Setting up Windows Subsystem For Linux (WSL) on Windows 10

Many programming courses at DigiPen use programming tools implemented in Linux. Just like Windows and macOS, Linux is an operating system originating from another popular operating system called Unix. On a historical note, C was invented in the early 1970's to implement the Unix operating system.

There are three main reasons why you should learn Linux. First, Linux is everywhere. Whether you are aware of it or not, you're already using Linux every moment of every day. If you've an Android cell phone, you're using a variant of Linux. If you've an Apple cell phone, you're using iOS which is a variant of Unix from which Linux itself is derived. The web and internet are powered by servers running Linux. Pretty much every modern supercomputer is powered by Linux. In-vehicle infotainment systems, airplane in-flight entertainment systems, consumer electronics including smart TVs, DVRs, voice controlled devices such as Amazon Echo and Google Home, home security systems, and numerous industrial applications such as robots, assembly lines, and mass-transit systems are entirely powered by Linux. Second, Linux is free. Linux is an open-source operating system, with source code distributed by the Free Software/Open Source community. Anyone who receives it can make changes and redistribute it. No one company or individual owns Linux, which was developed, and is still being debugged and improved, by thousands of corporate-supported and volunteer programmers. Third, it is good for your career. Because Linux is free, versatile, and present everywhere, experience in Linux is a necessary requirement for most software developers and engineers. The basic exposure to Linux gained in this course will begin your journey in gaining and developing job skills that are in demand and well paid.

Computers in DigiPen labs have the Windows Subsystem for Linux (WSL) that provides a Linux environment directly on Windows 10. This document is for those intending to clone the DigiPen lab setup on your personal Windows 10 based computer. Unfortunately, neither DigiPen nor I own Apple-based computers and therefore it is not possible to specify a method to clone the DigiPen programming lab environment for such computers.

Downloading and installing WSL and Ubuntu on your personal Windows computers

1. Microsoft provides comprehensive documentation on installing WSL and a Linux distribution. You can follow the steps described there and then proceed to this section to install software in Linux. The following is an alternative method that in my opinion has fewer steps and is simpler to follow.
2. Before installing WSL, ensure your computer is up to date with the latest update of Windows 10.
3. Before installing any Linux distros for WSL, enable the "Windows Subsystem for Linux" feature. Do this by opening **PowerShell** as **administrator** (type **PowerShell** into the search bar, right-click **Windows PowerShell**, and then click **Run as administrator**) and run this command:

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
```

After running the command, your computer will install certain things. Restart your computer whether prompted by Windows or not. During the restart, more pieces of software will be installed.

4. Do the same as above except now you're enabling the "Virtual Machine Platform" feature on Windows:

   ```
   Enable-WindowsOptionalFeature -Online -FeatureName VirtualMachinePlatform
   ```

   Restart your computer whether prompted by Windows or not.

5. The next step is to download the latest Linux distribution which is Ubuntu 20.04 LTS. Open **PowerShell** as **administrator** again and run this command:

   ```
   Invoke-WebRequest -Uri https://aka.ms/wslubuntu2004 -OutFile Ubuntu.appx
   -UseBasicParsing
   ```

   The distribution will take a few minutes to download and will be saved on your computer as `Ubuntu.appx`.

6. To install the distribution, run the following command in **PowerShell**:

   ```
   Add-AppxPackage .\Ubuntu.appx
   ```

7. There are two versions of WSL and you should aim to install WSL 2 rather than WSL 1. Download and install the latest WSL 2 Linux kernel update package for $x64$ machines (the most common CPU for desktops and laptops).

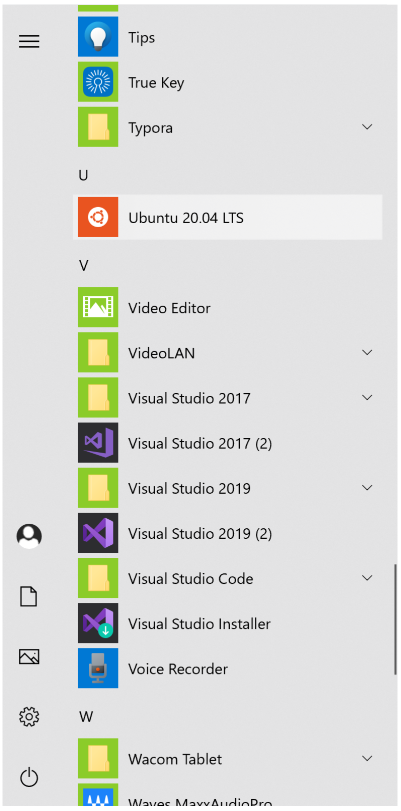
8. Again, reboot your computer whether prompted or not.

9. Use **PowerShell** to tell Windows that you want the Linux distribution to use WSL 2 as the default version:

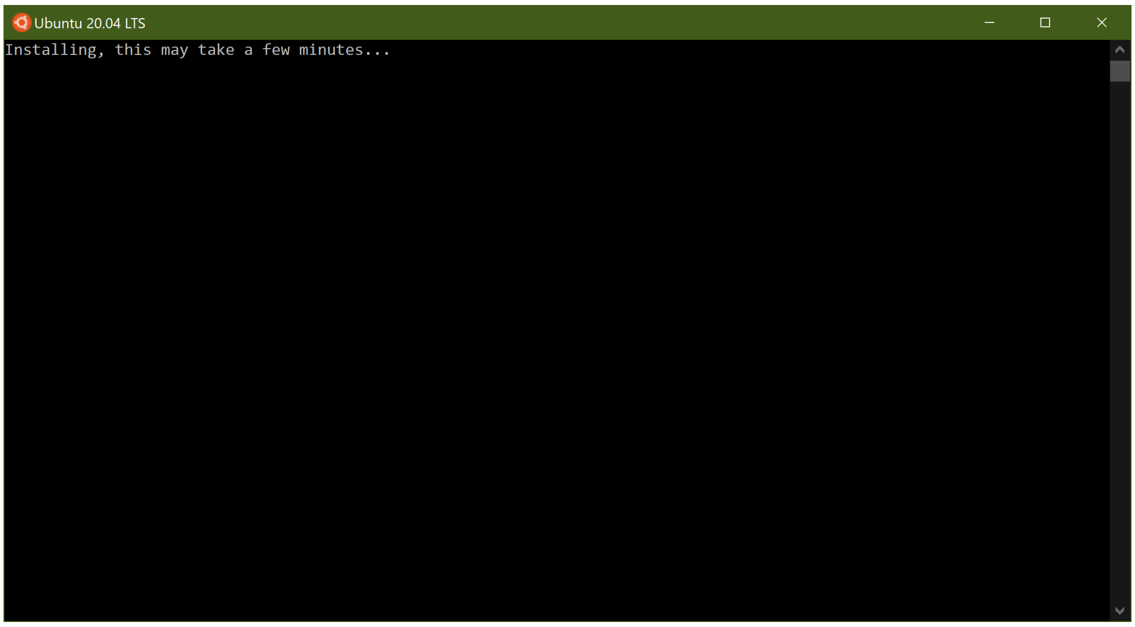
   ```
   wsl --set-default-version 2
   ```

Setting up WSL 2

1. Once WSL 2 and Linux distribution of Ubuntu 20.4 LTS have been downloaded, you're ready to install and enable Linux. Choose **Ubuntu 20.4 LTS** app from the **Start** menu:

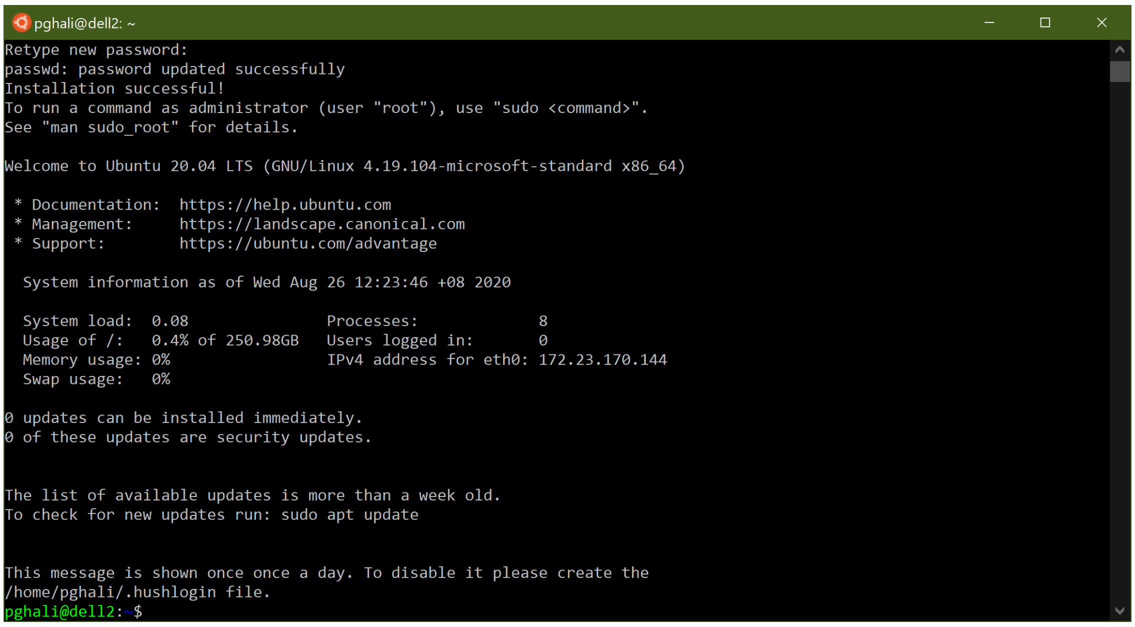


2. The app will open a bash shell that will install Linux with the command window looking like this:



