How to add a debugger to Visual Studio Code

ME 213

September 2025

Contents

EPFL

1 Introduction

This guide shows how to install a debugger and how to compile/run C code in one step on VS Code for both Windows and Linux users.
Please follow it from start to finish.

2 Requirements

For this to work you need to already have **Visual Studio Code** installed.
If you don't have it you can download it here.

Next we need a compiler and a debugger, we are going to use **gcc** and **gdb**.

2.1 Installing GCC and GDB

2.1.1 For Windows

To install **gcc** and **gdb** on Windows we need to use **MinGW** via **MSYS2**.

To do so you can follow this VS code video that is based on this tutorial.
NOTE : Use "Shift + Insert" to paste the command line in the terminal.

Now to make sure **gcc** and **gdb** have been correctly installed type the following commands into the terminal:

```
gcc --version
gdb --version
```

If everything is installed properly it should look like this:

```
PS C:\Users\pbign> gcc --version
gcc.exe (Rev2, Built by MSYS2 project) 14.2.0
Copyright (C) 2024 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

PS C:\Users\pbign> gdb --version
GNU gdb (GDB) 15.2
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

2.1.2 For Linux

To install **GCC** first open the terminal and enter the following commands:

```
sudo apt update
sudo apt install gcc -y
```

You can make sure gcc is installed correctly by typing:

```
gcc --version
```

You should get the following message:

```
pbign@ThinkPad:~$ gcc --version
gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

pbign@ThinkPad:~$
```

Finally to install **GDB** enter next commands in the terminal.

```
sudo apt-get update
sudo apt-get install gdb
```

You can then verify that gdb is correctly installed by typing:

```
gdb --version
```

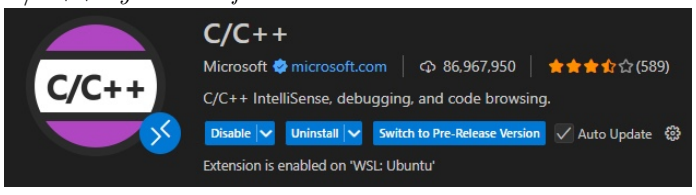
It should look like this:

```
pbign@ThinkPad:~$ gdb --version
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
pbign@ThinkPad:~$
```

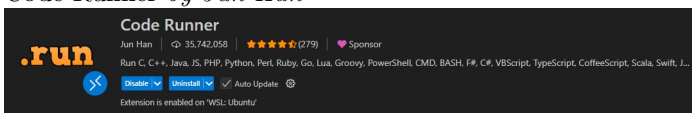
3 Compile and run code

Open VS Code (for Windows users make sure it is **not** opened through WSL) and access the extensions page by pressing "Ctrl + Shift + X" or by clicking on this icon on the left panel  .
In this tab you need to install the following extensions:

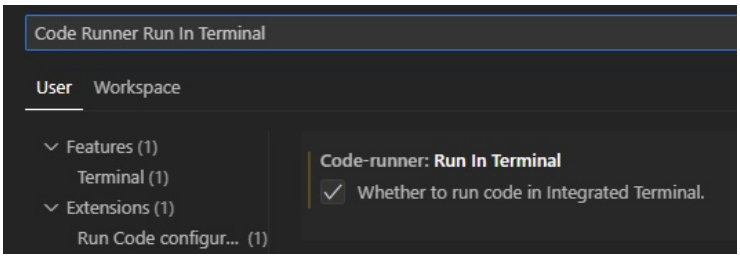
- C/C++ *by Microsoft*



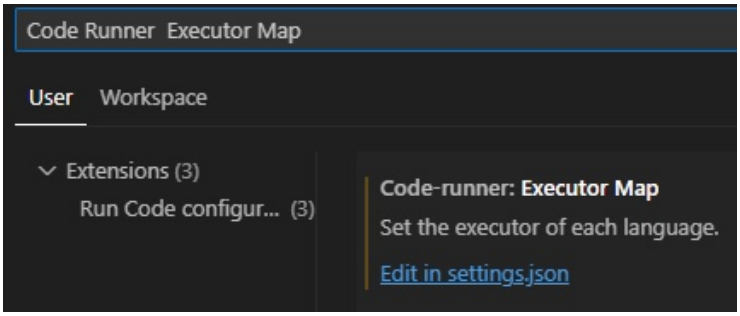
- Code Runner *by Jun Han*



We are now going to modify the settings of the "Code Runner" extension to fit our needs, first open the settings of VS Code by pressing "Ctrl + ,", search for "Code Runner Run In Terminal" and make sure it is activated.



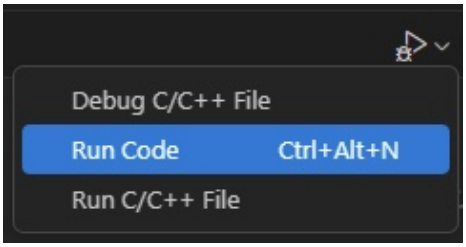
Furthermore, if you are using specific libraries or need to include arguments when running the code, you need to modify the compile or run commands. To do so, open the VS Code settings with "Ctrl + ,", then search for *Code Runner Executor Map* to edit the *settings.json* file.



In the file you can change the *"c":* line to modify the commands that are executed when compiling and running the code.
Those commands are separated by "&&", the first one sets the directory, the second compiles the code, and the last runs the program.
For example to add *-lm* to the compilation command and *Argument* to the run command you need to edit it like this:

