Cambricon CNStudio 用户手册

发布 *0.7.0*

Cambricon

2020 年 02 月 18 日

Cambricon 寒武纪

目录

Cambricon 寒武纪

表格目录

1 版权声明

免责声明

中科寒武纪科技股份有限公司（下称“寒武纪”）不代表、担保（明示、暗示或法定的）或保证本文件所含信息，并明示放弃对可销售性、所有权、不侵犯知识产权或特定目的适用性做出任何和所有暗示担保，且寒武纪不承担因应用或使用任何产品或服务而产生的任何责任。寒武纪不应对因下列原因产生的任何违约、损害赔偿、成本或问题承担任何责任：（1）使用寒武纪产品的任何方式违背本指南；或（2）客户产品设计。

责任限制

在任何情况下，寒武纪都不对因使用或无法使用本指南而导致的任何损害（包括但不限于利润损失、业务中断和信息损失等损害）承担责任，即使寒武纪已被告知可能遭受该等损害。尽管客户可能因任何理由遭受任何损害，根据寒武纪的产品销售条款与条件，寒武纪为本指南所述产品对客户承担的总共和累计责任应受到限制。

信息准确性

本文件提供的信息属于寒武纪所有，且寒武纪保留不经通知随时对本文件信息或对任何产品和服务做出任何更改的权利。本指南所含信息和本指南所引用寒武纪文档的所有其他信息均“按原样”提供。寒武纪不担保信息、文本、图案、链接或本指南内所含其他项目的准确性或完整性。寒武纪可不经通知随时对本指南或本指南所述产品做出更改，但不承诺更新本指南。

本指南列出的性能测试和等级要使用特定芯片或计算机系统或组件来测量。经该等测试，本指南所示结果反映了寒武纪产品的大概性能。系统硬件或软件设计或配置的任何不同会影响实际性能。如上所述，寒武纪不代表、担保或保证本指南所述产品将适用于任何特定用途。寒武纪不代表或担保测试每种产品的所有参数。客户全权承担确保产品适合并适用于客户计划的应用以及对应用程序进行必要测试的责任，以期避免应用程序或产品的默认情况。

客户产品设计的脆弱性会影响寒武纪产品的质量和可靠性并导致超出本指南范围的额外或不同的情况和/或要求。

知识产权通知

寒武纪和寒武纪的标志是中科寒武纪科技股份有限公司在美国和其他国家的商标和/或注册商标。其他公司和产品名称应为与其关联的各自公司的商标。

本指南为版权所有并受全世界版权法律和条约条款的保护。未经寒武纪的事先书面许可，不可以任何方

式复制、重制、修改、出版、上传、发布、传输或分发本指南。除了客户使用本指南信息和产品的权利，根据本指南，寒武纪不授予其他任何明示或暗示的权利或许可。未免疑义，寒武纪不根据任何专利、版权、商标、商业秘密或任何其他寒武纪的知识产权或所有权对客户授予任何（明示或暗示的）权利或许可。

- 版权声明
- © 2020 中科寒武纪科技股份有限公司保留一切权利。

2 前言

2.1 版本记录

表 2.1: 版本记录

文档名称	Cambricon CNStudio 用户手册
版本号	V0.7.0
作者	Cambricon
修改日期	2020.02.24

3 概述

CNStudio 是基于 Visual Studio Code（简称 VSCode）的 BANG C 语言编程插件，利用 VSCode 强大的编辑和可视化操作，使编写 BANG C 语言更加方便。CNStudio 目前提供的主要功能包含：语法高亮、自动补全、程序调试。

注意:

颜色主题为 **Dark（Visual Studio）** 会影响 CNStudio 的高亮显示，建议使用其他颜色主题。

4 插件安装

4.1 环境依赖

1. VSCode 版本要求 1.28.0 及以上版本。
2. 依赖的 VSCode 其他插件：
   - ms-vscode.cpptools。
     该插件支持 Host 端 cpp 程序的自动补全、语法检查、智能提示的功能。
   - vscode remote ssh。
     现阶段 Windows 系统下无法直接使用 VSCode 的调试功能。在 Windows 系统下，安装该插件连接远程 Linux 服务器，可实现在 windows 系统中打开远程服务器文件，并且能够获得与 linux 中相同的代码补全，代码导航，调试功能。关于远程调试功能请参考SSH 远程调试 章节。
3. 依赖编译 BANG C 的 CNCC、CNAS 编译器，调试 BANG C 的 CNGDB 调试器。