```
Installing, this may take a few minutes...
```

3. After a few minutes it will prompt you for a username and password. Congratulations! You are now running Linux (inside of Windows)! You can tell because the command prompt has changed:



```
Retype new password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 20.04 LTS (GNU/Linux 4.19.104-microsoft-standard x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Aug 26 12:23:46 +08 2020

  System load:  0.08               Processes:             8
  Usage of /:   0.4% of 250.98GB   Users logged in:       0
  Memory usage: 0%                 IPv4 address for eth0: 172.23.170.144
  Swap usage:   0%

0 updates can be installed immediately.
0 of these updates are security updates.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


This message is shown once once a day. To disable it please create the
/home/pghali/.hushlogin file.
pghali@dell2:~$
```

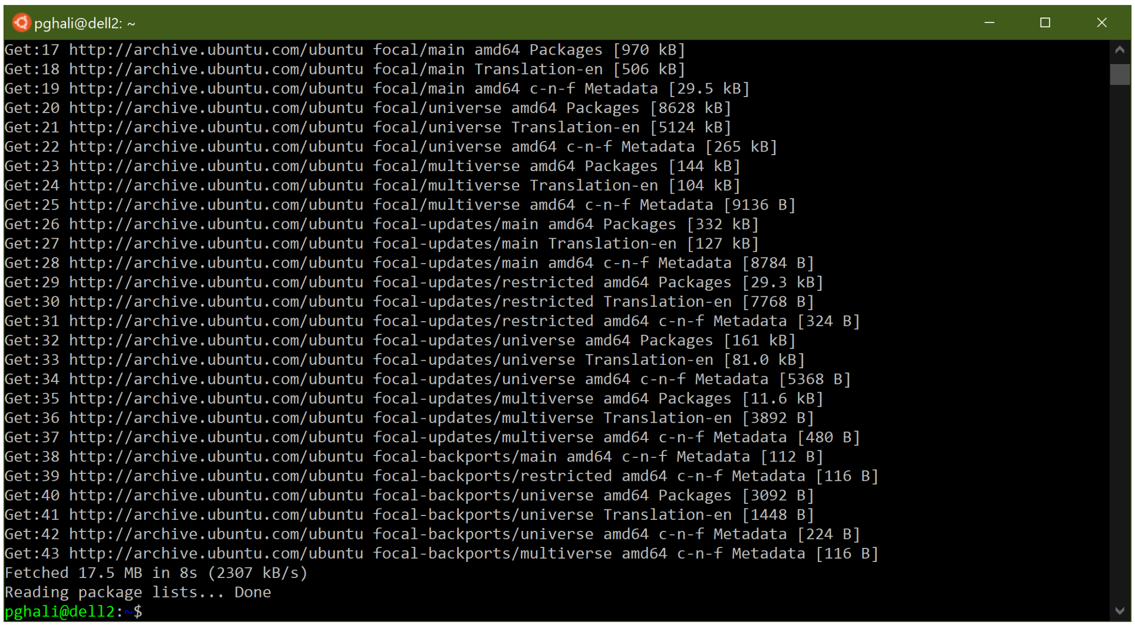
Installing software in WSL Linux

We'll require GNU C and C++ compiler drivers, Clang C and C++ compiler drivers, Valgrind for detecting memory management bugs, and Doxygen which is a documentation system for most programming languages including C and C++. The process is illustrated in the following steps.

1. Begin by updating the package repository. Since updating the installed package database requires superuser privileges, you'll need to use `sudo`, as in:

   ```
   pghali@dell2:~$ sudo apt-get update
   ```

   It will prompt for your password (the one you just created) and then run for a minute or so and then tell you it's done:



```
Get:17 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [970 kB]
Get:18 http://archive.ubuntu.com/ubuntu focal/main Translation-en [506 kB]
Get:19 http://archive.ubuntu.com/ubuntu focal/main amd64 c-n-f Metadata [29.5 kB]
Get:20 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [8628 kB]
Get:21 http://archive.ubuntu.com/ubuntu focal/universe Translation-en [5124 kB]
Get:22 http://archive.ubuntu.com/ubuntu focal/universe amd64 c-n-f Metadata [265 kB]
Get:23 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [144 kB]
Get:24 http://archive.ubuntu.com/ubuntu focal/multiverse Translation-en [104 kB]
Get:25 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 c-n-f Metadata [9136 B]
Get:26 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [332 kB]
Get:27 http://archive.ubuntu.com/ubuntu focal-updates/main Translation-en [127 kB]
Get:28 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [8784 B]
Get:29 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [29.3 kB]
Get:30 http://archive.ubuntu.com/ubuntu focal-updates/restricted Translation-en [7768 B]
Get:31 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 c-n-f Metadata [324 B]
Get:32 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [161 kB]
Get:33 http://archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [81.0 kB]
Get:34 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 c-n-f Metadata [5368 B]
Get:35 http://archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [11.6 kB]
Get:36 http://archive.ubuntu.com/ubuntu focal-updates/multiverse Translation-en [3892 B]
Get:37 http://archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 c-n-f Metadata [480 B]
Get:38 http://archive.ubuntu.com/ubuntu focal-backports/main amd64 c-n-f Metadata [112 B]
Get:39 http://archive.ubuntu.com/ubuntu focal-backports/restricted amd64 c-n-f Metadata [116 B]
Get:40 http://archive.ubuntu.com/ubuntu focal-backports/universe amd64 Packages [3092 B]
Get:41 http://archive.ubuntu.com/ubuntu focal-backports/universe Translation-en [1448 B]
Get:42 http://archive.ubuntu.com/ubuntu focal-backports/universe amd64 c-n-f Metadata [224 B]
Get:43 http://archive.ubuntu.com/ubuntu focal-backports/multiverse amd64 c-n-f Metadata [116 B]
Fetched 17.5 MB in 8s (2307 kB/s)
Reading package lists... Done
pghali@dell2:~$
```

2. Once the installed package database is updated, these packages can be upgraded using the following command:

   ```
   pghali@dell2:~$ sudo apt-get upgrade
   ```

After determining disk space requirements for the upgrade, you'll be prompted to type `y` to continue or `n` to stop the upgrade. Make sure to type `y`. The upgrade process will take a few minutes ...

3. Once the Linux distribution is completely upgraded, you're ready to install certain programming and software development tools that will be required in your programming courses:

```
pghali@dell2:~$ sudo apt-get install build-essential gcc g++ clang
```

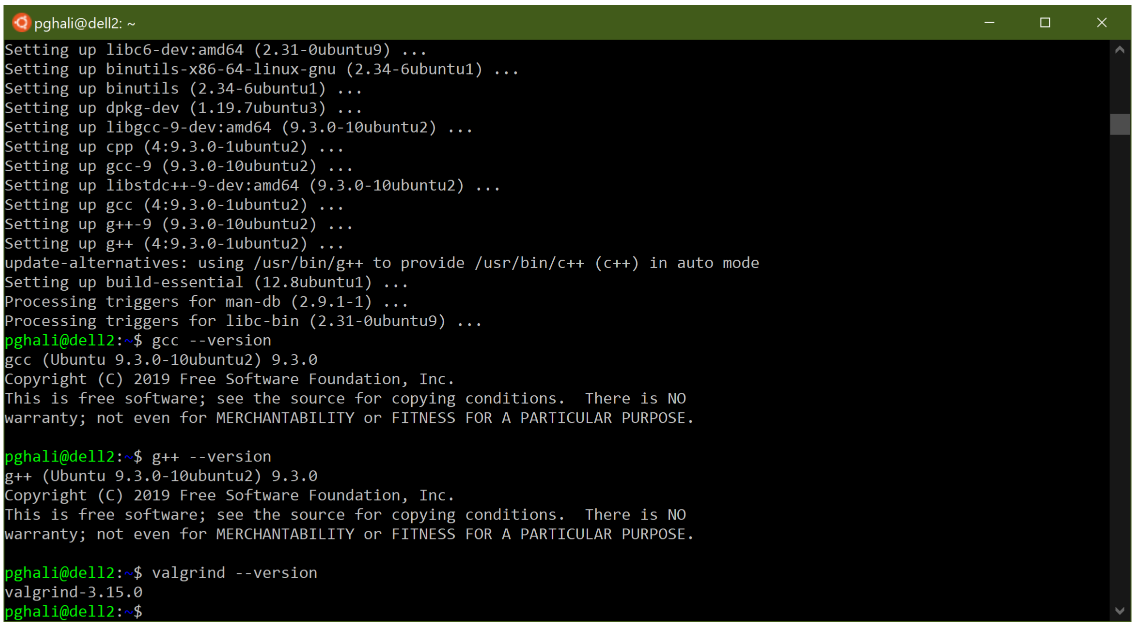
The command will gather the necessary packages, determine disk space requirements, and prompt you for a `y` or `n`. Make sure to type `y`. The upgrade process will take a few minutes ...

This will install GNU C and C++ compiler drivers, Clang C and C++ compiler drivers, and other tools necessary to build executables such as `make`. Clang has become a popular compiler for languages in the C family. It is used extensively in Apple and is reputed to generate faster compiles with lower memory requirements and better diagnostics compared to GNU C/C++ compilers.

At the conclusion of this command, you've just setup a development environment and installed two different sets of C and C++ compiler drivers! Confirm that the compilers are correctly installed by running the following commands in the shell:

```
pghali@dell2:~$ gcc --version
pghali@dell2:~$ g++ --version
pghali@dell2:~$ clang --version
```

You will see something like this:



```
Setting up libc6-dev:amd64 (2.31-0ubuntu9) ...
Setting up binutils-x86-64-linux-gnu (2.34-6ubuntu1) ...
Setting up binutils (2.34-6ubuntu1) ...
Setting up dpkg-dev (1.19.7ubuntu3) ...
Setting up libgcc-9-dev:amd64 (9.3.0-10ubuntu2) ...
Setting up cpp (4:9.3.0-1ubuntu2) ...
Setting up gcc-9 (9.3.0-10ubuntu2) ...
Setting up libstdc++-9-dev:amd64 (9.3.0-10ubuntu2) ...
Setting up gcc (4:9.3.0-1ubuntu2) ...
Setting up g++-9 (9.3.0-10ubuntu2) ...
Setting up g++ (4:9.3.0-1ubuntu2) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Setting up build-essential (12.8ubuntu1) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9) ...
pghali@dell2:~$ gcc --version
gcc (Ubuntu 9.3.0-10ubuntu2) 9.3.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

pghali@dell2:~$ g++ --version
g++ (Ubuntu 9.3.0-10ubuntu2) 9.3.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

