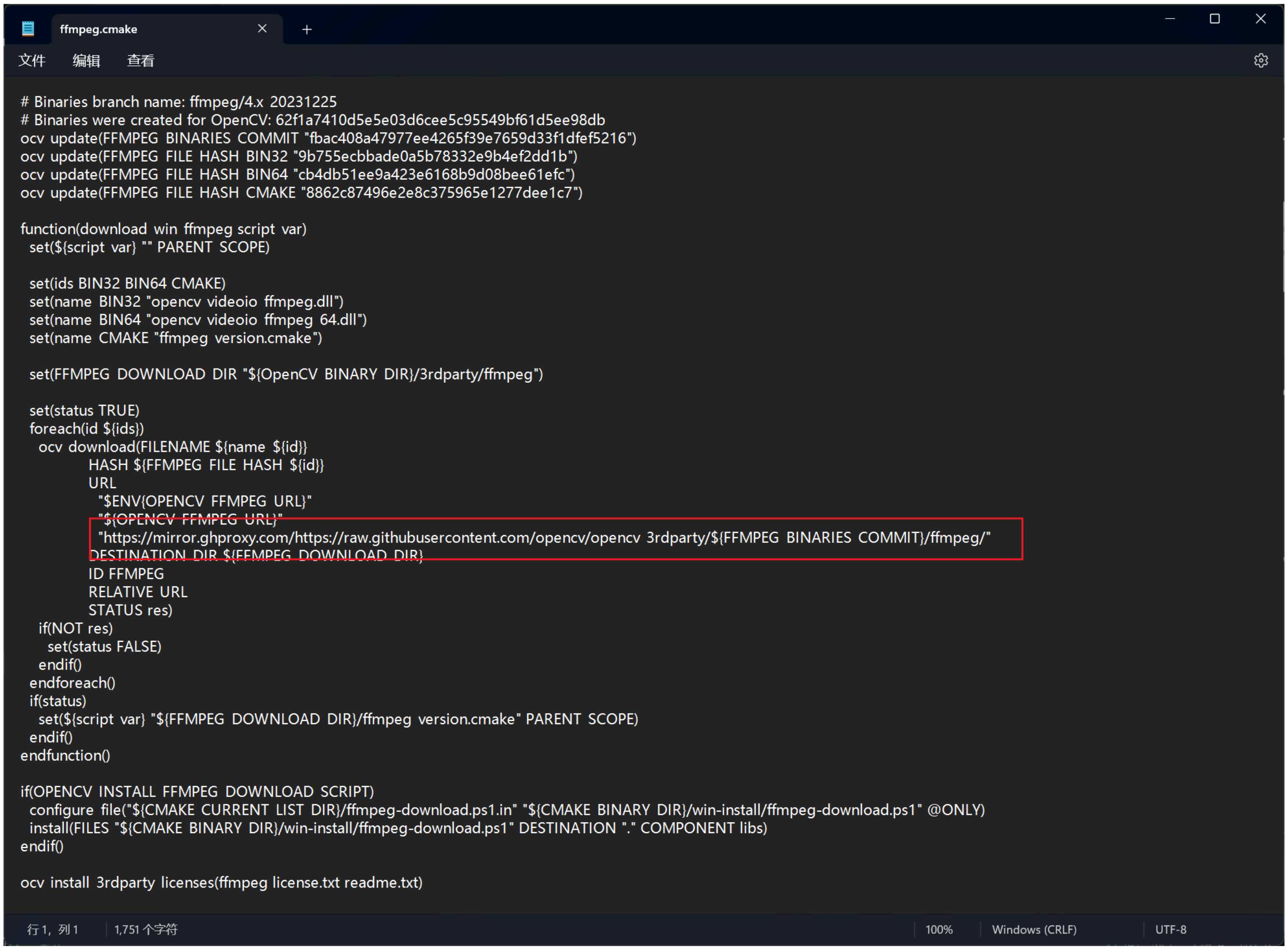
把这一条网站

```
"https://raw.githubusercontent.com/opencv/opencv_3rdparty/${FFMPEG_BINARIES_COMMIT}/ffmpeg/"
```

修改

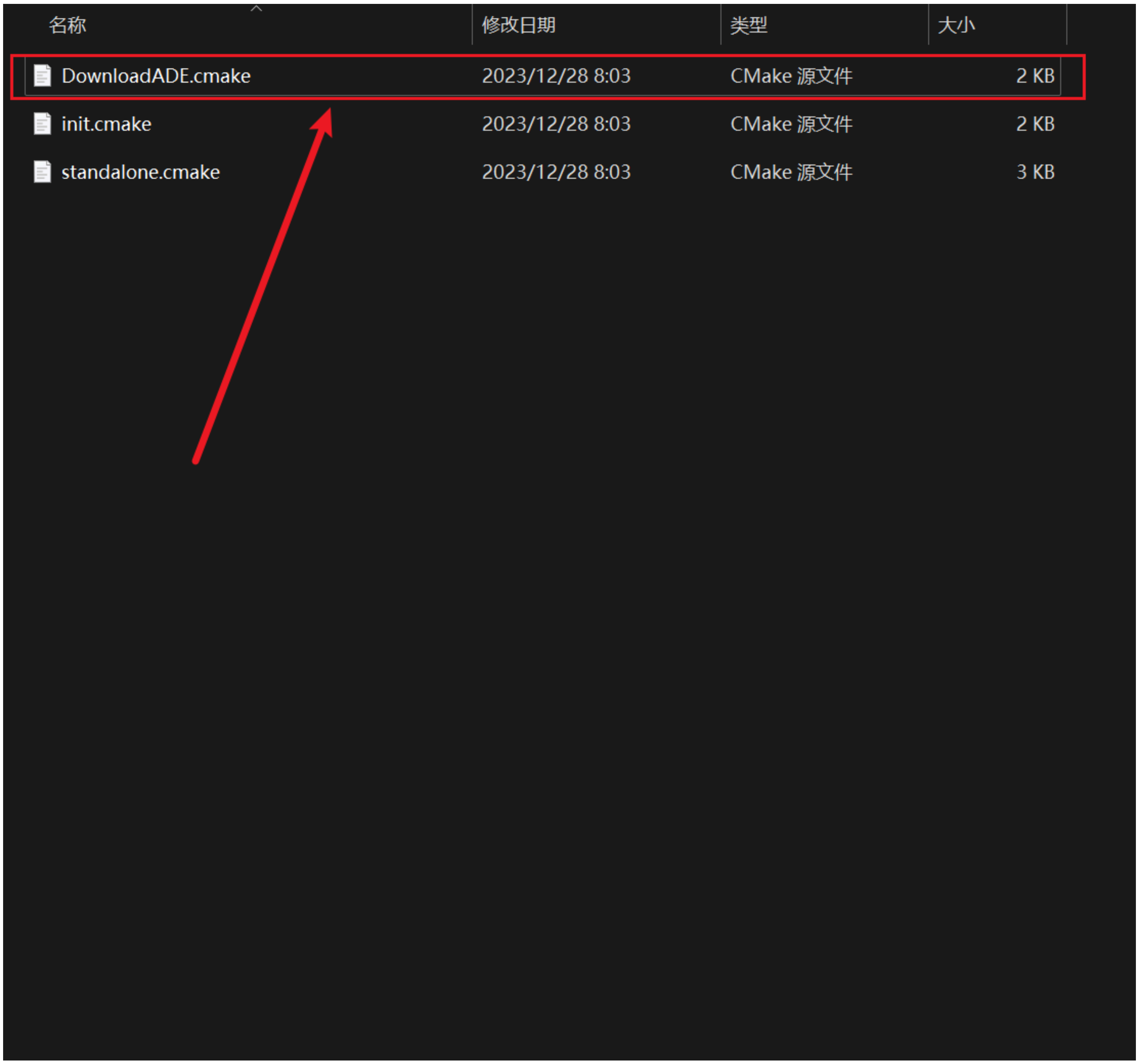
为

```
"https://mirror.ghproxy.com/https://raw.githubusercontent.com/opencv/opencv_3rdparty/${FFMPEG_BINARIES_COMMIT}/ffmpeg/"
```

