Windows Subsystem for Linux (WSL)

Intro. to Computing

`https://mandarm.github.io/computing.html`

Installation

- Open Windows PowerShell (WPS).
- Run `wsl --install`.
  - If/when it says `Downloading: Ubuntu`, type Control-C to abort.
  - Otherwise, let it install the Virtual Machine Platform.
- Run `wsl --update`.
  (should say most recent version is already installed)
- If you do not already have a copy of
  `https://releases.ubuntu.com/26.04/ubuntu-26.04-wsl-amd64.wsl`,
  download it.
- Double-click the file to launch the installer, OR run
  `wsl --install --name ubuntu26 --from-file .\ubuntu-26.04-wsl-amd64.wsl`
- Close the window or type Control-D to exit.
- Run `wsl` in WPS to start WSL again.
- Run `sudo apt update && sudo apt upgrade`.

File locations

- **How do Windows locations appear in WSL and vice versa?**
  - `C:\` ≡ `/mnt/c`
  - `/` ≡ `\\wsl.localhost\Ubuntu-26.04` or `\\wsl$` (in Windows File explorer address bar)
- **General guideline:** for best performance, avoid working across environments (Windows and WSL)
  - When programming in WSL, store files under `/home/student` (or `/home/cssc` ...)
    not `/mnt/c/Users/CSSC/Project` or `C:\Users\CSSC\Project`
  - Exception (for now): edit your Python programs using Windows; run them using WSL.
- **Warning re: file/directory names:**
  Windows → case-insensitive, Linux → case-sensitive

Mixing Windows and WSL commands

- Using Windows executables from WSL: include .exe extension
  Examples:
  ```
  explorer.exe .
  notepad.exe .bashrc
  notepad.exe .bash_aliases
  notepad.exe "C:\temp\foo.txt"
  notepad.exe C:\\temp\\foo.txt
  ```
  BUT `notepad.exe ~/.bash_aliases` **did not work for me!**
- Using Linux programs from Windows: add `wsl` before command name
  Examples:
  ```
  wsl ls -l
  ```

My personal experience: does not always work as expected.

Integrating with VS Code - I

1. Install Visual Studio Code.
2. Install WSL extension.

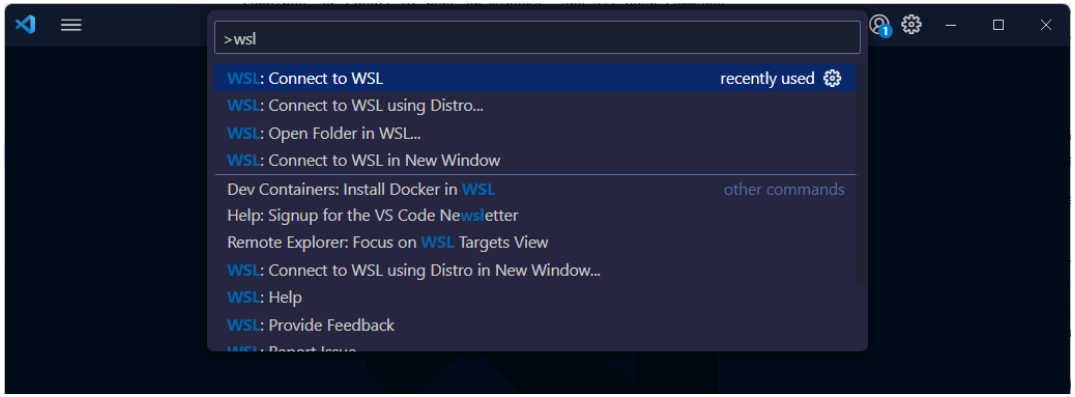
(a) Activity Bar (left side) $\longrightarrow$ Extensions

(b) Search for WSL

(c) Install

Integrating with VS Code - II

3. Connect to WSL in VS Code.



   Command Palette (Ctrl + Shift + P) $\longrightarrow$ *WSL: Connect to WSL* or *WSL: Connect to WSL using Distro*

4. Use `mkdir` to create a folder (say `python`) for your programs in your home directory.
   Create a file (say `hello.py`) in the folder; write `print("hello")` in the file and save it.

5. In VSCode, Command Palette (Ctrl + Shift + P) $\longrightarrow$ *WSL: Open folder in WSL*.

Integrating with VS Code - III

6. In VSCode, Explorer (Ctrl + Shift + E) $\longrightarrow$ `hello.py`.
7. Install the Python extension (incl. Python debugger and Pylance).
8. In `hello.py`, set breakpoints on some lines by left-clicking in the gutter to the left of the line number or pressing by F9.
9. To start debugging, press F5 to run your application. When prompted for a run configuration, choose Python File.
10. Output should appear in the debug console.

References

- **Overview:** learn.microsoft.com/en-us/windows/wsl/about
- **All docs:** learn.microsoft.com/en-us/windows/wsl
- **Running GUI apps** (e.g., a simple editor, *meld* — program for viewing differences between two similar files)

  learn.microsoft.com/en-us/windows/wsl/tutorials/gui-apps
- **Visual Studio Code**

  learn.microsoft.com/en-us/training/modules/developing-in-wsl