pghali@dell2:~$ valgrind --version
valgrind-3.15.0
pghali@dell2:~$
```

4. You'll need an editor to write C source code. The free and open source Visual Studio Code is an excellent choice because it works in both Windows 10 and in Linux. To install Code, follow the instructions specified here.

5. Finally, install memory debugger Valgrind, documentation system Doxygen, and an open-source graph drawing toolkit called Graphviz (which will be used by Doxygen to generate hierarchy graphs):

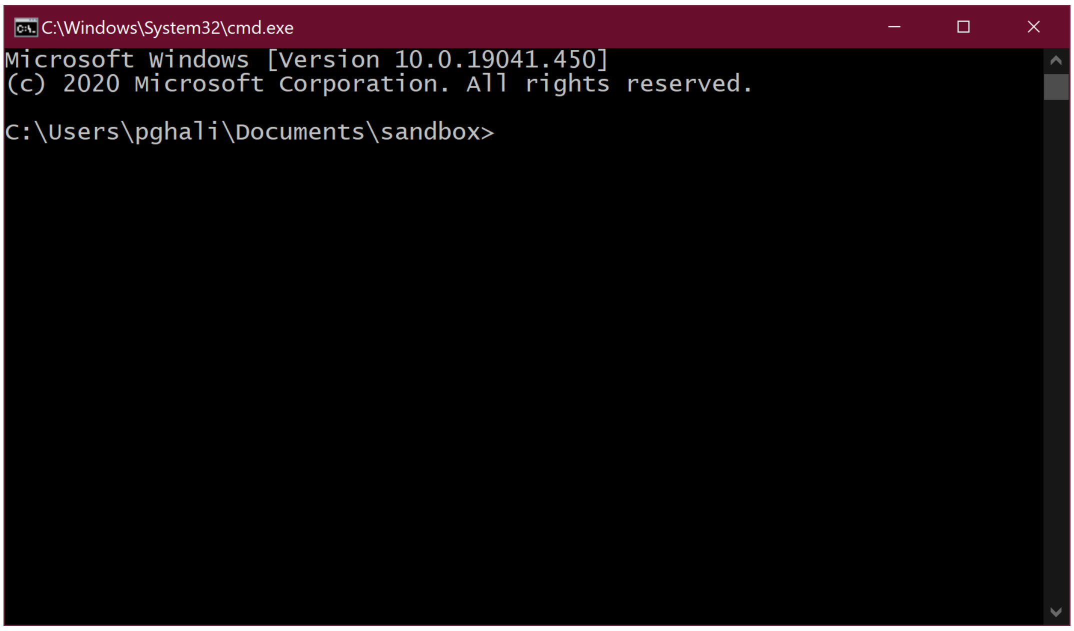
```
pghali@dell2:~$ sudo apt-get install valgrind doxygen graphviz
```

Confirm these packages are correctly installed by running the following commands in the shell:

```
pghali@dell2:~$ valgrind --version
pghali@dell2:~$ doxygen --version
pghali@dell2:~$ dot -V
```

Testing your setup

1. I'll suppose that your Windows username is `pghali`. In Windows, create a subfolder called `sandbox` in folder `C:\Users\pghali\Documents`. Open a command prompt in that folder (click in the address bar of folder `sandbox` to select its contents; then type `cmd` and press Enter). Now, you're at the command line, but still in Windows:

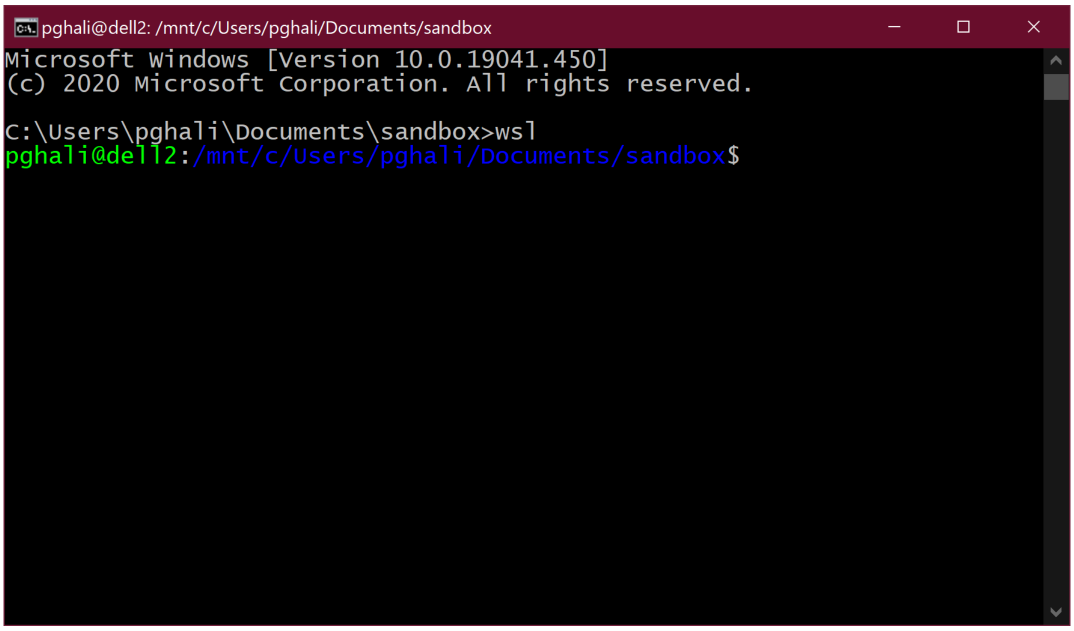


C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.19041.450]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\pghali\Documents\sandbox>
```

