

VsCode + Qt6.8.1 + Ninja + Cmake安装

一、安装Qt+Ninja+Cmake

安装包我 (zzzzhhhh) 直接提供一份。如果已经安装，可以通过根目录下的MaintenanceTool.exe修改。

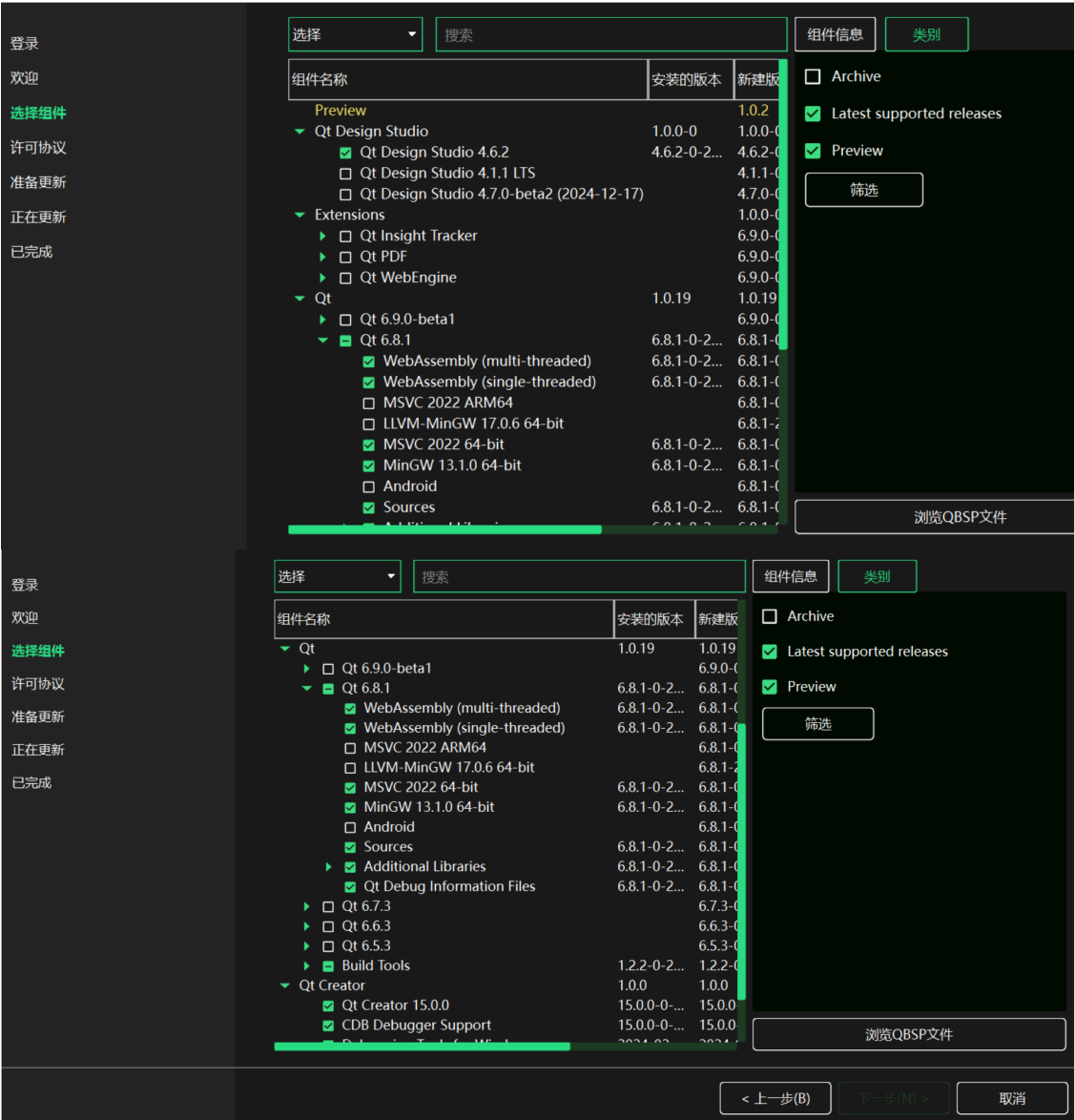
进入安装界面后勾选以下组件 qt是自帶Cmake与Ninja，所以可以直接通过Qt一起安装。

