THE C-LINUX PROGRAMMING ENVIRONMENT: INSTALLATION AND HOW TO USE

Updated:

8/26/2020, by Yong Zhang

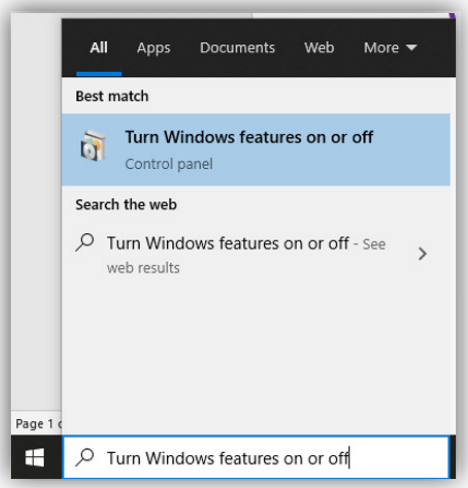
Requirements:

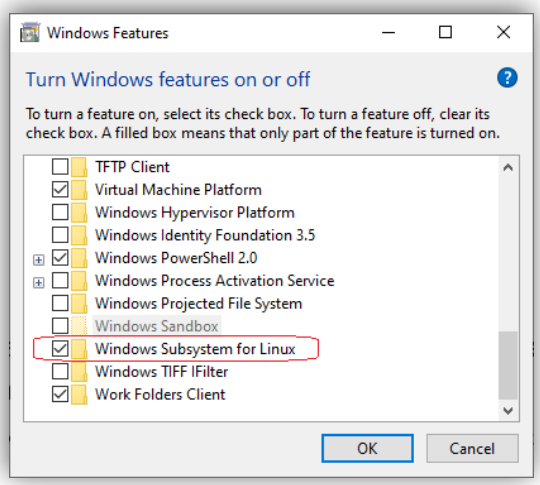
- You will need a x86 PC running Windows 10, or a PC running Linux or MacOS
- Windows 10 needs to be updated to include the Windows 10 Fall Creators update, released October 2017. This update includes the **Windows Subsystem for Linux** which is needed to run the Ubuntu terminal.

Installation: Windows 10 Instructions

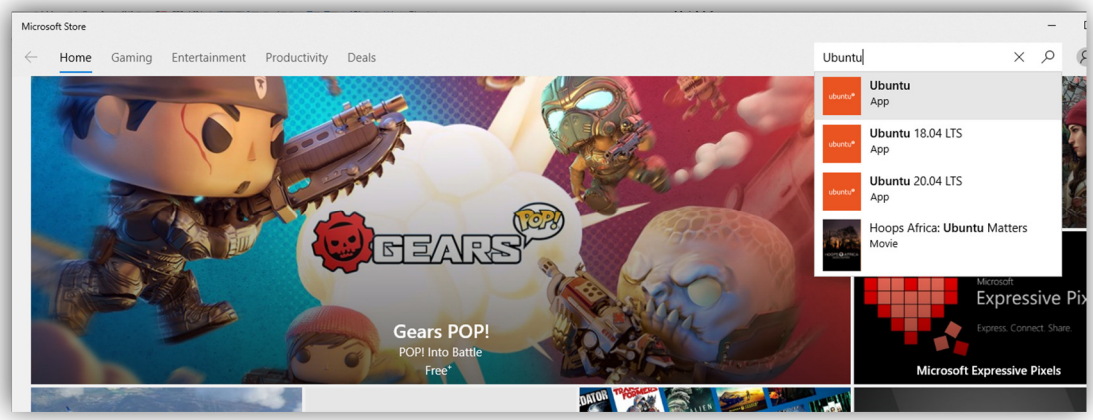
Installing Windows Subsystem for Linux (WSL)

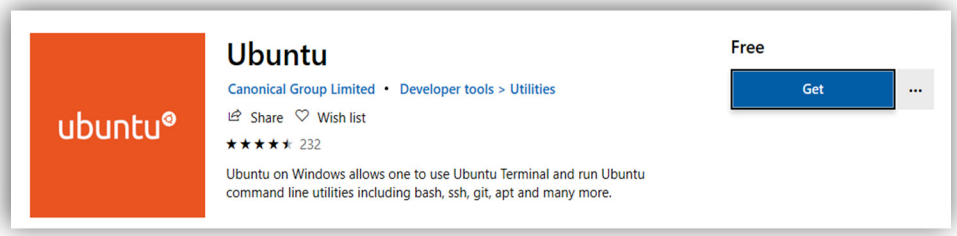
- Besides Windows Start button, use the search box, search for "**Turn Windows features on or off"**, click the match result to open the "Turn Windows features on or off" setting in Control Panel.
- Scroll down until you see **Windows Subsystem for Linux**. Check the box next to it and click OK. This might take a few minutes.

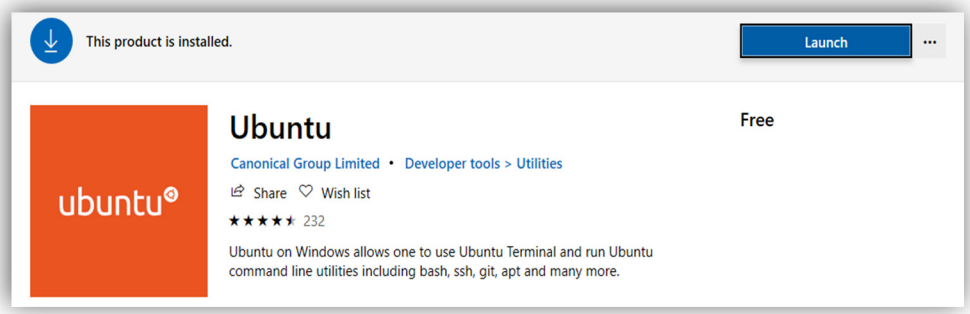




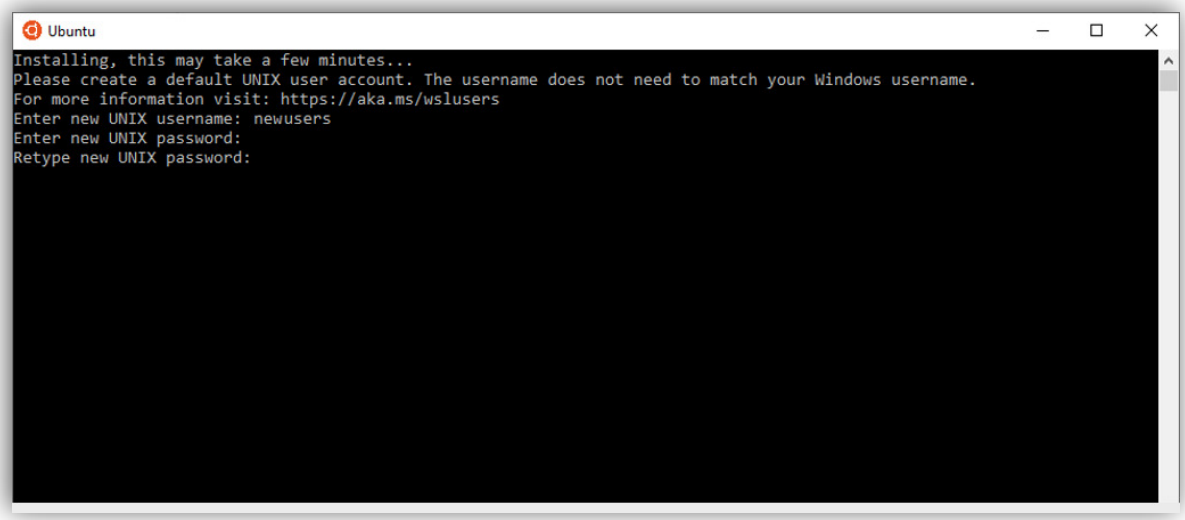
- Open the **Microsoft Store**. You can do that by clicking the Start button and locate Microsoft Store from the list of applications, or use the search box and type **Store** to find it.
- Inside Microsoft Store, search for **Ubuntu** and then select the **Ubuntu** app from the result list.
- Click the **Get** button and wait for it to download. You might be asked to create a free Microsoft account or login with one.
- Click the Launch button to open up the Ubuntu Subsystem app and finish installing it.

