

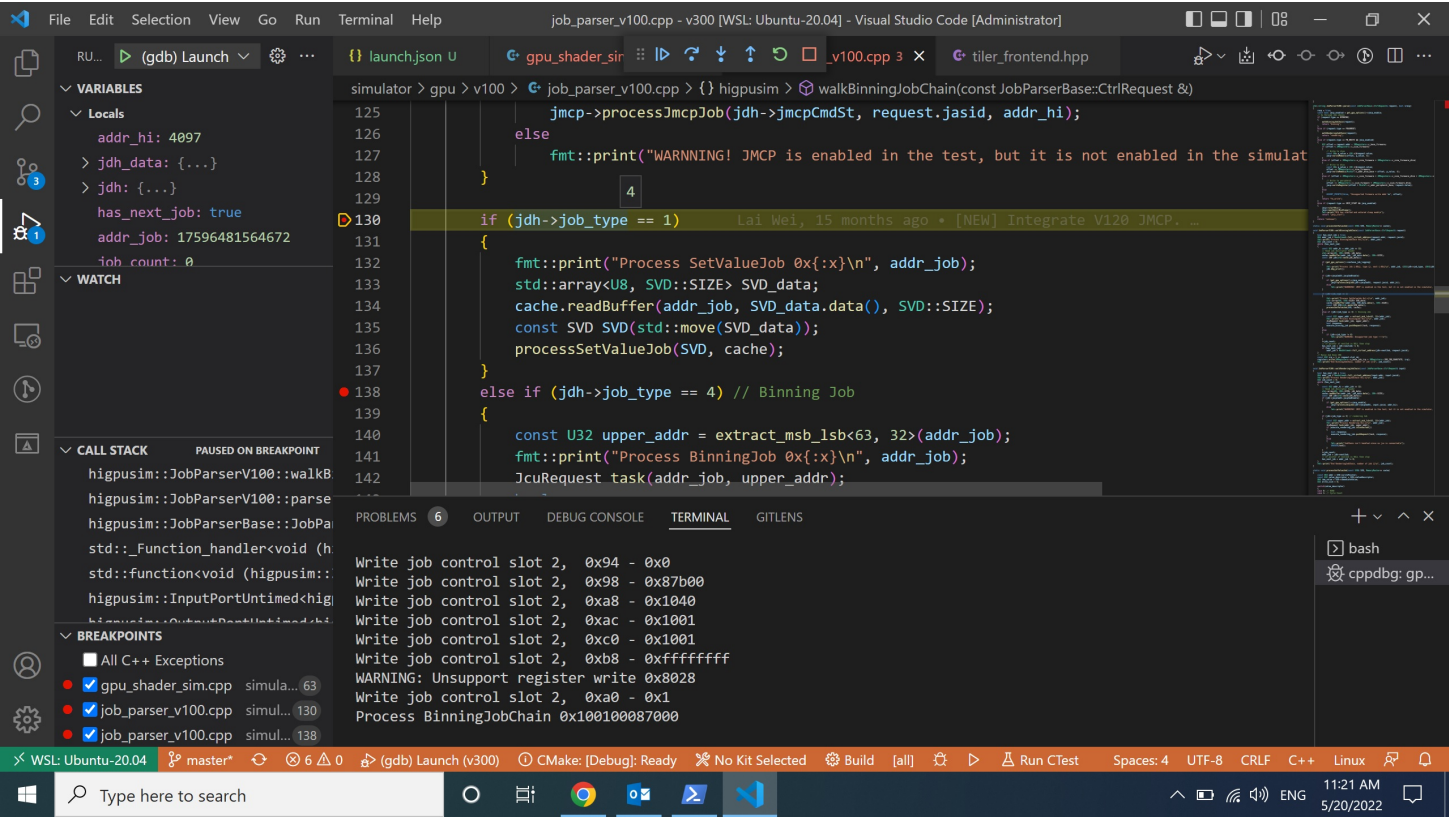
# How to setup a WSL development environment

If you are working on a Linux-based project (such as the GPU Simulator) you may be using the Windows Subsystem for Linux (WSL). WSL does not provide a Linux-standard GUI and so GUI programs cannot run as they normally do.

Visual Studio Code provides a remote development model which can be used to bypass this problem. This document explains how you may setup your development environment to develop and debug the Linux-based GPU Simulator project using Visual Studio Code and WSL.

