vsc_guide.md

Visual Studio Code Guide

Author: Wu Chenggang

Only read this guide after you have done things in c_env.pdf.

Visual Studio Guide is an open source lite weight editor that can replace the IDE with the help of various plugins, just like Emacs and Atom. With good settings, you can use it more easily and faster than other IDEs.

Installation

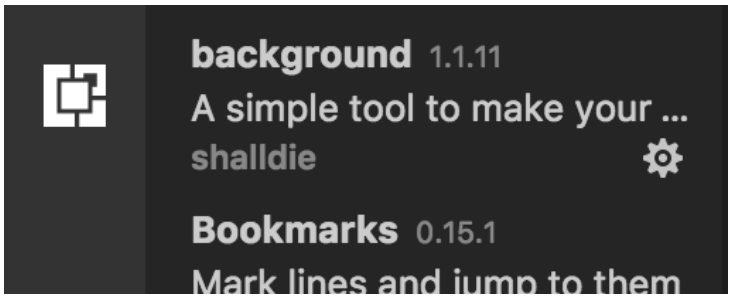
Visit VSC and install it. It's cross-platform.

Install Plugins

Here is a list of recommended plugins for you:

- background
- Bookmarks
- C/C++
- C++ Intellisense
- Code Spellchecker
- Dash
- Expand Selection To Scope
- Git History (git log)
- Project Manager
- vscode-icons
- vscode-pdf

You can search and install them from the bottom icon in the left column.



You may need to read the description and the manual of the plugins to know how to use and config them.

The plugins of C/C++ are necessary for debugging. The plugin Project Manager is very useful to find your work directory.

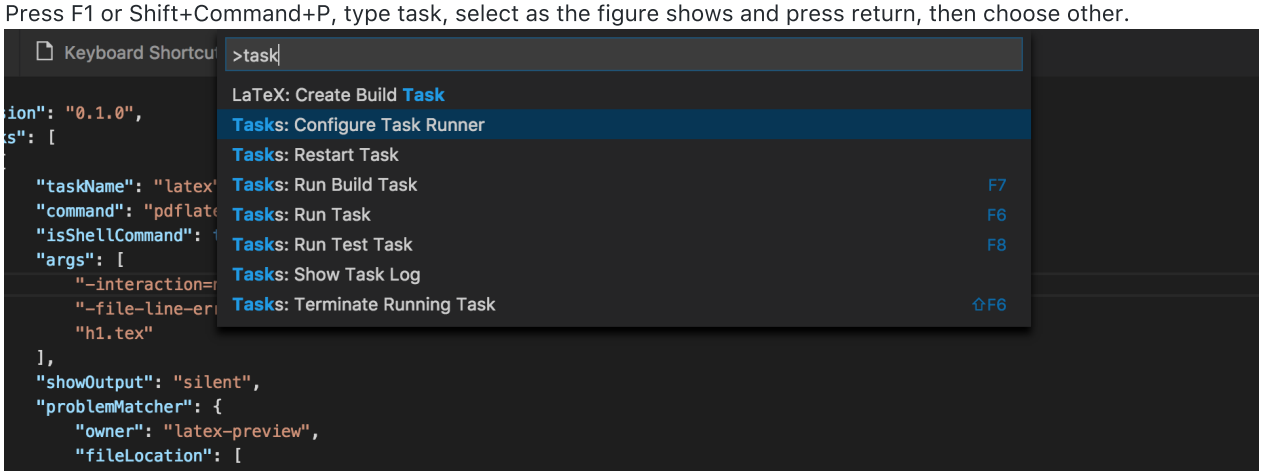
Debugging with VSC

You need to see the c_set_env guide first to prepare the C/C++ environment first.

You can see the official manual for some more detail.

Mac

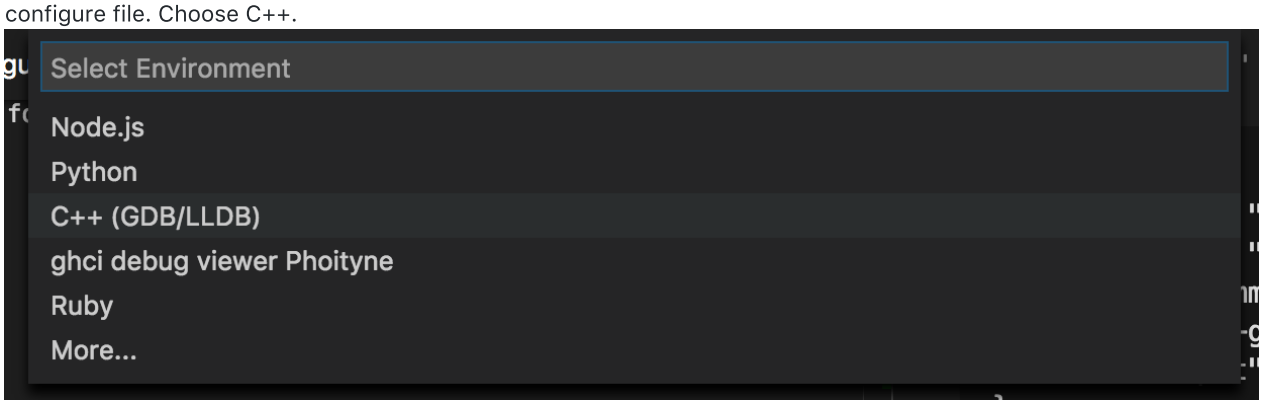
Press F1 or Shift+Command+P, type task, select as the figure shows and press return, then choose other.



Here is a sample config file for tasks.json:

```
{
    "version": "0.1.0",
    "command": "gcc",
    "isShellCommand": true,
    "args": ["-g", "${file}"]
    "showOutput": "always"
}
```

Then press the debug button in the left column and press the green triangle. It will prompt you if you don't have a configure file. Choose C++.