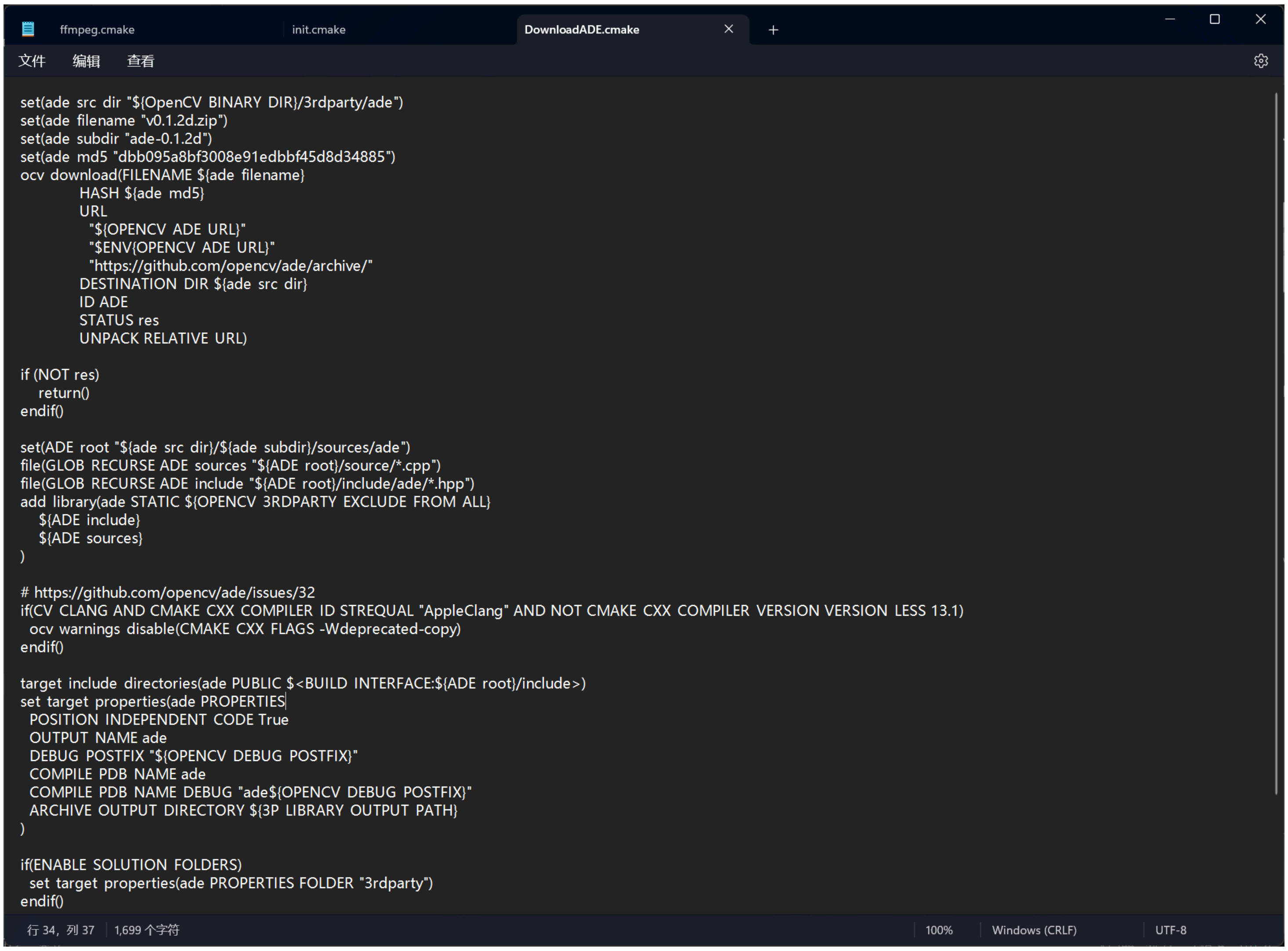


```
# Binaries branch name: ffmpeg/4.x 20231225
# Binaries were created for OpenCV: 62f1a7410d5e5e03d6cee5c95549bf61d5ee98db
ocv update(FFMPEG BINARIES COMMIT "fbac408a47977ee4265f39e7659d33f1dfef5216")
ocv update(FFMPEG FILE HASH BIN32 "9b755ecbbade0a5b78332e9b4ef2dd1b")
ocv update(FFMPEG FILE HASH BIN64 "cb4db51ee9a423e6168b9d08bee61efc")
ocv update(FFMPEG FILE HASH CMAKE "8862c87496e2e8c375965e1277dee1c7")

function(download win ffmpeg script var)
  set(${script var} "" PARENT SCOPE)

  set(ids BIN32 BIN64 CMAKE)
  set(name BIN32 "opencv videoio ffmpeg.dll")
  set(name BIN64 "opencv videoio ffmpeg 64.dll")
  set(name CMAKE "ffmpeg version.cmake")

  set(FFMPEG DOWNLOAD DIR "${OpenCV BINARY DIR}/3rdparty/ffmpeg")

  set(status TRUE)
  foreach(id ${ids})
    ocv download(FILENAME ${name ${id}}
               HASH ${FFMPEG FILE HASH ${id}}
               URL
                 "$ENV{OPENCV FFMPEG URL}"
                 "${OPENCV FFMPEG URL}"
                 "https://mirror.ghproxy.com/https://raw.githubusercontent.com/opencv/opencv 3rdparty/${FFMPEG BINARIES COMMIT}/ffmpeg/"
               DESTINATION DIR ${FFMPEG DOWNLOAD DIR}
               ID FFMPEG
               RELATIVE URL
               STATUS res)
    if(NOT res)
      set(status FALSE)
    endif()
  endforeach()
  if(status)
    set(${script var} "${FFMPEG DOWNLOAD DIR}/ffmpeg version.cmake" PARENT SCOPE)
  endif()
endfunction()

if(OPENCV INSTALL FFMPEG DOWNLOAD SCRIPT)
  configure file("${CMAKE CURRENT LIST DIR}/ffmpeg-download.ps1.in" "${CMAKE BINARY DIR}/win-install/ffmpeg-download.ps1" @ONLY)
  install(FILES "${CMAKE BINARY DIR}/win-install/ffmpeg-download.ps1" DESTINATION "." COMPONENT libs)
endif()

ocv install 3rdparty licenses(ffmpeg license.txt readme.txt)
```