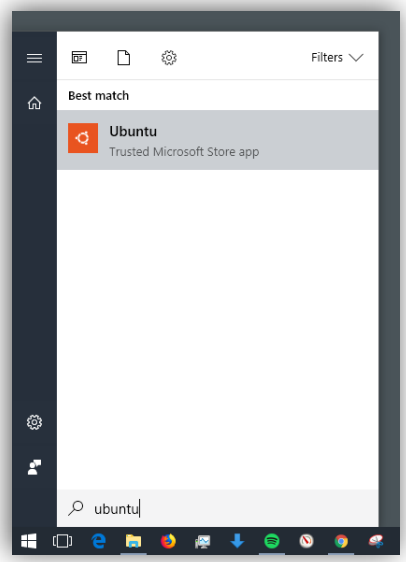


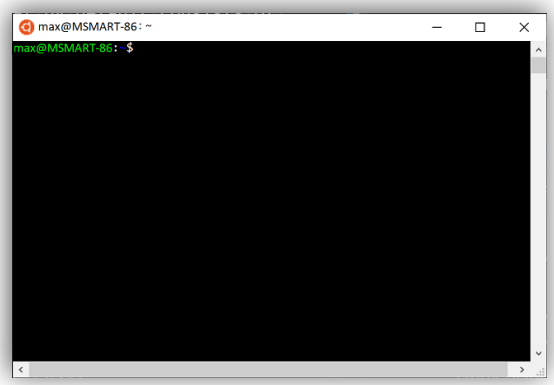




- Ubuntu requires you to create a user account. Enter the username (no spaces) and password (with confirmation) of this account. This is a brand-new account you are creating, so please select a set of new username and password for it. Remember this password. It's OK when nothing shows on the screen when you type the password; just keep typing it and then press ENTER when you are done.
- Ubuntu can now be launched in the same way as any other Windows 10 application, such as searching for and selecting **Ubuntu** in the Start menu. Once your click the Ubuntu application, it will open the **Bash shell for WSL**. It is a command-line based user interface, just like Windows command prompt. The shell appears with a command prompt that by default consists of your user name and computer name, and puts you in your home directory (see the following screenshots for how it looks like)
- If you want to find out more, please visit this link: https://ubuntu.com/tutorials/ubuntu-on-windows


Ubuntu
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: newusers
Enter new UNIX password:
Retype new UNIX password:


Filters
Best match
Ubuntu
Trusted Microsoft Store app
ubuntu


max@MSMART-86: ~
max@MSMART-86: $

Installing Visual Studio Code (VS Code)

- Visit the official Visual Studio Code website: https://code.visualstudio.com/download, download the installer for Windows 10, and install it on the Windows side (not in WSL). This is the code editor for editing C programs for this class. **Note:** When prompted to **Select Additional Tasks** during installation, be sure to check the **Add to PATH** option so you can easily open a folder in WSL using the `code` command.
- Install the **Remote Development extension**. This extension pack allows you to open any folder in a container, on a remote machine, or in the WSL and take advantage of VS Code's full feature set. Visit its official website, click the **Install** button, it will open the installation link with VS Code. Just follow the instructions to install it inside VS Code:

  https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.vscode-remote-extensionpack

- Install the **Remote-WSL extension**. Remote-WSL is an extension to VS Code to let you use VS Code on Windows to build Linux applications that run on the WSL. To install it, visit the official website, click Install button and it will open the installation link with VS Code. Just follow the instructions to install it inside VS Code:

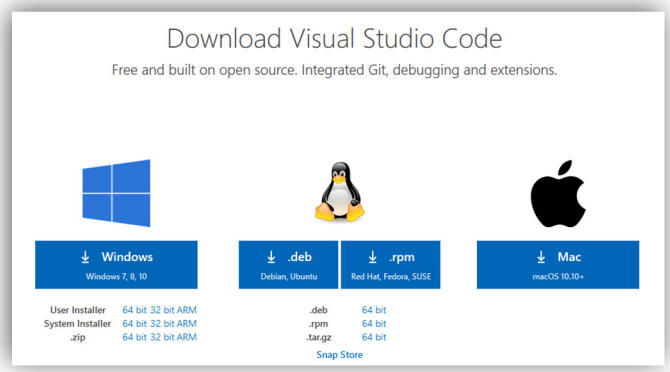
  https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-wsl

Installation: Linux Instructions for VS Code

- Visit the official Visual Studio Code website: https://code.visualstudio.com/docs/setup/linux, on how to install and launch VS Code on Linux.

Installation: MacOS Instructions for VS Code

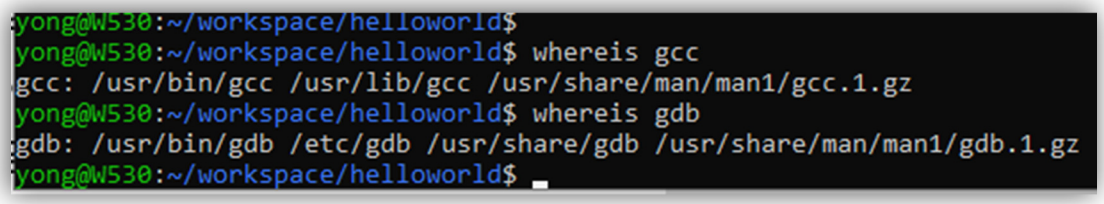
- Visit the official Visual Studio Code website: https://code.visualstudio.com/docs/setup/mac, on how to install and launch VS Code on MacOS.



