VSCode for Python & C++ windows

- 本文档介绍如何在mac和windows下配置visual studio code，用于python和c++的教学。
- 这个配置过程，对于缺少windows和mac使用经验的同学来说，可能有些痛苦，但是这个过程做一遍就可以了；付出这些代价，换回的是自由，小巧和强大、通用的vscode for pyton, c++ and everything！！！
- 因为使用的都是开源跨操作系统软件，可能对中文、空格等文件或目录（文件夹）名称不友好，请避免安装到有中文后者空格的目录中。

待下载软件列表

这些软件请从各自官网下载，下文提供百度网盘链接的，可以直接通过链接下载。

1. visual studio code
    a. windows (https://pan.baidu.com/s/1l2zo-ERG8uhI0YggBiYJnA 提取码: ha85),
    b. mac (官网下载)
2. MinGW64，mac系统不需要本文件。
    a. 官网下载：在https://sourceforge.net/projects/mingw-w64/files/ 下载 **MinGW-W64 GCC-8.1.0下的** x86_64-posix-seh，别的版本应该也行，但是我没测试过。
    b. 百度网盘：https://pan.baidu.com/s/1pOnRBenzO4BR8wtVzMwkBg，提取码：qkyx
    c. 备注：下载后，是一个后缀为.7z的压缩文件：x86_64-posix-seh.7z，可以通过winrar等软件解压，可以自己百度，安装喜欢的解压软件。
3. Python > 3.10，官网下载，或者
    a. python的windows 64位安装包，链接: https://pan.baidu.com/s/12KXyCt_t1h0QPrEpQGApaQ 提取码: v1rr