2. 找到目标文件夹 `F:\opencv\sources\modules\gapi\cmake`

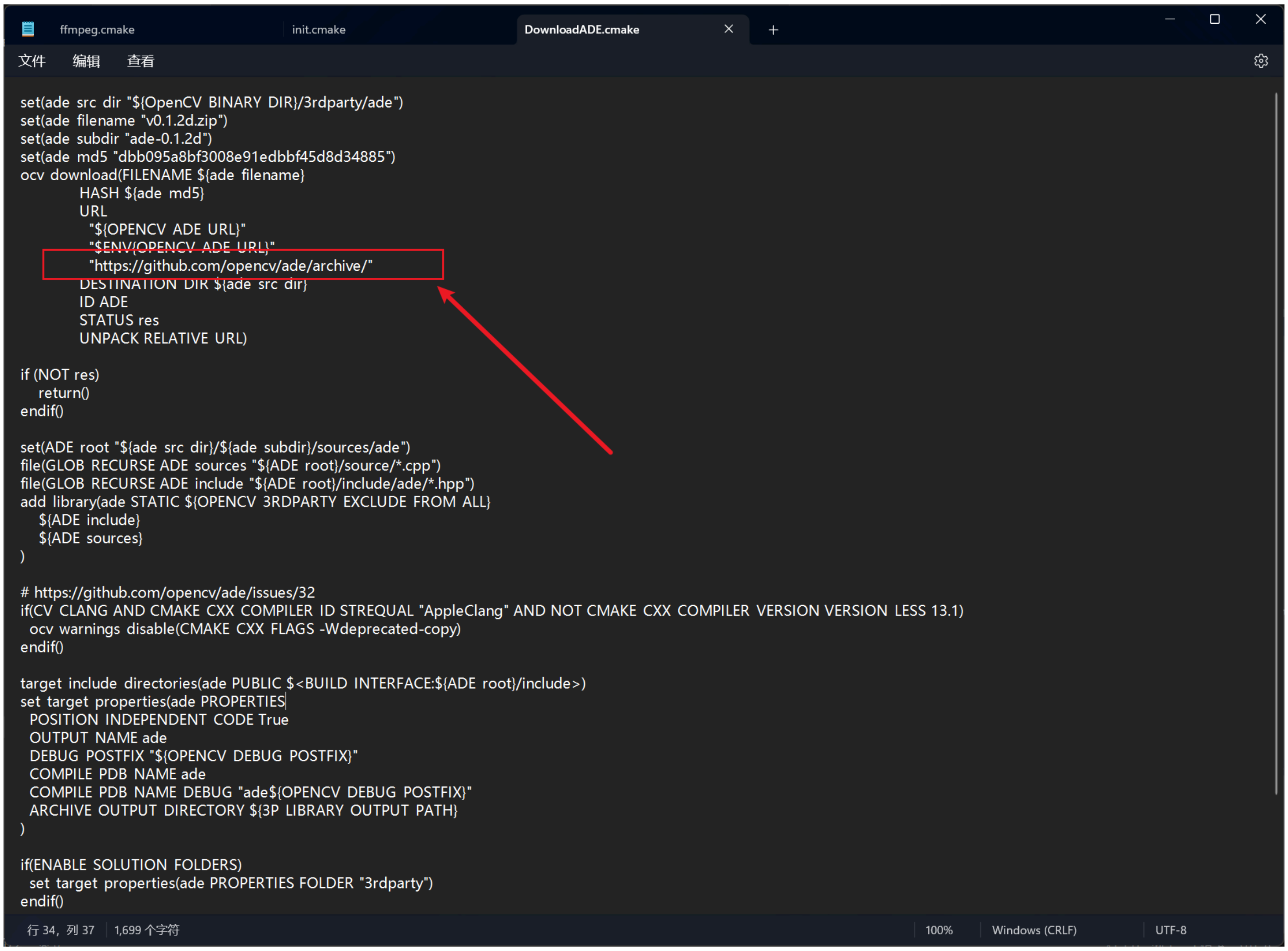


名称	修改日期	类型	大小
DownloadADE.cmake	2023/12/28 8:03	CMake 源文件	2 KB
init.cmake	2023/12/28 8:03	CMake 源文件	2 KB
standalone.cmake	2023/12/28 8:03	CMake 源文件	3 KB



```
set(ade src dir "${OpenCV BINARY DIR}/3rdparty/ade")
set(ade filename "v0.1.2d.zip")
set(ade subdir "ade-0.1.2d")
set(ade md5 "dbb095a8bf3008e91edbbf45d8d34885")
ocv download(FILENAME ${ade filename}
          HASH ${ade md5}
          URL
            "${OPENCV ADE URL}"
            "$ENV{OPENCV ADE URL}"
            "https://github.com/opencv/ade/archive/"
          DESTINATION DIR ${ade src dir}
          ID ADE
          STATUS res
          UNPACK RELATIVE URL)

if (NOT res)
   return()
endif()

set(ADE root "${ade src dir}/${ade subdir}/sources/ade")
file(GLOB RECURSE ADE sources "${ADE root}/source/*.cpp")
file(GLOB RECURSE ADE include "${ADE root}/include/ade/*.hpp")
add library(ade STATIC ${OPENCV 3RDPARTY EXCLUDE FROM ALL}
   ${ADE include}
   ${ADE sources}
)

# https://github.com/opencv/ade/issues/32
if(CV CLANG AND CMAKE CXX COMPILER ID STREQUAL "AppleClang" AND NOT CMAKE CXX COMPILER VERSION VERSION LESS 13.1)
  ocv warnings disable(CMAKE CXX FLAGS -Wdeprecated-copy)
endif()

target include directories(ade PUBLIC $<BUILD INTERFACE:${ADE root}/include>)
set target properties(ade PROPERTIES
  POSITION INDEPENDENT CODE True
  OUTPUT NAME ade
  DEBUG POSTFIX "${OPENCV DEBUG POSTFIX}"
  COMPILE PDB NAME ade
  COMPILE PDB NAME DEBUG "ade${OPENCV DEBUG POSTFIX}"
  ARCHIVE OUTPUT DIRECTORY ${3P LIBRARY OUTPUT PATH}
)

if(ENABLE SOLUTION FOLDERS)
  set target properties(ade PROPERTIES FOLDER "3rdparty")
endif()
```

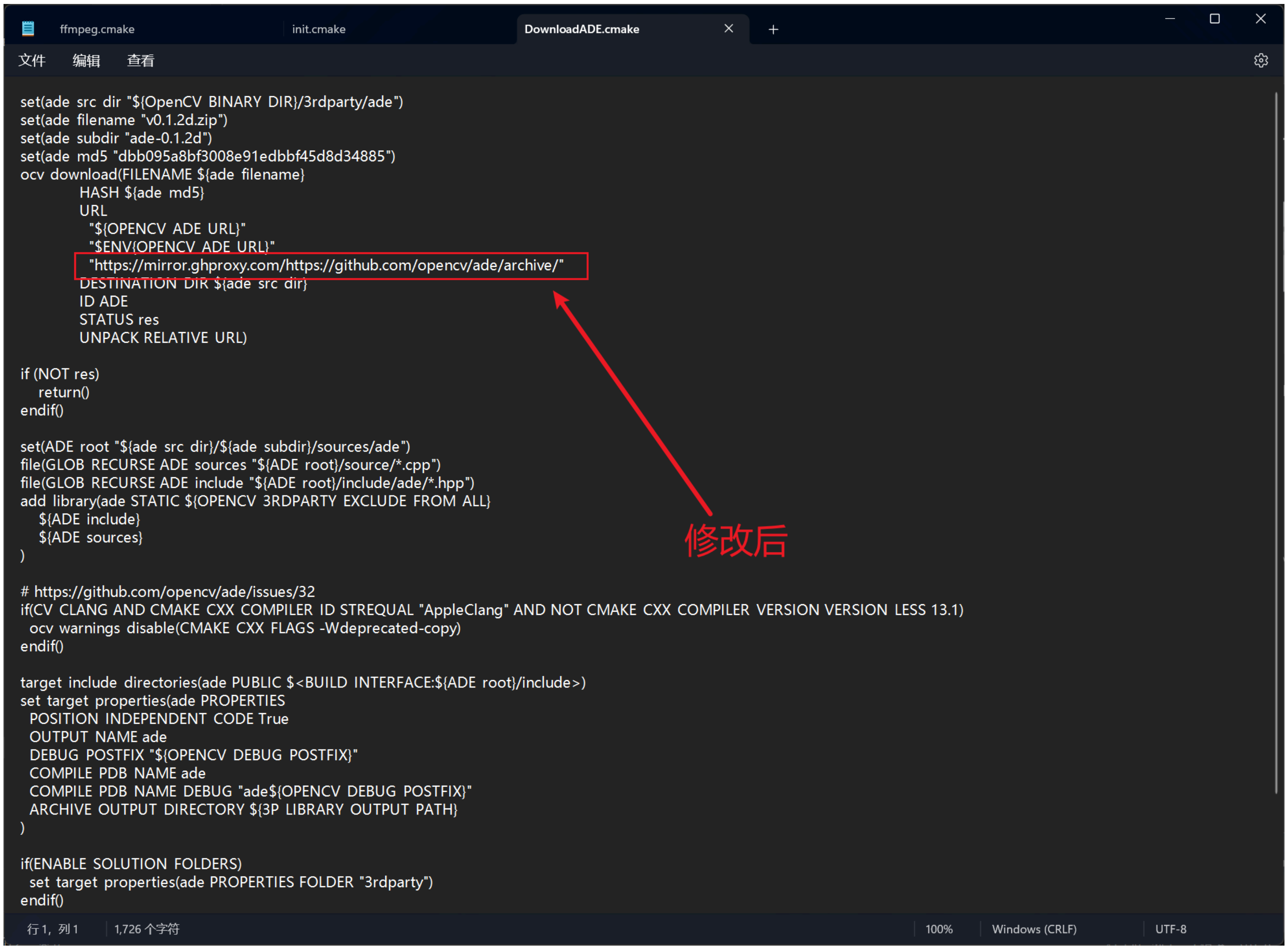
同理使用代理服务修改



```
set(ade src dir "${OpenCV BINARY DIR}/3rdparty/ade")
set(ade filename "v0.1.2d.zip")
set(ade subdir "ade-0.1.2d")
set(ade md5 "dbb095a8bf3008e91edbbf45d8d34885")
ocv download(FILENAME ${ade filename}
          HASH ${ade md5}
          URL
            "${OPENCV ADE URL}"
            "$ENV{OPENCV ADE URL}"
            "https://github.com/opencv/ade/archive/"
          DESTINATION DIR ${ade src dir}
          ID ADE
          STATUS res
          UNPACK RELATIVE URL)

if (NOT res)
    return()
endif()

set(ADE root "${ade src dir}/${ade subdir}/sources/ade")
file(GLOB RECURSE ADE sources "${ADE root}/source/*.cpp")
file(GLOB RECURSE ADE include "${ADE root}/include/ade/*.hpp")
add library(ade STATIC ${OPENCV 3RDPARTY EXCLUDE FROM ALL}
    ${ADE include}
    ${ADE sources}
)

# https://github.com/opencv/ade/issues/32
if(CV CLANG AND CMAKE CXX COMPILER ID STREQUAL "AppleClang" AND NOT CMAKE CXX COMPILER VERSION VERSION LESS 13.1)
  ocv warnings disable(CMAKE CXX FLAGS -Wdeprecated-copy)
endif()

target include directories(ade PUBLIC $<BUILD INTERFACE:${ADE root}/include>)
set target properties(ade PROPERTIES
  POSITION INDEPENDENT CODE True
  OUTPUT NAME ade
  DEBUG POSTFIX "${OPENCV DEBUG POSTFIX}"
  COMPILE PDB NAME ade
  COMPILE PDB NAME DEBUG "ade${OPENCV DEBUG POSTFIX}"
  ARCHIVE OUTPUT DIRECTORY ${3P LIBRARY OUTPUT PATH}
)

if(ENABLE SOLUTION FOLDERS)
  set target properties(ade PROPERTIES FOLDER "3rdparty")
endif()
```

把 `"https://github.com/opencv/ade/archive/"` 修改为 `"https://mirror.ghproxy.com/https://github.com/opencv/ade/archive/"`

修改后



```
set(ade src dir "${OpenCV BINARY DIR}/3rdparty/ade")
set(ade filename "v0.1.2d.zip")
set(ade subdir "ade-0.1.2d")
set(ade md5 "dbb095a8bf3008e91edbbf45d8d34885")
ocv download(FILENAME ${ade filename}
          HASH ${ade md5}
          URL
            "${OPENCV ADE URL}"
            "$ENV{OPENCV ADE URL}"
            "https://mirror.ghproxy.com/https://github.com/opencv/ade/archive/"
          DESTINATION DIR ${ade src dir}
          ID ADE
          STATUS res
          UNPACK RELATIVE URL)

if (NOT res)
   return()
endif()

set(ADE root "${ade src dir}/${ade subdir}/sources/ade")
file(GLOB RECURSE ADE sources "${ADE root}/source/*.cpp")
file(GLOB RECURSE ADE include "${ADE root}/include/ade/*.hpp")
add library(ade STATIC ${OPENCV 3RDPARTY EXCLUDE FROM ALL}
   ${ADE include}
   ${ADE sources}
)

# https://github.com/opencv/ade/issues/32
if(CV CLANG AND CMAKE CXX COMPILER ID STREQUAL "AppleClang" AND NOT CMAKE CXX COMPILER VERSION VERSION LESS 13.1)
  ocv warnings disable(CMAKE CXX FLAGS -Wdeprecated-copy)
endif()

target include directories(ade PUBLIC $<BUILD INTERFACE:${ADE root}/include>)
set target properties(ade PROPERTIES
  POSITION INDEPENDENT CODE True
  OUTPUT NAME ade
  DEBUG POSTFIX "${OPENCV DEBUG POSTFIX}"
  COMPILE PDB NAME ade
  COMPILE PDB NAME DEBUG "ade${OPENCV DEBUG POSTFIX}"
  ARCHIVE OUTPUT DIRECTORY ${3P LIBRARY OUTPUT PATH}
)

if(ENABLE SOLUTION FOLDERS)
  set target properties(ade PROPERTIES FOLDER "3rdparty")
endif()
```