4.2 软件包获取

当前 CNStudio 插件只提供离线安装包，不支持在线安装，安装包是扩展名为 vsix 的文件。

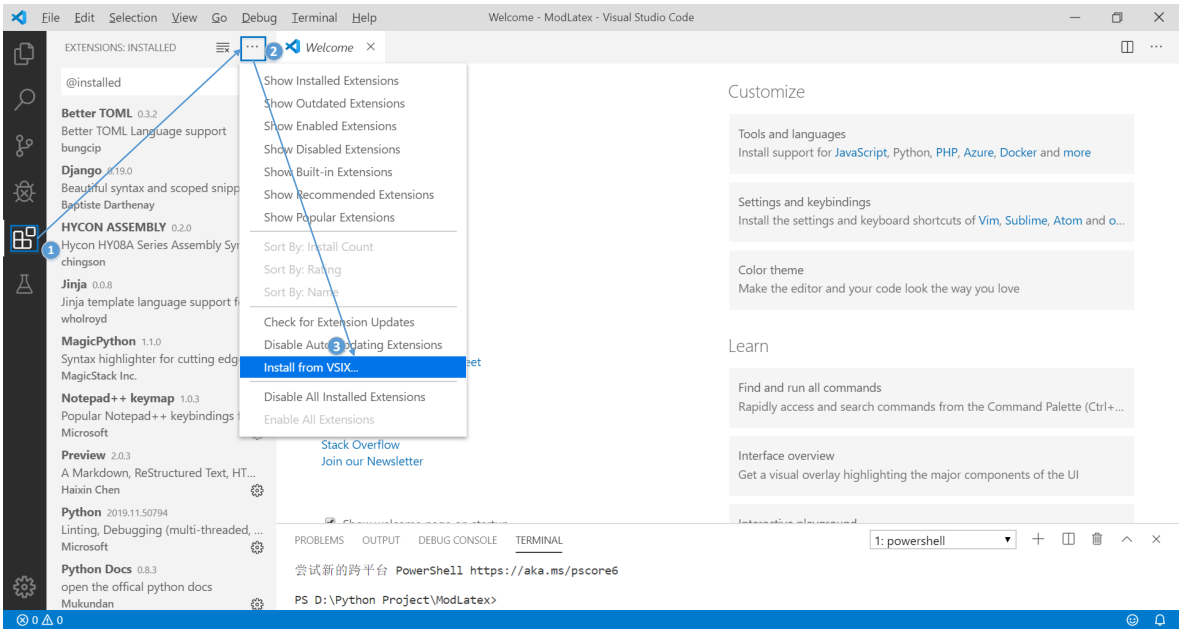
4.3 插件安装

CNStudio 是 VSCode 的插件，因此 VSCode 插件的安装步骤，都是基于 VSCode 的操作。在安装 CNStudio 之前，先将 CNStudio 的插件保存到自定义目录下，该目录没有特别要求，由用户定义。

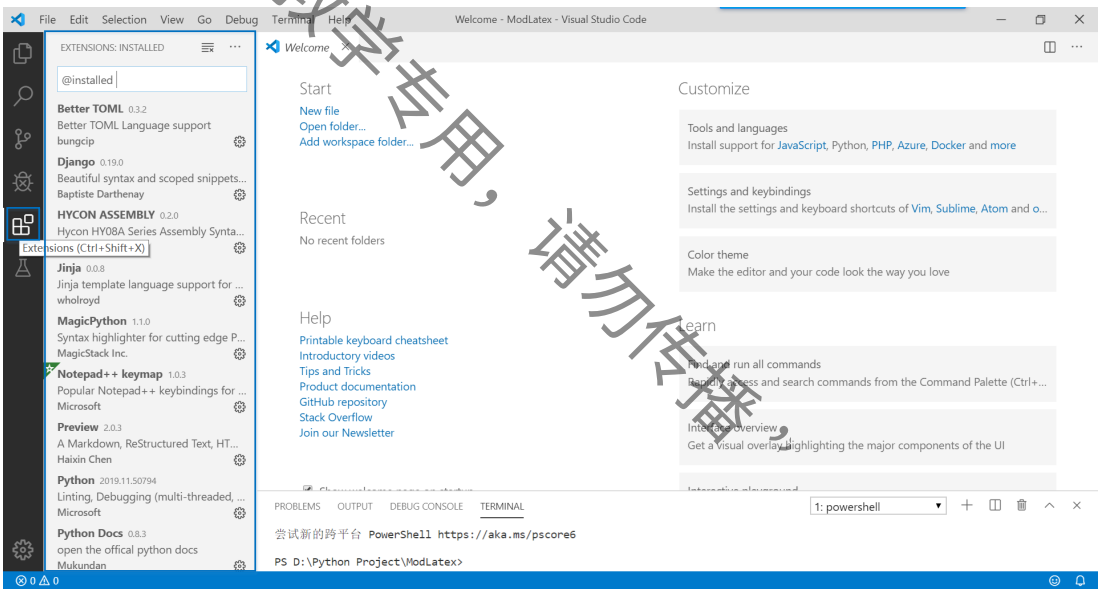
注意:

- 不同版本 VSCode 的图标可能会不一样，但图标的文字提示是相同的，可以按文字提示选择相应操作。
- 如果是通过 remote SSH 功能链接到远程服务器的方式使用 VSCode，请在 SSH 连接到远程服务的状态下安装插件。