We will cover:

1. How to enable and run WSL
2. How to enable and test Internet access on WSL
3. How to setup your WSL environment to access Huawei internal gitlab
4. Enabling HTTPS access for Git in WSL (Ubuntu 18.04), Cygwin and MSYS2
5. Installing and running Visual Studio Code on WSL
6. Installing the required VSCode "Extensions" to work with WSL
7. Setting up your code-base for VSCode's CPPTOOLS extension
8. Debugging your code using VSCode and WSL



## How to enable and run WSL

On Windows 10:

1. Press the Windows key, type "Windows features" and choose "Turn Windows features on or off"
2. Make sure that the "Windows Subsystem for Linux" option is ticked and click on Ok
3. You may need to wait for the feature to be enabled and restart the computer

Next, you will need to install an operating system to run on the WSL (We use Ubuntu 20.04 LTS at the time of this writing) - follow the following steps:

1. Press on the Windows key and type "Store" and open the "Microsoft Store" application
2. It may seem like the Store needs you to login with a Microsoft account. However, be informed that this is NOT required. You may choose to ignore or close or cancel the login request and the store application will continue to work as normal.
  - You may not be able to skip this step. As of 2022-06-06 I found I could not continue without logging in. Either register an account with a burner gmail (for example) address or, if you're a contractor, use your own Microsoft account. You may even have an account from Minecraft etc.
3. From the store, search for "Ubuntu 20.04" and then click on "Get"
  - If this step stalls e.g. downloading appears to be taking an infinite amount of time then go to "Library" in the left hand panel of the MS Store and "update all". Anecdotaly it proved necessary to update the Store itself to get Ubuntu to DL.
4. In some cases (such as Ubuntu) you will see the option to "Install" the application after the download operation is complete). Please note that neither downloading nor installing Ubuntu 20.04 from Microsoft Store will require

logging into the Store with a Microsoft account. You can ignore the login requests.

5. During the installation process you may be asked for a username and a password. Choose any that suits you but you must remember the password.
6. You will have to restart your computer after the Ubuntu image is installed

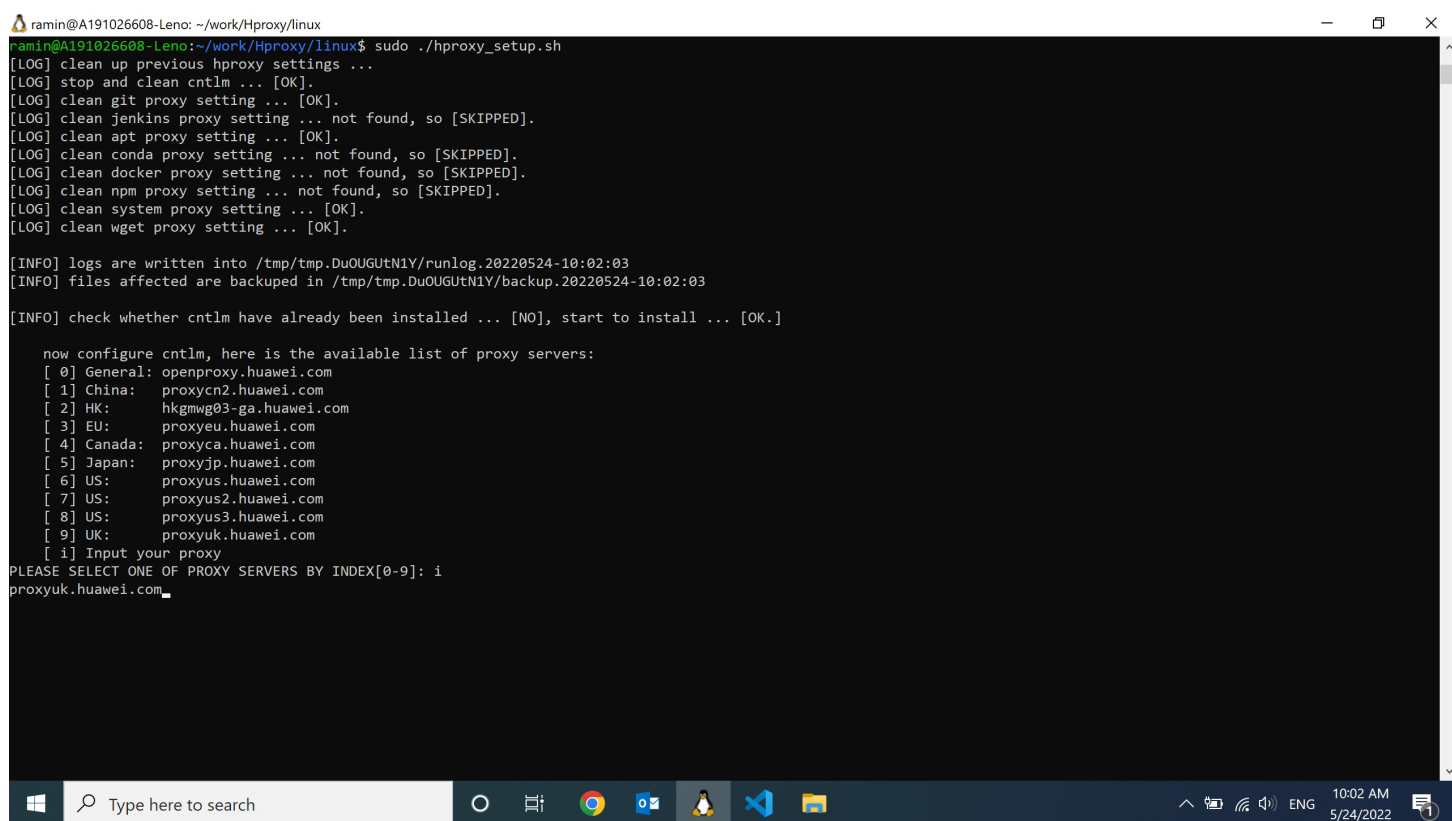
Once you are done with the above steps you can run WSL by pressing on the Windows key, typing "WSL" and opening the WSL program that will appear in your options.