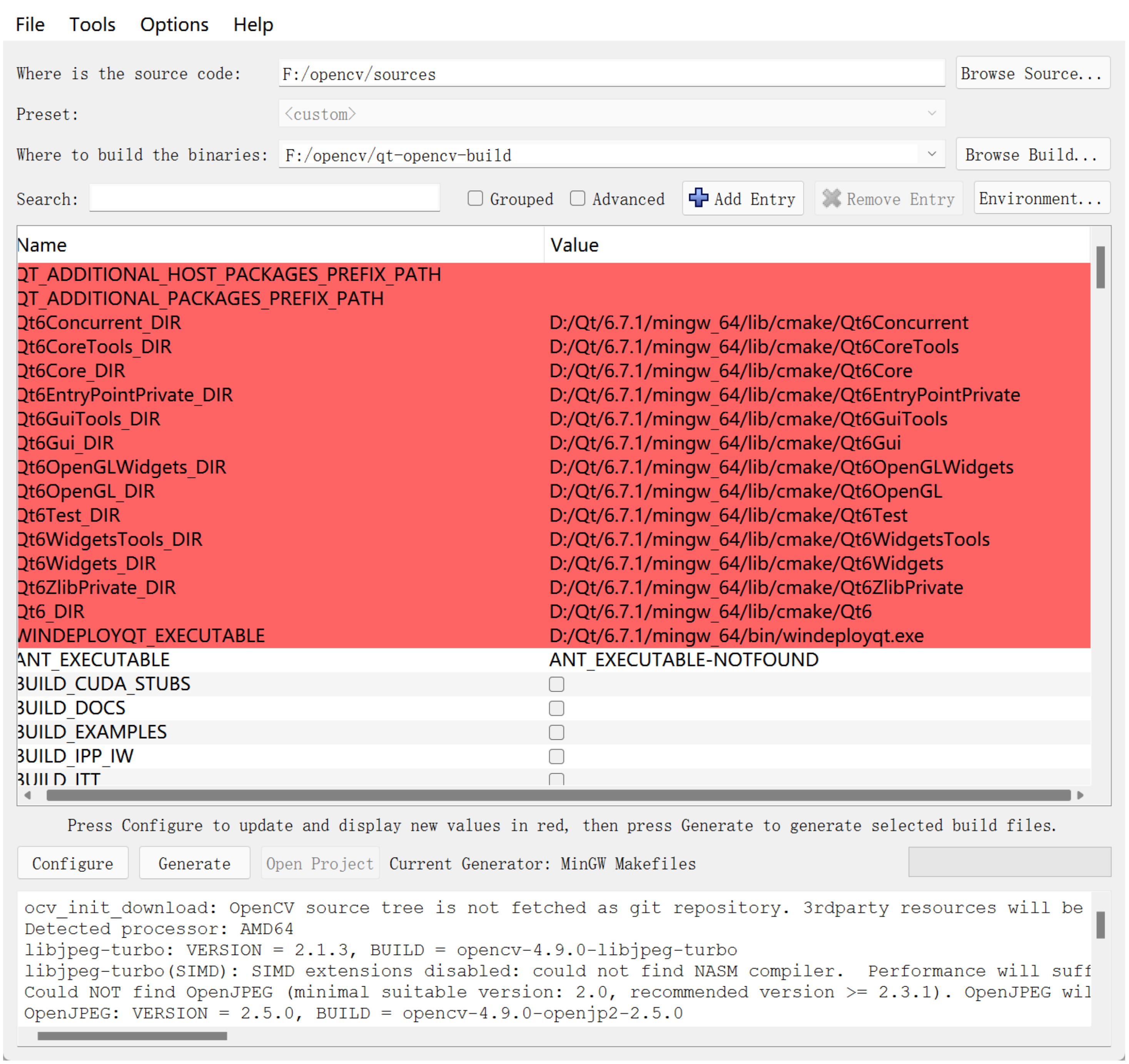
此时查阅后就不在有报错和警告

Step3. 继续配置QT

参考下面表格

Name	Value
WITH_OPENGL	选中
WITH_QT	选中
WITH_IPP	不选

选择完成之后点击 `Configure`



Where is the source code: F:/opencv/sources

Preset: <custom>

Where to build the binaries: F:/opencv/qt-opencv-build

Search:

Name	Value
QT_ADDITIONAL_HOST_PACKAGES_PREFIX_PATH	
QT_ADDITIONAL_PACKAGES_PREFIX_PATH	
Qt6Concurrent_DIR	D:/Qt/6.7.1/mingw_64/lib/cmake/Qt6Concurrent
Qt6CoreTools_DIR	D:/Qt/6.7.1/mingw_64/lib/cmake/Qt6CoreTools
Qt6Core_DIR	D:/Qt/6.7.1/mingw_64/lib/cmake/Qt6Core
Qt6EntryPointPrivate_DIR	D:/Qt/6.7.1/mingw_64/lib/cmake/Qt6EntryPointPrivate
Qt6GuiTools_DIR	D:/Qt/6.7.1/mingw_64/lib/cmake/Qt6GuiTools
Qt6Gui_DIR	D:/Qt/6.7.1/mingw_64/lib/cmake/Qt6Gui
Qt6OpenGLWidgets_DIR	D:/Qt/6.7.1/mingw_64/lib/cmake/Qt6OpenGLWidgets
Qt6OpenGL_DIR	D:/Qt/6.7.1/mingw_64/lib/cmake/Qt6OpenGL
Qt6Test_DIR	D:/Qt/6.7.1/mingw_64/lib/cmake/Qt6Test
Qt6WidgetsTools_DIR	D:/Qt/6.7.1/mingw_64/lib/cmake/Qt6WidgetsTools
Qt6Widgets_DIR	D:/Qt/6.7.1/mingw_64/lib/cmake/Qt6Widgets
Qt6ZlibPrivate_DIR	D:/Qt/6.7.1/mingw_64/lib/cmake/Qt6ZlibPrivate
Qt6_DIR	D:/Qt/6.7.1/mingw_64/lib/cmake/Qt6
WINDEPLOYQT_EXECUTABLE	D:/Qt/6.7.1/mingw_64/bin/windeployqt.exe
ANT_EXECUTABLE	ANT_EXECUTABLE-NOTFOUND
BUILD_CUDA_STUBS	☐
BUILD_DOCS	☐
BUILD_EXAMPLES	☐
BUILD_IPP_IW	☐
BUILD_ITT	☐

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: MinGW Makefiles

```
ocv_init_download: OpenCV source tree is not fetched as git repository. 3rdparty resources will be
Detected processor: AMD64
libjpeg-turbo: VERSION = 2.1.3, BUILD = opencv-4.9.0-libjpeg-turbo
libjpeg-turbo(SIMD): SIMD extensions disabled: could not find NASM compiler.  Performance will suff
Could NOT find OpenJPEG (minimal suitable version: 2.0, recommended version >= 2.3.1). OpenJPEG wil
OpenJPEG: VERSION = 2.5.0, BUILD = opencv-4.9.0-openjp2-2.5.0
```