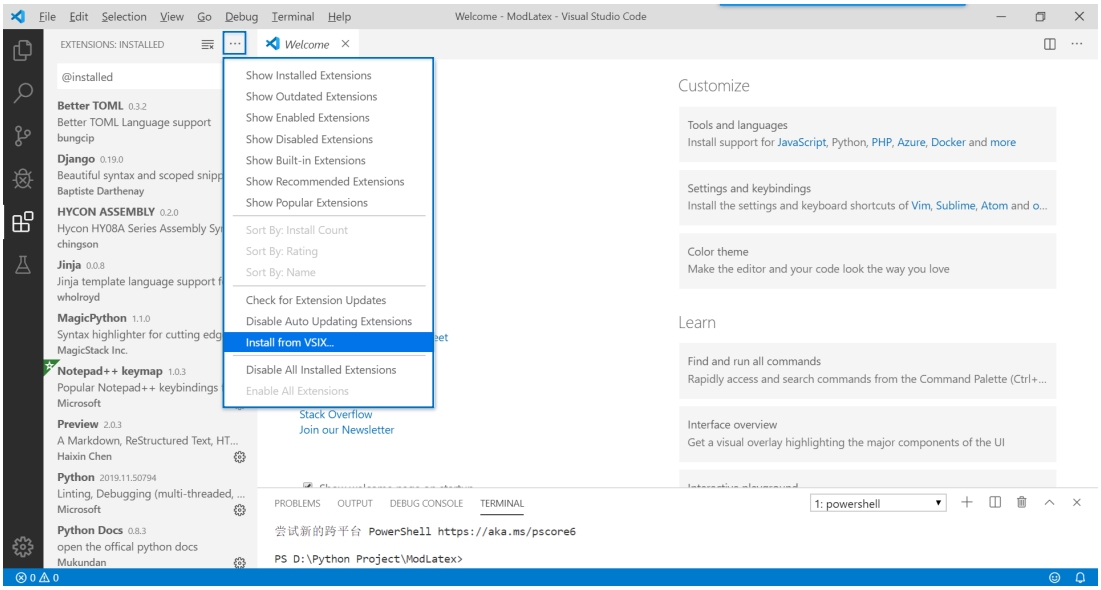
安装 CNStudio 插件的过程主要分为以下三个步骤，如下图所示。



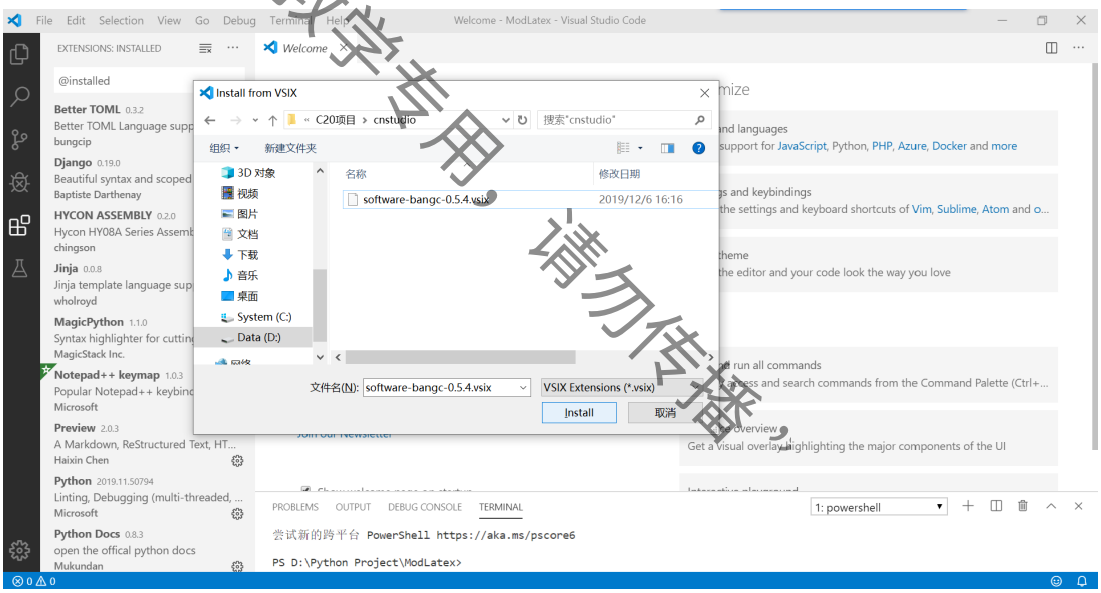
· 步骤一、点击下图中的“Extensions”图标，打开安装插件界面，下图蓝色矩形标注的界面。



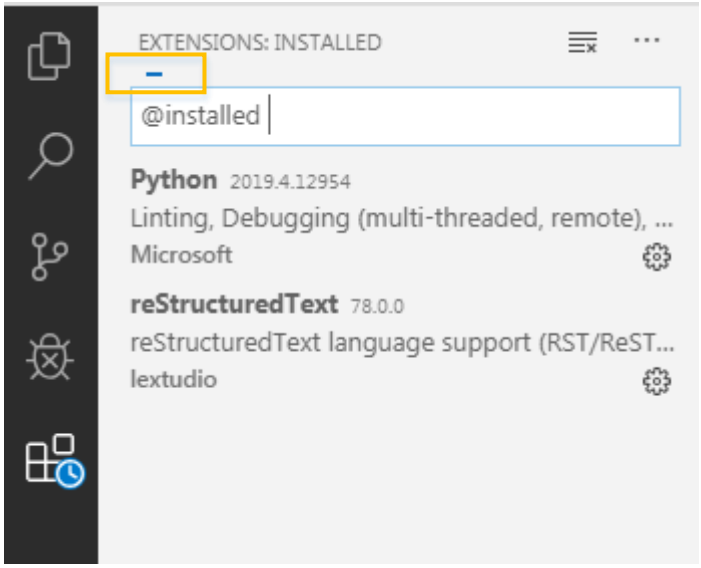
· 步骤二、点击安装插件界面右上角的“···”图标，在弹出的菜单中选择“install from VSIX”。



· 步骤三、在弹出的对话框中，选择要安装的 CNStudio 插件，点击“install”按钮进行安装，如下图所示：



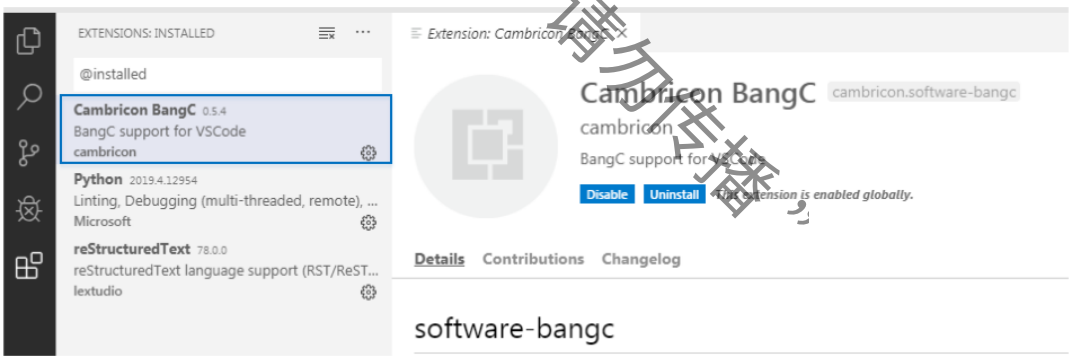
· 步骤四、在安装插件界面会看到插件安装过程，如下图黄框所标识的蓝色短线在不停的循环移动，如果看不到蓝色短线，则表明插件安装结束，重启 vscode 即可。