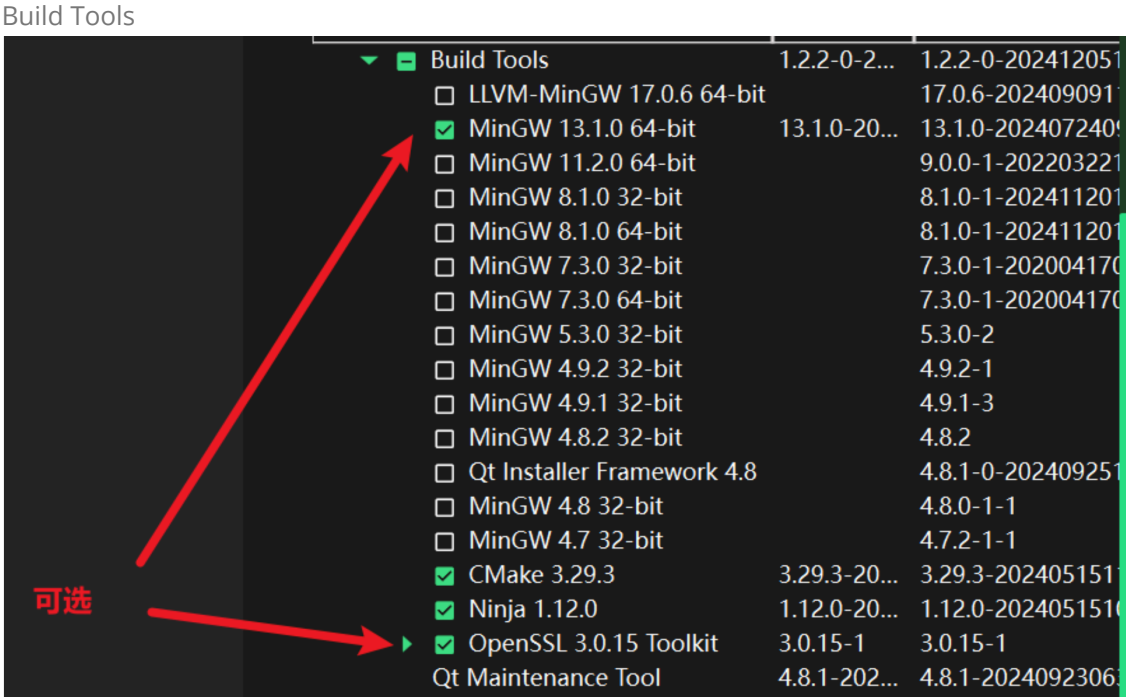
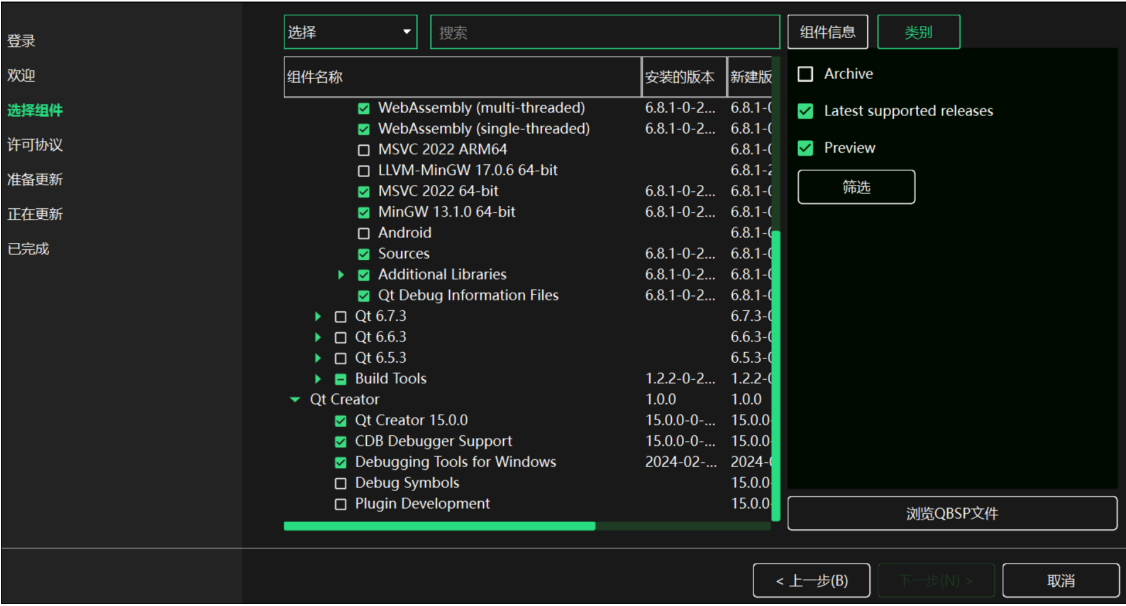
下载qit 6.8.1时建议使用国内源，下载速度更快

```
C:\Windows\System32\cmd.e  X  +  v
Microsoft Windows [版本 10.0.22621.4317]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\zdt\Desktop>qt-online-installer-windows-x64-4.8.1.exe --mirror https://mirrors.aliyun.com/qt/
```