b. mac下的python安装包：链接: https://pan.baidu.com/s/1l-_zgMyJqGtNmftx0XQYug 提取码: mqpk

环境配置开始！！！

安装步骤

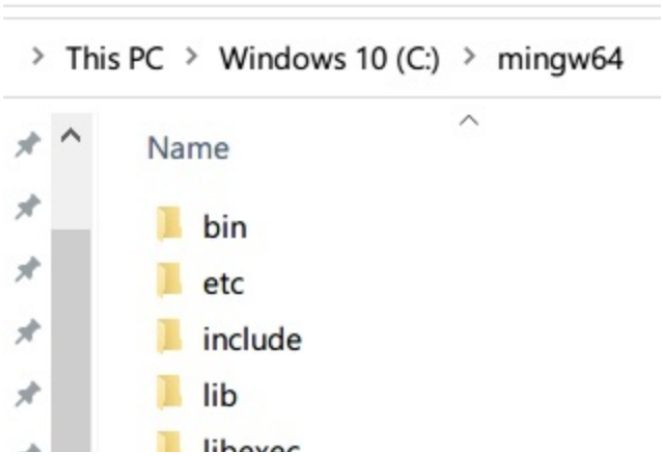
1. 安装visual studio code

不建议安装中文版，或汉化扩展包，后期对词定位费劲。

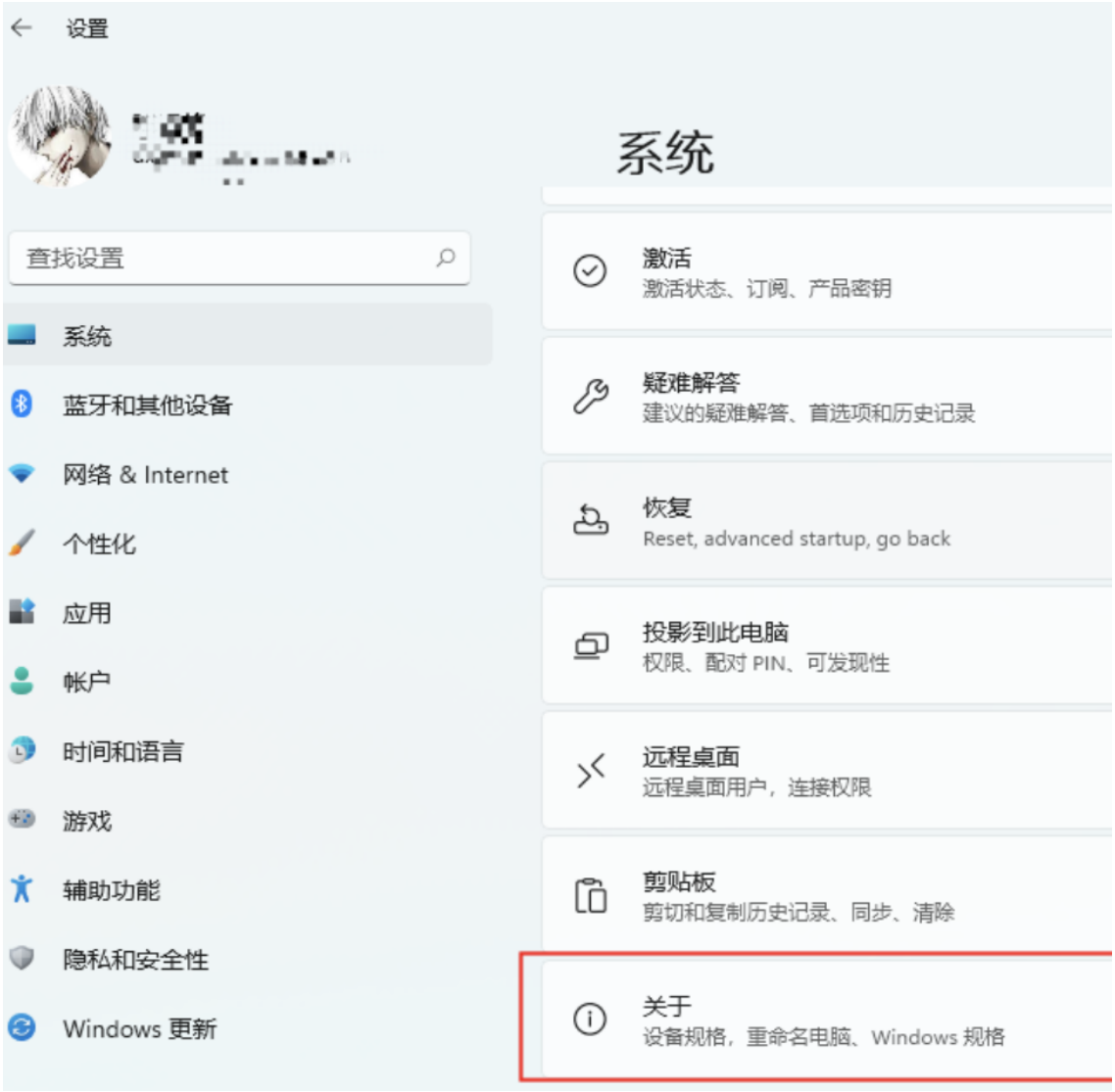
2. 安装mingw for windows

如果是mac系统，请跳到步骤3.

1. 解压x86_64-posix-seh.7z到指定目录，使得解压后的bin目录是： C:\mingw64\bin 或者 D:\mingw64\bin。

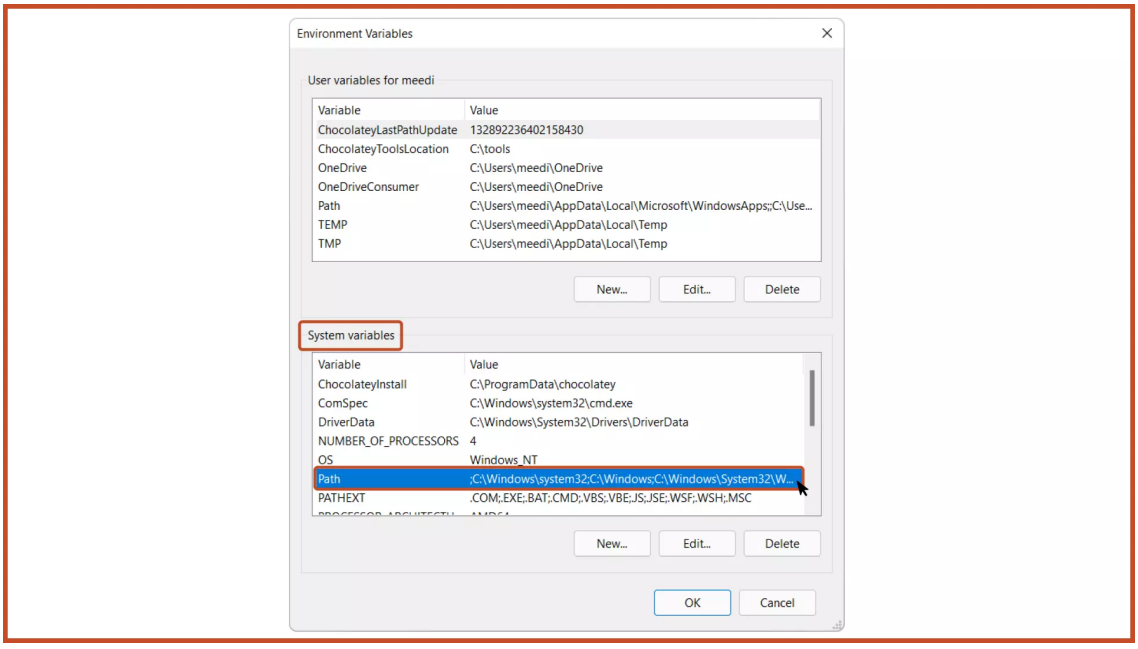


2. 把C:\mingw64\bin添加到 **环境变量path** 中，保证gcc.exe等可执行程序可以被其他软件找到
   a. 请根据你的windows的版本，自行百度搜索。假设你是windows 11，则百度”windows 11 设置环境变量”。下文给出一种方式的关键信息。
   b. 设置→系统→关于->高级系统设置