· 步骤五、重启 VSCode，检查插件是否安装成功。

在安装插件界面的顶部输入框，输入“@installed”，可以看到已安装的全部插件，CNStudio 插件名称为“Cambricon BangC”。在已安装插件中看到“Cambricon BangC”说明安装成功，如下图中蓝色矩形框所标识的插件。

如果看不到“Cambricon BangC”，则说明安装失败，需要重新安装插件或联系插件工程师解决。



4.4 插件升级

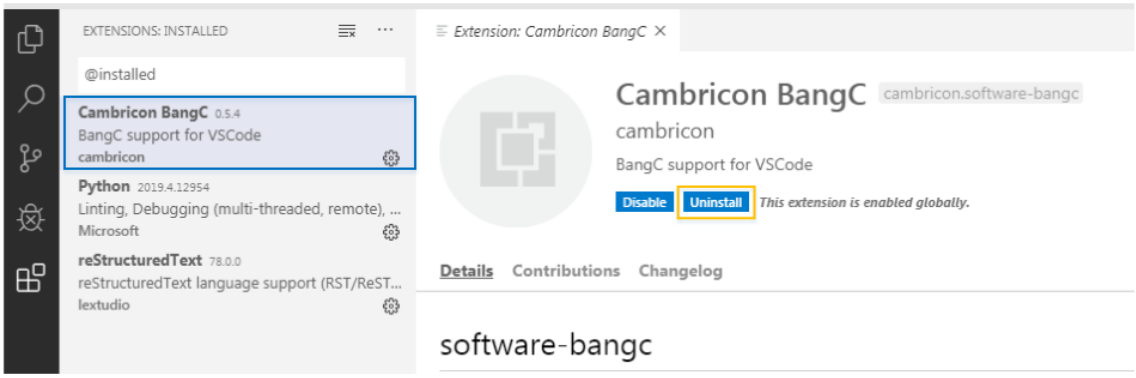
升级插件有两种方法：

1. 先卸载老版本插件，再安装新版本插件，直到安装成功。详细的安装步骤参见插件安装 。
2. 直接安装新版本插件，安装成功后，老版本会被覆盖。

4.5 插件卸载

VSCode 卸载插件的过程比较方便，卸载 CNStudio 插件的详细步骤如下所示：

- 步骤一、参考插件安装的步骤五 找到已安装的 CNStudio 插件，并选择 CNStudio 插件，如下图所示：



- 步骤二、选择 CNStudio 插件后，可以看到插件的详细信息，并能看到“Disable”和“Uninstall“两个按钮，如上图黄色框所标识的按钮。
  - Disable：禁止插件生效。执行该操作后，插件将不会再生效，但插件不会被卸载。如果再次使用该插件，再“Enable”即可。如果只是短暂的禁止 CNStudio 插件运行，可以执行 **Disable** 操作。
  - Uninstall：卸载插件。执行该操作后，插件将被卸载。如果想彻底卸载 CNStudio 插件，请执行 **Uninstall** 操作。
- 步骤三、卸载插件后重启 VSCode，在已安装插件中看不到“Cambricon BangC”，即插件卸载成功。

5 编程指南

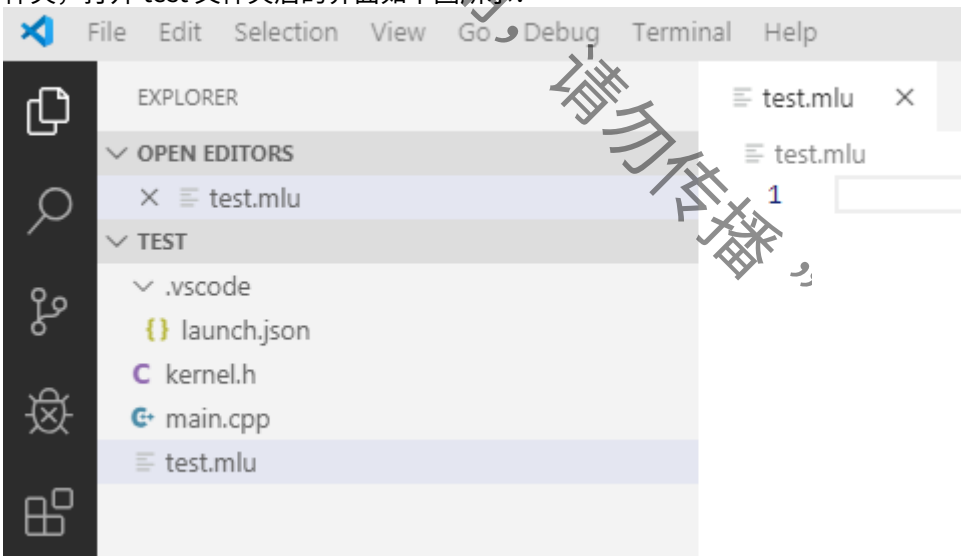
本章节以实际案例演示编写调试 BANG C 的流程。

BANG C 程序源文件后缀名为“*.mlu”。安装 CNStudio 插件后，vscode 会自动识别 mlu 文件。

5.1 工程创建

下面以 test 工程为例描述创建工程的步骤。

- 步骤一、新建文件夹 test，并在 test 文件夹下新建 mlu 端程序 test.mlu，Host 端程序 main.cpp，以及头文件 kernel.h。
- 步骤二、打开 VSCode，选择下面的菜单“File”-“Open Folder…”，选择 **步骤一**新建的 test 文件夹，打开 test 文件夹后的界面如下图所示：



