

【C/C++ 新手大礼包 之二】

龙爸写给初学 C/C++ 的朋友们的**新手入门指南**，扫平入门的几个大障碍的**极简教程**：

1. **安装 C/C++ 编译环境**： [《C/C++ 运行环境安装配置指南——初学者极简版》](#) | [PDF版下载](#)
2. **用 VSCode 写你的第一个 Hello World**： [极简安装、配置中文语言包、C/C++ 扩展](#) | [PDF版下载](#)
3. **算法入门练习题**： [陪小学生学习 C/C++ 练习题](#)（陆续完善中）
4. **方便设置 Path 环境变量的脚本**： 在 Path 环境变量中查看、查找、添加、删除——[使用说明](#) | [PDF版下载](#)

原文发布、维护于龙爸[陪孩子学习 Python](#)、[C++](#) 的练习项目：

- [coffeescholar/C_CPP-Learning](#)，作者：[爱学习的龙爸](#)
- 欢迎意见、建议和帮助，谢谢 🤗❤❤
- 如果对你有帮助，请支持 Star~ ✨✨✨
- 转载时请保留声明和出处，谢谢 🙏🙏🙏

用 VSCode 写你的第一个 Hello World：极简安装、配置中文语言包、C/C++ 扩展

本教程不同于其它教程，相对更加简单，也不容易出现意外。

但因为面向小学生、初中生小朋友，所以详细步骤会比较详细，超龄的小朋友请根据自身情况适当快进。

现有的部分教程写的比较复杂，尽量选择较新的学习，因为版本更新后一些配置可能变化了或者不需要了。

很多教程的方法比较复杂的原因：

下载安装 MinGW64 或 Cygwin 或 msys2，下载安装、设置系统环境变量时，经常遇到问题，安装过程也略微复杂，还要设置系统环境变量。

本教程简化之处：

- 用 Scoop 自动下载、安装 GNU 的 gcc/g++/gdb，不需要通过类似上面的三个应用或者其它应用；
- 用 Scoop 自动设置系统环境变量，省去了复杂的设置操作；
- 用 Scoop 随时更新到最新版本。

1. 背景简介

编写代码，可以用记事本等文本编辑软件——大牛，更常见是使用专门为编程设计的代码编辑软件——普通人。

代码编辑软件有很多，其中功能相对比较全面、强大的，带有编译、调试、源代码管理甚至项目管理和团队协作等等的，称为集成开发环境（IDE，Integrated Development Environment）。

集成开发环境 IDE 也有很多，其中比较出名和被广泛使用的主要有开源、免费的微软公司 VSCode 和 JetBrains 商业授权的 CLion 等等。

微软公司有付费购买商业授权的 IDE——被软件开发行业尊称戏称为宇宙第一 IDE 的 Visual Studio 系列，非常强大，伴随了本教程作者的几乎整个软件生涯。

这里的 VSCode 全称为 Visual Studio Code，正是微软公司后来推出的免费开源 IDE——但两者是不同的软件。开源社区基于免费开源的 VSCode 还延申出了不同的开源版本，以后可以再深入了解。

另外一家专门提供软件开发工具的著名公司 JetBrains：

JetBrains 公司有很多非常棒的软件开发工具，比如各种编程语言的 IDE，如 Java 开发人员最熟悉的 IntelliJ IDEA，等等。该公司的大部分软件需要付费购买商业授权才能使用，例如学习 Python 也可以用 PyCharm，非常不错。在校大学生可以通过向 JetBrains 公司申请教育授权来免费使用，其它人则可以通过创建和维护开源项目、公益项目来申请免费试用。

借此文凭吊一下曾经非常著名的提供软件开发工具的公司 `Borland`：

`Turbo Pascal`、`Turbo C`、`Borland C++`、`Delphi`，

`Visual J++`、`C#/.NET`、`Typescript`（不好意思，串台了）。

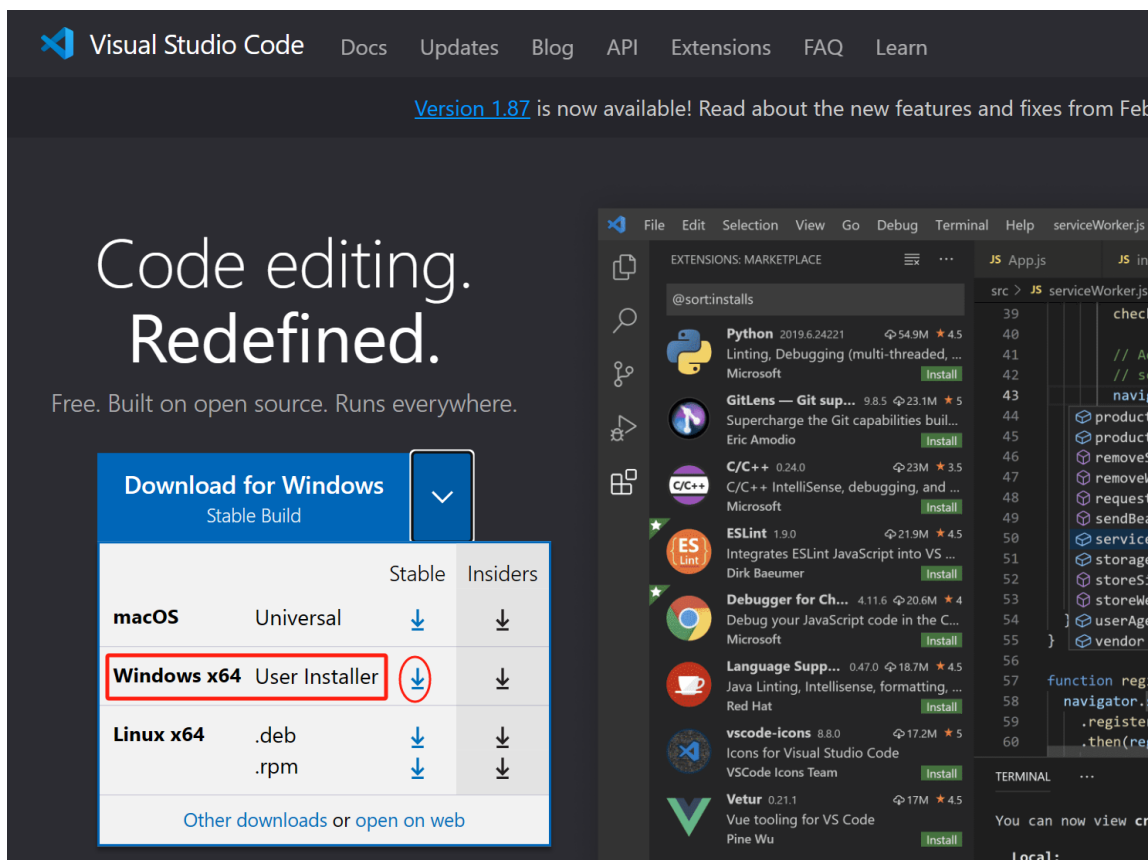
对于初学者，推荐用 `微软公司` 的 `VSCode`（全称：`Visual Studio Code`）。