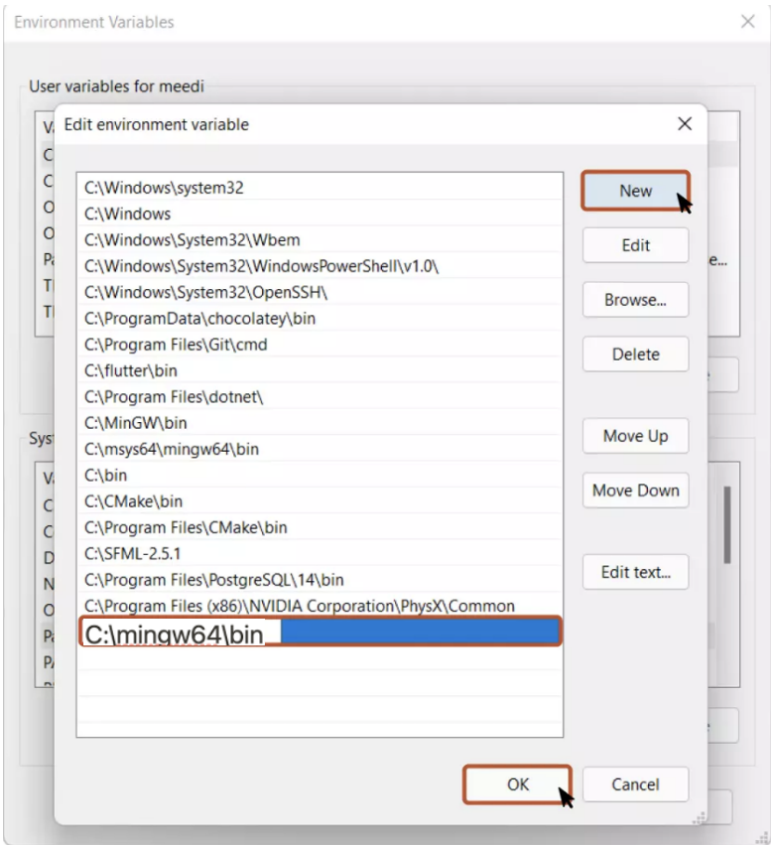

设置
查找设置
系统
蓝牙和其他设备
网络 & Internet
个性化
应用
帐户
时间和语言
游戏
辅助功能
隐私和安全性
Windows 更新
系统
激活
激活状态、订阅、产品密钥
疑难解答
建议的疑难解答、首选项和历史记录
恢复
Reset, advanced startup, go back
投影到此电脑
权限、配对 PIN、可发现性
远程桌面
远程桌面用户，连接权限
剪贴板
剪切和复制历史记录、同步、清除
关于
设备规格，重命名电脑、Windows 规格


查找设置
系统
蓝牙和其他设备
网络 & Internet
个性化
应用
系统 › 关于
设备名称
处理器
Intel(R) Core(TM) i7-10750H CPU @
机带 RAM
16.0 GB (15.8 GB 可用)
设备 ID
产品 ID
系统类型
64 位操作系统, 基于 x64 的处理器
笔和触控
相关链接
域或工作组
系统保护
高级系统设置

c. Double click the “Path” variable in the System Variables section.



d. Click the “New” button, enter the file path: C:\mingw64\bin, and click the “OK” button.

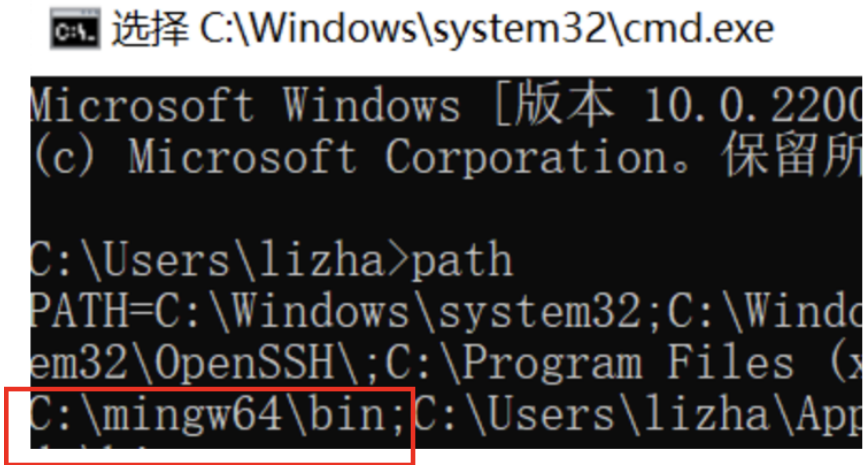


e. 一路ok，click下去，直到下面这个窗口中的“确定” click完毕。

f. 新打开一个cmd窗口，检查Path是否修改成功；已经打开的cmd窗口或者其他软件要重启后，才能用反映上述设置。（Restart your software, such as a new cmd window, or vs code, to make changes take effect.）

   i. window+R进入cmd

   ii. 在该窗口输入path，回车换行，查看当前path是否被成功修改



输出的PATH=….中，如果包含C:\mingw64\bin（或者你的mingw64\bin所在目录），则成功；反之失败，这说明你在执行上述b—e的操作时有误，需要重复这几个操作。

3. 验证mingw64安装成功，即C:\mingw64\bin目录下存在gcc.exe, g++.exe, gdb.exe
    a. 新建一个cmd窗口，在该窗口中运行如下两个命令，没有出现““xxx不是内部或外部命令”，则表示成功：
        i. `gcc -v`
        ii. `gdb -v`

3. 安装clang for mac

如果是windows系统，请跳到步骤4.

