

CMake+VSCode编译运行C++程序简单教程



Spike

10 人赞同了该文章

前言

本教程旨在帮助不想使用vs，但是又想使用VScode的调试功能而不想手动编译的人，只能最低限度满足要求，很多地方还不完善，如不支持源代码在不同文件夹下编译，不包含CMake语法的教程，需要进阶操作请自行搜索CMake语法，

环境安装

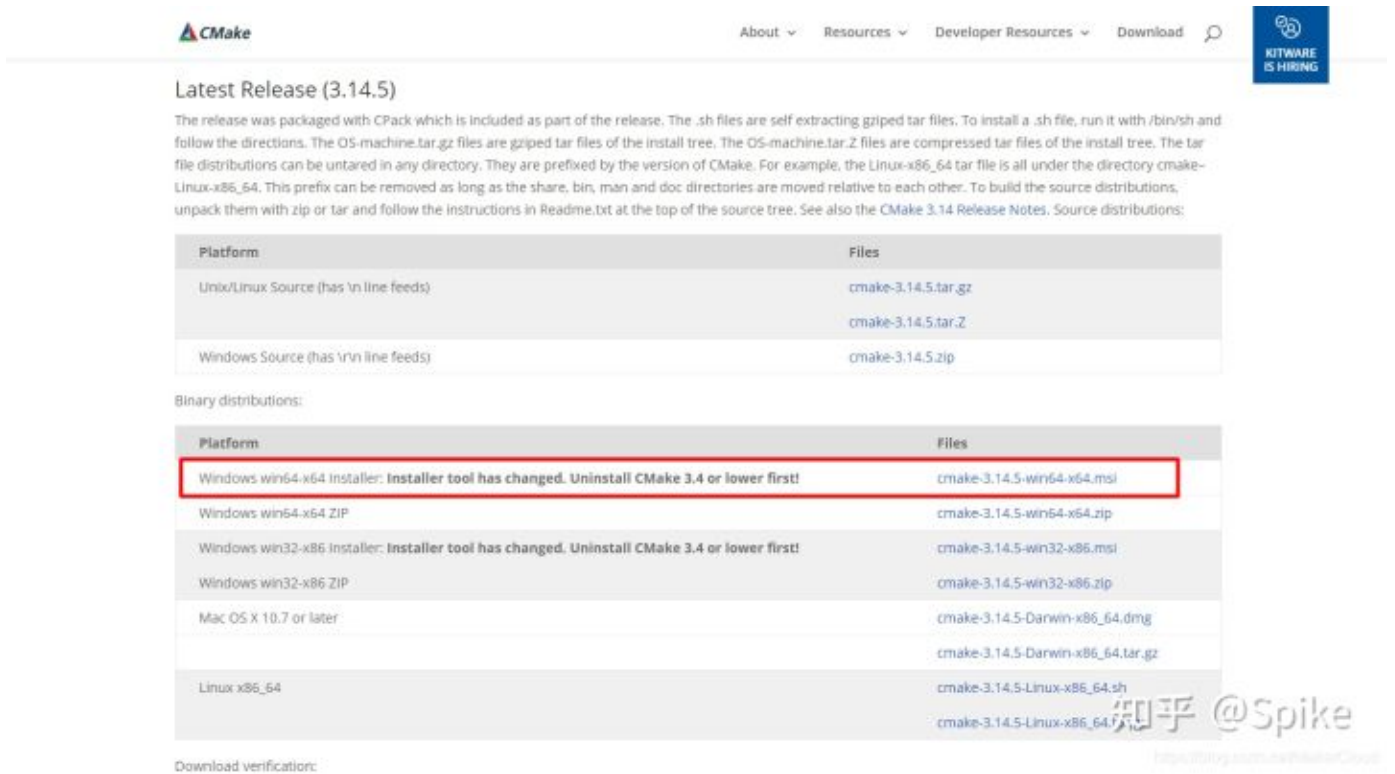
安装和配置MinGw

- MinGw中包含gcc,g++等多种编译器，可以在windows上使用（windows无法直接安装gcc），下载链接：sourceforge.net/project...
解压时尽量解压在某个盘的根目录。
- 将MinGW下bin目录的路径添加到环境变量中
- 进入bin文件夹，找到mingw32-make.exe，复制一份，将其中一份重命名为make.exe（还是保存在bin文件夹中）
- 验证是否配置成功，cmd中输入
gcc -v
make -v
可以轻易判断是否安装成功

安装Cmake

下载链接：cmake.org/download/

选择一个后缀为.msi的安装包下载，尽量选最上面的，比较稳定，如图



Latest Release (3.14.5)