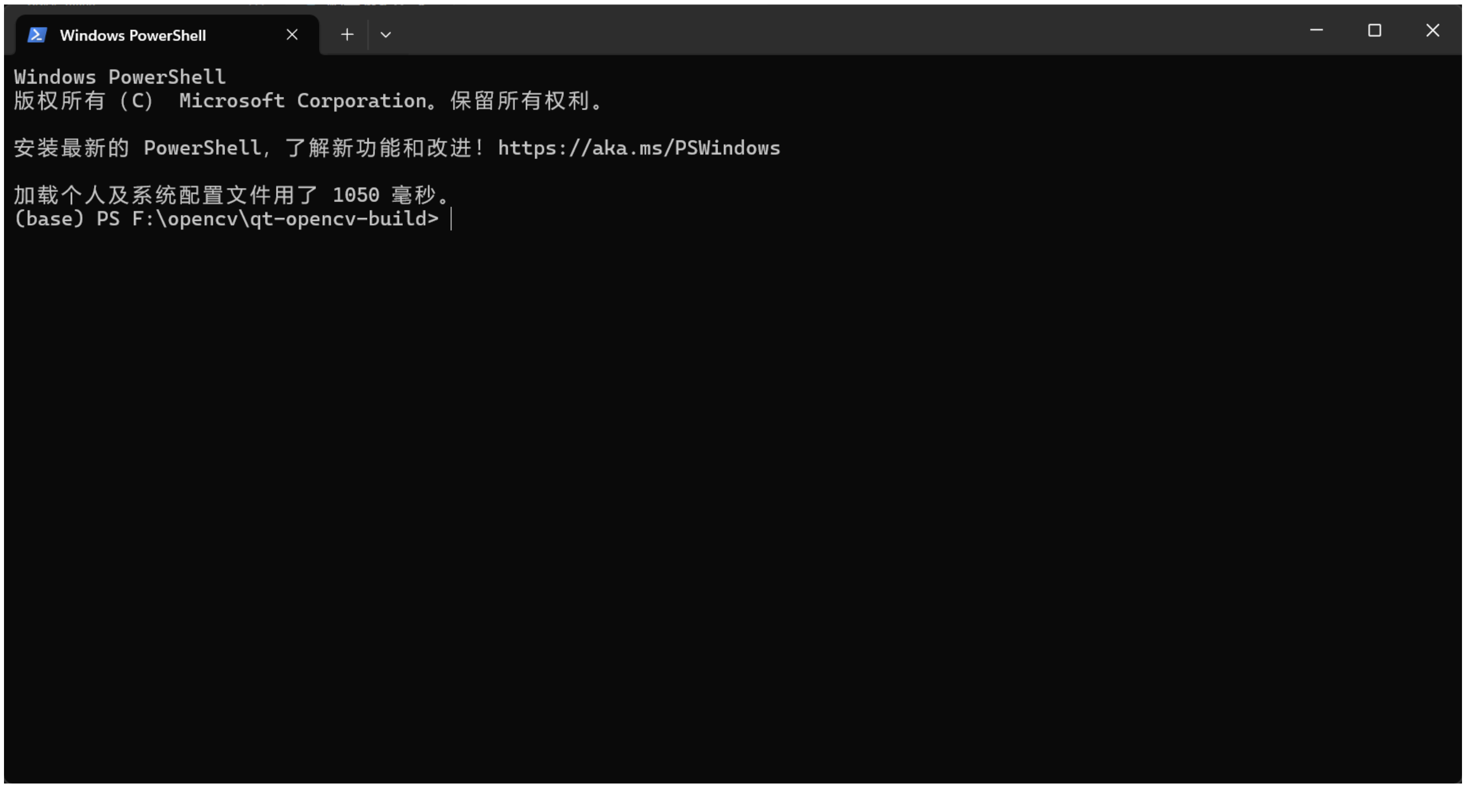
我们是 `Qt6` 的版本，所以不管 `Qt5` ，将 `Qt6_DIR`
后面的路径设置为: `D:/Qt/6.7.1/mingw_64/lib/cmake/Qt6`
注意表中的路径，一定要和所对应的路径一样,如果没有自动填写好，需要一个个去找

再次点击 `Configure`

`Configuring done` 之后点击 `Generate`

Step4. Mingw编译

打开终端，进入输出文件夹



```
Windows PowerShell
版权所有 (C)  Microsoft Corporation。保留所有权利。

安装最新的 PowerShell，了解新功能和改进！https://aka.ms/PSWindows

加载个人及系统配置文件用了 1050 毫秒。
(base) PS F:\opencv\qt-opencv-build>
```