打开一个terminal（终端），如iTerm等，执行

```
xcode-select --install
```

4. 安装C++扩展(extension)

系统会推荐很多extension让你安装，一个都不要！！

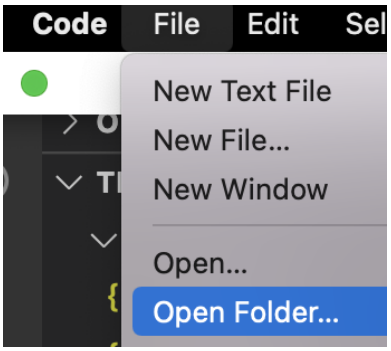
1. 必装：C/C++：又名 cpptools，提供Debug和Format功能
2. 选装：Code Runner：右键即可编译运行单文件，很方便；但无法Debug。

5. 测试c++环境是否安装成功

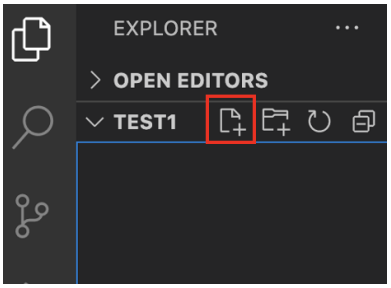
参考：**Debug a C++ project in VS Code，略有调整。不能翻墙的，看我发的视频。**

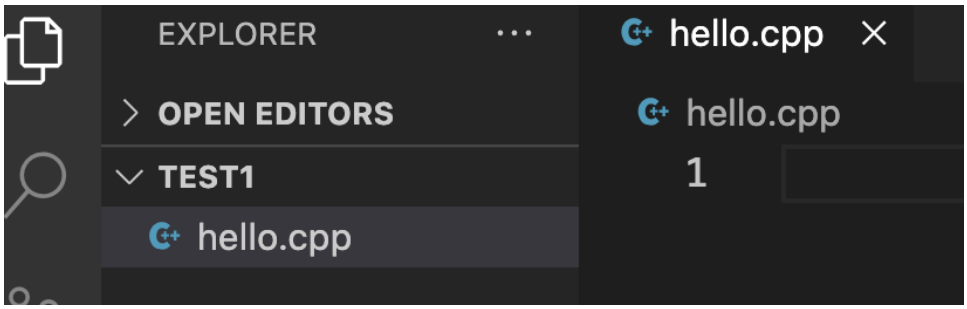
这个视频对应的网页介绍是：https://code.visualstudio.com/docs/cpp/config-mingw#_run-helloworldcpp。 它的helloworld.cpp调试起来更有意思。说明的也很好，无比对照着看一下。本文侧重于根据实际情况，介绍tasks.json和launch.json，本文的hello.cpp过于简单，只是为了测试环境搭建成功。

1. 新建一个目录（文件夹），如d:\test1，请自行百度“**在D盘根目录下新建一文件夹操作的过程”**
2. 在vscode中，打开上述目录



3. 在Explorer中可以看到，给目录下没有任何文件；接下来为这个目录增加一个hello.cpp:





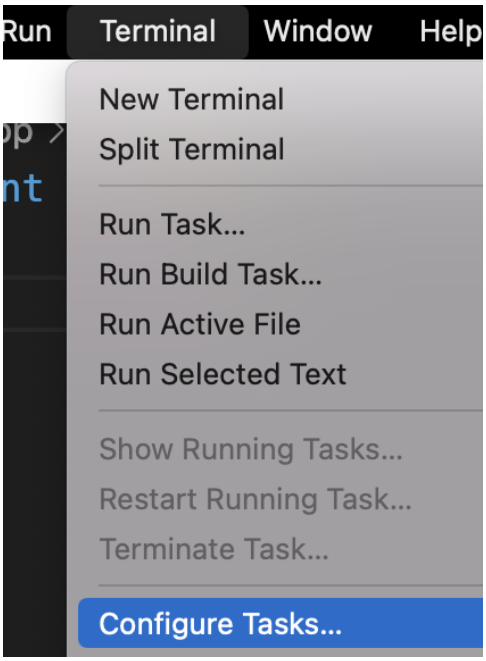
4. 为该cpp添加代码如下：

```
int main() {
return 0;
}
```