How to Use: Linux Programming Environment

Setting Up the Linux Environment

- Although you will be using VS Code to edit your source code, you'll be compiling the source code on Linux using the gcc compiler. You'll also debug on Linux using GDB. These tools are not installed by default on Ubuntu, so you have to install them. Fortunately, that task is quite easy! From the WSL command prompt, first run `sudo apt-get update` to update the package lists for Ubuntu. An out-of-date distro can sometimes interfere with attempts to install new packages.
- From the same command prompt, install the GNU compiler tools (gcc, g++, make, etc) and the DGC debugger by typing `sudo apt-get install build-essential gdb` command. When running the commands, Ubuntu may ask your password to verify your account information. Check out this link: https://packages.ubuntu.com/xenial/build-essential, if you need more information about build-essential.
- Verify that the installations are successful by locating gcc and gdb, see examples in the screenshot below. If the filenames are not returned from the `whereis` command, try running the update command again.



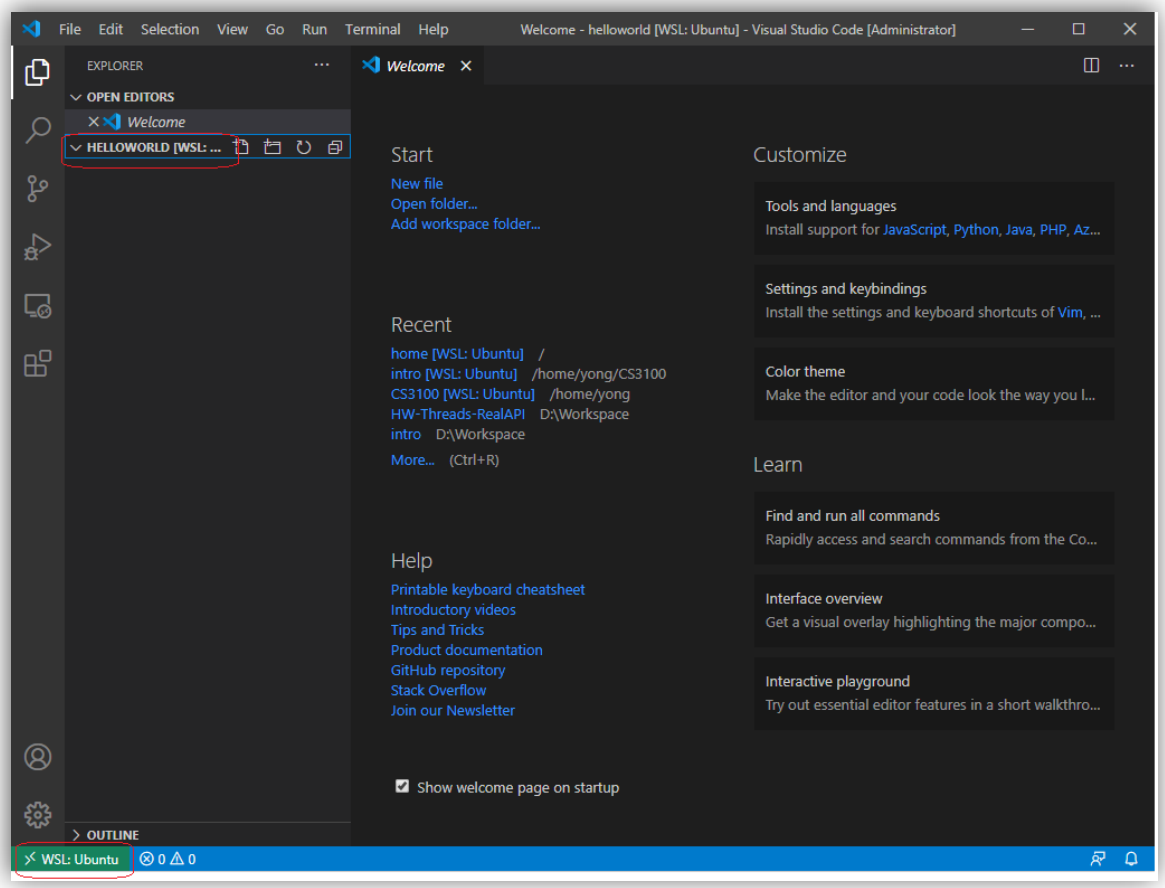
How to Use: VS Code

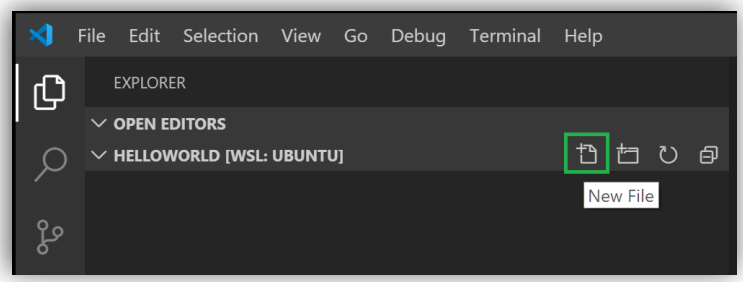
How to Start VS Code with WSL

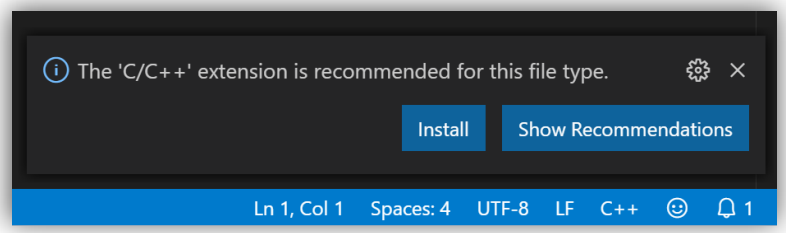
- Start the Ubuntu application to open the Batch shell for WSL. Make a directory **workspace** (or any name you prefer) in your home directory using the command `mkdir workspace`. This is the folder that will contain all of your programming projects for this class. Run the command `cd workspace` to navigate to the workspace directory, then run the command `mkdir helloworld` to create your first C programming project folder. Run the command `cd helloworld` to navigate to the **helloworld** project directory.
- Once you are in the **helloworld** directory, type `code .` command to launch VS Code. The `'.'` argument tells VS Code to open the current folder. You'll see a message about "Installing VS Code Server". VS Code is downloading and installing a small server on the Linux side that the desktop VS Code will then talk to. This is a one-time installation. VS Code will then start and open the **helloWorld** folder. The File Explorer shows that VS Code is now running in the context of WSL with the title bar **[WSL: Ubuntu]** at the bottom left corner of VS Code.

```
yong@W530:~$ ls
CS3100  workspace
yong@W530:~$ cd workspace/
yong@W530:~/workspace$ mkdir helloworld
yong@W530:~/workspace$ cd helloworld
yong@W530:~/workspace/helloworld$ code .
```

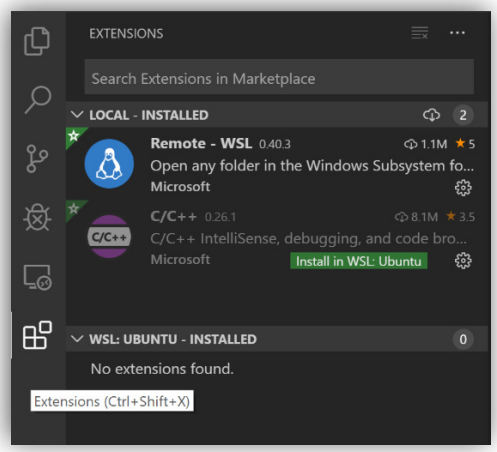
- In the File Explorer title bar, select the **New File** button and name the file `helloworld.c`. Once you create the file and VS Code detects it is a C/C++ language file, you may be prompted to install the Microsoft C/C++ extension if you don't already have it installed. Choose **Install** and then **Reload Required** when the button is displayed in the Extensions view to complete installing the C/C++ extension. If you already have C/C++ language extensions installed locally on Windows 10 in VS Code, you'll need to go to the Extensions view (Ctrl+Shift+X) and install those extensions into WSL. Locally installed extensions can be installed into WSL by selecting the **Install in WSL** button and then **Reload Required**.