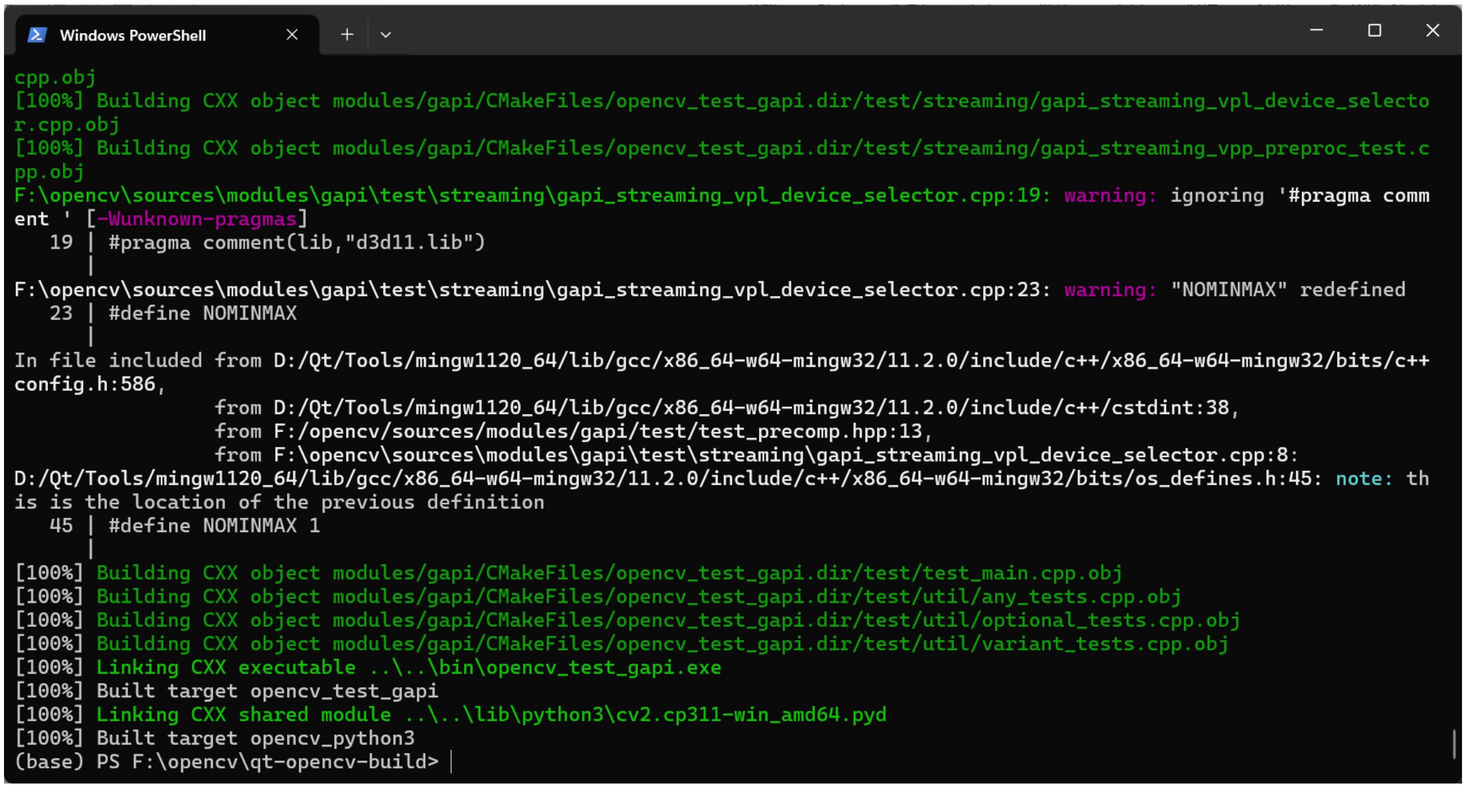
输入如下指令开始编译（-j 16 多核编译）

输入 `mingw32-make -j 16`

可以根据自己电脑是几核的来设定，如果不清楚建议使用

```
mingw32-make -j 4
```

接下来就是漫长的等待，只要没有出现报错就可以

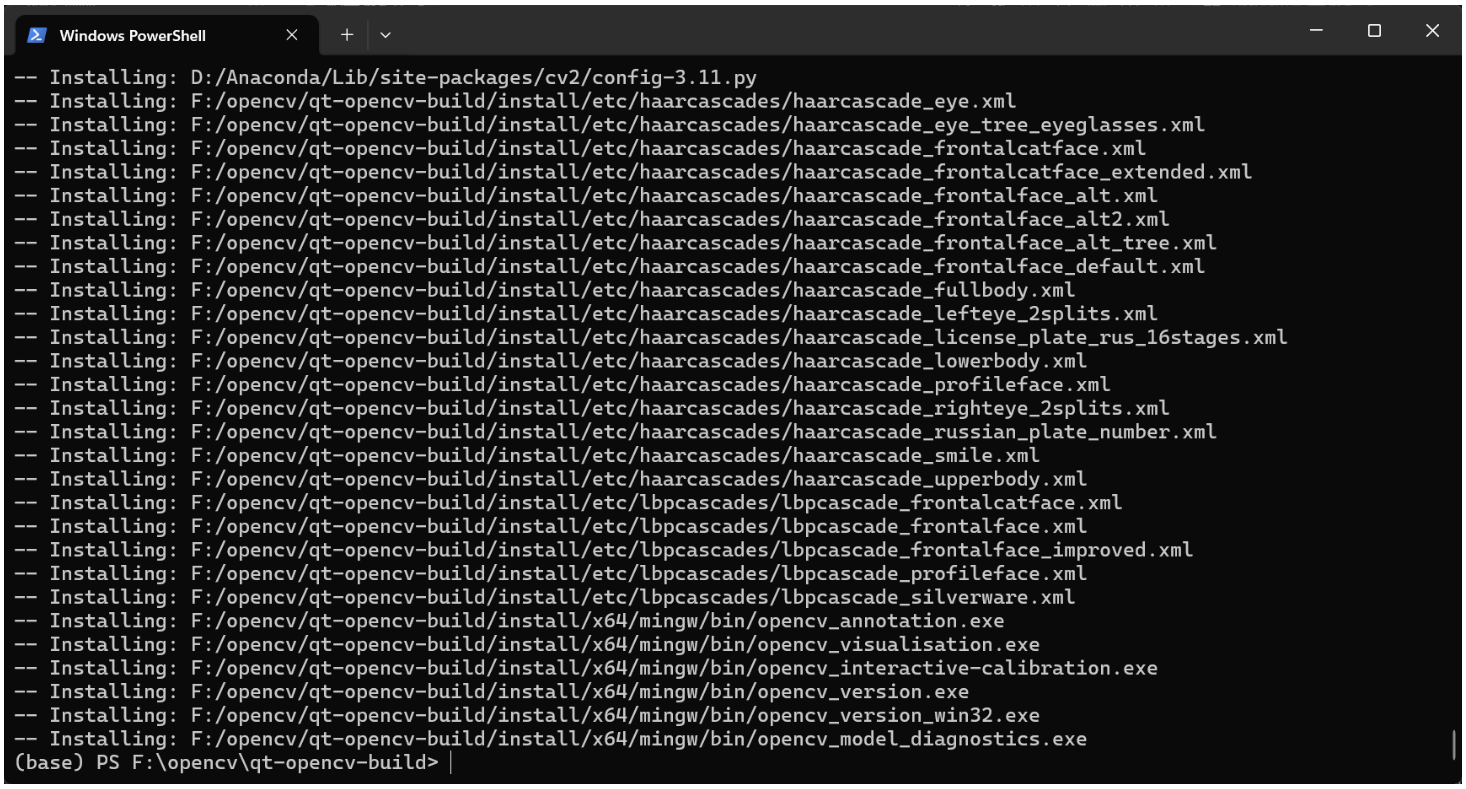


```
cpp.obj
[100%] Building CXX object modules/gapi/CMakeFiles/opencv_test_gapi.dir/test/streaming/gapi_streaming_vpl_device_selecto
r.cpp.obj
[100%] Building CXX object modules/gapi/CMakeFiles/opencv_test_gapi.dir/test/streaming/gapi_streaming_vpp_preproc_test.c
pp.obj
F:\opencv\sources\modules\gapi\test\streaming\gapi_streaming_vpl_device_selector.cpp:19: warning: ignoring '#pragma comm
ent ' [-Wunknown-pragmas]
   19 | #pragma comment(lib,"d3d11.lib")
      |
F:\opencv\sources\modules\gapi\test\streaming\gapi_streaming_vpl_device_selector.cpp:23: warning: "NOMINMAX" redefined
   23 | #define NOMINMAX
      |
In file included from D:/Qt/Tools/mingw1120_64/lib/gcc/x86_64-w64-mingw32/11.2.0/include/c++/x86_64-w64-mingw32/bits/c++
config.h:586,
                 from D:/Qt/Tools/mingw1120_64/lib/gcc/x86_64-w64-mingw32/11.2.0/include/c++/cstdint:38,
                 from F:/opencv/sources/modules/gapi/test/test_precomp.hpp:13,
                 from F:\opencv\sources\modules\gapi\test\streaming\gapi_streaming_vpl_device_selector.cpp:8:
D:/Qt/Tools/mingw1120_64/lib/gcc/x86_64-w64-mingw32/11.2.0/include/c++/x86_64-w64-mingw32/bits/os_defines.h:45: note: th
is is the location of the previous definition
   45 | #define NOMINMAX 1
      |
[100%] Building CXX object modules/gapi/CMakeFiles/opencv_test_gapi.dir/test/test_main.cpp.obj
[100%] Building CXX object modules/gapi/CMakeFiles/opencv_test_gapi.dir/test/util/any_tests.cpp.obj
[100%] Building CXX object modules/gapi/CMakeFiles/opencv_test_gapi.dir/test/util/optional_tests.cpp.obj
[100%] Building CXX object modules/gapi/CMakeFiles/opencv_test_gapi.dir/test/util/variant_tests.cpp.obj
[100%] Linking CXX executable ..\..\bin\opencv_test_gapi.exe
[100%] Built target opencv_test_gapi
[100%] Linking CXX shared module ..\..\lib\python3\cv2.cp311-win_amd64.pyd
[100%] Built target opencv_python3
(base) PS F:\opencv\qt-opencv-build>
```

编译完成

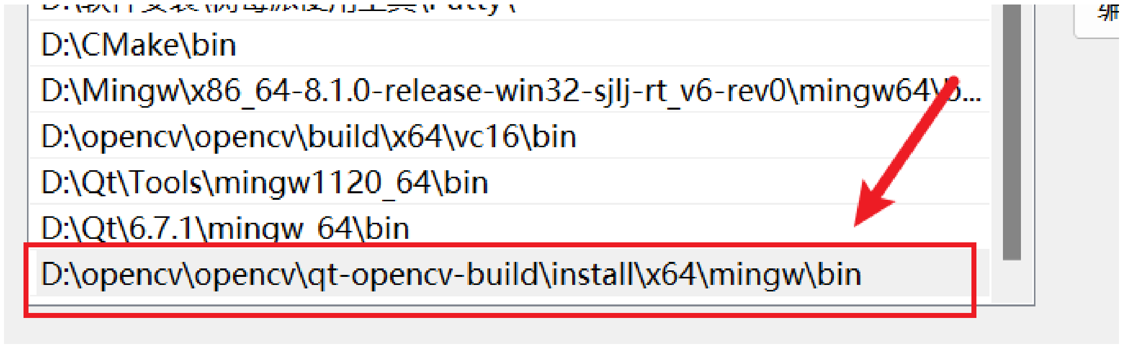
编译完成之后，输入如下指令安装:

```
mingw32-make install
```



```
-- Installing: D:/Anaconda/Lib/site-packages/cv2/config-3.11.py
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_eye.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_eye_tree_eyeglasses.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_frontalcatface.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_frontalcatface_extended.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_frontalface_alt.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_frontalface_alt2.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_frontalface_alt_tree.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_frontalface_default.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_fullbody.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_lefteye_2splits.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_license_plate_rus_16stages.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_lowerbody.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_profileface.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_righteye_2splits.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_russian_plate_number.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_smile.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/haarcascades/haarcascade_upperbody.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/lbpcascades/lbpcascade_frontalcatface.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/lbpcascades/lbpcascade_frontalface.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/lbpcascades/lbpcascade_frontalface_improved.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/lbpcascades/lbpcascade_profileface.xml
-- Installing: F:/opencv/qt-opencv-build/install/etc/lbpcascades/lbpcascade_silverware.xml
-- Installing: F:/opencv/qt-opencv-build/install/x64/mingw/bin/opencv_annotation.exe
-- Installing: F:/opencv/qt-opencv-build/install/x64/mingw/bin/opencv_visualisation.exe
-- Installing: F:/opencv/qt-opencv-build/install/x64/mingw/bin/opencv_interactive-calibration.exe
-- Installing: F:/opencv/qt-opencv-build/install/x64/mingw/bin/opencv_version.exe
-- Installing: F:/opencv/qt-opencv-build/install/x64/mingw/bin/opencv_version_win32.exe
-- Installing: F:/opencv/qt-opencv-build/install/x64/mingw/bin/opencv_model_diagnostics.exe
(base) PS F:\opencv\qt-opencv-build>
```

配置环境变量

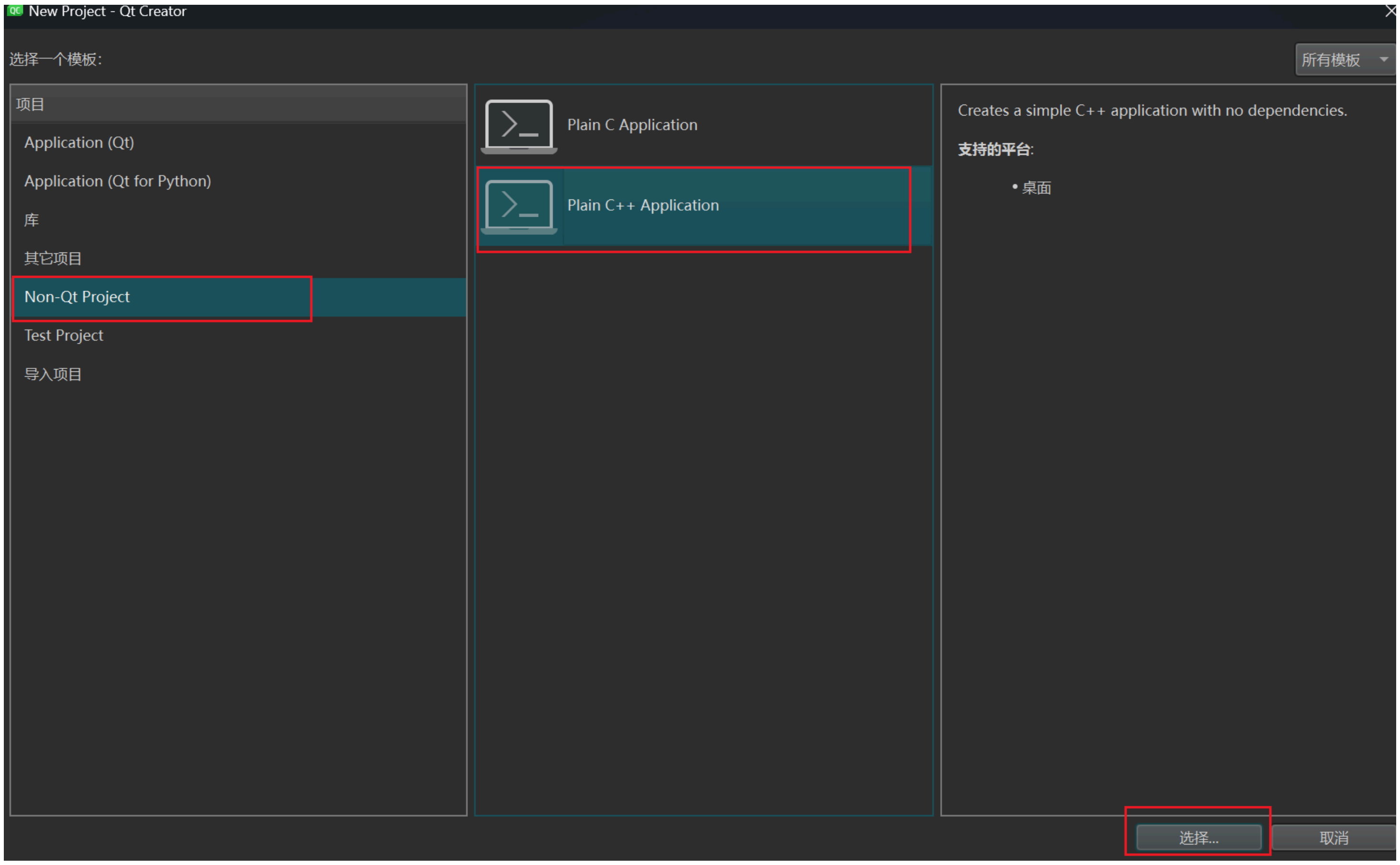


```
D:\CMake\bin
D:\Mingw\x86_64-8.1.0-release-win32-sjlj-rt_v6-rev0\mingw64\b...
D:\opencv\opencv\build\x64\vc16\bin
D:\Qt\Tools\mingw1120_64\bin
D:\Qt\6.7.1\mingw_64\bin
D:\opencv\opencv\qt-opencv-build\install\x64\mingw\bin
```