2. 下载安装 VSCode

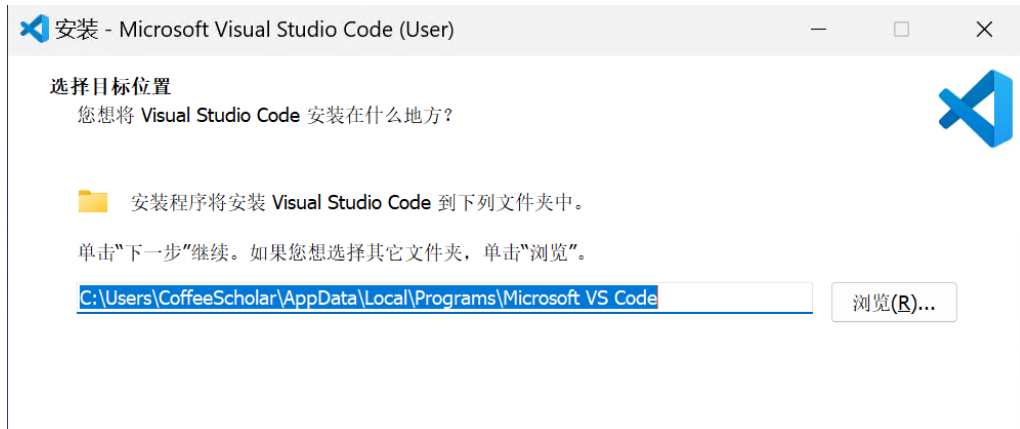
VSCode 官网：<https://code.visualstudio.com/>

不记得没关系，浏览器输入 `VSCode` 搜索，地址与上面相同那一条就是官网了。——下载速度很快，不必从其它网站下载二手的。

1. 下载后安装，默认选择为当前用户安装，



建议不要安装到默认的 `c` 盘，这里为了演示简化：——路径中不要出现中文，避免以后可能遇到的一些莫名其妙的麻烦，不必要（`GNU C/C++` 编译和调试器等等，对 `Windows` 和中文环境的兼容性虽然已经很好了，但情况很复杂）