现在(20250107)选择下载Qt6.8.1





安装QT结束后，进行环境变量配置

QT6_PATH

E:\QT6.8

```
E:\QT6.8\Tools
E:\QT6.8\Tools\Ninja
E:\QT6.8\Tools\CMake_64\bin
E:\QT6.8\Tools\QtCreator\bin
E:\QT6.8\6.8.1
E:\QT6.8\6.8.1\msvc2022_64
E:\QT6.8\6.8.1\msvc2022_64\bin
```

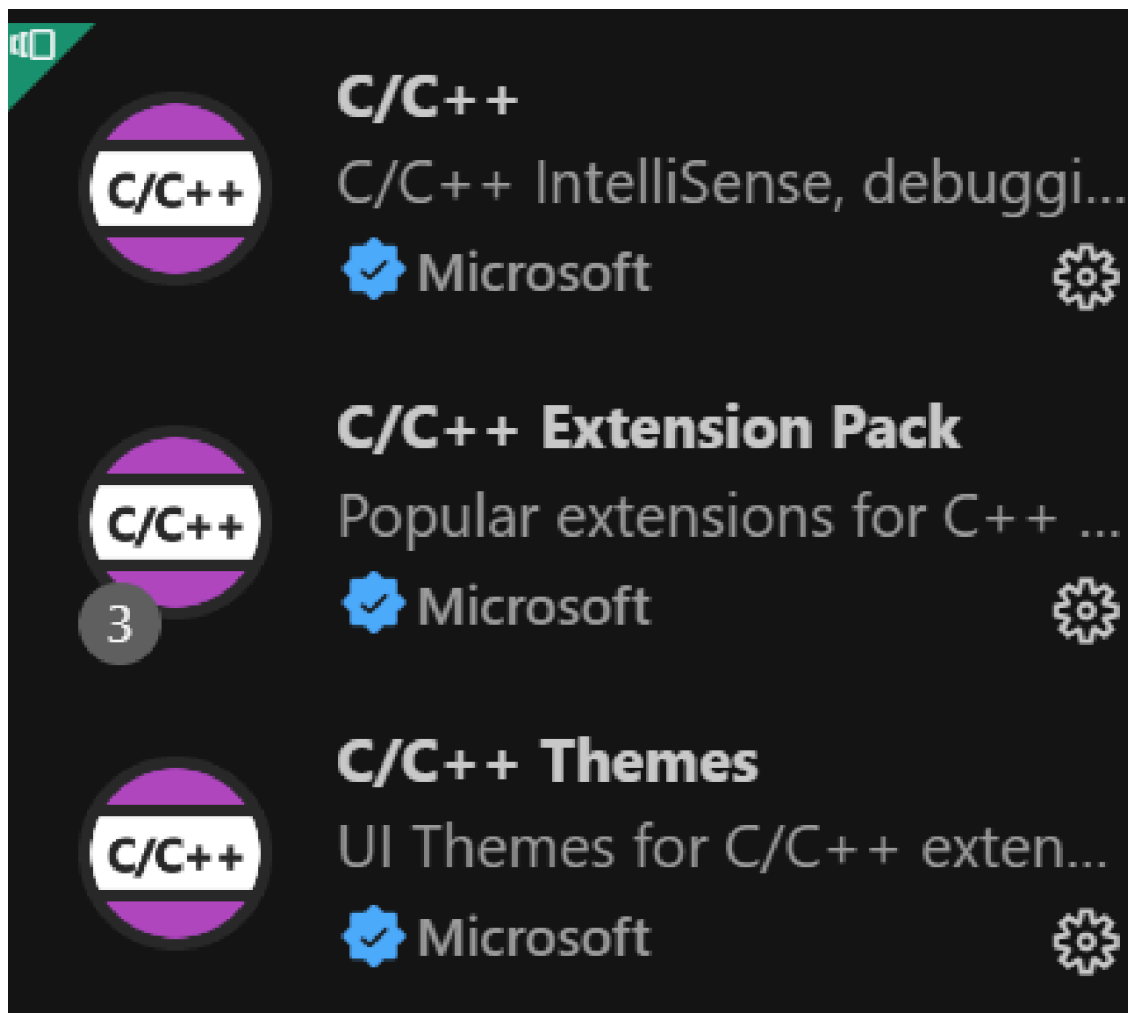
在Path中，尽量遵循一定的顺序，优先级较高的路径应该放在前面。

二、安装VsCode

点击链接[VsCode](#)进入官网，直接下载Windows版本就行。

VsCode进入后需要安装插件，不配置插件中的路径。

如果要使用Clangd就要把c/c++系列替换为Clangd与Clangd-format（可选）





Chinese (Simplified) (简体... 中文(简体)



Microsoft



CMake

CMake language support for ...
twxs



CMake Tools

Extended CMake support in ...



Microsoft



现在建议都用Clangd!!!



Clang-Format

⌚ 8ms

Use Clang-Format in Visual Stu...

Xaver Hellauer



clangd

⌚ 67ms

C/C++ completion, navigation,...



LLVM



注：LLVM是一些代码提示，代码补全，编译预生成，编译套件工具。

在我们的工程中的：


clangd.exe（代码静态检查，错误提示，代码补全），clang-format.exe（自动格式化整理代码）

如果大家配不好，也不需要配，可以用你自己喜欢的插件，在开发中不是必要的唯一。

工程中的必要插件是C/C++插件，cmake tools插件。


除了上面两个插件，其他的都不是必须的。


Qt Qml
Qt Qml Support
Qt Group

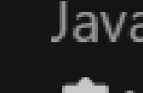


Console Ninja

JavaScript console.log outp...

 Wallaby.js