六、QT中使用opencv

Step1.新建Qt工程并配置

创建新的项目进行测试

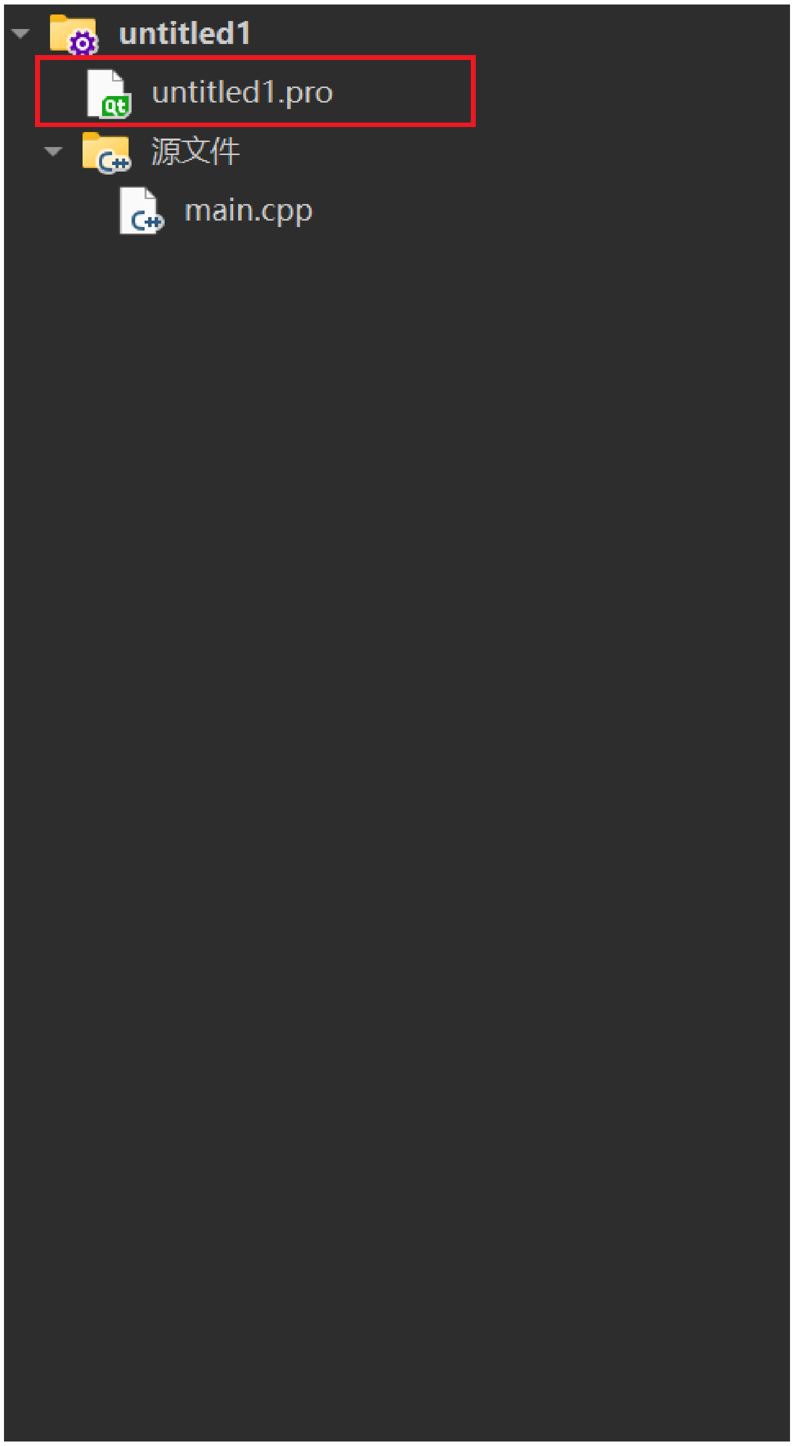


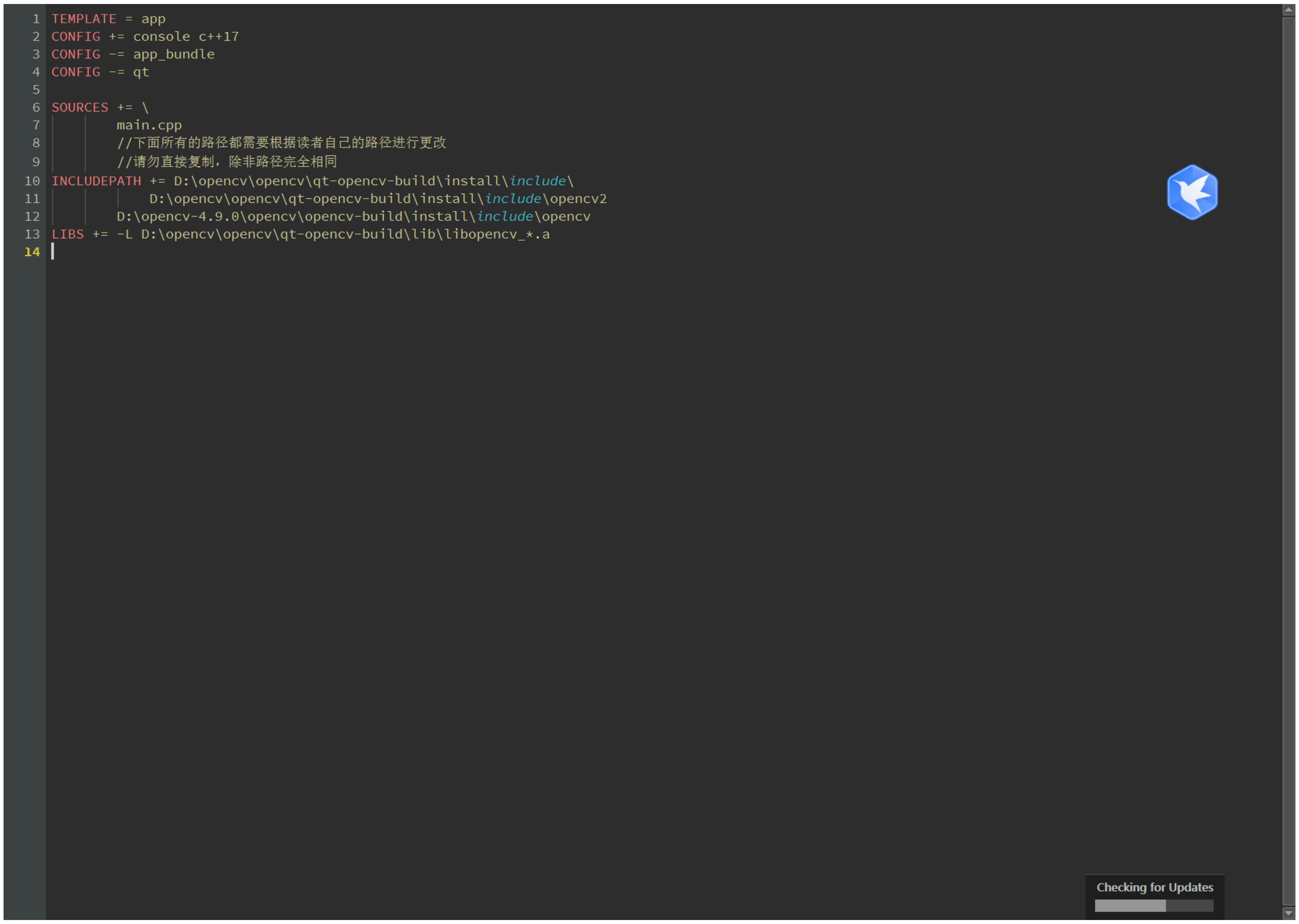
Creates a simple C++ application with no dependencies.

