EN.540.635 "Software Carpentry"

How to Install Windows Subsystem for Linux

Introduction

There are many Operating Systems (OS) out there, with many intresting niches. We can categorize these OS into their Kernels: Windows (XP, Vista, 10, etc), Unix (MacOS, NetBSD, SunOS, etc), and Linux (Ubuntu, Arch, CentOS, etc). The Unix kernel, first released in 1971, was soon to be a well known and established kernel for OS development (leading to the development of MacOS). Wanting to develop a unix like kernel, but keeping it free to use, Linus Torvalds wrote the Linux kernel in 1991. This was done as a project while he was an Undergraduate, and was originally not intended to be ground breaking, as can be seen by the following direct quote (link):

> Hello everybody out there using minix -
>
> I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).
>
> I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)
>
> Linus (torv...@kruuna.helsinki.fi)
>
> PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

However, history has shown this to be provably false! Linux is now extensively used, from every android device to popular OS's such as Ubuntu, Linux can be found in one's everyday life. For researchers that use supercomputers, they will be hard pressed to find one that does not run on some version of Linux.

As Linux was written to be as "Unix-like" as possible (at first), those using MacOS can "learn" how to use Linux simply by playing around on their terminals. Those using Windows, however, a more "drastic" approach is necessary. In the past this entailed dual-booting one's computer with a Linux Distro (such as Ubuntu); however, now a simpler alternative exists: the Windows Subsystem for Linux (WSL).

Enable the WSL feature

First, you must enable the WSL feature. If you have already done this in the past, you may skip these steps.

1. Open up the PowerShell as an Administrator. This can be done by going to the start menu, typing "powershell", right clicking it, and selecting "Run as administrator".
2. Type the following command into powershell and hit enter: Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
3. If need be, restart the computer afterwards.

Install Linux

You may now install any version of Linux you wish to use. To do so, simply use the Microsoft Store.

1. Open the Microsoft Store. This can be done by going to the start menu, typing "microsoft store", and selecting the option.
2. In the search bar (top right of the store), type "linux" and hit enter.
3. On the top left you should see "Departments" and "All departments" as the choice, switch this to "Apps".

4. There are a variety of options to choose from. I recommend going with the "Ubuntu 18.04 LTS" option.

After installing, you can simply run the application by going to your start menu and searching for it. In my case, I installed Ubuntu, so I can open it by going to the start menu, searching "ubuntu", and choosing the option. NOTE! I recommend right-clicking and pinning this to the taskbar for easy access later. On your first opening of Ubuntu, it will take some time to initialize appropriately.

Setup

When you first install Ubuntu, not everything is setup perfectly. One such thing is that you will never get a display out of the defaults. That is, if you were to try plotting something in python using WSL, the display will not be found. That is because the default setup is command-line only! To fix this, you need to setup something called an X-Server. I recommend going with VcXsrv (link). After installing, you should run this to essentially tell WSL that "hey, I exist and can show the user things". This will need to be run every time you reboot your computer (it is a "server" which means it needs to be "on" to be used).

Note - you will still have errors though! The server now exists, but WSL is blissfully unaware of it. You can tell your linux OS that it exists by setting something called an "Environment Variable" (specifically the DISPLAY variable). This is done by editing a file and adding the following to the end of the file (on its own line):

```
export DISPLAY=:0
```

In order to edit a file in the command line, use one of the built in programs (I recommend vim); however, as this is getting outside the realm of "Install WSL" and into the realm of "Use Linux", let's just say you can enter the following command to do your bidding for now (note, **do NOT copy paste** as the command will not work if you do that):

```
echo "export DISPLAY=:0" >> ~/.bashrc
```

This command uses the **echo** function to print the line in question to the terminal; however, by using the > symbol, we are *piping* the output to a file, and by using two in a row (no spaces) we are *appending* to the file (whereas only one would overwrite the entire file!). The file in question we are editing is stored in our *home directory* (denoted as ~), and is a hidden file (denoted by how it starts with a period). The ~/.bashrc file is a *resource file* that holds the startup information for our *shell* (another term for our terminal). There are a variety of shells out there, such as ash, bash, csh, tcsh, dash, zsh, and more! Bash is the most known and utilized though, so we will stick to that in this course.

Additional Resources:

- Installing WSL (link).
- A History of Unix and Linux (link).
- Vim Cheatsheet (link).
- Vim "Masterclass" (link).