- 步骤三、如上图所示，打开 test 文件夹后，可以看到新建的 test.mlu、main.cpp 和 kernel.h 文件，打开相应的文件进行编程。

小技巧:

- 编写 mlu 程序时，可以使用 tab 键对代码自动补全。
- 打开文件夹后，可以选择“File”-“Save Workspace As…”，将打开的工程保存为 workspace，下次可以直接选择“File”-“Open Workspace…”打开工程。

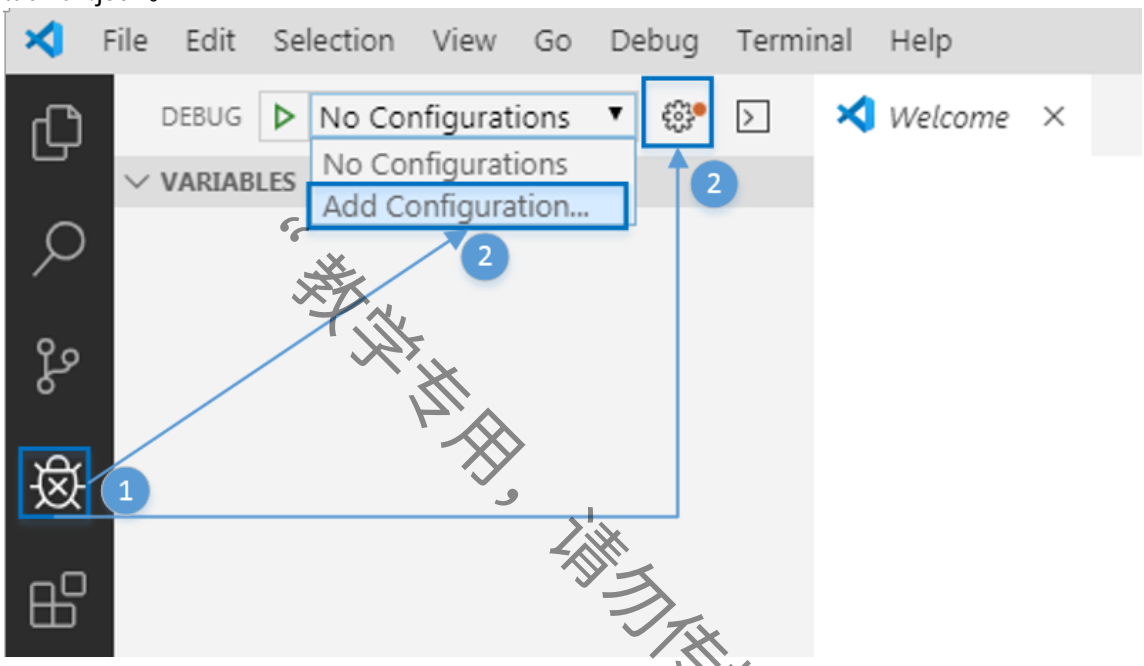
· 步骤四、工程创建结束。

5.2 工程调试

5.2.1 本地调试

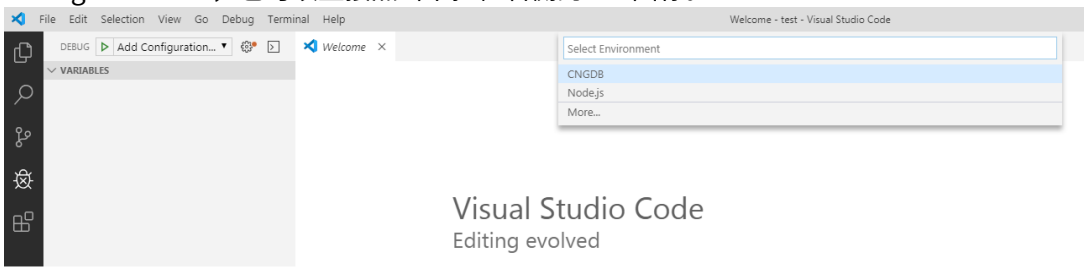
· 步骤一、添加调试配置文件。

如果 VSCode 没有添加过任何配置文件，在创建工程后按下图中”1->2”的顺序调出默认配置文件 launch.json。



具体操作流程如下：

1. 点击  （不同版本图标可能会有所不同），调出 Debug 界面。
2. 如图中步骤“2”所指的位置，调出默认配置选项，可以在 Debug 界面的下拉框中选择“Add Configuration…”，也可以直接点击下拉框右侧的  图标。