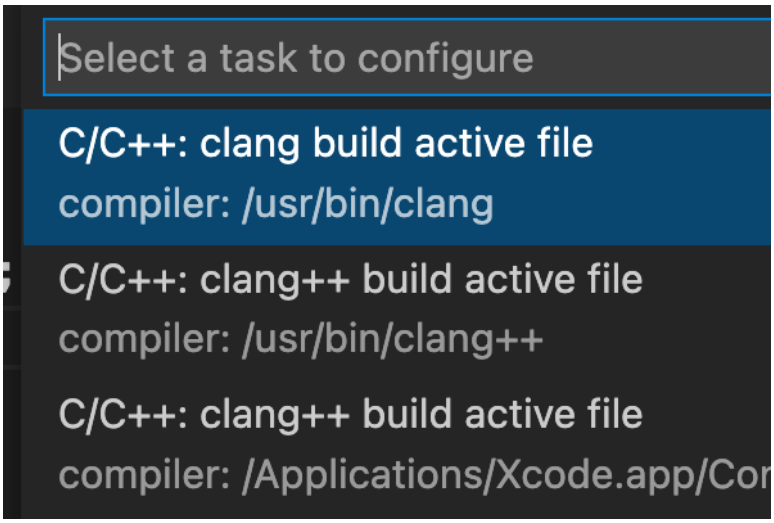
    1. 别忘了，按ctrl+s，保存你的修改。

5. 建立一个`tasks.json` (build instructions)，负责把cpp编译成exe

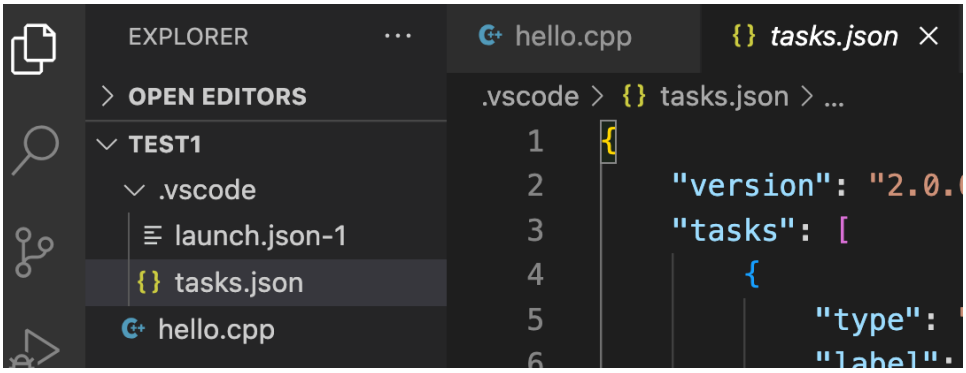
    a. 在hello.cpp被激活（选中）的状态下，Terminal/Configure Tasks



    b. mac选择“C/C++: clang++ build active file”; windows选择“C/C++: g++.exe build active file”



c. 生成的tasks.json位于test1/.vscode目录下：



d. mac的tasks.json内容如下：

```
{
"version": "2.0.0",
"tasks": [
{
"type": "cppbuild",
"label": "C/C++: clang build active file",
"command": "/usr/bin/clang",
"args": [
"-fdiagnostics-color=always",
"-g",
"${file}",
"-o",
"${fileDirname}/${fileBasenameNoExtension}"
],
"options": {
"cwd": "${fileDirname}"
```

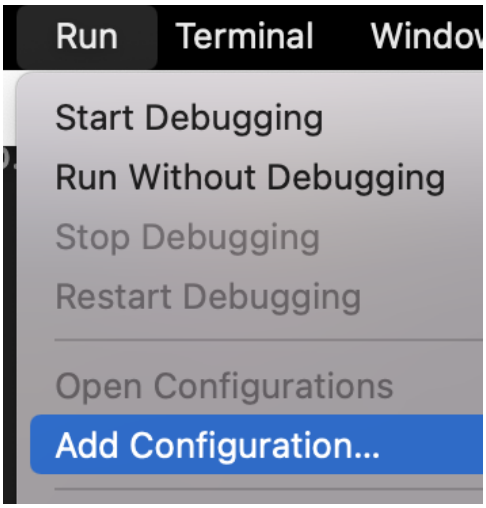
```
},
"problemMatcher": [
"$gcc"
],
"group": "build",
"detail": "compiler: /usr/bin/clang"
}
]
}
```