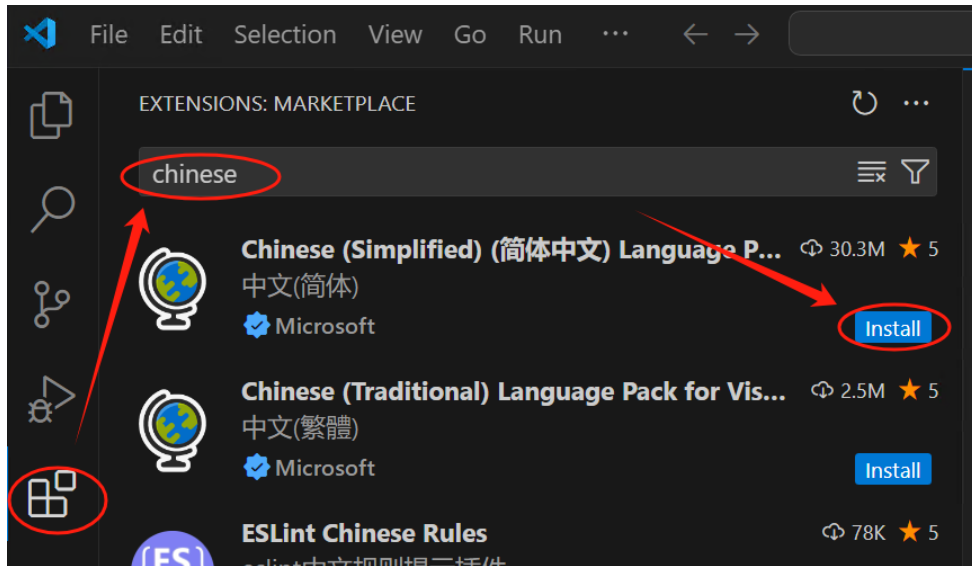


选中这两项方便一点：

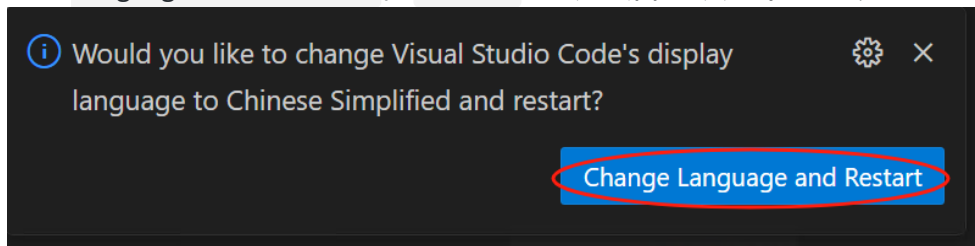


2. 安装中文语言包：

- 安装完成后，运行 VSCode，按下快捷键：Ctrl-Shift-X 打开 Extensions（扩展）——或者点左边工具栏积木方块样子的图标；
- 在顶部的搜索栏输入：Chinese 并稍等符合条件的扩展包列表加载，
- 选择下面的 Chinese (Simplified)（简体中文）Language Pack for Visual Studio Code，
- 选中并在右侧点击 Install，安装中文语言包扩展：



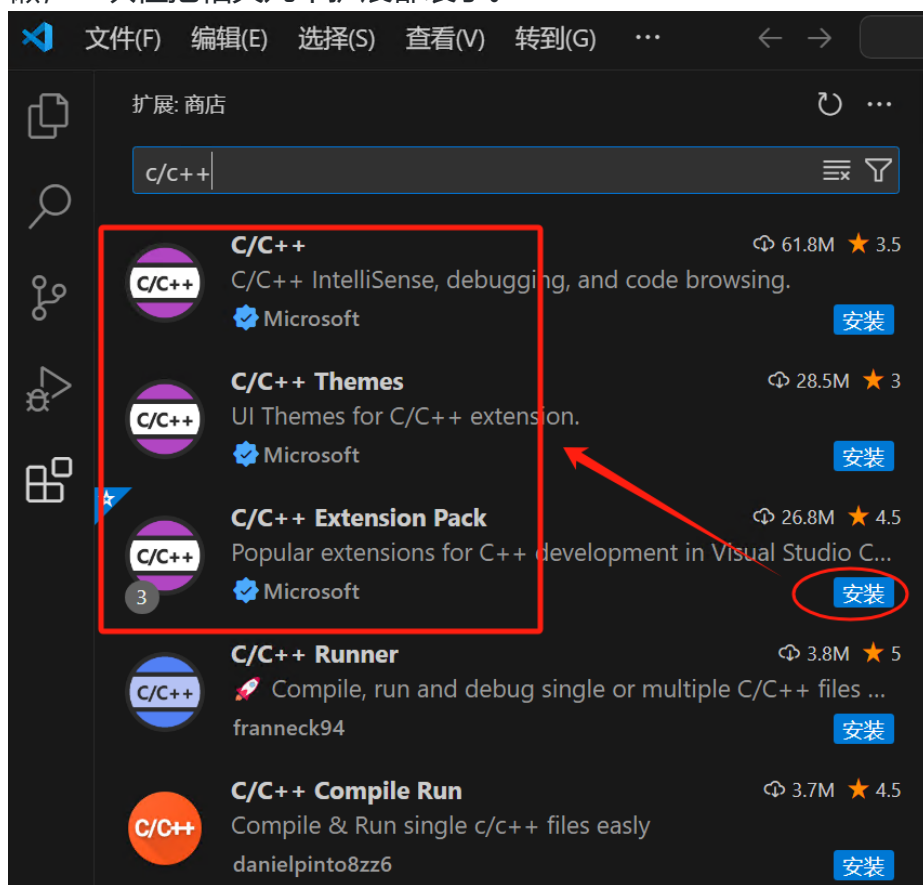
- 安装完成后，VSCode 右下角会弹出消息提示 更改语言并重新启动 ， 点击 Change Language and Restart ， VSCode 自动重启后就是中文环境了。



3. (此处跳过，后面自动安装) 手动安装、配置 C/C++ 扩展：

- VSCode 重启后，按下快捷键：Ctrl-Shift-X 打开 Extensions (扩展)，在顶部的搜索栏输入：C/C++，选择 C/C++ Extension Pack 扩展包，并在右侧点 安装 ——偷

懒，一次性把相关几个扩展都装了。



- 以后可以装 `C/C++ Runner`，或者 `C/C++ Compile Run` 扩展之一，——它们会自动找到之前安装的 `gcc/g++/gdb` 并自动配置，可以 一键编译、调试和运行 C/C++ 代码，省去了很多麻烦，**甚至可以省去后面的步骤**，——但，**暂时不要安装这些扩展**，先把基础配置做好，**免得以后遇到问题不知道如何处理**。类似的扩展还有：`Code Runner` ——不仅可以用于 C/C++，还可用于运行 `Java`、`C#`、`Python` 等等常用开发语言的代码。