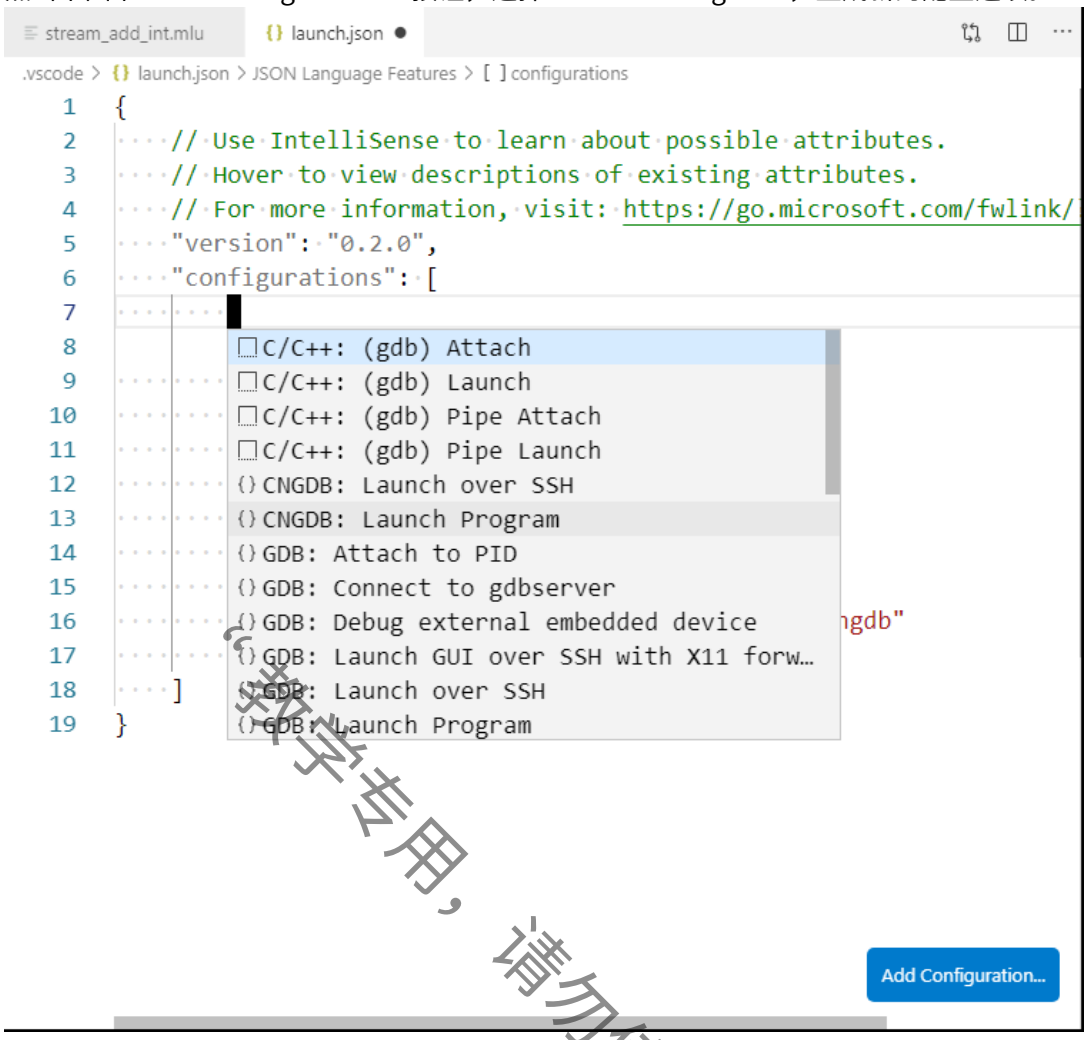
3. 在调出默认配置选项后，选择“CNGDB”，就可以打开默认配置文件 launch.json，配置文件中已经添加了 BANG C 编译的默认配置。

提示:

该默认配置需要修改后，方可使用。

---

4. 点击下图中“add configuration”按钮，选择“Launch Program”，生成新的配置选项。




```
{
   "type": "cngdb",
   "request": "launch",
   "name": "CNGDB Launch Program",
   "target": "./a.out",
   "cwd": "${workspaceRoot}",
   "valuesFormatting": "parseText",
   "cngdbpath": "${PATH}/cngdb",
   "env": {
       "LD_LIBRARY_PATH": "${libcnrt.so}"
   },
   "preLaunchTask": "build" //可选项
}
```

- target: 要调试的目标程序，需要填写相对路径。
- cngdbpath: cngdb 所在路径。
- LD_LIBRARY_PATH：libcnrt.so 所在路径

· 步骤二、编译工程。

打开 mlu 工程，编译调试版本，编译过程如下：

提示:

以下命令在 Shell 下执行。

```
cncc \*.mlu -o \*.o -g --bang-mlu-arch=\$TARGET
gcc \*.cpp \*.o -g -I \${PATH_include} -L \${PATH_lib} -lcnrt
```

CNStudio 支持自动编译，实现自动编译的步骤如下所示：

1. 添加配置文件，添加配置文件的过程参见步骤一添加调试配置文件 ，在打开的 launch.json 配置文件中增加”preLaunchTask”：“build”选项。
2. 在工程目录的.vscode 文件夹添加 tasks.json。

   tasks.json 文件写入如下内容：

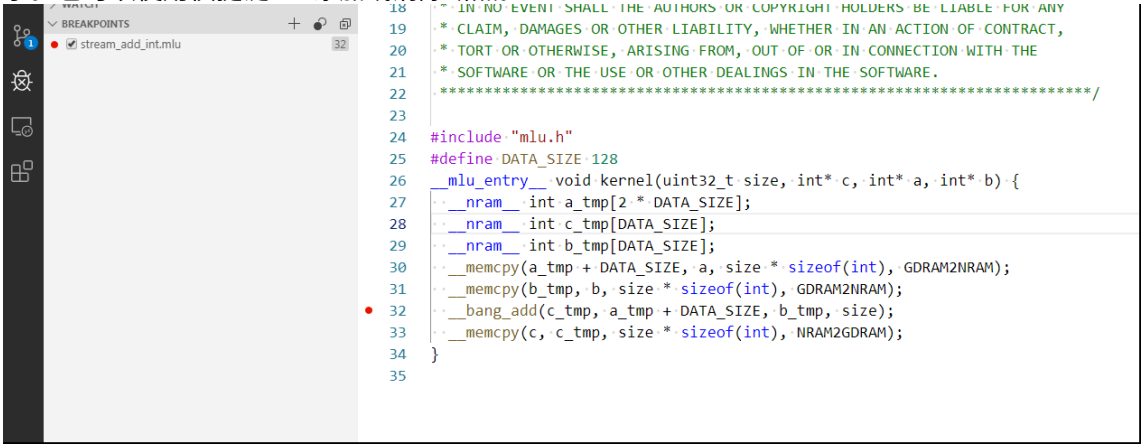
```
{
   "version": "2.0.0",
   "tasks": [
       {
           "type": "shell",
           "label": "cncc",
           "command": "/.../llvm/build/bin/cncc",
           "args": [
               "-c",
              "-g",
               "*.mlu",
               "--bang-mlu-arch=MLU270"
           ],
           "problemMatcher": [
               "$gcc"
           ],
           "group": "build"
       },
       {
           "type": "shell",
           "label": "g++",
           "command": "/usr/bin/g++",
           "args": [
               "-g",
               "\*.cpp",
               "*.o",
```

```
                "-I",
                "/.../Bang_testcase/MLU200/neuware/include",
                "-L",
                "/.../Bang_testcase/MLU200/neuware/lib",
                "-lcnrt"
                //"-o",
                //"${fileDirname}/${fileBasenameNoExtension}"
            ],
            "problemMatcher": [
                "$gcc"
            ],
            "group": "build",
            "dependsOn": ["cncc"]
        },
        {
            "label": "build",
            "type": "shell",
            "command": "rm",
            "args": ["*.o"],
            "problemMatcher": [
                "$gcc"
            ],
            "dependsOn": ["g++"]
        }
    ]
}
```

