Setup of Visual Studio Code for editing and debugging

Niclas Börlin

January 16, 2023

Contents

1 Overview

The following are suggestions for the respective platforms:

- Linux users
  1. Install VS code (section Install in Ubuntu)
     (a) run locally or
     (b) remote via SSH
- Windwos user
  1. Install VS code and WSL2 (WSL2 is needed for `valgrind` support) (section Install in Windows/WSL)
     (a) run locally under WSL2 or
     (b) remote via SSH
- macOS users
  1. Install VS code (Install in macOS)
     (a) run remote via SSH (no `valgrind` support under macOS)
     (b) there have been rumours of `valgrind` support for intel-based hardware, but the rumours are so far

2 Install VS code

2.1 Install in Ubuntu

1. Install VS Code

   1.1. Via the graphical user interface (GUI)

   1.1.1. Open Software Center/Ubuntu Software

   1.1.1.1. Search for `code`.

   1.1.1.2. Select code (Visual Studio Code). You may have to scroll down a few items.

   1.1.1.3. Press `Install`.

   1.2. Or: Via the command line interface (CLI)

   1.2.1. Open a terminal and execute the following command:

   ```
   sudo snap install code --classic
   ```

2. Open a terminal and execute the following command to launch the VS code environment

   ```
   code
   ```

2.2 Install in Windows/WSL

- Follow the instructions here to install VS code: [1].
- Then follow the instructions here to install and configure WSL: [2]

2.3 Install in macOS

- Follow the instructions here: [3]

[1] `https://code.visualstudio.com/docs/setup/windows`
[2] `https://code.visualstudio.com/docs/cpp/config-wsl`
[3] `https://code.visualstudio.com/docs/setup/mac`

2.4 Other

Global instructions are found here: [4].

3 Install C/C++ extensions

1. Start VS code
2. Install the C/C++ extension
    - 2.1. Inside VS Code, press Ctrl-Shift-X (hold down the Ctrl and Shift keys and press the X key) to enter the Extensions View
    - 2.2. Enter `C/C++` in the search window
    - 2.3. Select the `C/C++ extension` from Microsoft
        - 2.3.1. You may install the `C/C++ Extension pack` from Microsoft instead (apparantly, otherwise VS code will pester you repeataedly with suggestions to install it...).
    - 2.4. Press the Install button
    - 2.5. Exit VS code by selecting File/Exit (Ctrl-Q)

4 Configure global editor settings

This HOWTO was inspired by the guide in [5].

4.1 Set global editor settings

1. Inside VS Code, select File/Preferences/Settings
    - 1.1. Write `tab size` in the search window
        - 1.1.1. Enter `8` as the Tab size
    - 1.2. Enter `detect indentation` in the search window
        - 1.2.1. Uncheck `Editor: Detect Indentation`
    - 1.3. Enter `insert spaces` in the search window
        - 1.3.1. Uncheck `Editor: Insert Spaces`
    - 1.4. Exit VS code (Ctrl-Q).
2. You should now have a file `settings.json` with the following content:

```
{
"editor.tabSize": 8,
"editor.insertSpaces": false,
"editor.detectIndentation": false
}
```

3. The file is stored in

    Linux `~/.config/Code/User/settings.json`

    Windows/WSL (probably same as Linux)

    macOS (probably same as Linux)

[4] https://code.visualstudio.com/docs/setup/setup-overview
[5] https://code.visualstudio.com/docs/cpp/config-linux

5 Work locally (Linux/Windows/WSL)

5.1 Create directories

In a terminal in Linux or WSL, run the following commands:

```
mkdir -p ~/edu/doa/projects/helloworld
cd ~/edu/doa/projects/helloworld
code . # This will start code in the current directory
```

If asked, check ”I trust the authors of the parent dir” and answer ”Yes, I trust the authors” (it’s you!).

5.2 Work with a minimal source code example

5.2.1 Create an example source code

1. Select File/New (Alt-Ctrl-N)
2. Name the file `helloworld.c` (not `.cpp`)
3. Enter the following into the file:

   ```
   #include <stdio.h>

   int main(void)
   {
       printf("Hello, world!\n");

       return 0;
   }
   ```

4. Press Ctrl-S (File/Save) to save the file.

   Now, continue in section Compile.