## How to enable and test Internet access on WSL

This can be slightly tricky. Please follow the following steps:

**NOTE:** You may have to repeat some of these steps each time you close and open WSL and find that you have lost the Internet connection!

1. First, you would need to download our internal [Hproxy\\_tool](#). One easy way is to get the zip file from gitlab and place it under your WSL home directory.
2. You can copy the unzipped directory to \\wsl\$\Ubuntu-20.04\home\$USER\
3. Ensure that your WSL window is closed
4. Open powershell in Admin mode (press on the Windows key, type power, and choose Run as Administrator)
5. In powershell run the command "wsl --shutdown"
6. Now, disconnect SPES, any VPN and disable all the WiFi/Ethernet
7. Open a new WSL window (you may type "wsl" in the same powershell instance)
8. Now, re-enable all the WiFi/Ethernet
9. Wait until the Internet connection is well established on Windows
10. You can now go to the Hproxy directory that we copied earlier inside your WSL window: cd ~/Hproxy
11. From there type: cd linux && sudo ./hproxy\_setup.sh # Will ask for your WSL password for the sudo root access operation
12. After a few moments you will see a prompt that looks like below. Here, choose the option [i] (type i) and press enter.
13. Now type "proxyuk.huawei.com" and press enter
14. If successful, you will be asked for your corporate account ID and password after which your WSL will be successfully connected to the Internet using Huawei proxy servers



```
ramin@A191026608-Leno: ~/work/Hproxy/linux
ramin@A191026608-Leno:~/work/Hproxy/linux$ sudo ./hproxy_setup.sh