2. Type `wsl` on the command line and you'll be switched to Linux. Again, it is easy to see because your command prompt has changed:



pghali@dell2: /mnt/c/Users/pghali/Documents/sandbox

```
Microsoft Windows [Version 10.0.19041.450]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\pghali\Documents\sandbox>wsl
pghali@dell2:/mnt/c/Users/pghali/Documents/sandbox$
```

Remember command `wsl` as it is the command you use to switch into Linux from Windows. Very conveniently, you will still be in the same directory `sandbox` as you were in Windows. To get out of Linux and return back to Windows, simply type `exit` while you're in Linux. If you happen to type `exit` while you're in Windows, it will close the command prompt and you'll have to reopen it.

Notice the directory you're in after switching to Linux. My example shows that in Windows you were in folder `C:\Users\pghali\Documents\sandbox` while in Linux the directory is `/mnt/c/Users/pghali/Documents/sandbox`. When you're in Linux, the entire C drive for Windows is located in `/mnt/c`. Also notice that directories are separated by a forward slash `/` in Linux, while Windows uses a backslash `\`. Finally, notice that in Windows, the term *folder* is used to indicate the repository of files and other subfolders. In Linux, the more classical term *directory* is used to indicate the same thing as a folder.

3. In the Linux shell, run Code:

```
pghali@dell2:/mnt/c/Users/pghali/Documents/sandbox$ code main.c
```

   This will run the Visual Studio Code programming editor with a new file `main.c`. Enter the following C code in this file:

```c
#include <stdio.h>

int main(void) {
  printf("Hello World!!!\n");
  return 0;
}

```

4. The convenient feature of using Windows and WSL is that a file `main.c` created in Linux is available for further editing in Windows with both Code and other editors.
5. Let's compile and link the C source file `main.c` with GNU C compiler driver (note for brevity I'm replacing the lengthy `/mnt/c/Users/pghali/Documents/sandbox` with the character `~` in the following and rest of the command prompts):

```
pghali@dell2:~$ gcc -std=c11 -pedantic-errors -Wstrict-prototypes -Wall -
Wextra -Werror main.c -o main.out
```

6. If your software setup is correct and you typed the above C code in `main.c`, the GNU C compiler driver will compile source file `main.c` to an object file and link the C library to create an executable file `main.out`. You can run this executable like this to see the output from the program:

```
pghali@dell2:~$ ./main.out
Hello World!!!
```

7. Although text files are transferable between Linux and Windows, the binary executable `main.out` is formatted to correctly load and execute in Linux but not in Windows. Try switching back to Windows shell (by typing `exit`) and running `main.out` - you'll find your computer is unable to execute the Linux executable in Windows. On the other hands, Windows-compatible programs can be run from the bash shell in Linux. For example, try running a primitive text editor supplied by Windows called `Notepad`:

```
pghali@dell2:~$ Notepad.exe
```

8. Let's compile and link source file `main.c` with Clang driver:

```
pghali@dell2:~$ clang -std=c11 -pedantic-errors -Wstrict-prototypes -Wall
-Wextra -Werror main.c -o maincl.out
```

A nice feature of Clang is that it uses very similar options as GNU C compiler driver. Run executable `maincl.out` to get the text `Hello World!!!` to the shell:

```
pghali@dell2:~$ ./maincl.out
Hello World!!!
```