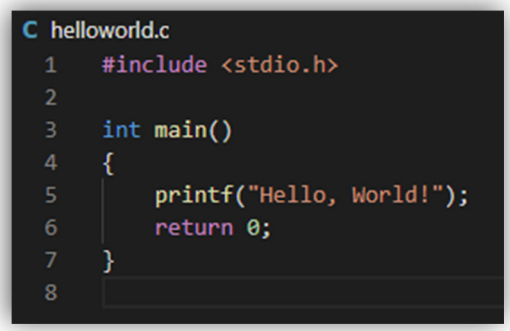


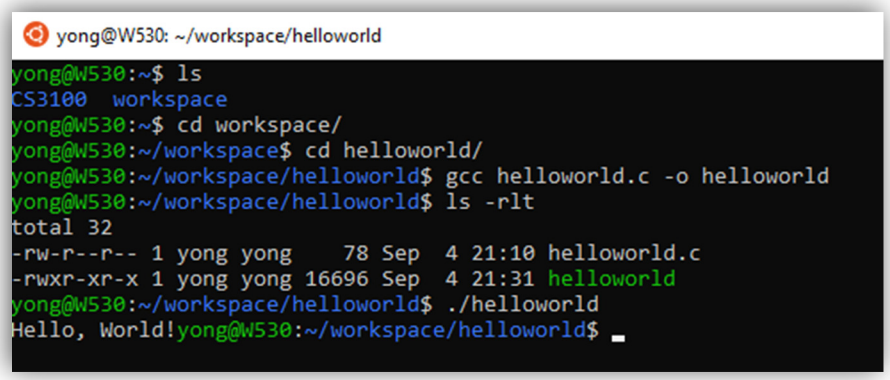




- Add hello world source code (see screenshot below for the code). Press `Ctrl+S` to save the file. You can also enable Auto Save to automatically save your file changes, by checking **Auto Save** in the main **File** menu. To compile the C program, from the WSL command prompt, run `gcc helloworld.c -o helloworld` command. To run the program, from WSL command prompt, use the command `./helloworld`.
- Check out the VS Code documents: https://code.visualstudio.com/docs, on how to use VS Code as your source code editor, especially https://code.visualstudio.com/docs/getstarted/introvideos.





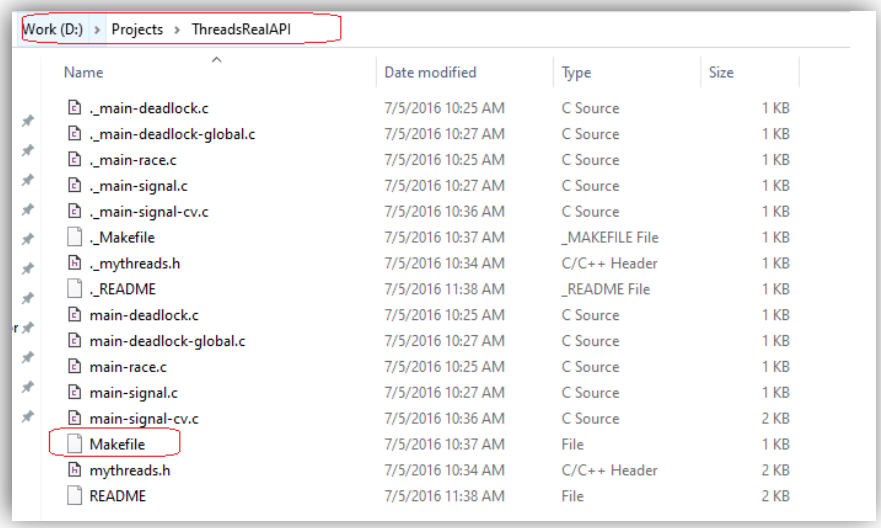


How to Close WSL Session from VS Code

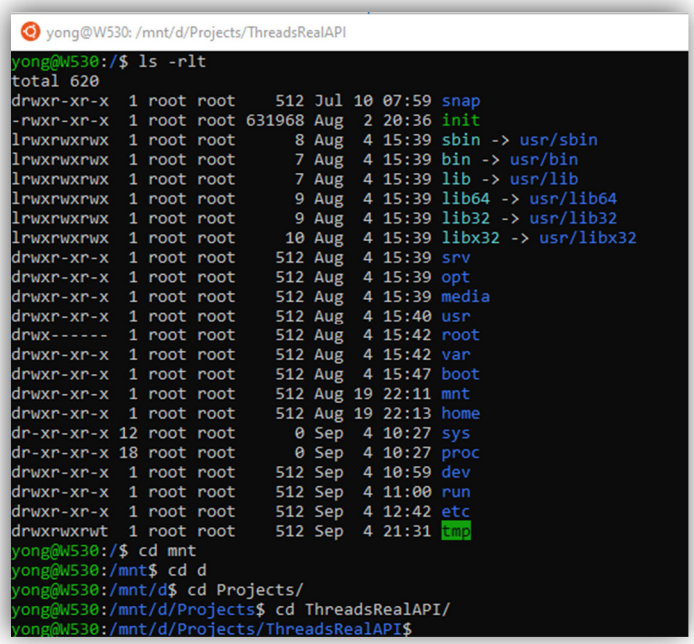
- When you are done working in WSL, you can close your remote session with the **Close Remote Connection** command available in the main **File** menu or the Command Palette (Ctrl+Shift+P). This will restart VS Code running locally. You can easily reopen your WSL session from the **File** > **Open Recent** list by selecting folders with the **[WSL]** suffix.