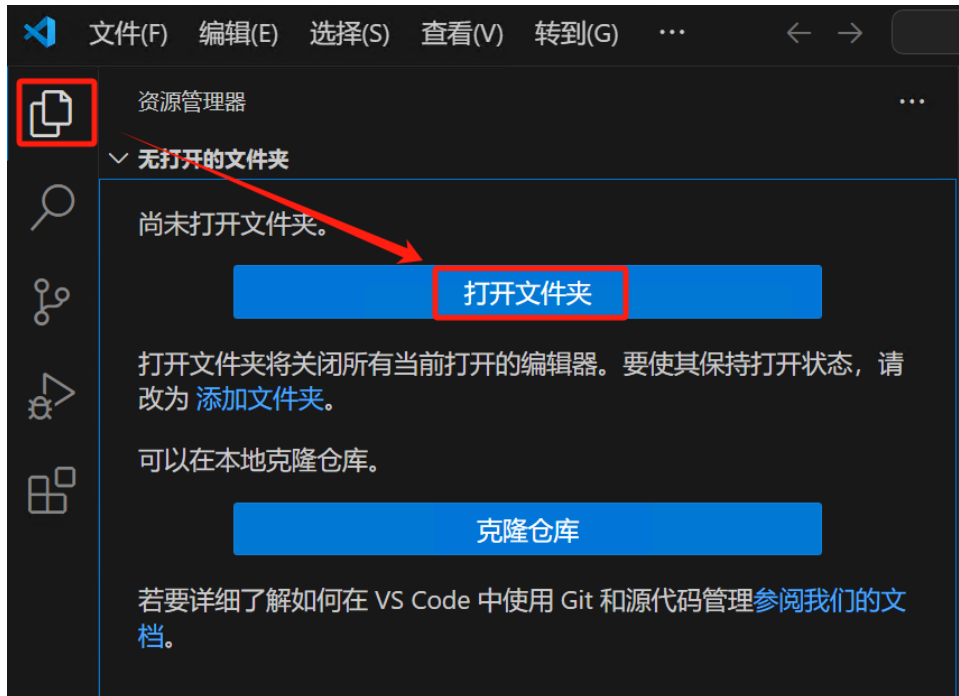
有更多好用的扩展，以后慢慢熟悉、筛选、磨合，选择多了有时候也挺麻烦。

3. 编写、运行你的第一段 `C/C++` 代码

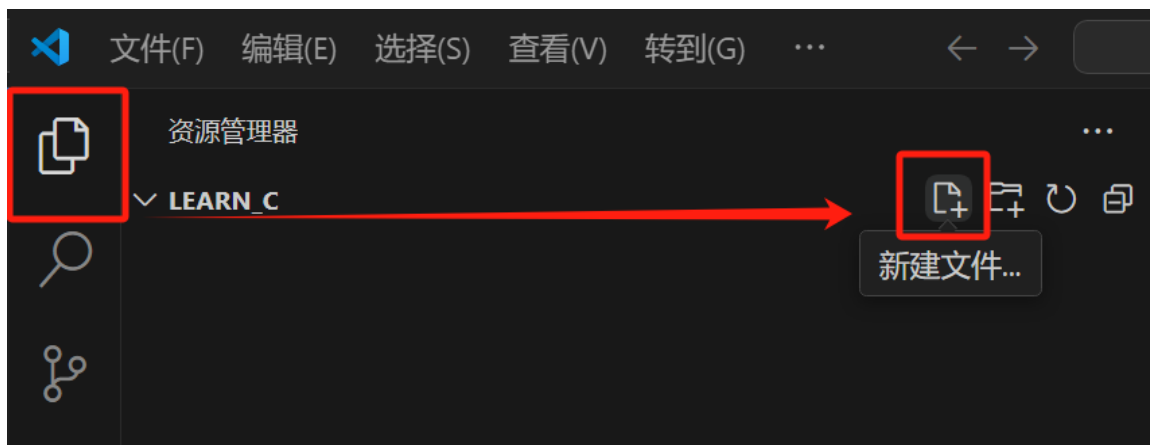
接下来，可以编写、运行你的第一段 `C/C++` 代码了：

- 1. 打开文件夹：** 在 `VSCode` 中选择一个文件夹来保存你的代码，例如：`D:\MyCode\Learn_C`；

注意：**路径和文件名中不要带空格、中文**，避免后续编译、调试时无法预料的错误。目前发现，主要是调试时对**中文路径和中文文件名**支持仍有问题（不是 `gdb` 的问题，初步测试判断是 `VSCode` 终端环境有差异，待深入排查）。