e. windows的tasks.json，内容如下：

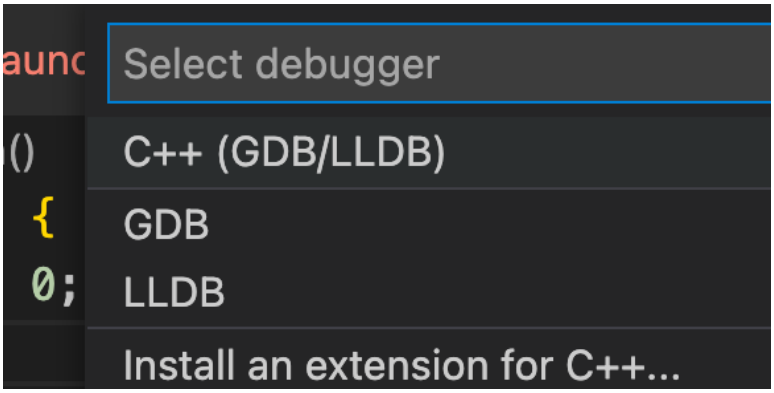
```
{
  "tasks": [
    {
      "type": "cppbuild",
      "label": "C/C++: g++.exe build active file",
      "command": "C:\\msys64\\mingw64\\bin\\g++.exe",
      "args": [
        "-fdiagnostics-color=always",
        "-g",
        "${file}",
        "-o",
        "${fileDirname}\\${fileBasenameNoExtension}.exe"
      ],
      "options": {
        "cwd": "${fileDirname}"
      },
      "problemMatcher": ["$gcc"],
      "group": {
        "kind": "build",
        "isDefault": true
      },
      "detail": "Task generated by Debugger."
    }
  ],
  "version": "2.0.0"
}
```

6. 建立一个launch`.json` (debugger settings)，负责通过运行生成的exe，调试你写的cpp。

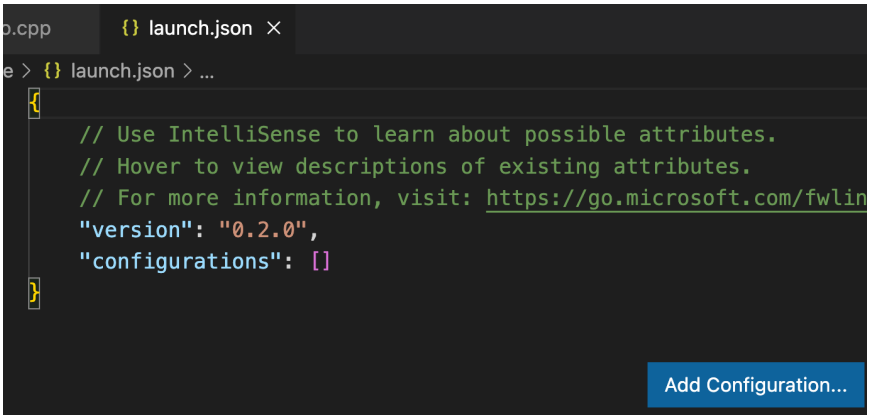
   a. 选中hello.cpp，然后Run/Add configuration



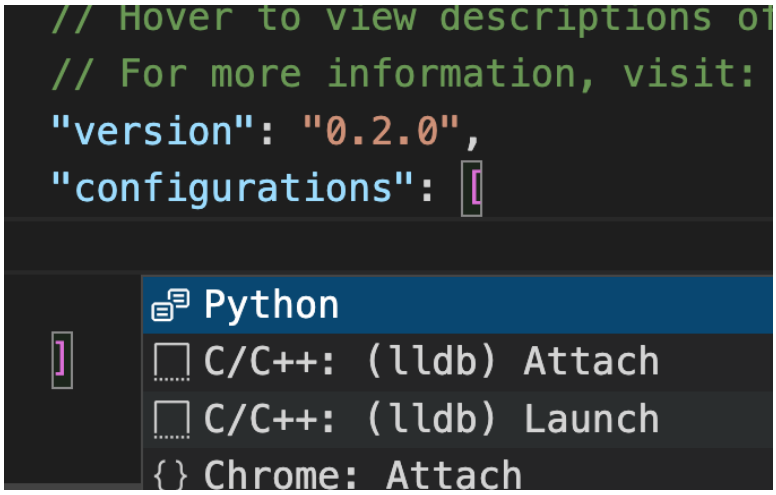
b. mac选择 “C++ (GDB/LLDBB”)，windows选择“**C/C++: g++ build and debug active file**”



c. 生成launch.json如果如下：



d. 则还需要单击 右下角蓝色按钮“add configuration”，然后mac选择“C/C++: (lldb) Launch”，windows选择“**C/C++: g++ build and debug active file”**



e. 注意以下几行内容，如果不一样或者没有，请按照下文修改：

   i. 修改program行，成为"program": "${fileDirname}/bin/${fileBasenameNoExtension}",

   ii. mac增加一行"preLaunchTask": "C/C++: clang++ build active file"；windows则增加"preLaunchTask": "C/C++: g++.exe build active file"。preLaunchTask后面的两个，分别是mac和windows下的tasks.json文件的label，请看上文。