How to Use VS Code and Makefile

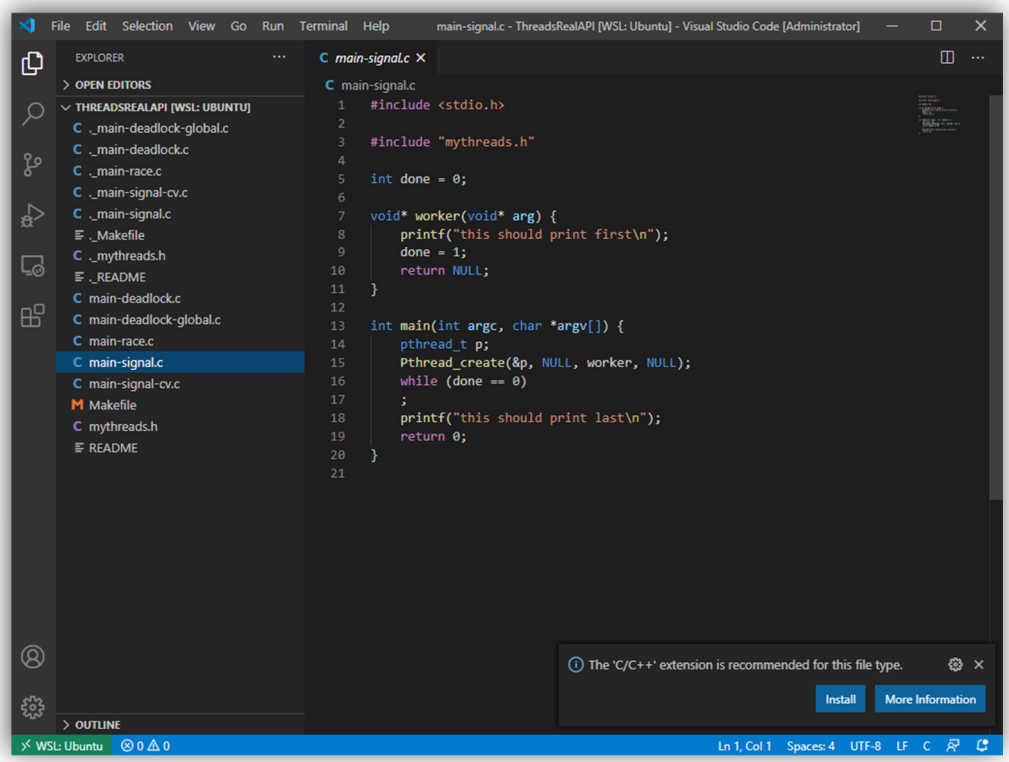
- If your C project uses `Makefile` to organize code compilation, you may use the `make` command to compile all the C source code in the project. On Windows 10, Save the project files to your project folder, such as D:\projects\ThreadRealAPI. Note: your project should have a file named `Makefile`.



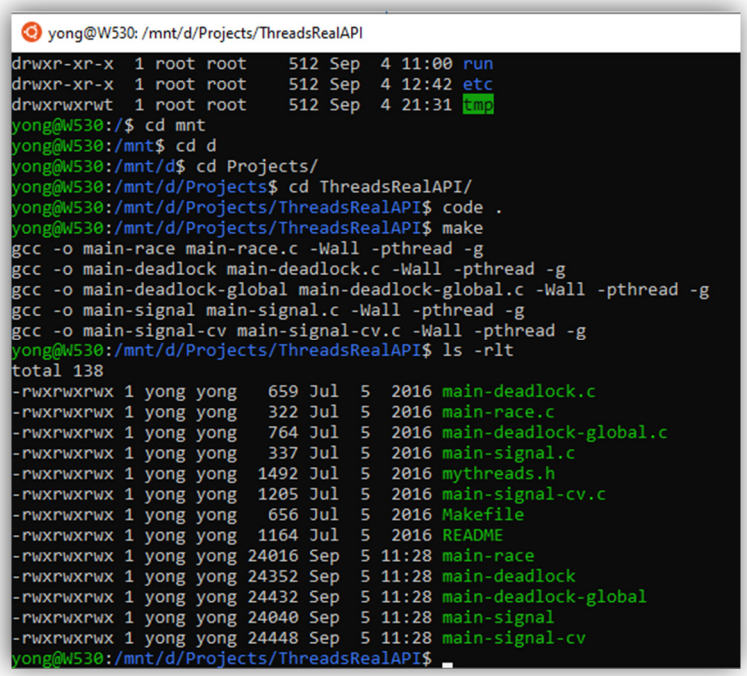
- From WSL command prompt, navigate to the folder /mnt/d/projects/ ThreadRealAPI. This Linux folder corresponds to the folder D:\projects\ ThreadRealAPI in your Windows 10 environment.

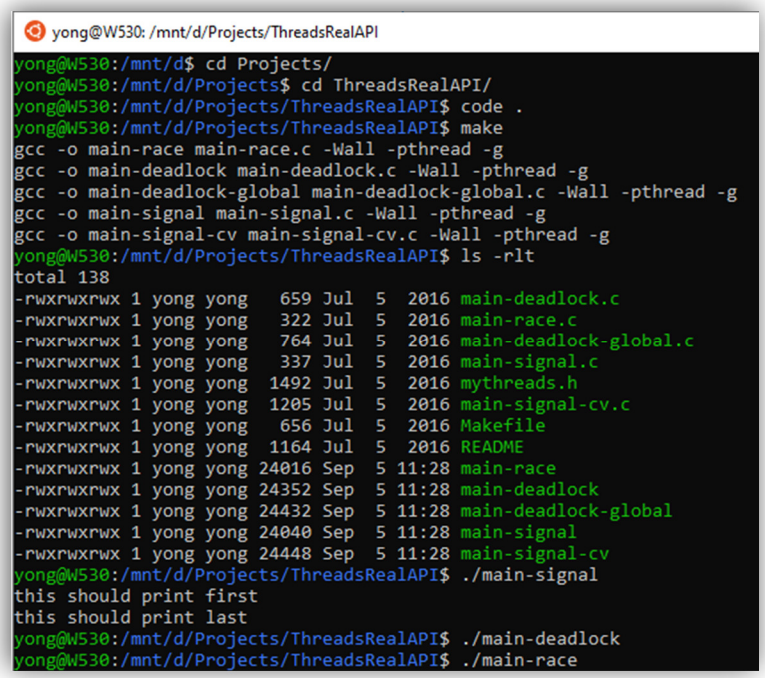


- In WSL command prompt, run `code .` command from this folder to open the project with VS Code, you should see all the C programs from the left file explorer pane. You may view the code by clicking a C program from the left pane, then you will see the source code open on the right editing pane. You may edit the code using the editing pane. Press `Ctrl+S` to save the changes.



- To compile the programs, run `make` command from the WSL command prompt. **Note:** this command will compile all the C programs in the whole project. For this project, you should see that five executable files are created.
- To run a program, from the WSL command prompt, run `./programName`, for example, you may run the main-signal program by executing the command `./main-signal`.