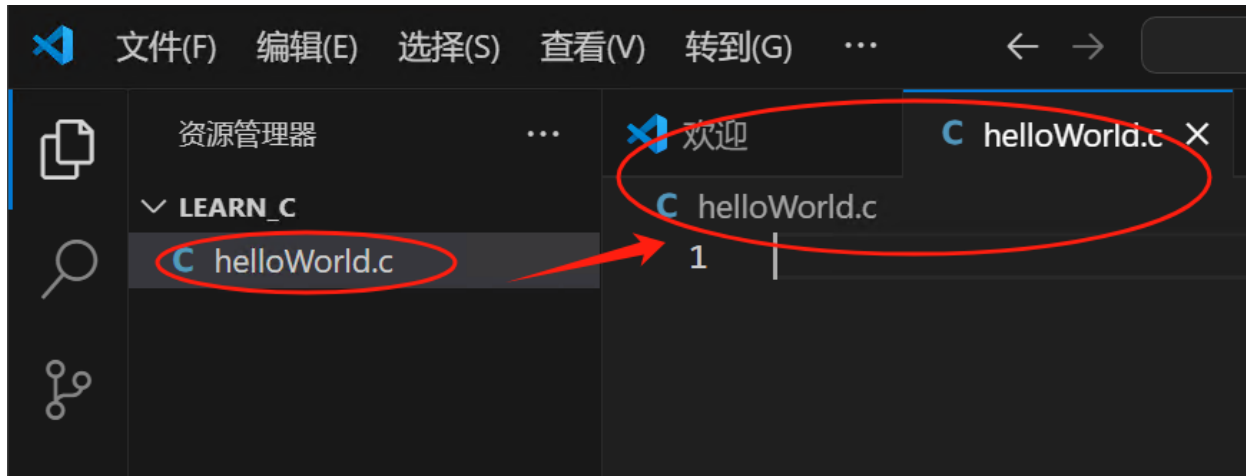


2. **新建文件**：在 VSCode 左侧的 资源管理器 中，点右上角的 新建文件 ，

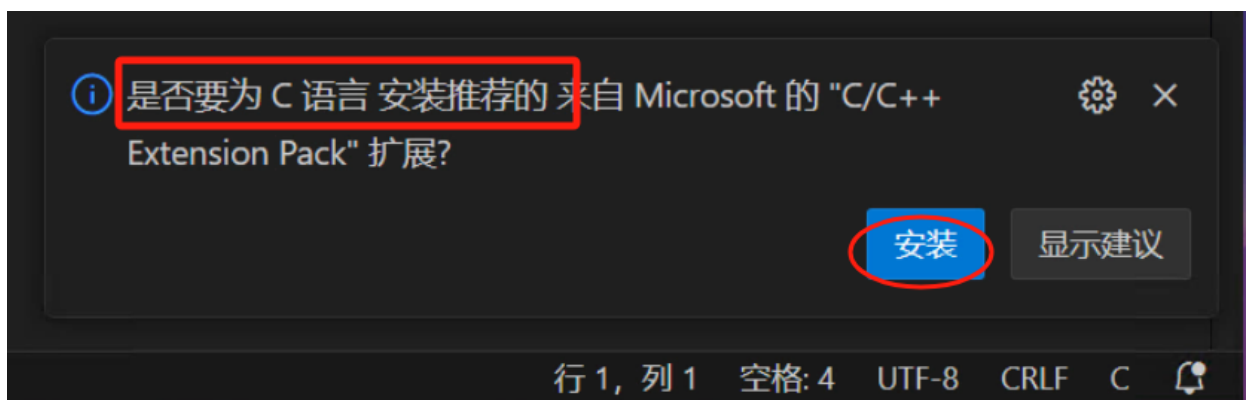


并为创建的文件取个名字比如 `HelloWorld.c`（英文），不需要内容

——注意扩展名用 `.c` 或 `.cpp`，VSCode 会自动识别并启用对应语言的扩展：



稍等，右下角提示安装推荐的扩展包：



——接受，等扩展安装完毕，然后【重新运行 VSCode】——关闭，再运行。

3. **编写代码**：重新运行后，在右侧编辑区域编写代码并保存（按 `Ctrl + S`），

粘贴下方 程序员打开数字世界的第一个魔法口令：

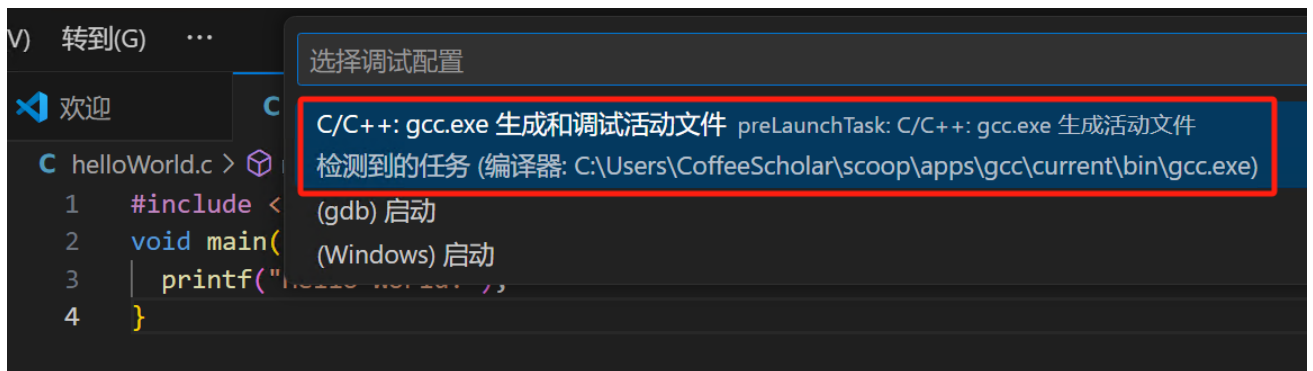
```
#include <stdio.h>
void main() {
    printf("Hello World!");
}
```

4. **运行代码**：点击编辑区域右上角类似 `播放` 按钮，运行和调试：

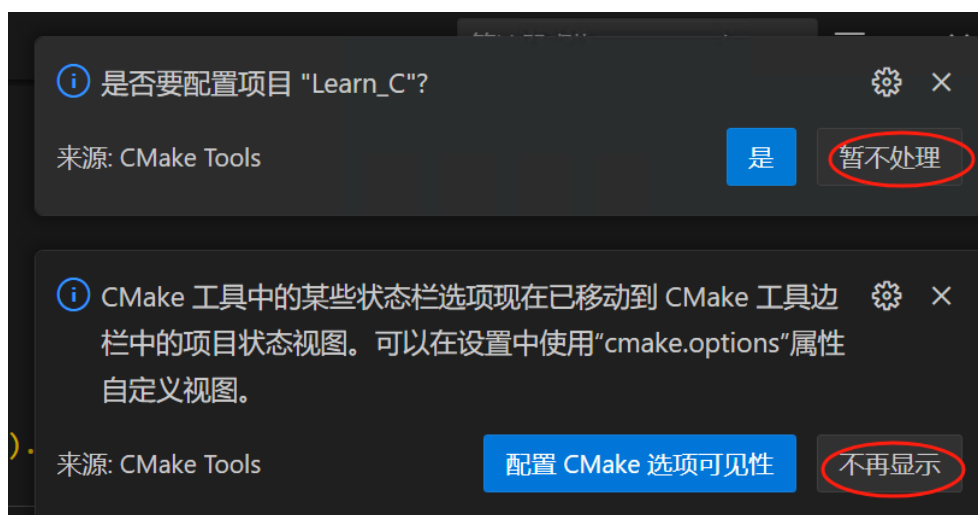


5. 自动配置：VSCode 会自动提示配置编译器，选中默认项：——VSCode 已经识别到之前安装并在 Path 环境变量配置过的 gcc.exe：

——这里发现之前自动装好扩展必须重启一下 VSCode，否则 VSCode 经常无法自动识别 gcc



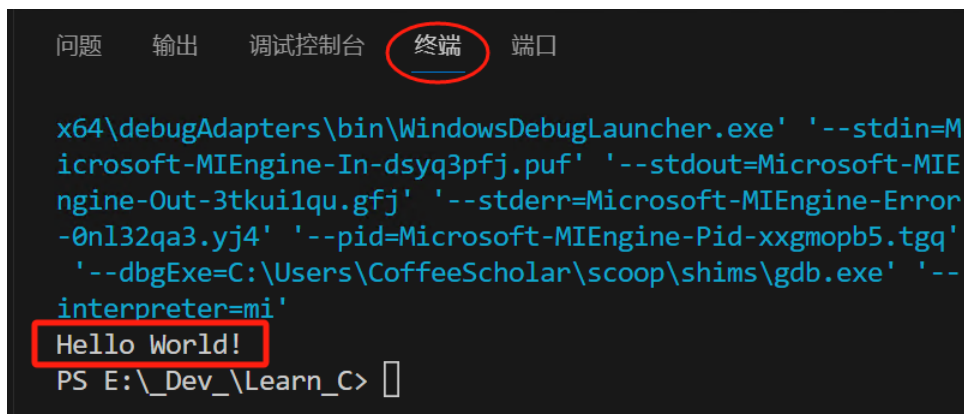
- 关闭右下角的提示（来源为 CMake Tools 的提示——短期内用不上 CMake，有个印象就好）：



6. 自动产生任务配置文件：左侧 资源管理器 看到多了一个文件夹、两个文件：

- `.vscode` : `VSCode` 自动创建的用于存放配置文件的文件夹;
- `tasks.json` : 负责编译代码为可执行的 `exe` 文件的配置; ——正是其它教程需要手工配置的文件之一, 这里**自动生成了, 以后再深入研究**;
- `helloWorld.exe` : `gcc` 对你的代码进行编译生成的 可执行文件, 调试、运行都离不开它。

7. **查看运行结果**: 在底部输出窗口能看到类似下面的运行结果:

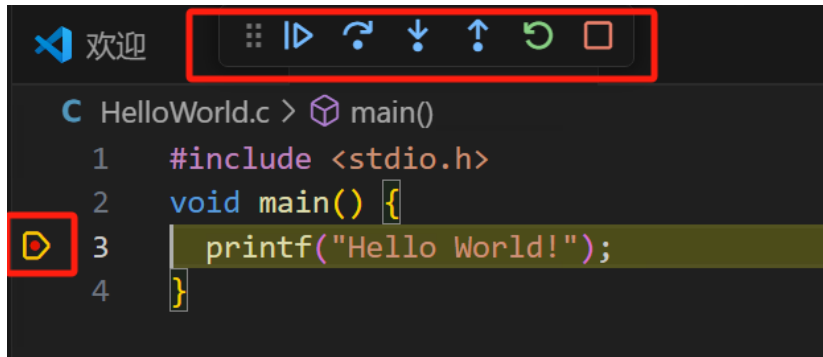


The screenshot shows the VS Code terminal window with the 'Terminal' tab selected. The terminal output displays the command prompt 'PS E:_Dev_\Learn_C>' followed by 'Hello World!'. The 'Hello World!' text is highlighted with a red box. Above the terminal, the 'Terminal' tab is also highlighted with a red circle.

8. **运行和调试**: 下拉菜单中选 `运行`, `VSCode` 会记住上一次的操作, 以后不用每次都按旁边的下拉菜单来选择。——运行和调试有什么区别? 后续调试代码会讲到:

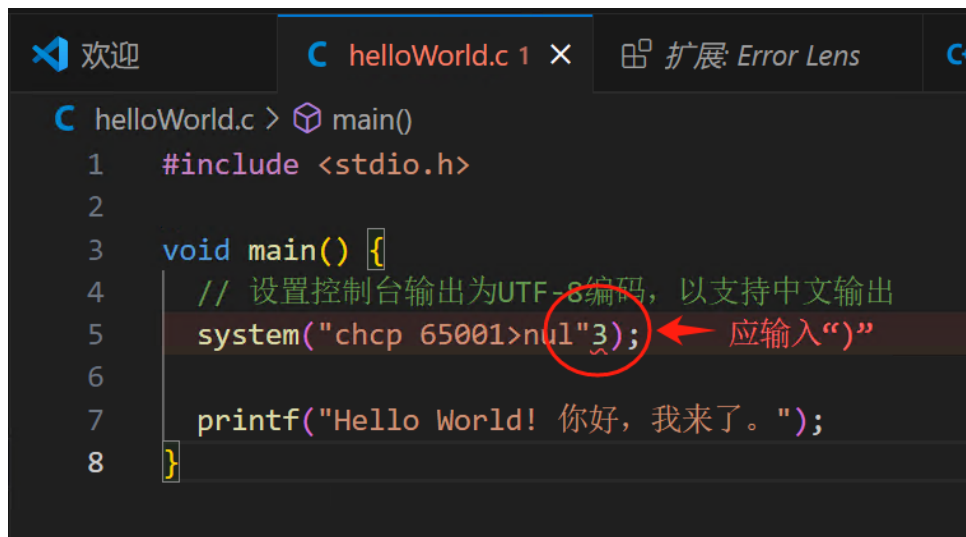


已经了解调试的同学, 可以加断点调试测试一下配置是否正常, 代码会停在加了断点这一行, 并等待处理:



9. **完事，收尾：**这时候装几个常用扩展（按照前面装语言包的方式）：

- **必装扩展：** Error Lens ——在出错的代码旁显示错误信息，方便定位错误位置（默认在终端输出窗口）。下面的例子中故意制造了一个错误：



- **必装扩展：** Error Gutters ——配合上一个扩展，在警告、出错的代码前显示图标：

The screenshot shows the VS Code editor with a file named `helloWorld.c`. The code is as follows:

```

1  #include <stdio.h>
2
3  void main() {
4      // 设置控制台输出为UTF-8编码，以支持中文输出
5      system("chcp 65001>nul"); 应输入")
6
7      if( 应输入表达式
8      {
9          if( 应输入表达式
10         {
11             if( 应输入表达式
12             {
13                 printf( 函数调用中的参数太少
14             } 应输入";"
15         }
16     }
17     printf("Hello World! 你好，我来了。");
18 }

```

The Output window shows the following error messages:

- expected ')' before numeric constant gcc [行 5, 列 26]
- implicit declaration of function 'system' [-Wimplicit-function-declaration] gcc [行 5, 列 3]
- 应输入")" C/C++(18) [行 5, 列 26]
- 应输入表达式 C/C++(29) [行 7, 列 6]
- 应输入表达式 C/C++(29) [行 9, 列 8]
- 应输入表达式 C/C++(29) [行 11, 列 10]
- 函数调用中的参数太少 C/C++(165) [行 13, 列 16]
- 应输入";" C/C++(65) [行 14, 列 7]

○ 推荐扩展：C/C++ Compile Run

——编译、调试和运行 C/C++ 代码的帮助扩展，**自动配置**已经安装的 `gcc/g++/gdb`

——快捷键：F5 编译、调试运行，F6 编译、运行

——其实有它，入门学习基本够用了，可以跳过以上所有步骤。

○ 推荐扩展：Code Runner（注意，不是另外一个 C/C++ Runner）——同上，二选一即可

——快捷键：Ctrl + Alt + N

这个扩展支持更多的语言，如 `Java`、`C#`、`Python` 等等，因为同时在学习 `Python` 所以装了这个扩展：



- 可选扩展：Indent-Rainbow ——用彩虹色显示缩进，方便对齐——在 Python 中非常有用； ——类似的，还有彩虹括号、彩虹花括号，已经内置了：



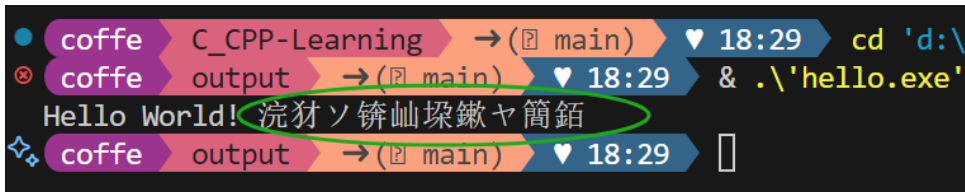
- 更多更多，层出不穷，眼花缭乱，还是缓一缓，专注于【先入门】吧。

4. 常见问题：中文输出乱码

假如给前面的代码的输出内容加上中文：

```
#include <stdio.h>
void main() {
    printf("Hello World! 你好，我来了。");
}
```

运行，查看下方输出内容中出现看不懂的乱字符，俗称乱码，其实是终端环境没能正常识别字符集的问题：



解决方法有两种：


- 改变 VSCode 的终端相关设置——在其它电脑上运行需要重新配置；
- 代码中进行处理——与环境设置无关。

前一种方法改变了终端运行环境设置，还可能会干扰其它功能，所以推荐后一种方法：

```
#include <stdio.h>
#include <stdlib.h>

void main() {
    // 设置控制台输出为UTF-8编码，以支持中文输出
    system("chcp 65001>nul");

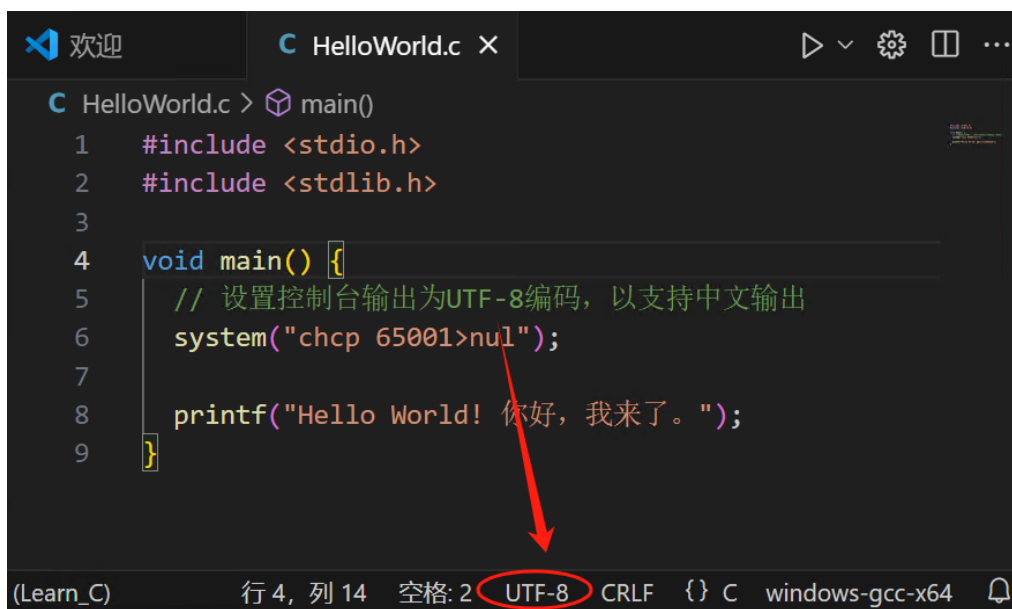
    printf("Hello World! 你好，我来了。");
}
```



```
C HelloWorld.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void main() {
5      // 设置控制台输出为UTF-8编码，以支持中文输出
6      system("chcp 65001>nul");
7
8      printf("Hello World! 你好，我来了。");
9  }
```