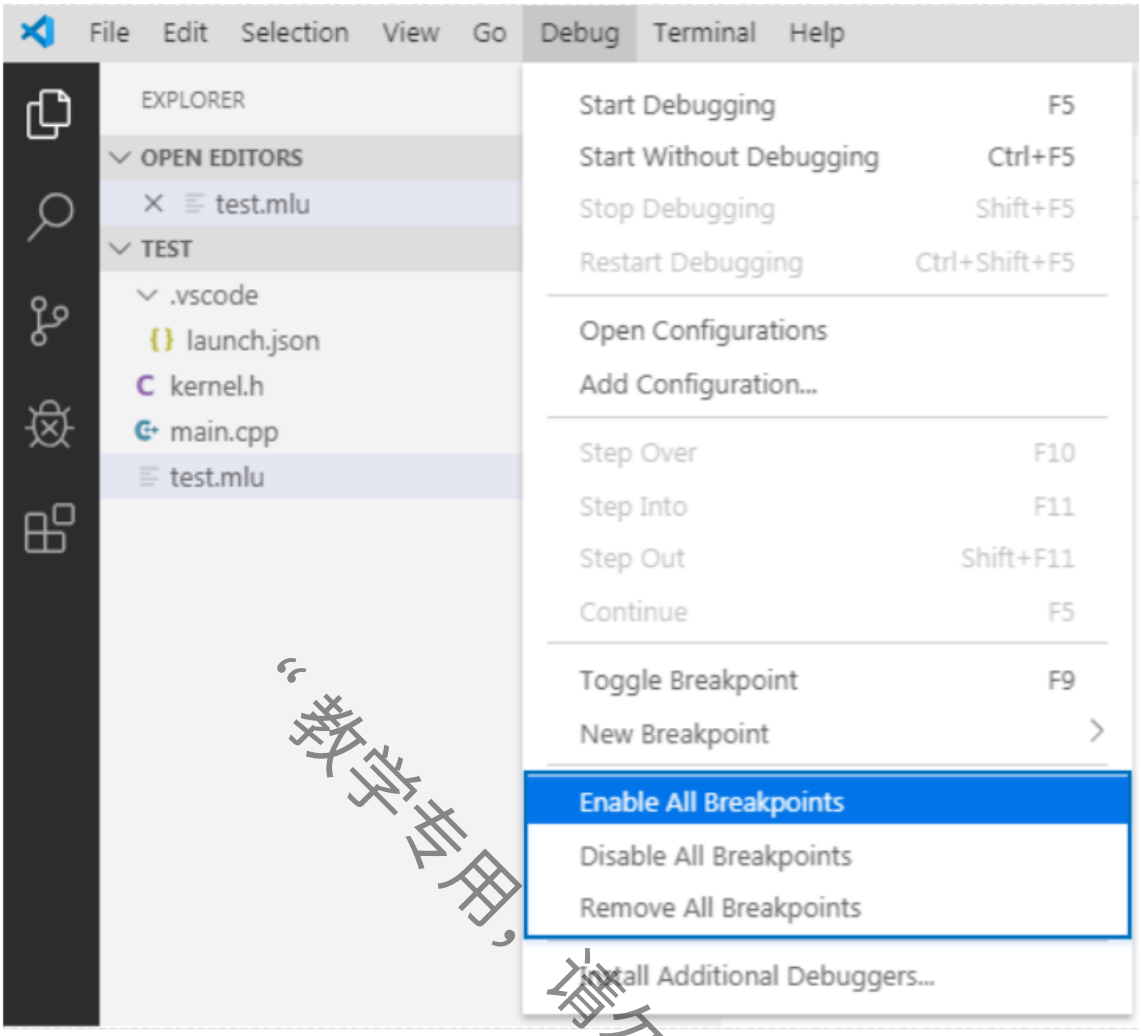

3. 经过以上配置，在 VSCode 中直接按“F5”进行调试，调试前自动完成预编译。

· 步骤三、添加断点。

使用鼠标在要添加断点的代码行左侧点击，如果行首出现了小红点，说明添加上了断点，如下图所示。也可以使用快捷键 **F9** 添加或消除断点。

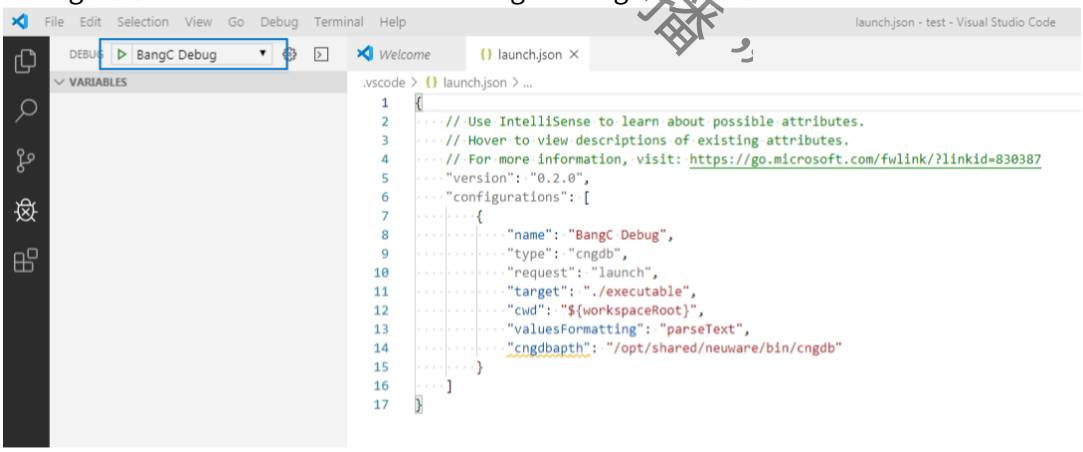


在“Debug”菜单中管理断点，如图所示：



- 步骤四、开始调试。

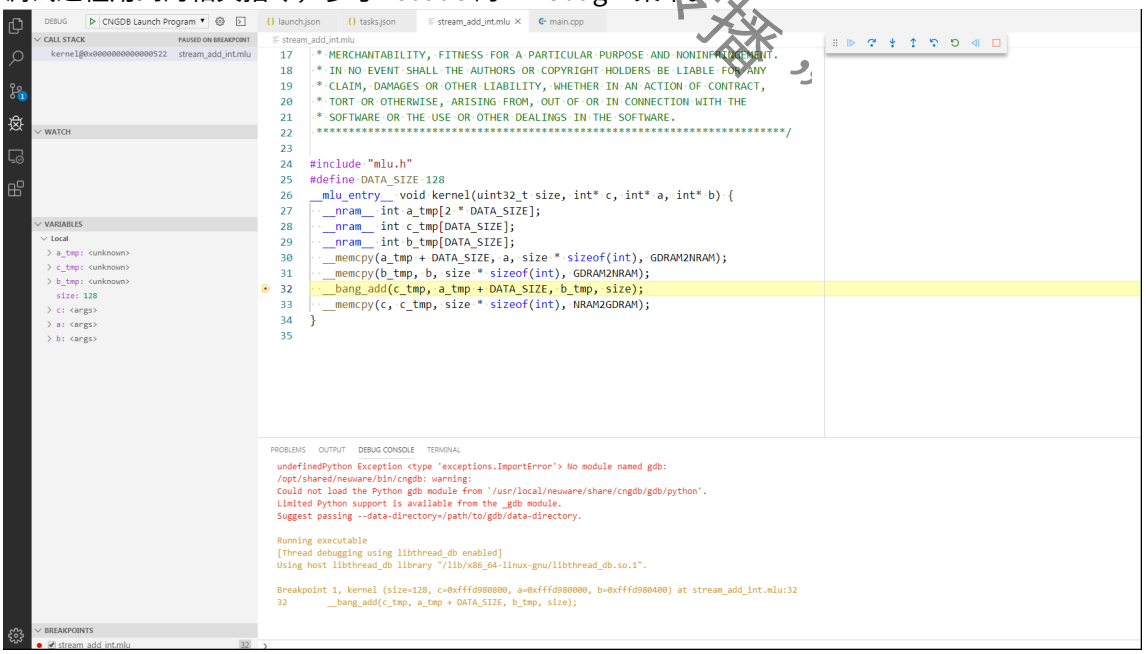
1. Debug 界面的下拉框显示调试选项为“BangC Debug”，如下图所示：



在 debug 配置中选择 CNGDB Launch Program，然后点击左侧的开始按钮 或者按快捷键 F5，开始 debug。

- 步骤五、调试过程。

  调试过程用到的相关指令，参考 vscode 的“Debug”菜单。



“Debug”界面信息栏相关信息解释如下：

  - BREAKPIONTS: 显示当前添加的断点，及断点是否启用的状态。

- CALL STACK: 显示函数调用栈。
- VARIABLES: 显示变量。
- WATCH: 在 WATCH 栏添加变量名，监控相关变量的值。

5.2.2 SSH 远程调试

注解:

使用 CNStudio 进行 SSH 远程调试前，请参考 VSCode 官网完成远程调试的相应配置。

SSH 远程调试的基本过程与本地调试相同。需要更改调试配置为“CNGDB Launch Program (SSH)”, 配置如下：

```
{
   "type": "cngdb",
   "request": "launch",
   "name": "CNGDB Launch Program (SSH)",
   "target": "./a.out",
   "cwd": "${workspaceRoot}",
   "cngdbpath": "/opt/shared/neuware/bin/cngdb",
   "ssh": {
       "host": "",
       "cwd": " ",
       "user": "remote_user",
       "password": "",
       "keyfile": "/home/my_user/.ssh/id_rsa",
       "bootstrap": " "
   },
   "valuesFormatting": "parseText",
   "preLaunchTask": "build" //可选项
}
```


SSH 中配置选项：

- host：远程连接的机器 IP；
- cwd：远程机器的工程文件夹；
- user：登录名；
- password：密码，使用 keyfile 时该选项可以删除；
- keyfile：密钥，需配置 SSH 免密登录，将本地机器的 id_rsa.pub 写入 host 机器的 ~/.ssh/authorized_keys 中；
- bootstrap：调试之前的脚本过程，将 libcnrt.so 路径写入 LD_LIBRARY_PATH 中。

6 附录: 快捷键介绍

表 6.1: 快捷键介绍

快捷键	描述
F5	继续调试
Ctrl+F5	重启
F10	单步
F11	单步并进入函数内部
Shift+F5	停止调试