How to Use: Working with GitHub in VS Code

Getting started with GitHub Pull Requests and Issues

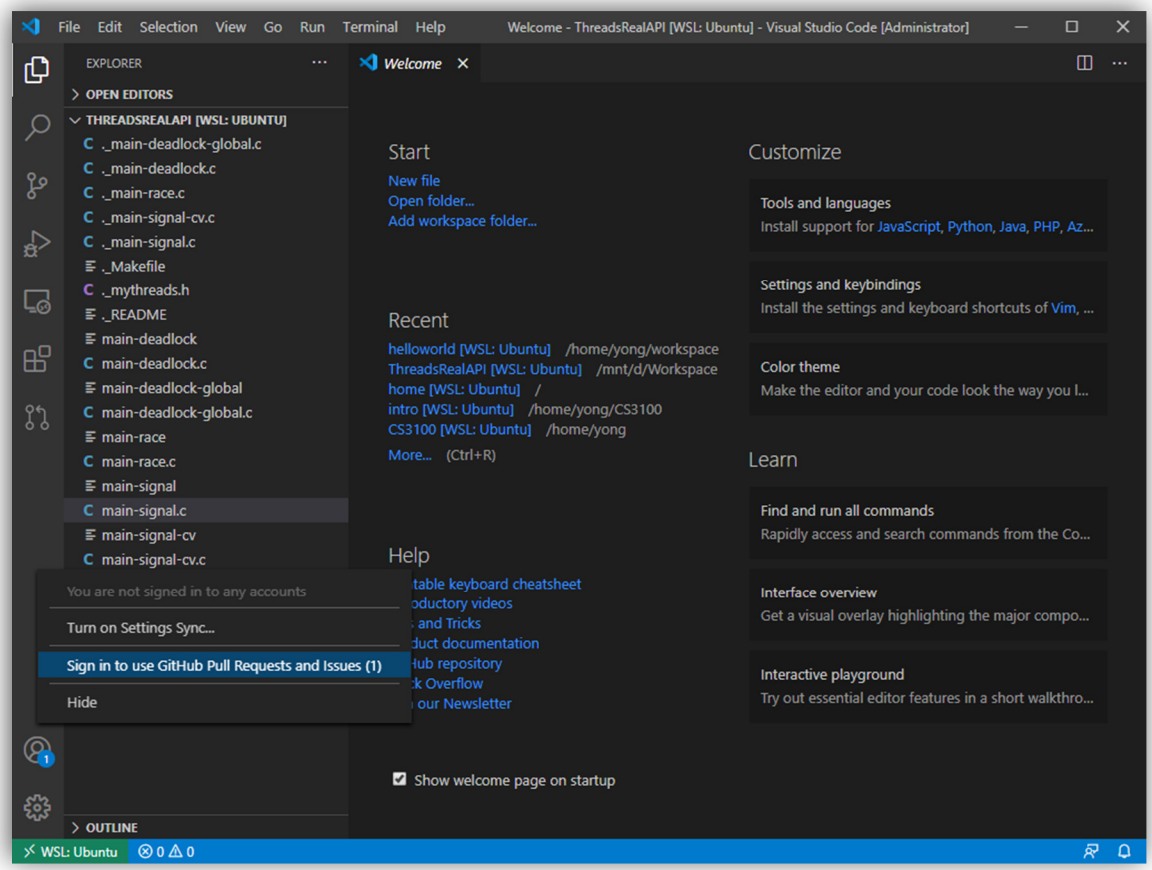
- Create a free account with GitHub if you do not have an account with GitHub:

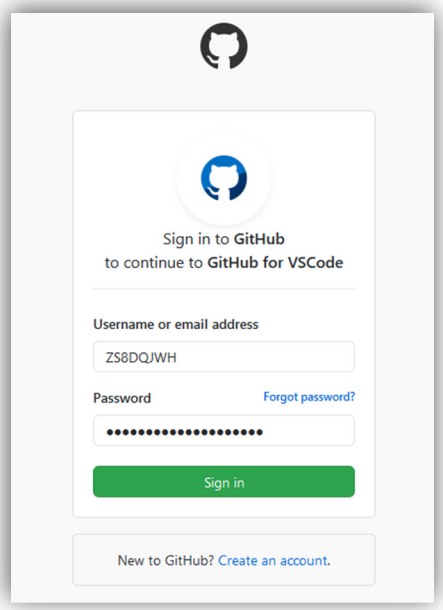
  https://docs.github.com/en/github/getting-started-with-github/signing-up-for-a-new-github-account

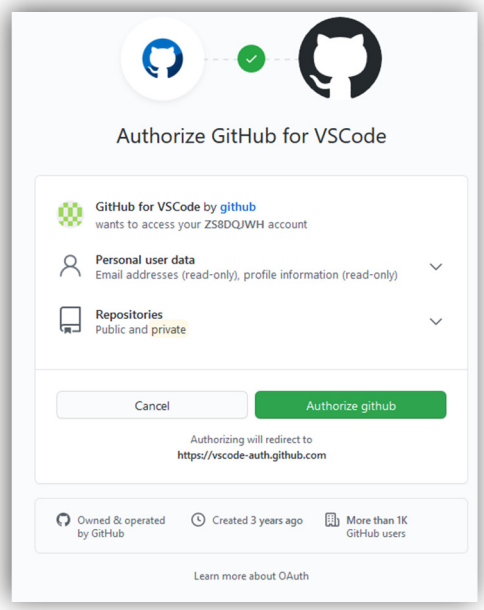
- Install the **GitHub Pull Requests and Issues extension** to VS Code:

  https://marketplace.visualstudio.com/items?itemName=GitHub.vscode-pull-request-github

- Sign in from VS Code by clicking the bottom-left account button, this will open the authentication web page for GitHub in the browser to authorize github for VS Code. Note: if an error occurred and you received error code 801 with Firefox browser, you may try to use Google Chrome browser to sign in. Once it succeeds, you will be redirected to VS Code. If you aren't redirected, you can add the token manually by following the instructions given by the authorization webpage.