   iii. windows还需要修改"miDebuggerPath": "gdb.exe",

   iv. 修改后mac的如下：

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "(lldb) Launch",
            "type": "cppdbg",
            "request": "launch",
            "program": "${fileDirname}/bin/${fileBasenameNoExtension}",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${fileDirname}",
            "environment": [],
            "externalConsole": false,
```

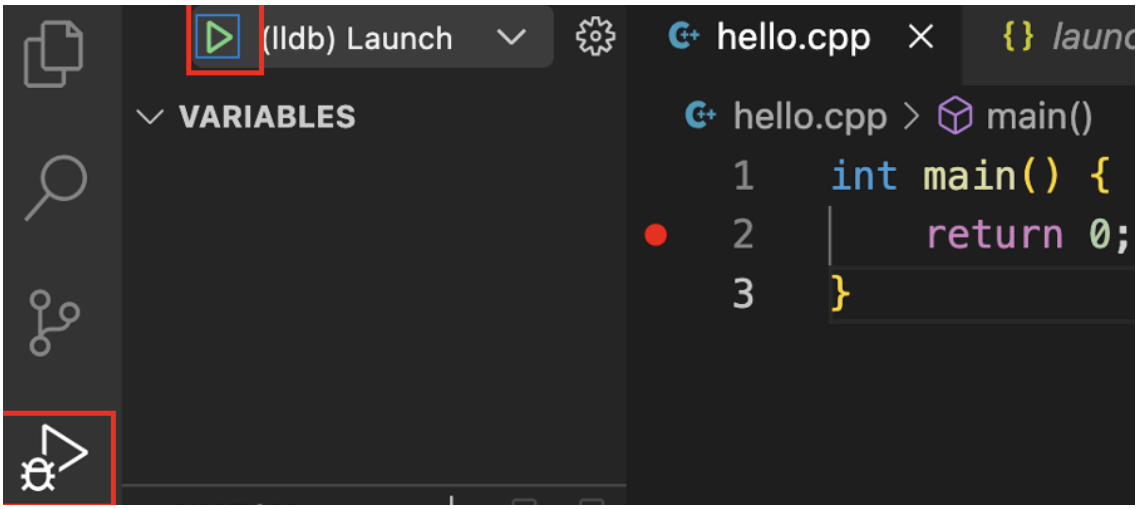
```
            "MIMode": "lldb",
            "preLaunchTask": "C/C++: clang++ build active file"
        }

    ]
}
```

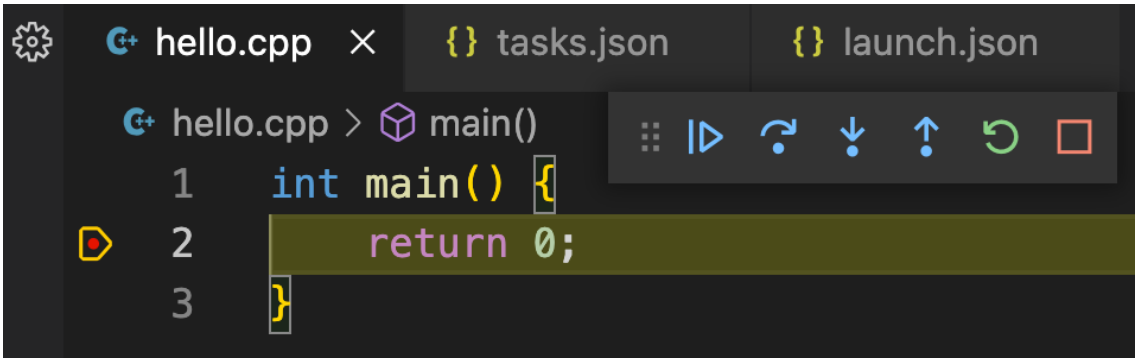
f. 修改后，windows的如下

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "C/C++: g++.exe build and debug active file",
      "type": "cppdbg",
      "request": "launch",
      "program": "${fileDirname}\\${fileBasenameNoExtension}.exe",
      "args": [],
      "stopAtEntry": false,
      "cwd": "${fileDirname}",
      "environment": [],
      "externalConsole": false,
      "MIMode": "gdb",
      "miDebuggerPath": "C:\\msys64\\mingw64\\bin\\gdb.exe",
      "setupCommands": [
        {
          "description": "Enable pretty-printing for gdb",
          "text": "-enable-pretty-printing",
          "ignoreFailures": true
        }
      ],
      "preLaunchTask": "C/C++: g++.exe build active file"
    }
  ]
}
```

g. 选中hello.cpp设置断点，调试代码

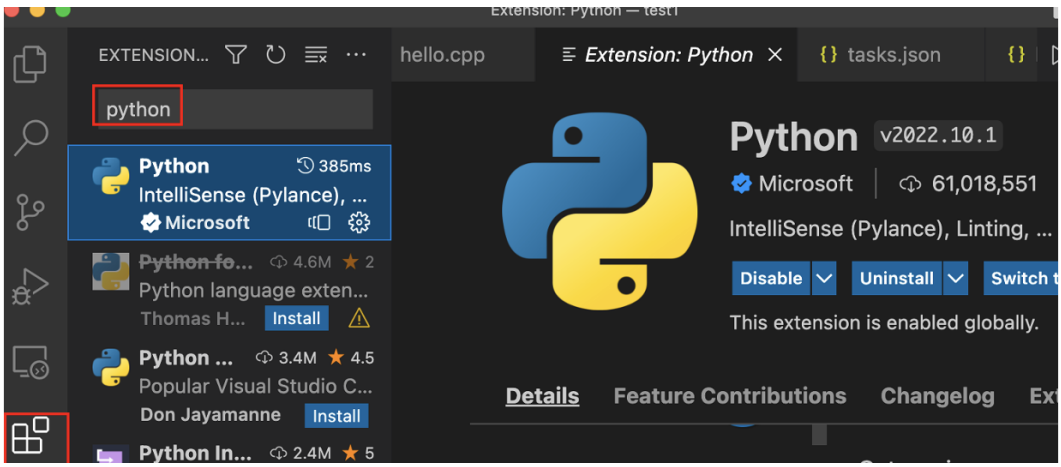


如果在红色断点那一行，程序暂停，出现如下调试工具栏，则成功。



6. 安装Python扩展(extension)