支持的平台:

- 桌面

修改下面这个配置文件(后缀为.pro)


untitled1
untitled1.pro
源文件
main.cpp



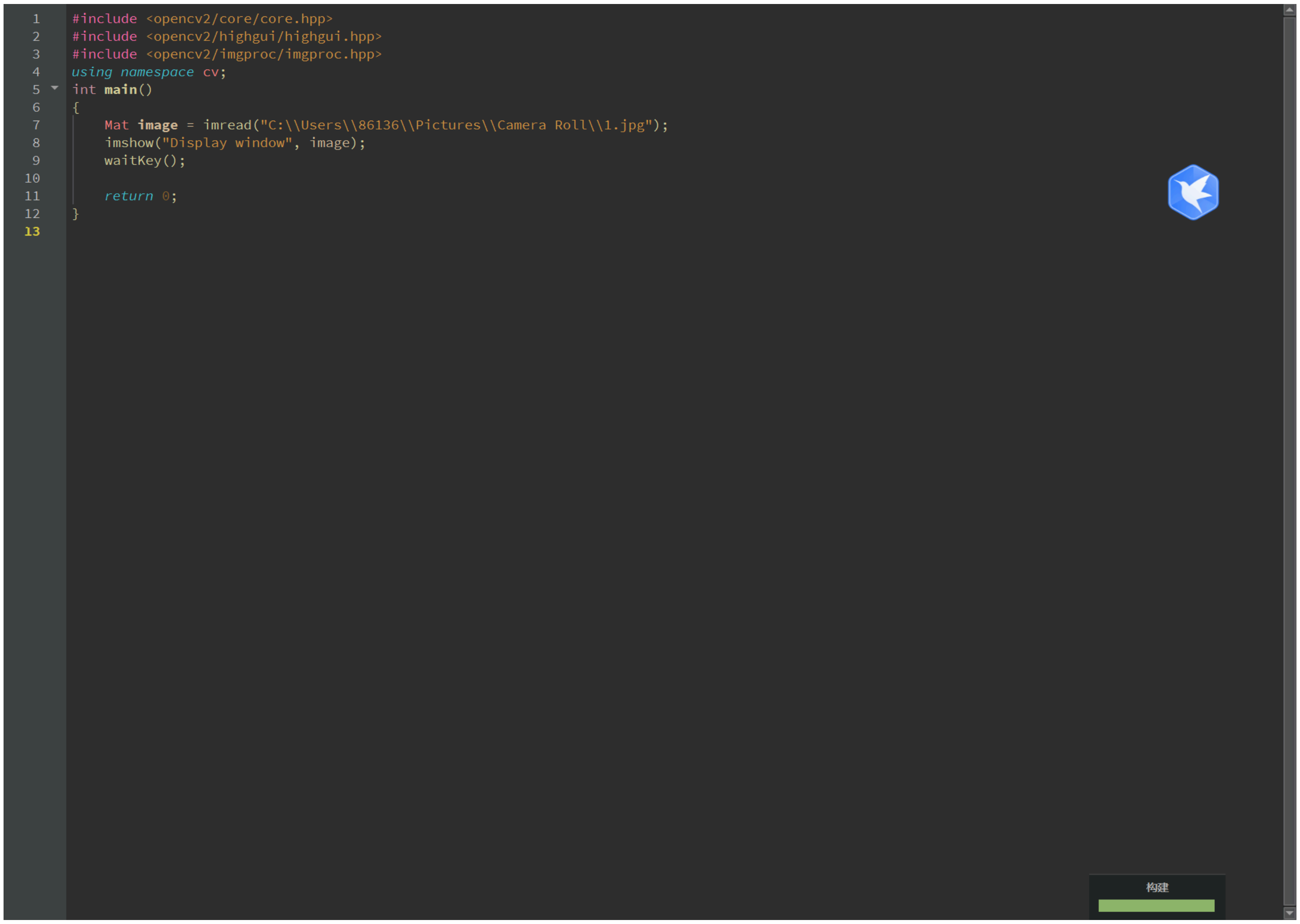
```
TEMPLATE = app
CONFIG += console c++17
CONFIG -= app_bundle
CONFIG -= qt

SOURCES += \
        main.cpp
        //下面所有的路径都需要根据读者自己的路径进行更改
        //请勿直接复制，除非路径完全相同
INCLUDEPATH += D:\opencv\opencv\qt-opencv-build\install\include\
            D:\opencv\opencv\qt-opencv-build\install\include\opencv2
        D:\opencv-4.9.0\opencv\opencv-build\install\include\opencv
LIBS += -L D:\opencv\opencv\qt-opencv-build\lib\libopencv_*.a

```

Checking for Updates

添加路径(注意换成自己文件的路径)

```
INCLUDEPATH += D:\opencv\opencv\qt-opencv-build\install\include\
               D:\opencv\opencv\qt-opencv-build\install\include\opencv2
LIBS += -L D:\opencv\opencv\qt-opencv-build\lib\libopencv_*.a
```

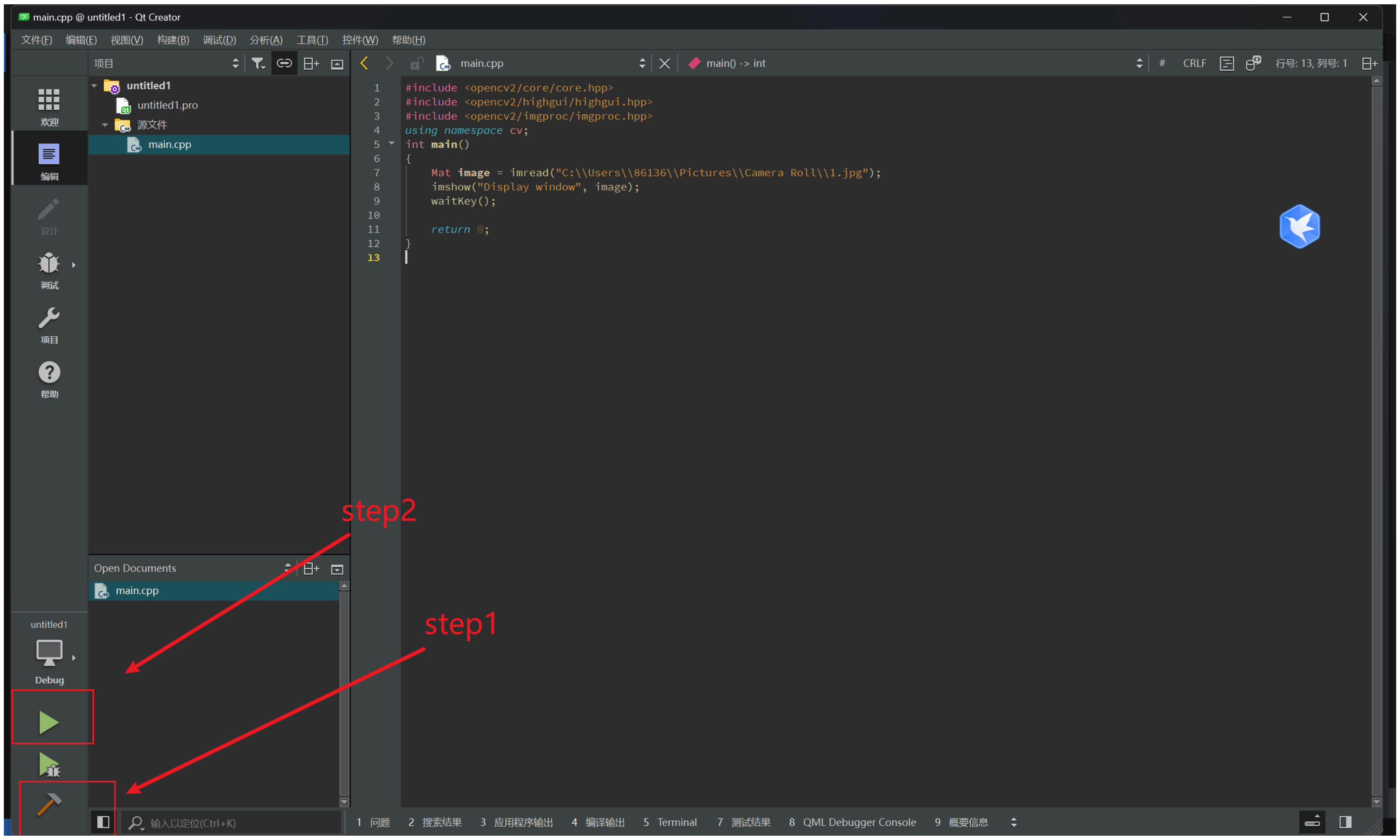
Step2.测试



```cpp
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
using namespace cv;
int main()
{
    Mat image = imread("C:\\Users\\86136\\Pictures\\Camera Roll\\1.jpg");
    imshow("Display window", image);
    waitKey();

    return 0;
}
```

测试代码

```cpp
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
using namespace cv;
int main()
{
    Mat image = imread("C:\\Users\\86136\\Pictures\\Camera Roll\\1.jpg");    //换成自己的路径
    imshow("Display window", image);
    waitKey();

    return 0;
}
```


main.cpp @ untitled1 - Qt Creator
文件(F) 编辑(E) 视图(V) 构建(B) 调试(D) 分析(A) 工具(T) 控件(W) 帮助(H)
项目
untitled1
untitled1.pro
源文件
main.cpp
欢迎
编辑
设计
调试
项目
帮助
main() -> int
CRLF
行号: 13, 列号: 1
Open Documents
untitled1
Debug
step2
step1
输入以定位(Ctrl+K)
1 问题 2 搜索结果 3 应用程序输出 4 编译输出 5 Terminal 7 测试结果 8 QML Debugger Console 9 概要信息

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
using namespace cv;
int main()
{
    Mat image = imread("C:\\Users\\86136\\Pictures\\Camera Roll\\1.jpg");
    imshow("Display window", image);
    waitKey();

    return 0;
}
```