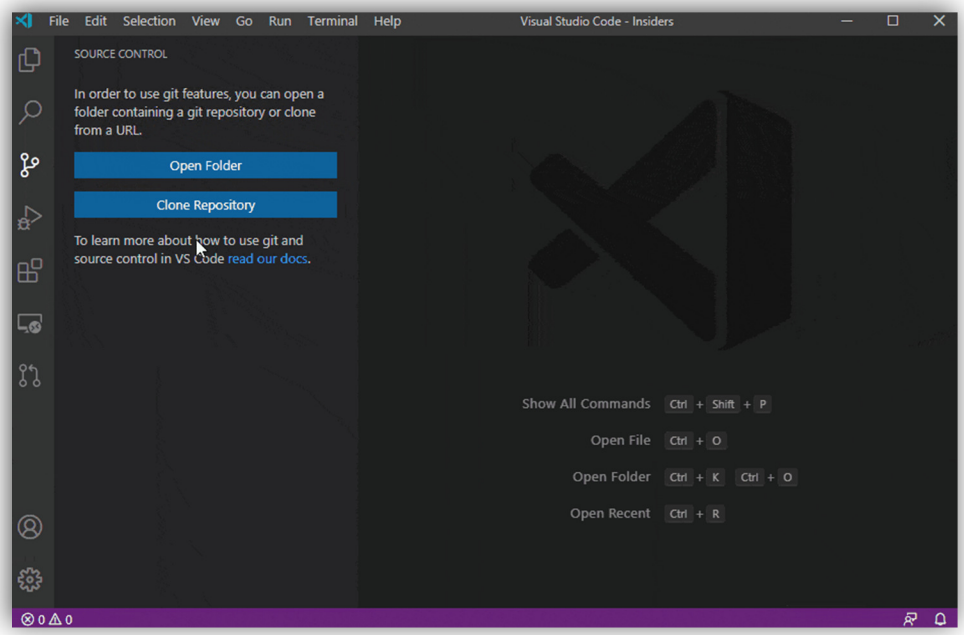




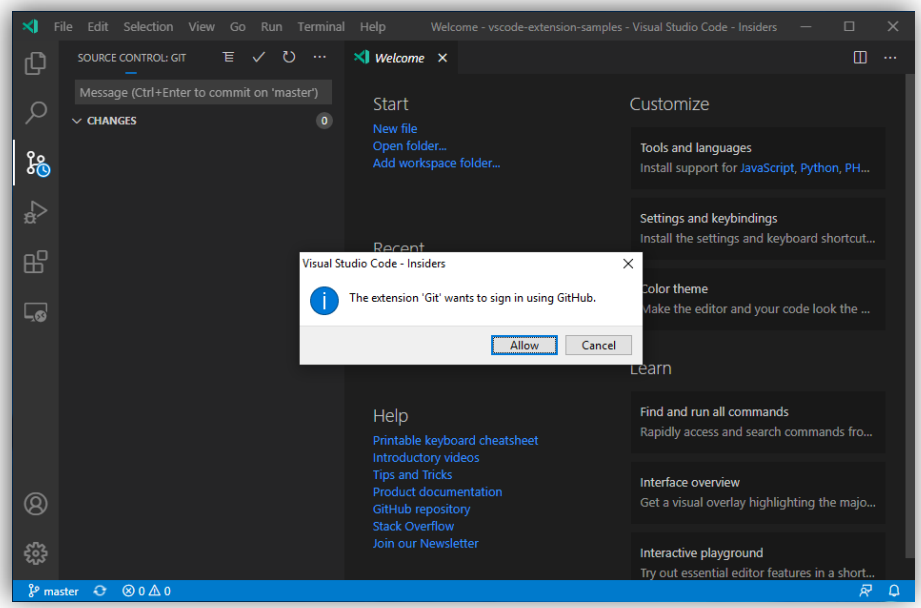


Setting up a repository

- Cloning a repository in VS Code: you can search for and clone a repository from GitHub using the **Git: Clone** command in the Command Palette (Ctrl+Shift+P) or by using the **Clone Repository** button in the Source Control view (available when you have no folder open).

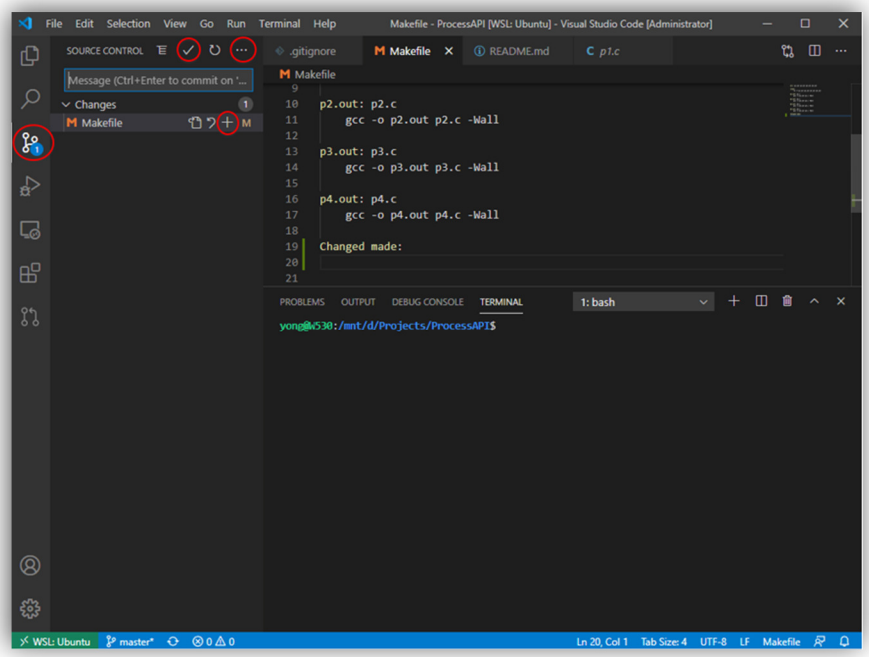


- Authenticating with an existing repository: enabling authentication through GitHub happens when you run any Git action in VS Code that requires GitHub authentication, such as pushing to a repository that you're a member of or cloning a private repository. You don't need to have any special extensions installed for authentication; it is built into VS Code so that you can efficiently manage your repository. When you do something that requires GitHub authentication, you'll see a prompt to sign in. Follow the steps to sign into GitHub and return to VS Code. Note that there are several ways to authenticate to GitHub, including using your username and password with two-factor authentication (2FA), a personal access token, or an SSH key.