[LOG] clean up previous hproxy settings ...
[LOG] stop and clean cntlm ... [OK].
[LOG] clean git proxy setting ... [OK].
[LOG] clean jenkins proxy setting ... not found, so [SKIPPED].
[LOG] clean apt proxy setting ... [OK].
[LOG] clean conda proxy setting ... not found, so [SKIPPED].
[LOG] clean docker proxy setting ... not found, so [SKIPPED].
[LOG] clean npm proxy setting ... not found, so [SKIPPED].
[LOG] clean system proxy setting ... [OK].
[LOG] clean wget proxy setting ... [OK].

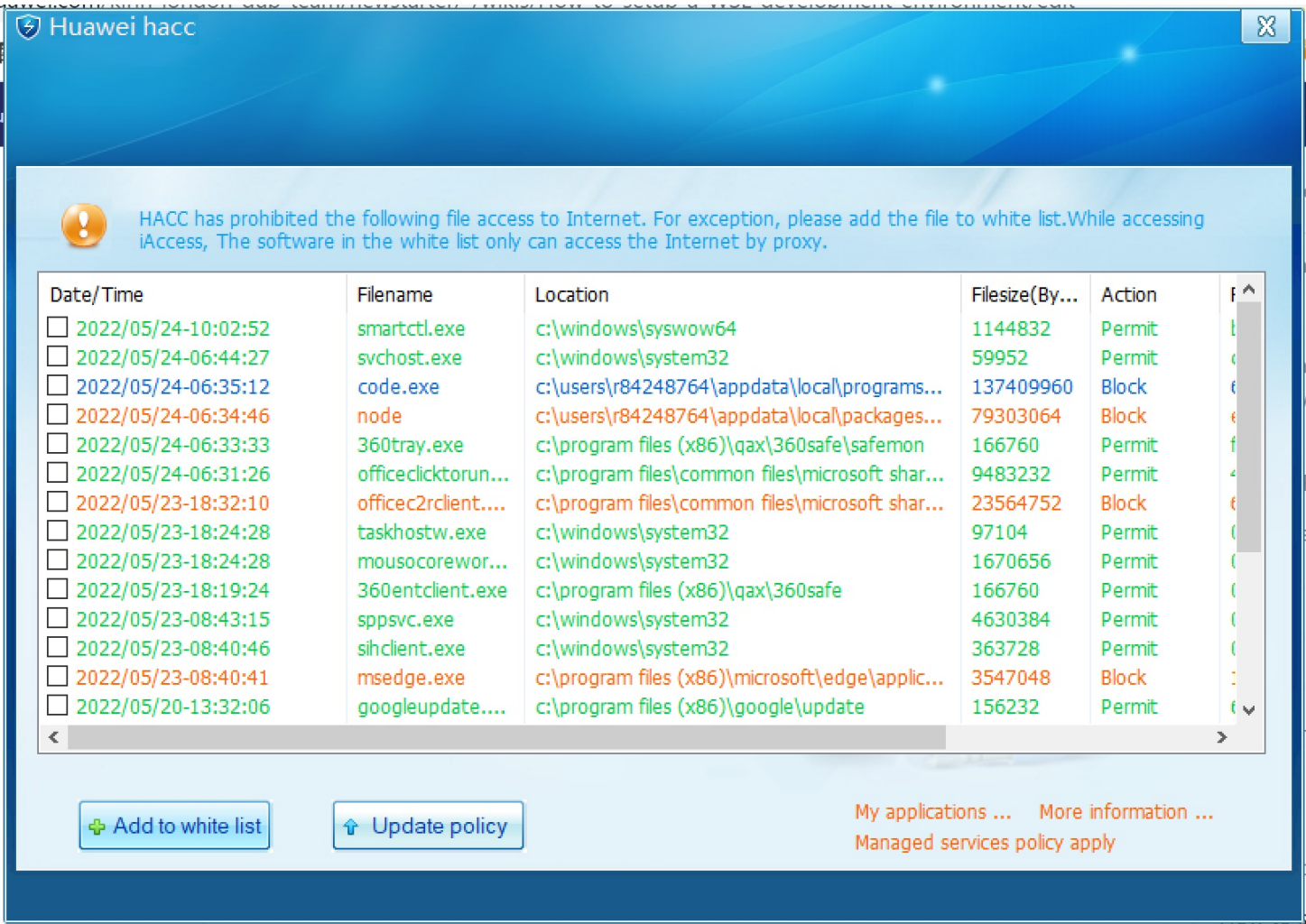
[INFO] logs are written into /tmp/tmp.DuOUGUtN1Y/runlog.20220524-10:02:03
[INFO] files affected are backuped in /tmp/tmp.DuOUGUtN1Y/backup.20220524-10:02:03

[INFO] check whether cntlm have already been installed ... [NO], start to install ... [OK.]

now configure cntlm, here is the available list of proxy servers:
[ 0] General: openproxy.huawei.com
[ 1] China: proxycn2.huawei.com
[ 2] HK: hkgmwg03-ga.huawei.com
[ 3] EU: proxyeu.huawei.com
[ 4] Canada: proxycan.huawei.com
[ 5] Japan: proxyjp.huawei.com
[ 6] US: proxyus.huawei.com
[ 7] US: proxyus2.huawei.com
[ 8] US: proxyus3.huawei.com
[ 9] UK: proxyuk.huawei.com
[ i] Input your proxy
PLEASE SELECT ONE OF PROXY SERVERS BY INDEX[0-9]: 1
proxyuk.huawei.com_
```

If the process fails (for example you may see a message about 'iSafe' blocking your Internet access) you will need to follow the following steps and repeat all the above steps again:

1. Open SPES
2. Go to the Security tab
3. Click to open the Host Firewall view (option on the right hand side of SPES)
4. Click on "Update policy" button
5. Open a powershell window in Admin mode and run the following commands.



**NOTE:** At the end of this list you must restart your computer and repeat the steps at the top of this section again


```
wsl --shutdown
netsh int ip reset all
netsh winhttp reset proxy
netsh winsock reset
ipconfig /flushdns
reboot
```

To test that you are connected to the Internet via Huawei proxy:

- 1. In WSL use this command: wget <http://archive.ubuntu.com/ubuntu/dists/focal/InRelease>
- 2. This must download the file or your connection is not working

## How to setup your WSL environment to access Huawei internal gitlab

Password-less access from WSL into internal Gitlab servers:

- 1. Generate a new pki RSA key pair on WSL and press enter when asked for password to create them without a password: ssh-keygen -b 2048 -t rsa # place them under ~/.ssh/
- 2. Cat the public part: cat ~/.ssh/id\_rsa.pub
- 3. Go to gitlab's SSH Key preferences page: <https://gitlab-uk.rnd.huawei.com/-/profile/keys>
- 4. Copy and paste the public key from 'ssh-rsa' all the way to the last character onto the SSH Key page
- 5. Give it a title and a deadline date and press on Add Key button
- 6. On the WSL window, add this new key to your ~/.ssh/config file; You may use  [this attached config file](#) as an example.
- 7. Now when using git your WSL must try these keys to login to the gitlab servers without requiring password

**TODO:** Show how to update the apt repository and install git and cmake on WSL

## Enabling HTTPS access for Git in WSL (Ubuntu 18.04), Cygwin and MSYS2

To get projects from sites similar to github.com anonymously, you will require Git to access the HTTPS protocol.

Unfortunately, WSL 1, when used together with a Ubuntu 18.04 image, may be missing some required system libraries to allow Git accessing HTTPS protocols.



If you try you may see errors such as the one below:

```
$ git clone https://github.com/glfw/glfw.git

Cloning into 'glfw'...
git-remote-https: /usr/lib/x86_64-linux-gnu/libcurl.so.4: version CURL_OPENSSL_3 not found (required by /usr/lib/x86_64-linux-gnu/libcurl.so.4)
```



Follow these steps to fix this problem:


1. Download the .deb packages compatible with your Ubuntu image: [libcurl3 7.58.0-2ubuntu2 amd64.deb](#) and [libssl1.0.0 1.0.2n-1ubuntu5 amd64.deb](#)
2. In case the above links are missing you can use the attached files:  [libcurl3 7.58.0-2ubuntu2 amd64.deb](#) and  [libssl1.0.0 1.0.2n-1ubuntu5 amd64.deb](#)
3. Copy them over to a directory inside WSL such as ~/ (From Windows: \wsl\$\Ubuntu-20.04\home\YOUR\_USERNAME)
4. Open a WSL window and go to where you copied the files: cd ~/
5. Unpack them into separate directories: dpkg-deb -R libcurl3\_7.58.0-2ubuntu2\_amd64.deb ./libcurl3; dpkg-deb -R libssl1.0.0\_1.0.2n-1ubuntu5\_amd64.deb ./libssl100;
6. copy the required libraries to your system: sudo cp ./libcurl3/usr/lib/x86\_64-linux-gnu/libcurl.so.4.5.0 /lib/x86\_64-linux-gnu/; sudo cp ./libssl100/usr/lib/x86\_64-linux-gnu/\* /lib/x86\_64-linux-gnu/;
7. Make sure your system points to libcurl.so.4.5.0: sudo rm /lib/x86\_64-linux-gnu/libcurl.so.4; sudo ln -s /lib/x86\_64-linux-gnu/libcurl.so.4.5.0 /lib/x86\_64-linux-gnu/libcurl.so.4
8. Ensure the soft link was created correctly: ls -la /lib/x86\_64-linux-gnu/libcurl.so.4
9. Use this argument when invoking git: -c http.sslVerify=false

On **Cygwin**, you will only need to install git and make sure the proxy is used correctly. Here are some examples:

```
$ # This will ask your Huawei corporate password:
$ curl -v --proxy proxyuk.huawei.com:8080 --proxy-user <YOU_USER_ID> google.com
```

```
$ # This will ask your Huawei corporate password:
$ git -c http.sslVerify=false -c http.proxy=<YOUR_USER_ID>@proxyuk.huawei.com:8080 clone https://git
```

On **MSYS2**, there is a similar requirement for using the Internet and accessing HTTPS protocol:

1. First, ensure you export proxy environment variables: export HTTP\_PROXY="<YOUR\_USER\_ID>@proxyuk.huawei.com:8080" && export HTTPS\_PROXY=\$HTTP\_PROXY && export http\_proxy=\$HTTP\_PROXY && export https\_proxy=\$HTTP\_PROXY
2. Update your ssl certificates on MSYS2. Follow instructions in  [this PDF file](#) or on this Stackoverflow page: <https://stackoverflow.com/questions/69348953/certificate-error-when-trying-to-install-msys2-packages-on-windows-server>
3. Test proxy settings. This should work: curl google.com
4. Now you can access the Internet: pacman -Syu mingw-w64-x86\_64-toolchain
5. If iSafe or firewall blocks pacman then please follow the above 'Update policy' steps to request it to be unlocked and try the above step again

**IMPORTANT NOTE:** Your system might reset the soft link /lib/x86\_64-linux-gnu/libcurl.so.4 in some cases - such as when using apt install

**NOTE:** You may be choose to make this ssl git argument permanent: git config http.sslVerify "false"

## Installing and running Visual Studio Code on WSL

**IMPORTANT NOTE:** Do *NOT* install the any of the required Extensions directly on a Windows instance of VSCode. Please remove these extensions if they are installed before the following steps. This is important!

**IMPORTANT NOTE:** You *MUST* have established an Internet connection via Huawei proxy for these steps to work

1. Find a [VSCode installable download](#) and install it on your Windows
2. Open a WSL window and ensure you do have an Internet connection first
3. Create a test directory: mkdir ~/test && cd ~/test
4. Open VSCode from inside WSL now by typing: code . # Notice the dot
5. In VSCode, first install the "Remote Development" Extension by Microsoft
6. Once this step is done, close VSCode and re-open it in WSL: code .
7. Ensure you see the "WSL: Ubuntu20.04" green marker at the bottom left side of the VSCode window (see the screenshot at the top of this page)
8. Now install the following Extensions only and only by clicking on "Install in WSL: Ubuntu 20.04": "C/C++" by Microsoft, and "C/C++ Extension Pack" by Microsoft, optionally "GitLens"
9. Close and re-open VSCode from inside WSL: code .


## Building and setting up your code-base for VSCode's C/C++ Extension

1. Close VSCode instances
2. Open a WSL window

3. Clone the project that yo want in a directory such as ~/work/: cd ~/work/ && git clone <https://gitlab-uk.rnd.huawei.com/kirin-london-gup-team/v300/-/tree/master/simulator>
4. Go to the cloned repository's directory: cd ~/work/v300/
5. Open VSCode from here inside WSL: code .
6. This way you should be able to use VSCode to edit your code

## Debugging your code using VSCode and WSL

---

1. Go to where you want to start a build: cd ~/work/v300/simulator
2. Create a build directory: mkdir build && cd build
3. Enable DEBUG symbols when using cmake: cmake -DENABLE\_ALL\_TESTS=ON -DENABLE\_ZIP=ON -DCMAKE\_BUILD\_TYPE=Debug ..
4. Build your project: make -j4
5. Make sure that your project has built properly and runs on the command line independently of VSCode.
6. Go back to the project root: cd ~/work/v300/
7. Open VSCode from inside WSL from inside your project root: cd ~/work/v300/ && code .
8. Create a new launch file: ~/work/v300/.vscode/launch.json
9. You may use  [the attached file](#) to fill the content of the launch file but make sure you replace the phrase "YOUR\_USER" with your correct user-name and fix the paths to match your environment
10. Ensure that the arguments being passed to your executable are correct inside the launch file
11. Your project is now ready to be debugged (including breakpoints and step through) from inside the WSL instance of VSCode - just open the "Run and Debug" tab from the tabs on the left side of VSCode and find the "Launch" option from the drop down list at the top of the "Run and Debug" side panel

Your debugger would usually stop at the entry point (main function). Alternatively, you may test by placing break points and running your code.