新增的两行代码中：`chcp 65001` 表示把系统的字符集切换为 `65001` (UTF-8) ,

——与 VSCode 编辑代码时的默认字符集一致：



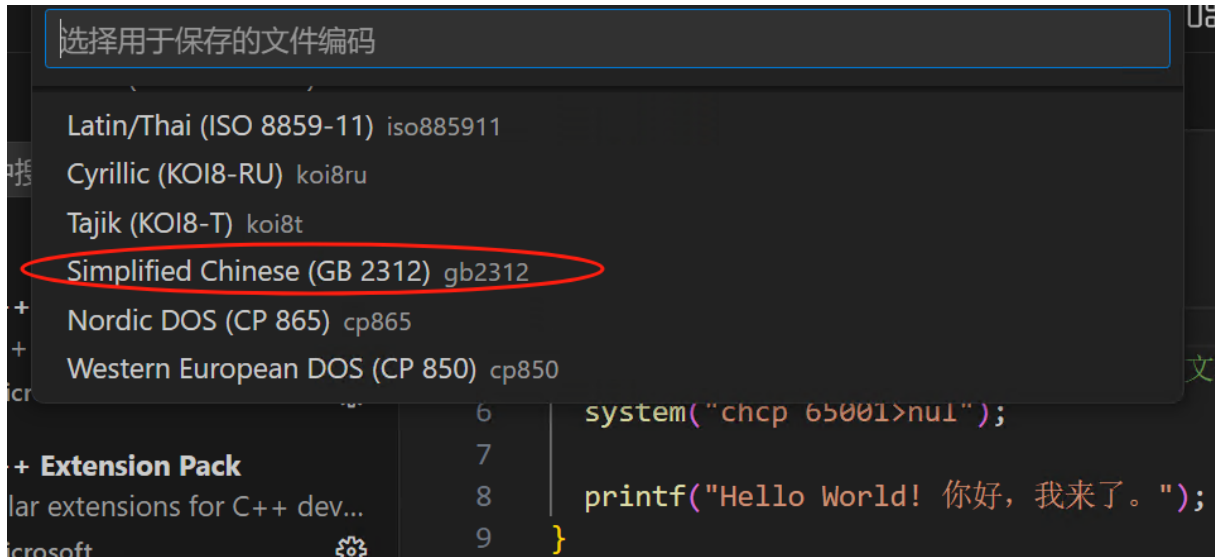
```
C HelloWorld.c > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void main() {
5      // 设置控制台输出为UTF-8编码，以支持中文输出
6      system("chcp 65001>nul");
7
8      printf("Hello World! 你好，我来了。");
9  }
```

(Learn_C) 行 4, 列 14 空格: 2 **UTF-8** CRLF {} C windows-gcc-x64

所以，如果你的编辑环境的编码为 `GB2312` 的话，代码中的 `65001` 改为 `936` 也能正常输出。

——最好保持 `UTF-8` 适用性更广泛。

——以后你会学习到字符集相关的知识，这是很容易出现混乱的若干历史包袱之一，属于 **程序员必经心魔**。



Windows 系统中默认也是 936，很多教程教你去修改 Windows 系统的默认字符集为 Unicode，——显然，比更改 VSCode 终端环境参数的影响更大，会带来更多无法预料的麻烦。

知道背后的原理，以后能少很多麻烦，或者遇到问题知道原因

5. 大功告成，新手，接任务吧

目前已经能让你开始学习了，后续还需要进一步了解和熟悉以下方面的配置：

`c_cpp_properties.json` 配置 C/C++ 语言的基本设置 `launch.json` 配置调试操作相关的设置 `tasks.json` 配置编译相关的设置

——具体细节可参考其它教程，有很多很多。

有了 Scoop，先顺手装两个强大的工具：

```
# 添加一个官方软件包仓库
scoop bucket add extras
# 安装两个非常强大的工具
scoop install everything ffmpeg
```

Everything，顾名思义，就是 everything 的意思，非常方便

5.1. 任务一：搜索了解一下 Everything

——它知道你的电脑上的每个文件/文件夹 `everything` 在哪个犄角旮旯里，快给它设置个快捷键吧。

5.2. 任务二：搜索了解一下 `ffmpeg`

——最牛的跨平台音视频全方位解决方案，以后你将发现它就像 `原力` 一样，`无处不在`。

——几乎所有的音视频转码、编辑工具，其实**都是调用它**实现的。