6 Configure VS code to run remotely over ssh (all platforms)

Inspired by [6].

6.1 Install

1. Install an SSH client
   1.1. See [7].
2. Install the Remote Development extension pack in VS code
   2.1. Start VS code
   2.2. Press Ctrl-Shift-X (hold down the Ctrl and Shift keys and press the X key) to enter the Extensions View
   2.3. Enter `Remote` in the search window
   2.4. Select the `Remote - SSH` extension from Microsoft
   2.5. Press the Install button
   2.6. Exit VS code by selecting File/Exit (Ctrl-Q)

[6] `https://code.visualstudio.com/docs/remote/ssh`.
[7] `https://code.visualstudio.com/docs/remote/troubleshooting#_installing-a-supported-ssh-client`

6.2 Login remote via SSH

1. Figure out you CS username
   1.1. This is the same username you use to log onto other systems and the CS department.
   1.2. It is on the form `c22abc`, `id21def`, `tfy19ghi`, `oi20jkl`, `ens21mno`, `hed21pqr`.
   1.3. In this example, I will use the user name `c22abc`.
2. Remember your password for your CS account
3. Open a terminal/powershell
   3.1. On the command line, write

   ```
   ssh c22abc@itchy.cs.umu.se
   ```

   3.2. If this is your first remote login, you may get an warning that the host is unknown. In this case, and "Yes" to add itchy.cs.umu.se to the list of known hosts.
   3.3. Once logged in, you should get a prompt similar to `c22abc@itchy:~$`, where `c22abc` is your login name, `itchy` is the host name, and ~ is the current directory, where ~ means your home directory at the CS file system.
   3.4. Verify that you can read files in this folder by entering the command `ls` at the prompt:

   ```
   c22abc@itchy:~$ ls
   ```

   3.5. You should see a list of folders and files in your home folder, one of which should be called `edu`. Now change the current directory to `edu`:

   ```
   c22abc@itchy:~$ cd edu
   c22abc@itchy:~/edu$ ls
   c22abc@itchy:~/edu$ ls -la
   ```

   The `-la` is a so called *switch* that tells `ls` to make a long (`l`) listing of all (`a`) files, including hidden files.
   3.6. If your `edu` folder is empty, the `ls` command will not show anything, but the `ls -la` command will show handles to the current directory (`.`) and the parent directory (`..`).
   3.7. Now create some directories to work with.

   ```
   c22abc@itchy:~/edu$ mkdir doa
   c22abc@itchy:~/edu$ mkdir doa/projects
   c22abc@itchy:~/edu$ mkdir doa/projects/helloworld
   ```

   3.8. Now you can log out from itchy by entering

   ```
   c22abc@itchy:~/edu$ exit
   Connection to itchy.cs.umu.se closed.
   ```

   3.9. You may exit your terminal/powershell session.

6.3 Set up VS code to use your SSH login credentials

1. Start VS code
   1.1. Press Ctrl-Shift-P or F1 and enter `remote-ssh` in the command palette window.
   1.2. Select `remote-SSH: Connect to Host...`
      1.2.1. Enter your ssh login credentials as above `c22abc@itchy.cs.umu.se`.
      1.2.2. If asked, select `Linux` as the remote platform type.
      1.2.3. Enter your password.
      1.2.4. VS code will download and set up a few things. Once it has done that, the text `SSH:itchy.cs.umu.se` should appear in green-ish at the bottom left.

        1.2.5. If you end up with multiple VS Code windows, it is probably best to close all windows that do not have the green-ish `SSH:itchy.cs.umu.se` at the bottom left corner.

   1.3. Select `File/Open a folder...` and enter `/home/c22abc/edu/doa/projects/helloworld`. You may have to enter your password again.

        1.3.1. If asked, check ”I trust the authors of the parent dir” and answer ”Yes, I trust the authors” (it’s you!).

   1.4. Select `File/New Text File (Ctrl-N)` and enter `/home/c22abc/edu/doa/projects/helloworld/helloworld.c`.

   1.5. Enter the following text in the file

```
#include <stdio.h>

int main(void)
{
    printf("Hello, world!\n");

    return 0;
}
```

   1.6. Press Ctrl-S to save the file.

   1.7. Press Ctrl-Shift-X to enter the extension panel.

        1.7.1. Enter `@category:debuggers C` in the search window.
        1.7.2. Select the `C/C++` entry by Microsoft.
        1.7.3. Press Install in `SSH: itchy.cs.umu.se`

   1.8. Switch back to the source file `helloworld.c`

   1.9. Press the play icon (Run and debug, Ctrl-Shift-D)

   1.10. If asked for a debug configuration, select `C/C++: gcc build and debug active file` and not any of the ones mentioning gcc with a specific version number.

   1.11. The program should compile and run. Switch to the `TERMINAL` window (right of `PROBLEMS`, `OUTPUT`, `DEBUG CONSOLE`).

        1.11.1. Verify the output is

```
Hello, world!
```

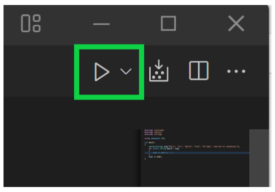
Now, continue in section Compile.

7 Compile

7.1 Compile and run

1. Open `helloworld.c` to make it the active file. The cursor should be blinking somewhere in the source text.

2. Press the Play button at the top right of the window



   2.1. If asked for a debug configuration, select `C/C++: gcc build and debug active file` and not any of the ones mentioning gcc with a specific version number.

2.2. The VS code program will do the following things:

2.2.1. Compile the source code in `helloworld.c` and create an executable file called `helloworld` (same name as source, but without an extension).

2.2.2. Run the executable file (`helloworld`).

3. Inspect the output

3.1. The output is printed to the terminal window. The terminal window may be hidden, so we must bring it to the front.

3.2. Select the `TERMINAL` panel at the bottom of the window (to the right of `PROBLEMS`, `OUTPUT`, `DEBUG CONSOLE`)

3.2.1. The panel should show some text, the first of which should read:

```
Hello, world!
```

3.2.2. There may be a second line similar to

```
[1] + Done    "/usr/bin/gdb" --interpreter=mi ..
```

3.2.3. Feel free to ignore this line for now.

7.2 Modify, compile and run

1. Modify the source

1.1. Go back to the `helloworld.c` window and add the line

```
printf("Normal exit.\n\n");
```

(note the double newlines) on the line before `return 0;`.

2. Press Ctrl-S to save the file.

3. Press the Play button again.

3.1. The VSCode program will re-compile the source code and re-run the resulting executable program. Some commands may flash by.

3.2. Verify that the new printout starts with

```
Hello, world!
Normal exit.
```

followed by a single blank line.

7.3 Set up compiler options required for the DoA course

1. Description

1.1. The VS Code program stores several settings in a file called `tasks.json`.

Linux/remote The file is stored in a directory called `.vscode` in your project folder (here: `~/edu/doa/projects/helloworld`)

Windows Probably the same place.

2. Open the configuration file

2.1. Switch to the explorer window (press the document icon to the upper left/Ctrl-Shift-E)

2.1.1. The window should contain a row `.vscode`

2.2. Press the small arrow to the left of `.vscode` to view the content of the directory

2.2.1. There should be a single file `tasks.json`

2.3. Press the `tasks.json` row

2.3.1. The file should open in the editor with a content similar to:

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

2.3.2. Note the line `"-g"` which is required for debugging (more below).

3. Modify the configuration file

   3.1. Description

   3.1.1. All assignments and excercises on the course requires two compiler flags: `-std=c99` (use the C99 standard) and `-Wall` (turn on all warning)

   3.2. Add the middle two lines below at the indicated place

```
"-fdiagnostics-color=always",
"-std=c99",
"-Wall",
"-g",
```

   3.3. Modify the following lines to read:

```
"label": "C/C++: gcc build active file with DoA options",
"detail": "Task updated to comply with DoA requirements."
```

   3.4. Press Ctrl-S to save the file.

   3.5. Press the cross icon to close the `tasks.json` source window.

4. Test compile and run with modified configuration file

   4.1. Description

   4.1.1. The VS code environment keeps track of which files have been changed, and compiles the source in the current project if it has been changed after the executable file was created.