```
"c":  "cd $dir && gcc $fileName -o $fileNameWithoutExt -lm && $dir$fileNameWithoutExt Argument",
```

You should now be able to compile and run C code in one action by pressing **Ctrl + Alt + N** or by clicking on this option in the top right.



If you encounter any issues try restarting VS Code.

4 Adding a debugger

To use the debugger open VS Code and create a folder named ".vscode" in the directory you want to work in, then in this new folder you need to create the files "tasks.json" and "launch.json".
In the *tasks.json* file you need to paste the following:

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "label": "gcc build",
            "type": "shell",
            "command": "gcc",
            "args": [
                "-g",
                "${file}",
                 "-o",
                "${fileDirname}/${fileBasenameNoExtension}"
            ]
        }
    ]
}
```

For the *launch.json* file, you need to paste:

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Debug C",
      "type": "cppdbg",
      "request": "launch",
      "program": "${fileDirname}/${fileBasenameNoExtension}",
      "args": [

      ],
      "cwd": "${fileDirname}",
      "MIMode": "gdb",
      "preLaunchTask": "gcc build"
    }
  ]
}
```

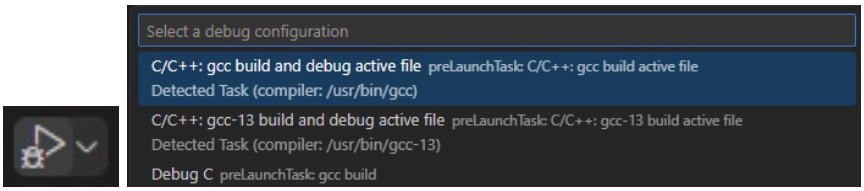
If you need to edit the compilation command modify the *"args":* variable in the *tasks.json* file.

To add arguments to the debugger when running the code modify the *launch.json* file and add arguments to the *"args"* variable like so :

```
"args":["Argument_1","Argument_2","Argument_3"],
```

You now have the ability to use a debugger in VS Code. To set breakpoints, click on the left gutter (red dot) next to a line:

To run the debugger click on the play button on the top right and for the first time select "Debug C".



From now on when you click on the debug button you will need to press "Enter" to launch the debugger with "Debug C", you can also just press **F5** to launch it.
To use the actions of the debugger click on the panel at the top: