Virtual Machine Installation Instruction

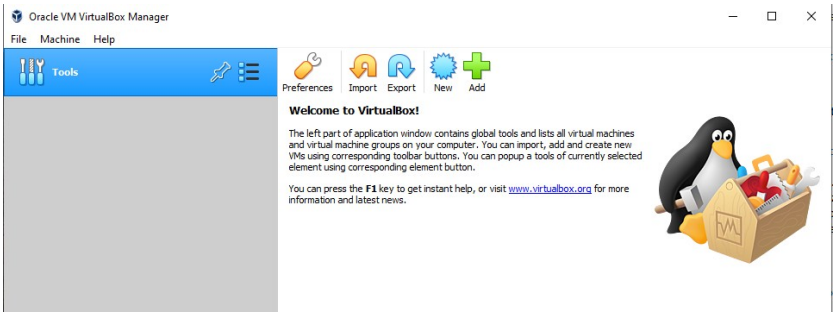
Testing code across different operating systems can get complicated! Setting up a virtual machine with the instructions below will ensure a consistent environment, so the instructors are less likely to have issues with your submitted code. Additionally, developing in a virtual machine can be useful to control development environments when you have multiple projects (or courses!) with different requirements. You will be developing on Linux using sublime and the command line. If you encounter issues with the virtual machine,.

1. Navigate to the following website and download the .iso file of the latest Ubuntu version:
   https://ubuntu.com/download/desktop
   N.B. the file is about 2.5 GB.

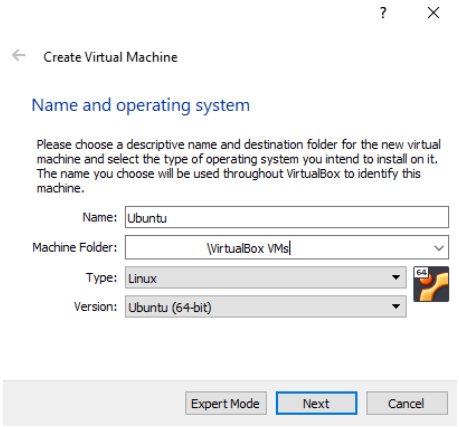
2. Navigate to the following website and download the latest version of VirtualBox that matches your OS (Windows or Mac) and architecture (x86 or x64):
   https://www.virtualbox.org/wiki/Downloads

3. Go ahead and install VirtualBox.

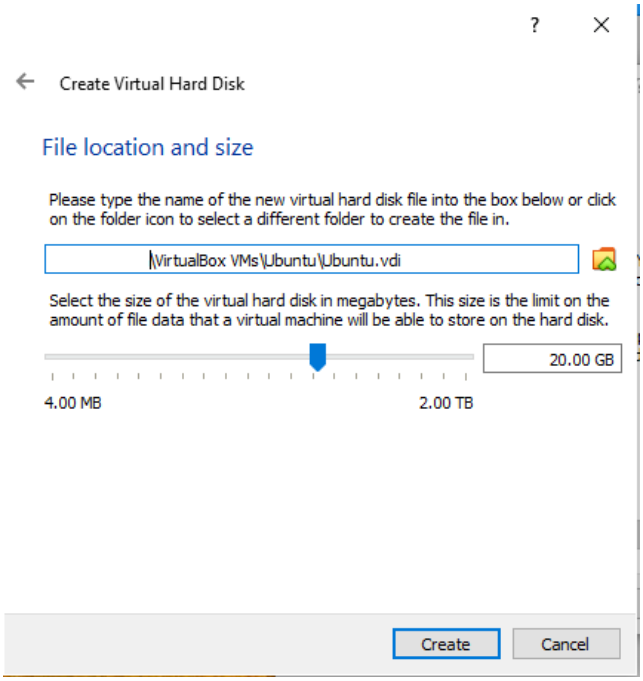
4. Create a Ubuntu Virtual Machine (VM):
   a. Open Virtual Box and in Virtual Box Manager, click “New”



   b. Name the VM “Ubuntu” to automatically populate the operating system Type and Version



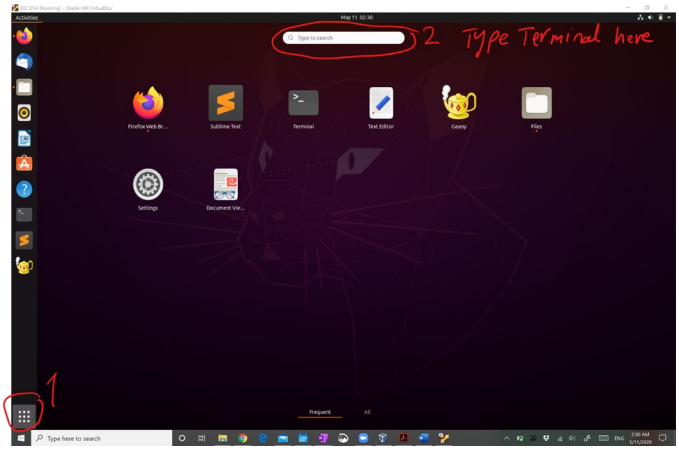
c. Continue the setup using default settings for Memory size and the virtual hard disk, **except make the hard disk size at least 20 GB** when you get to the "File location and size" screen.



d. After creating the VM, select it in the Virtual Box Manager and click Start. When prompted to select a virtual optical disk, browse to select the Ubuntu .iso file that you previously downloaded. Follow the instructions to install Ubuntu.
   i. Alternately, select the VM in the Virtual Box Manager, click Settings, go to Storage, then select Controller: IDE and add the .iso file. Then Start the VM and follow the instructions to install Ubuntu.
   ii. Note that if you reboot the machine without removing the installation media and a screen appears saying to eject the media and press 'Enter', just press Enter and the machine should start.

e. Log your user password for the virtual machine somewhere!

5. Once you are done creating the VM, run it and open a terminal (see the picture):



6. Follow the following guide to install Sublime Text on the VM. This will be the text editor used to write C++ code.
   https://linuxize.com/post/how-to-install-sublime-text-3-on-ubuntu-18-04/

7. Run the following commands from the terminal:

   ```
   > sudo apt-get install gcc

   > sudo apt-get install g++

   > sudo apt-get install cmake
   ```

8. Run the following command from the terminal to install git:

   ```
   > sudo apt install git-all
   ```

9. Your VM is now ready! See the following tutorials to familiarize yourself with the Ubuntu command line: https://ubuntu.com/tutorials/command-line-for-beginners#1-overview

10. If you find that your VM is running slow, you can increase the memory allocated to 4Gb in the VM settings.

Compiling With g++

1. Create a new .cpp file called “hello.cpp” in Sublime from the terminal:
   ```
   > subl hello.cpp
   ```

   Write the following or a similar “Hello World” program in hello.cpp and save the file:

   ```
   #include <iostream>

   int main() {
      std::cout << "Hello World!" << std::endl;
      return 0;
   }
   ```

2. To build the program from the terminal:
   ```
   > g++ -o hello hello.cpp
   ```

   where -o is the option to define the output file name, “hello” is the name of the executable output file, and “hello.cpp” is your top level cpp file.

3. Run the executable:
   ```
   > ./hello
   ```

   “Hello World!” should print in the terminal. Note that VSCode has a built in terminal (Terminal > New Terminal) where you can run the above commands.

Using CMake

We will use CMake to manage builds of projects including multiple files.

For projects in this course, the CMakeLists.txt file will be provided with any starter code.
Run “cmake .” from the terminal in the top level project directory (the same directory as CMakeLists.txt) to create a Makefile for the project. Then, run “make” in that directory any time to build the project:

```
> cmake .
```

```
> make
```

You will need to run make to rebuild after modifying any code. It is also good practice to run “make clean” before rerunning make, to remove any existing executables and avoid confusion if the new version does not compile:

```
> make clean
> make
```

To practice using CMake, create a file CMakeLists.txt in the same directory as your top level .cpp file with the following contents:

```
cmake_minimum_required(VERSION 3.5)
project(hello)
add_executable(hello hello.cpp)
```

The second line sets the project name and the third line compiles hello.cpp. Running “cmake .” from the terminal in this directory should create a Makefile for the project. Then, run “make” to build the project. The executable “hello” should be created similarly to using g++ above.