   4.1.2. However, the environment does not track the configuration file.

        4.1.3. Thus, we need to change the source file to trigger a re-compilation.

    4.2. In the `helloworld.c` source window, add a third newline to read

    ```
    printf("Normal exit.\n\n\n");
    ```

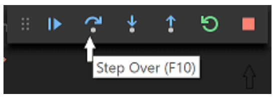
    4.3. Press Ctrl-S to save.

    4.4. Press the Play button to re-compile and run.

5. Verify that the compile and run went ok, now with two blank lines at the end.

8 Run the program in a debugger

1. Go back to `helloworld.c` so that it is the active file.
2. Set a breakpoint at the beginning of `main`
   - 2.1. Either: Click on the editor margin (left of the row numbers) of the line with the opening brace of `main` (probably line 4)
   - 2.2. or: Move the cursor to the line containg the opening brace of the `main` function head (probably line 4) and press the F9 key.
3. Run the program in the debugger
   - 3.1. Press the Play icon again or press the F5 key
     - 3.1.1. Since the source has not changed, no recompile is necessary.
     - 3.1.2. When the executable is started, we will enter the debugger and stop on a line at or near the breakpoint.
     - 3.1.3. It is now possible to debug the code using either the buttons at the top center of the window or pressing some function keys.
     - 3.1.4. The available commands are (left to right in the image below):



Continue (key F5) Run the code and stop in the debugger when the next breakpoint is encountered.

Step over (key F10) Run the code on the current line and stop in the debugger on the next line. If the line contains a function call, run the function until it returns and stop on the next line. Useful if you don't want to single-step the function, e.g. `printf`.

Step into (key F11) Run the code on the current line and stop in the debugger on the next line. If the line contains a function call, stop at the first executable line of the called function. Useful if you want to debug any of your own functions.

Step out (key Shift-F11) Run until the current function returns and stop on the next executable line in the calling function. Useful if you accidently pressed F11 when you intended to press F10.

Restart debugging (key Ctrl-Shift-F5) Quit the program at the current position and restart it.

Stop debugging (key Shift-F5) Quit the program without running the rest of it. Useful if you have seen a bug and want to fix it immediately.

9 Modify to more interesting code

1. Modify the code in helloworld.c to read

```
#include <stdio.h>

int main(void)
{
    int v[4] =  { 4, 2, 6, 3};

    int mx = v[0];
    for (int i=1; i<4; i++) {
if (mx < v[i]) {
    mx = v[i];
}
```

```
        }

        printf("The highest value was: %d.\n", mx);

        printf("Normal exit.");
        return 0;
    }
```

2. Save the code, set a breakpoint at the start of `main`, compile and run the program in the debugger.
   2.1. Notice what happens in the panel called `VARIABLES`, especially `Locals`.
   2.2. Enter the expression `v[i]` in the `WATCH` window. Notice how it changes as you step through the code.

10 Work with multiple files from the code base for the DoA course (not finished)

10.1 Download and unpack the code base

- Download the zip file `datastructures-v1.0.10.1.zip` from the `Filer/Datatypskod` folder on the Canvas course site.
- Unpack the code base into a known folder (locally or remote)

  ```
  mkdir ~/edu/doa/code_base
  cd ~/edu/doa/code_base
  unzip ~/Downloads/datastructures-v1.0.10.0.zip
  ```

  This should unpack the zip file into the directory =~/edu/doa/code$_{base}$/datastructures-v1.0.10.0

10.2 Create a new work folder and copy source files here

```
mkdir ~/edu/doa/projects/int_array_1d_mwe
cd ~/edu/doa/projects/int_array_1d_mwe
cp ~/edu/doa/code_base/datastructures-v1.0.10.0/src/int_array_1d/*.c .
```

This should copy two source files `int_array_1d.c` and `int_array_1d_mwe.c` to the current folder.

10.3 Open project in VS code

In VS code, select `File/Open folder...` and navigate to the new `int_array_1d_mwe` folder.