ninja-build

A syntax highlighter for Ninj...

Suraj Barkale



三、安装Visual Studio 2022

正在安装 - Visual Studio Community 2022 - 17.12.3

工作负荷 单个组件 语言包 安装位置

1 在选择安装项时需要帮助? [详细信息](#)

Python 开发 ☐

对 Python 进行编辑、调试、交互式开发和源代码管理。

Node.js 开发 ☐

使用 Node.js (一个由异步事件驱动的 JavaScript 运行时) 生成可缩放的网络应用程序。

桌面应用和移动应用 (5)

.NET Multi-platform App UI 开发 ☐

使用 C# 和 .NET MAUI 从单个基本代码库生成 Android、iOS、Windows 和 Mac 应用。

使用 C++ 的桌面开发 ☒

使用所选工具(包括 MSVC、Clang、CMake 或 MSBuild)生成适用于 Windows 的现代 C++ 应用。

.NET 桌面开发 ☐

将 C#, Visual Basic 和 F# 与 .NET 和 .NET Framework 一起使用, 生成 WPF、Windows 窗体和控制台应用程序。

Windows 应用程序开发 ☐

使用 WinUI 和 C# 为 Windows 平台创建应用程序, 或者选择性地使用 C++ 进行构建。

位置
E:\Soft\VisualStudio [更改...](#)

继续操作即表示你同意所选 Visual Studio 版本的[许可证](#)。我们还提供通过 Visual Studio 下载其他软件的功能。此软件单独进行许可, 如[第三方公告](#)或其随附的许可证中所述。继续即表示你同意这些许可证。

配置说明

以下内容为settings.json中的配置解释。

其中需要配置的环境变量为：

{env:CMAKE_PATH}: cmake.exe的路径，{env:CMAKE_PATH}需要在系统环境变量或系统用户变量中添加CMAKE_PATH，内容为cmake.exe的安装路径，例：E:\QT6.8\Tools\CMake_64\bin。

{env:VCPKG_ROOT}: vcpkg.cmake的路径，vcpkg.cmake路径为拉取vcpkg的路径下，例：E:\QTproject\Qtstructure\vcpkg_inner\vcpkg。

