Resources: Using the Windows Subsystem for Linux with a Windows Editor

Edit in Windows, Build and Run in Bash

You should use a good editor to create code. By that, I mean you should use an editor that allows you to focus on the code you want to create, without getting in your way.

Let’s say that editor is inside of Visual Studio, Sublime Text or Atom in the Windows OS. Meanwhile, you need to build and run your program in the Windows Subsystem for Linux.

Basic Ideas

- Choose a location in your Windows system to store the project.
- Make that location accessible to the bash prompt.
- Edit files using your editor in Windows.
- Build and run in the bash prompt.

Choose a location in your Windows system to store the project

For most people, this is in Documents or some other folder where they naturally put most files. It’s important to find the full path of the top-level folder. For example, it might be `C:\Users\fred\Documents\Projects\RubiksCube`.

Make that location accessible to the bash prompt

Open the bash prompt. Put a symbolic link to the project location, so you don’t have to type the long path all of the time.

```bash
ln -s '/mnt/c/Users/fred/Documents/Projects/RubiksCube' './RubiksCube'
```

The quotes are important if you have any spaces or special characters in the path name. The only problem is if you have quotes in your path. At this point, you should be able to open a bash prompt and issue `cd RubiksCube` to enter the work area. Try the `ls` command to list the contents of the folder.

```bash
  cd RubiksCube
  ls
```

Edit files using your editor in Windows

Edit/Create/Delete files in your project using your Windows editor.

Build and run in the bash prompt

In your bash prompt, `cd` into the project folder, as listed above. If you’re already in the directory, no need to do it again. Each bash prompt has its own current working directory, and they start at your bash home directory.

Assuming you’ve built a `Makefile` with correct project information, you should be able to build the project with:

```bash
  make
```

If the build is successful and builds the desired program, you should be able to run it here as well.

```bash
  ./rubiks_test
```

Where `rubiks_test` should be the name of the program you built with the `make` command.