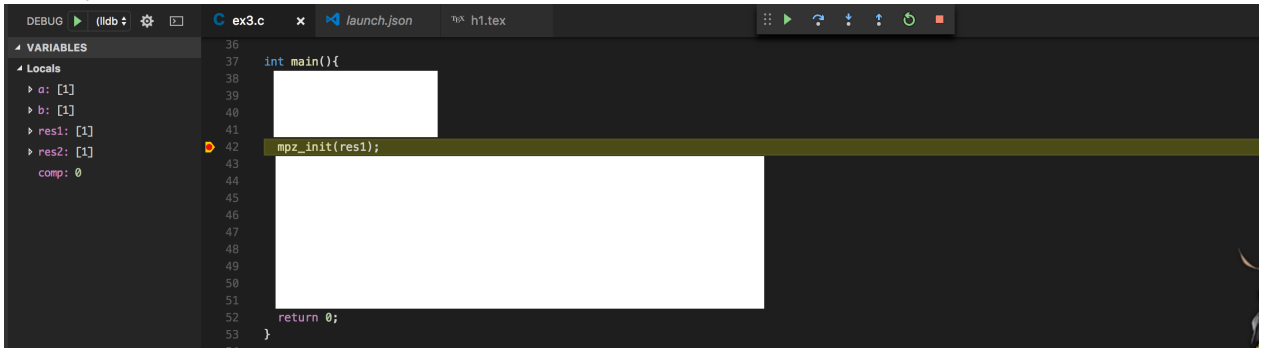
Here is an sample file of launch.json for you

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "(lldb) Launch",
            "type": "cppdbg",
            "request": "launch",
            "program": "${workspaceRoot}/a.out",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${workspaceRoot}",
            "environment": [],
            "externalConsole": false,
            "MIMode": "lldb",
            "preLaunchTask": "gcc"
        }
    ]
}
```

Notice that if you need some input like using `scanf`, you need to set externalConsole as true and start the terminal app. Then you can input from the external terminal.

You can try different config and test what are their usages according to the official manual. But for VG101, these two config files are enough. (You need to change gcc to g++ when using C++)

Now you can click on your .c file and press F5, and it will start debugging just like an IDE. The started program will be the file you clicked.



Linux

See the part of Mac for reference

Windows

See the part of Mac for reference. And here is the config file for MinGW and WSL. Student who has interest could try TDM-GCC on yourself. It's quite similiar to MinGW.

tasks.json:

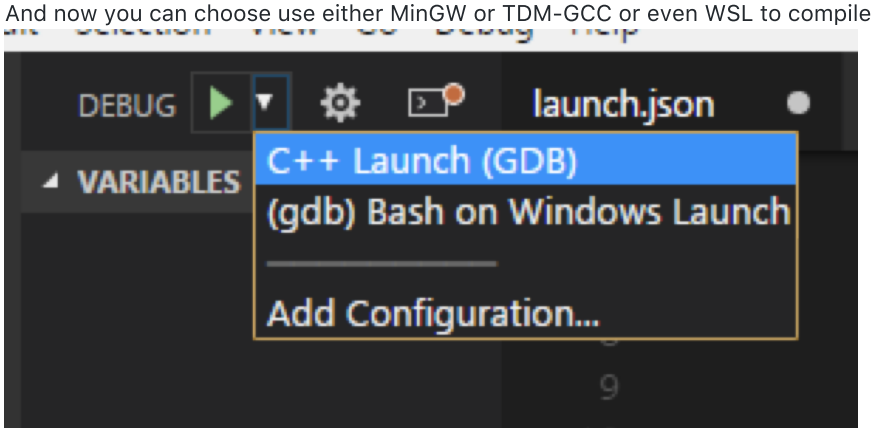
```
{
    "version": "0.1.0",
    "tasks": [{
            "command": "gcc",
            "taskName": "mingw",
            "args": ["-g", "${file}", "-o", "${fileBasenameNoExtension}.exe"]
        },
        {
            "taskName": "wsl",
            "command": "c:\\windows\\sysnative\\bash.exe",
            "args": [
                "-c", "gcc -g ${fileBasename}"
            ]
        }
    ]
}
```

launch.json:

```
{
    "version": "0.2.0",
    "configurations": [{
            "name": "C++ Launch (GDB)",
            "type": "cppdbg",
            "request": "launch",
            "targetArchitecture": "x86",
            "program": "${workspaceRoot}/${fileBasenameNoExtension}.exe",
            "miDebuggerPath": "gdb.exe",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${workspaceRoot}",
            "externalConsole": false,
            "preLaunchTask": "mingw"
        },
        {
            "name": "(gdb) Bash on Windows Launch",
            "type": "cppdbg",
            "request": "launch",
            "program": "/mnt/d/testwsl/a.out",
            "args": [],
            "stopAtEntry": false,
            "cwd": "/mnt/d/testwsl",
            "environment": [],
            "externalConsole": true,
            "windows": {
                "MIMode": "gdb",
```

```
                "setupCommands": [{
                    "description": "Enable pretty-printing for gdb",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                }]
            },
            "pipeTransport": {
                "debuggerPath": "/usr/bin/gdb",
                "pipeProgram": "c:\\windows\\sysnative\\bash.exe",
                "pipeArgs": ["-c"],
                "pipeCwd": ""
            },
            "sourceFileMap": {
                "/mnt/d": "d:\\"
            },
            "preLaunchTask": "wsl"
        }
    ]
}
```

And now you can choose use either MinGW or TDM-GCC or even WSL to compile and debug.



Now you can click on your .c file and press F5, and it will start debugging just like an IDE. The started program will be the file you clicked.