9. In Code, create a new C++ source file `main.cpp` and fill it with the following code:

```
#include <iostream>
#include <string>

int main() {
  std::cout << "Enter your name: ";
  std::string name;
  std::cin >> name;
  std::cout << "Hello " << name << "!!!" << std::endl;
}
```

10. Compile and link C++ source file `main.cpp` with GNU C++17 compiler driver:

```
pghali@dell2:~$ g++ -std=c++17 -pedantic-errors -Wall -Wextra -Werror
main.cpp -o maingpp.out
```

Run executable `maingpp.out`. The program will prompt for your name and then print a greeting:

```
pghali@dell2:~$ ./maingpp.out
Enter your name: Prasanna
Hello Prasanna!!!
pghali@dell2:~$
```

11. Repeat the exercise with Clang C++17 compiler driver:

```
pghali@dell2:~$ clang++ -std=c++17 -pedantic-errors -Wall -Wextra -Werror
main.cpp -o mainclpp.out
```

12. If you're able to install WSL and Ubuntu Linux distro and work in the Linux bash shell, edit text files using Code, compile and link using the GNU C/C++ and Clang C/C++ compiler drivers and execute the binary executables, then your computer is correctly setup for DigiPen programming courses.
13. Using Valgrind to detect memory-related errors and leaks and using Doxygen to document your source and header files will be discussed in other documents.

Updates: Installing `g++` and `clang++` for C++20

Installing `g++-11`

There is no stable release of `g++` compiler for C++20. Instead, we've to use a Personal Package Archive (PPA) software repository created by Ubuntu for the GNU toolchain. PPAs are often used to distribute pre-release software so that it can be tested by users. Begin by running a Python script that will add Ubuntu's PPA GNU toolchain repository to your system's list of repositories:

```
pghali@dell2:~$ sudo add-apt-repository -y ppa:ubuntu-toolchain-r/test
```

Install `g++-11` by using the following command:

```
pghali@dell2:~$ sudo apt install -y g++-11
```

Now you can verify that the installation finished successfully by checking `g++` version:

```
pghali@dell2:~$ g++-11 --version
```

Testing `g++-11`

Create a source file `main20.cpp` file and add the following lines of code:

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

void print(auto const& coll) {
  for (auto const& elem : coll) {
    std::cout << elem << ' ';
  }
  std::cout << '\n';
}

int main() {
  std::vector<std::string> coll {"Seattle", "Singapore", "Sydney", 
"Shanghai"};
  std::ranges::sort(coll);    // sort elements
  std::ranges::sort(coll[0]); // sort characters in first element
  print(coll);                // print container contents

  int arr[] {42, 0, 8, 15, 7};
  std::ranges::sort(arr); // sort values in array
  print(arr);
}
```

Compile and link the code in source file `main20.cpp`:

```
pghali@dell2:~$ g++-11 -std=c++20 -pedantic-errors -Wall -Wextra -Werror 
main20.cpp
```

If the program compiles, links, and executes to generate the following output:

```
Saeeltt Shanghai Singapore Sydney
0 7 8 15 42
```

you've installed the latest version of `g++`.

Uninstall `g++-11`

If you decide to completely remove `g++-11` and related dependencies, execute the following command:

```
pghali@dell2:~$ sudo apt purge --autoremove -y g++-11
```

Remove GNU toolchain repository and GPG key (GPG or GnuPG is a GNU implementation of a encryption technique called PGP):

```
pghali@dell2:~$ sudo rm -rf /etc/apt/trusted.gpg.d/ubuntu-toolchain-r_ubuntu_test.gpg
pghali@dell2:~$ sudo rm -rf /etc/apt/sources.list.d/ubuntu-toolchain-r-ubuntu-test-focal.list
```

Installing and Testing `clang++-12`

Just as with the GNU toolchain, there is no stable release of `clang++` - however, we can get pre-release version of `clang++-12`:

```
pghali@dell2:~$ sudo apt install clang-12 --install-suggests
```

Once this version is installed, you can check if the source file `main20.cpp` compiles:

```
pghali@dell2:~$ clang++-12 -std=c++20 -pedantic-errors -Wall -Wextra -Werror main20.cpp
```