The release was packaged with CPack which is included as part of the release. The .sh files are self extracting gzipped tar files. To install a .sh file, run it with /bin/sh and follow the directions. The OS-machine.tar.gz files are gzipped tar files of the install tree. The OS-machine.tar.Z files are compressed tar files of the install tree. The tar file distributions can be untared in any directory. They are prefixed by the version of CMake. For example, the Linux-x86_64 tar file is all under the directory cmake-Linux-x86_64. This prefix can be removed as long as the share, bin, man and doc directories are moved relative to each other. To build the source distributions, unpack them with zip or tar and follow the instructions in Readme.txt at the top of the source tree. See also the CMake 3.14 Release Notes. Source distributions:

Platform	Files
Unix/Linux Source (has \n line feeds)	cmake-3.14.5.tar.gz
	cmake-3.14.5.tar.Z
Windows Source (has \r\n line feeds)	cmake-3.14.5.zip

Binary distributions:

Platform	Files
Windows win64-x64 Installer: Installer tool has changed. Uninstall CMake 3.4 or lower first!	cmake-3.14.5-win64-x64.msi
Windows win64-x64 ZIP	cmake-3.14.5-win64-x64.zip
Windows win32-x86 Installer: Installer tool has changed. Uninstall CMake 3.4 or lower first!	cmake-3.14.5-win32-x86.msi
Windows win32-x86 ZIP	cmake-3.14.5-win32-x86.zip
Mac OS X 10.7 or later	cmake-3.14.5-Darwin-x86_64.dmg
	cmake-3.14.5-Darwin-x86_64.tar.gz
Linux x86_64	cmake-3.14.5-Linux-x86_64.sh
	cmake-3.14.5-Linux-x86_64.tar.gz

Download verification:

安装时记得勾选 “Add CMake to the system PATH for all users” ，这样就不用自己再配置环境变量了

安装后在cmd输入

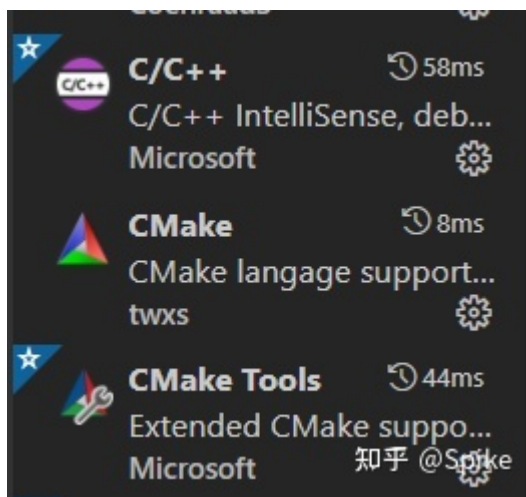
```
cmake -version
```

验证是否安装成功

VSCode 环境

需要的插件有

- C/C++
- CMake
- CMake Tools



CMake编写

手动编译

选择一个文件夹做工作区，在该文件夹下新建文件

CMakeLists.txt

输入以下内容

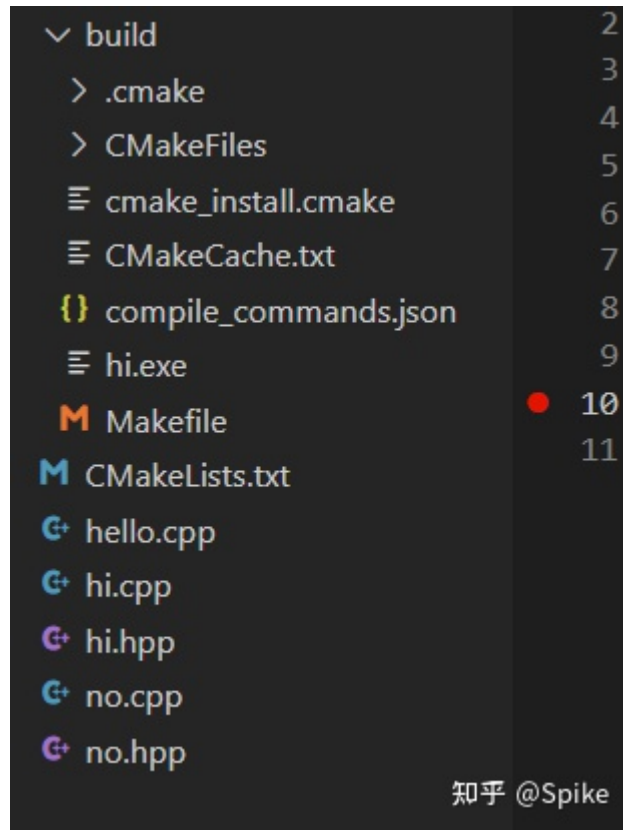
```
cmake_minimum_required(VERSION 3.0.0) //指定使用的CMake的最低版本
project(h) //指定一个项目名称
SET(CMAKE_BUILD_TYPE "Debug") //指定为调试模式，这样就可以用VSC的调试功能了
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -g")
aux_source_directory(. SRC_LIST) //前面的"."表示该目录下所有.cpp文件，把这些文件用"SRC_I
add_executable(hi ${SRC_LIST}) // "hi"是生成的可执行文件名称，需要自己指定
```

保存后可能vscode会直接开始自动build，创建build文件夹，同时也可以看到可执行文件也已经生成在build文件夹下，如果没有自动build，按ctrl+shift+p，然后选择CMake: select a kit，或在终端中输入如下命令：

```
mkdir build
cd build
cmake ../ //将CMakeLists翻译为Makefiles
```

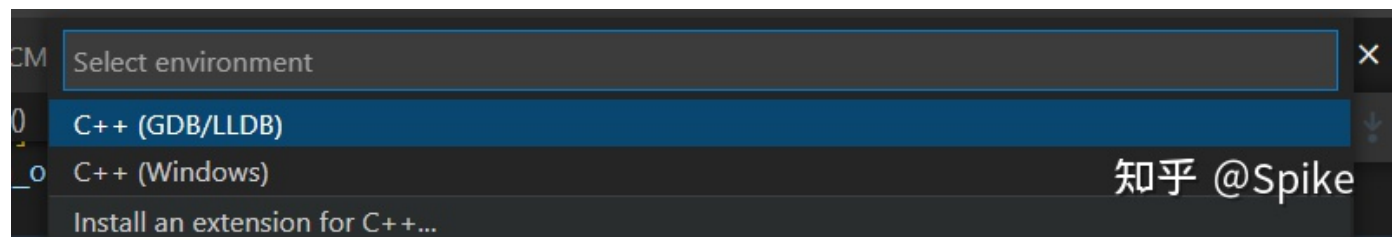
```
make          //编译,如果使用mingw编译,输入mingw32-make  
./xxx.exe    //执行程序
```

之后可以看到



启用调试

在主程序的页面，按f5，选择gdb，如图



点击后会生成launch.json

```

"version": "0.2.0",
  "configurations": [
    {
      "name": "g++.exe - Build and debug active file",
      "type": "cppdbg",
      "request": "launch",
      "program": "${workspaceFolder}/build/hi.exe", //这里需要改成该项目自己指定的
      "args": [],
      "stopAtEntry": false,
      "cwd": "${fileDirname}",
      "environment": [],
      "externalConsole": false,
      "MIMode": "gdb",
      "miDebuggerPath": "C:\\MinGW\\mingw64\\bin\\gdb.exe",
      "setupCommands": [
        {
          "description": "Enable pretty-printing for gdb",
          "text": "-enable-pretty-printing",
          "ignoreFailures": true
        }
      ],
      "preLaunchTask": "make" //这里修改为"make"
    }
  ]
}

```

之后配置task.json文件复制粘贴如下代码（从别的大佬那里copy来的）

```

{
  "version": "2.0.0",
  "options": {
    "cwd": "${workspaceRoot}/build"
  },
  "tasks": [
    {
      "label": "cmake",
      "type": "shell",
      "command": "cmake",
      "args": ["-G", "Unix Makefiles", "-DCMAKE_BUILD_TYPE=Debug", ".."]
    },
    {

```

```
    "label": "make",
    "group": {
        "kind": "build",
        "isDefault": true
    },
    "type": "shell",
    "command": "make",
    "args": []
}
]
```

之后就可以直接f5启用调试功能了

操作流程

先输入上述命令build（可以写个批处理文件执行），再按f5。

注意事项

CMakeLists.txt中指定的可执行文件名称与launch.json中所指定的必须相同

新建源代码后需要删除原来的build文件夹，重新build

待解决事项

- 可执行文件名称若为中文，理论上可以用set来将中文名称设置成变量传入执行表，还未具体操作
- 生成的可执行文件必须自己指定名称，而不能默认成主程序的名称
- 在一个工作区下临时创建一个新程序（可能用来调试），必须在CMakeLists.txt中重新指定需要编译链接的文件

发布于 10-04