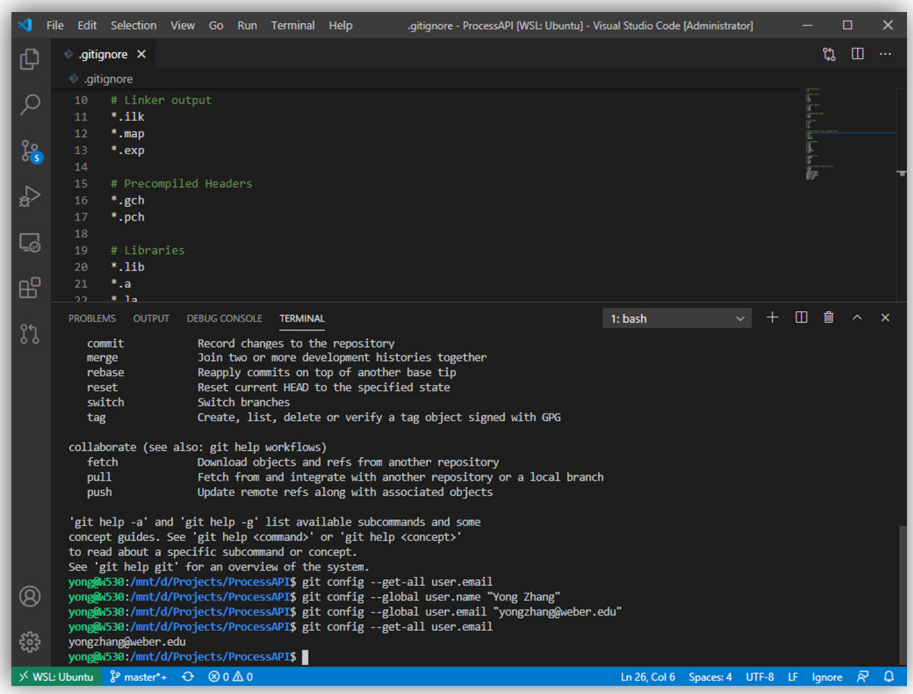
Stage, Commit and Push

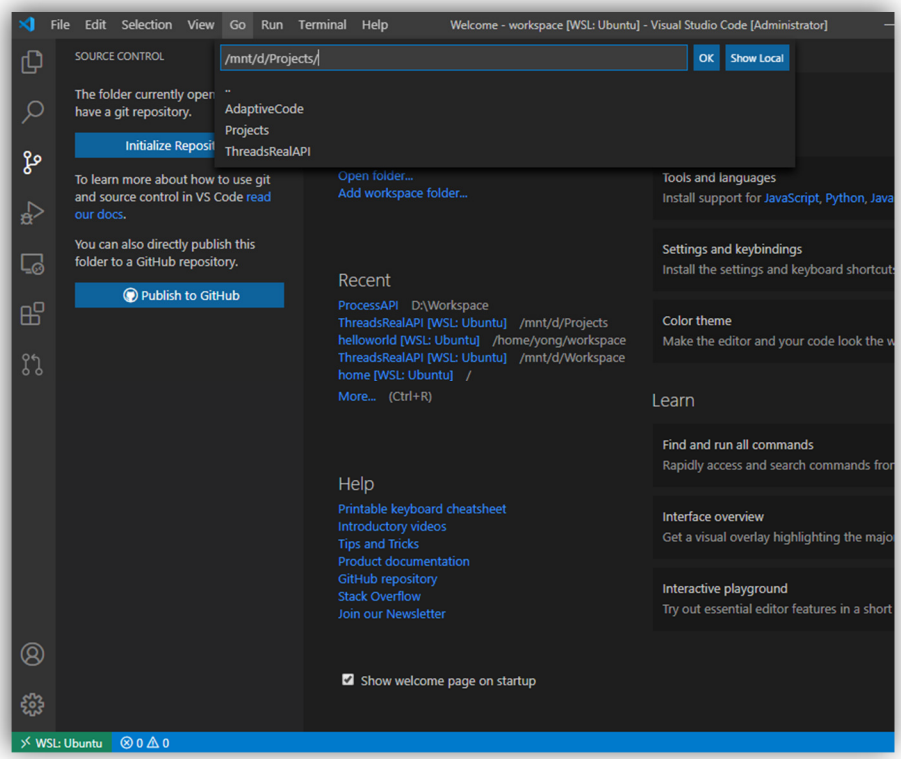
- If you are ready to share any changes you made to the central repository, you may do that in VS Code: click the source control icon on the left, click "+" to the right of the changed files to stage the change, then press the correct mark symbol to commit; lastly, press the "…" to find push command to push the changes to the central repository.

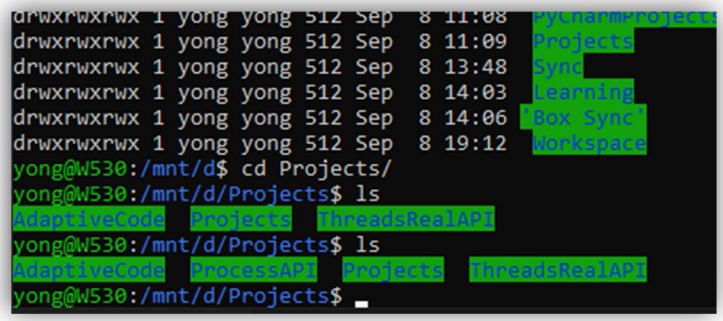


- If the git complains about your not configuring your user.name and user.email, open a terminal from within VS Code, run the following two commands, then the problem should be resolved.

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```




File Edit Selection View Go Run Terminal Help
Welcome - workspace [WSL: Ubuntu] - Visual Studio Code [Administrator]
SOURCE CONTROL
/mnt/d/Projects/
OK
Show Local
..
AdaptiveCode
Projects
ThreadsRealAPI
The folder currently open
have a git repository.
Initialize Reposit
To learn more about how to use git
and source control in VS Code read
our docs.
You can also directly publish this
folder to a GitHub repository.
Publish to GitHub
Open folder...
Add workspace folder...
Recent
ProcessAPI D:\Workspace
ThreadsRealAPI [WSL: Ubuntu] /mnt/d/Projects
helloworld [WSL: Ubuntu] /home/yong/workspace
ThreadsRealAPI [WSL: Ubuntu] /mnt/d/Workspace
home [WSL: Ubuntu] /
More... (Ctrl+R)
Help
Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation
GitHub repository
Stack Overflow
Join our Newsletter
Tools and languages
Install support for JavaScript, Python, Java
Settings and keybindings
Install the settings and keyboard shortcuts
Color theme
Make the editor and your code look the w
Learn
Find and run all commands
Rapidly access and search commands fron
Interface overview
Get a visual overlay highlighting the majo
Interactive playground
Try out essential editor features in a short
Show welcome page on startup
WSL: Ubuntu
0 0


drwxrwxrwx 1 yong yong 512 Sep 8 11:08 PycharmProjects
drwxrwxrwx 1 yong yong 512 Sep 8 11:09 Projects
drwxrwxrwx 1 yong yong 512 Sep 8 13:48 Sync
drwxrwxrwx 1 yong yong 512 Sep 8 14:03 Learning
drwxrwxrwx 1 yong yong 512 Sep 8 14:06 'Box Sync'
drwxrwxrwx 1 yong yong 512 Sep 8 19:12 Workspace
yong@W530:/mnt/d$ cd Projects/
yong@W530:/mnt/d/Projects$ ls
AdaptiveCode Projects ThreadsRealAPI
yong@W530:/mnt/d/Projects$ ls
AdaptiveCode ProcessAPI Projects ThreadsRealAPI
yong@W530:/mnt/d/Projects$

```
yong@W530:/mnt/d/Projects$ ls
AdaptiveCode  ProcessAPI  Projects  ThreadsRealAPI
yong@W530:/mnt/d/Projects$ cd ProcessAPI/
yong@W530:/mnt/d/Projects/ProcessAPI$ ls
Makefile  README.md  p1.c  p2.c  p3.c  p4.c
yong@W530:/mnt/d/Projects/ProcessAPI$ ls -rlt
total 10
-rwxrwxrwx 1 yong yong 174 Sep  8 19:17 Makefile
-rwxrwxrwx 1 yong yong 359 Sep  8 19:17 README.md
-rwxrwxrwx 1 yong yong 572 Sep  8 19:17 p1.c
-rwxrwxrwx 1 yong yong 646 Sep  8 19:17 p2.c
-rwxrwxrwx 1 yong yong 966 Sep  8 19:17 p3.c
-rwxrwxrwx 1 yong yong 884 Sep  8 19:17 p4.c
yong@W530:/mnt/d/Projects/ProcessAPI$ make
gcc -o p1 p1.c -Wall
gcc -o p2 p2.c -Wall
gcc -o p3 p3.c -Wall
gcc -o p4 p4.c -Wall
yong@W530:/mnt/d/Projects/ProcessAPI$ ls
Makefile  README.md  p1  p1.c  p2  p2.c  p3  p3.c  p4  p4.c
yong@W530:/mnt/d/Projects/ProcessAPI$ ./p1
hello world (pid:458)
hello, I am parent of 459 (pid:458)
hello, I am child (pid:459)
yong@W530:/mnt/d/Projects/ProcessAPI$
```

Pull requests

- From the **Pull Requests** view you can view, manage, and create pull requests.