{env:QT6_PATH}: Qt6.8.1的安装路径，例：E:\QT6.8。

{

```
// 禁用 C/C++ 扩展中内置的代码补全和跳转到定义功能
"C_Cpp.intelliSenseEngine": "disabled",

// 指定 clangd.exe 的路径（安装LLVM的路径下）
"clangd.path": "C:/Program Files/LLVM/bin/clangd.exe",
"clangd.arguments": [
    "--compile-commands-dir=${workspaceFolder}/build", // 编译命令的目录路径
    "--background-index", // 启动后台索引以提高检索速度
    "--clang-tidy" // 启动 clang-tidy 静态分析工具
],

// 指定 clang-format.exe 的路径（安装LLVM的路径下）
"clang-format.executable": "C:/Program Files/LLVM/bin/clang-format.exe",

// C/C++ 语言特定设置
"[cpp]": {
    "editor.defaultFormatter": "xaver.clang-format", // 默认使用的代码格式化工具为
xaver.clang-format
    "editor.formatOnSave": true // 开启保存时自动格式化代码的功能
},

// QML 语言特定设置
"[qml]": {
    "editor.defaultFormatter": null, // 不使用默认的代码格式化工具
    "editor.formatOnSave": false // 关闭保存时自动格式化代码的功能
},

// CMake 相关设置
"cmake.useCMakePresets": "never", // 禁用 cmake 预设
"cmake.configureOnOpen": true, // 打开项目时自动配置 cmake
"cmake.cmakePath": "${env:CMAKE_PATH}/cmake.exe", // 指定 cmake.exe 的路径
"cmake.generator": "Ninja", // 设定 cmake 生成器为 Ninja
"cmake.buildDirectory": "${workspaceFolder}/build", // 指定构建目录的位置
"cmake.configureSettings": {
    "CMAKE_TOOLCHAIN_FILE": "${env:VCPKG_ROOT}/scripts/buildsystems/vcpkg.cmake",
// 指定工具链文件的位置
    "CMAKE_PREFIX_PATH": "${env:QT6_PATH}/6.8.1/msvc2022_64/lib/cmake" // 指定
cmake 的前缀路径
},
"cmake.configureArgs": ["-DCMAKE_BUILD_TYPE=${buildType}"], // 配置构建类型参数

// Qt QML 相关设置
```

```
"qt-qml.qmls.customExePath": "qmls.exe" // 指定自定义的 QML 服务器可执行文件路径
```

```
}
```

四、成功

在CmakeLists.txt出按下CTRL+S就可以生成构建

```
[proc] 执行命令: E:/Qt6.8/tools/CMake_64/bin/cmake.exe --version
[proc] 执行命令: E:/Qt6.8/tools/CMake_64/bin/cmake.exe -E capabilities
[kit] 已成功从 C:\Users\la\AppData\Local\CMakeTools\cmake-tools-kits.json 加载 11 工具包
[variant] 已加载一值列表
[proc] 执行命令: chcp
[main] 正在配置项目: toges_infinity_station
[proc] 执行命令: E:/Qt6.8/tools/CMake_64/bin/cmake.exe -DCMAKE_BUILD_TYPE:String=Debug -DCMAKE_TOOLCHAIN_FILE:String=E:/Q/project/Qtstructure/cvpg_inno/cvpg/scripts/buildsystem/cvpg.cmake -DCMAKE_PREFIX_PATH:String=E:/Qt6.8/6.8.1/mvc2022_64/1th/cmake
DCMAKE_EXPORT_COMPILE_COMMANDS:BOOL=TRUE -DCMAKE_BUILD_TYPE=Debug --no-warn-unused-cli -Sf:/Q/project/Qtstructure/toges_infinity_station -B:/Q/project/Qtstructure/toges_infinity_station/build -G Ninja
[cmake] Not searching for unused variables given on the command line.
[cmake] build type: Debug
[cmake] Executable will be built in: E:/Q/project/Qtstructure/toges_infinity_station/x64-windows-Debug
[cmake] res/qml/home.qml;res/qml/main.qml
[cmake] -- Could NOT find WrapVulkanHeaders (missing: Vulkan_INCLUDE_DIR)
[cmake] -- Could NOT find WrapVulkanHeaders (missing: Vulkan_INCLUDE_DIR)
[cmake] -- Configuring done (9.7s)
[cmake] -- Generating done (0.2s)
[cmake] -- Build files have been written to: E:/Q/project/Qtstructure/toges_infinity_station/build
```

五、可能会遇到的问题

各位遇到问题可以告诉我补充