系统会推荐很多extension让你安装，一个都不要！！

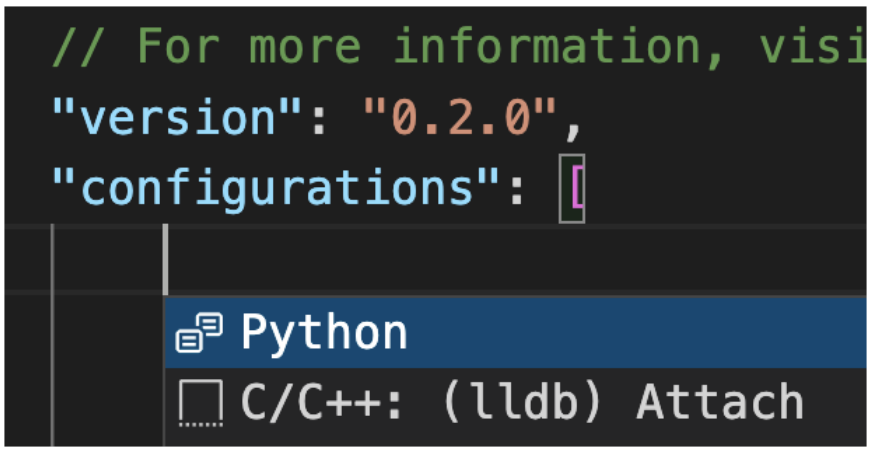


7. 测试Python环境是否安装成功

参考：https://code.visualstudio.com/docs/python/python-tutorial

过程和测试c++环境类似。通过理解测试c++环境，你应该具备看懂这个英文教程的能力。

但是，可以不新建目录，就在test1中增加hello.py文件，打开launch.json，”add configuration”，选择Python。

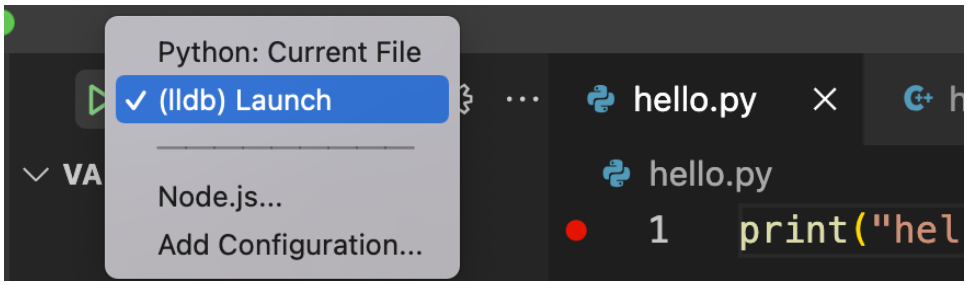


然后你的launch.json中就有了2个configuration，mac的如下：

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Python: Current File",
            "type": "python",
            "request": "launch",
            "program": "${file}",
            "console": "integratedTerminal",
            "justMyCode": true
        },
        {
            "name": "(lldb) Launch",
            "type": "cppdbg",
            "request": "launch",
            "program": "${fileDirname}/${fileBasenameNoExtension}",
            "args": [],
```

```
            "stopAtEntry": false,
            "cwd": "${fileDirname}",
            "environment": [],
            "externalConsole": false,
            "MIMode": "lldb",
            "preLaunchTask": "C/C++: clang build active file"
        }
    ]
}
```

调试python和c++的时候，可以动态选择对应的configuration



8. 为什么Python不用修改tasks.json？你可以回答不？

名词解释

1. 环境变量的作用
    a. 当你运行某些程序时，除了在当前文件夹中寻找外，还会到这些环境变量中去查找，比如“Path”就是一个变量，里面存储了一些常用的命令所存放的目录路径。
    b. 当启动cmd命令行窗口调用某一命令的时候，经常会出现“xxx不是内部或外部命令，也不是可运行的程序或批处理文件”如果你的拼写没有错误，同时计算机中确实存在这个程序，那么出现这个提示就意味着它不在默认路径里，或你有没有给出命令程序的路径（命令程序前面未带路径，注意有些命令程序可能不支持前面带路径）。

参考

Using GCC with MinGW

https://www.zhihu.com/question/30315894

https://blog.csdn.net/weixin_43492780/article/details/119876493 **保姆级教程**