UNIVERSITY OF CALIFORNIA, MERCED

BERKELEY · DAVIS · IRVINE · LOS ANGELES · MERCED · RIVERSIDE · SAN DIEGO · SAN FRANCISCO

SANTA BARBARA · SANTA CRUZ

CSE 30: Data Structures
Development Environment Setup

Spring 2020

1 Introduction

In this document, we will be setting up your workspace for the rest of the semester. This involves installing all the compilers, tools, and libraries that you will need in order to work with the programming examples and exercises that will be provided in the course.

This document contains instructions specific to operating systems. The support code you will be provided with has been tested on computers running Windows[1] 10, macOS 10.13 or later, and Ubuntu 18.04 and some of its variants.

2 Computer Setup

Please follow the guide relevant to you, based on the operating system installed on your laptop.

2.1 Windows

The following has been tested and verified to work on Windows 10. If you have another version of Windows, you may be able to follow a similar set of steps.

1. **Enable Virtualization** This is a crucial step, as it is needed by Ubuntu Shell, and it is perhaps the most difficult, as the process is slightly different for each laptop manufacturer. First, check to see if virtualization is enabled. To do this, open Task Manager → Performance, and see if Virtualization is Enabled or Disabled. You should see an interface similar to Fig. 1.

   If virtualization is enabled on your computer, you can proceed to the next step. Otherwise, you need to enable virtualization in your computer's BIOS. To enter the BIOS, on most computers, you should press F1 during bootup (immediately after powering on). It may be one of the other F keys, so you should check the exact instructions for your own laptop. We have tested with a Lenovo ideapad, and it is F1, on some machines it is F12. You can also find it by trial and error. Once you enter the BIOS setup, look for a configuration option called Intel Virtual Technology, and set it to Enabled. Save changes and go boot into Windows.

   Please be careful when fiddling in the BIOS setup. While this is unlikely, it is possible for you to damage your computer if you do something wrong. If you are uncomfortable with this step, please ask your Teaching Assistant to help you.

[1] In Windows, we will be using Ubuntu Shell, so we will not exactly build native Windows applications, but things will work.

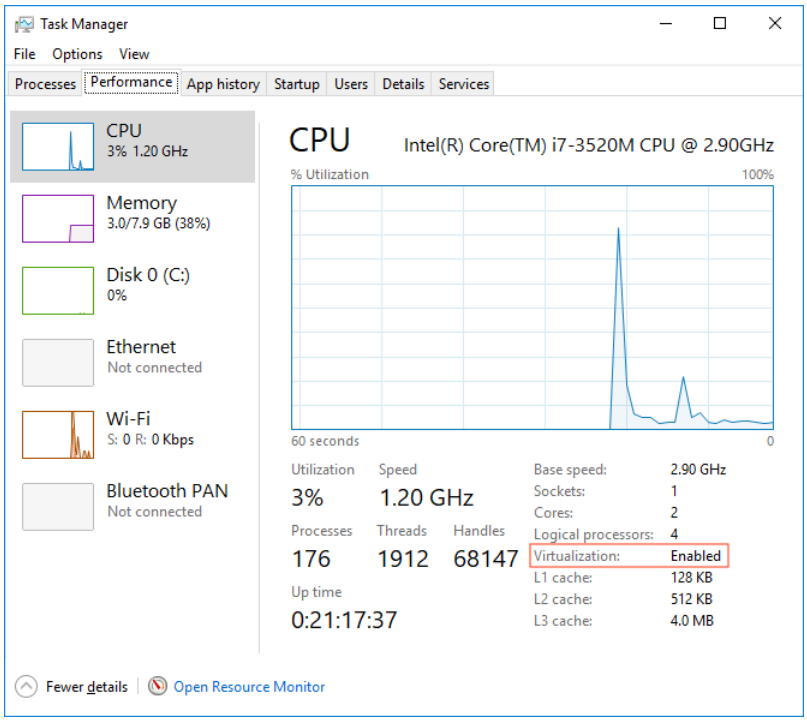

Figure 1: Task Manager window, highlighting virtualization is enabled.

2. **Enable Windows Subsystem for Linux** This is another pre-requisite for Ubuntu to run in your Windows environment. To do this, go to Control Panel → Programs → Turn Windows Features On Or Off, and select the option Windows Subsystem for Linux. This is shown in Fig. 2. Click OK, and you will be prompted to reboot your computer. This feature will not take effect until you reboot, so do it at this time.

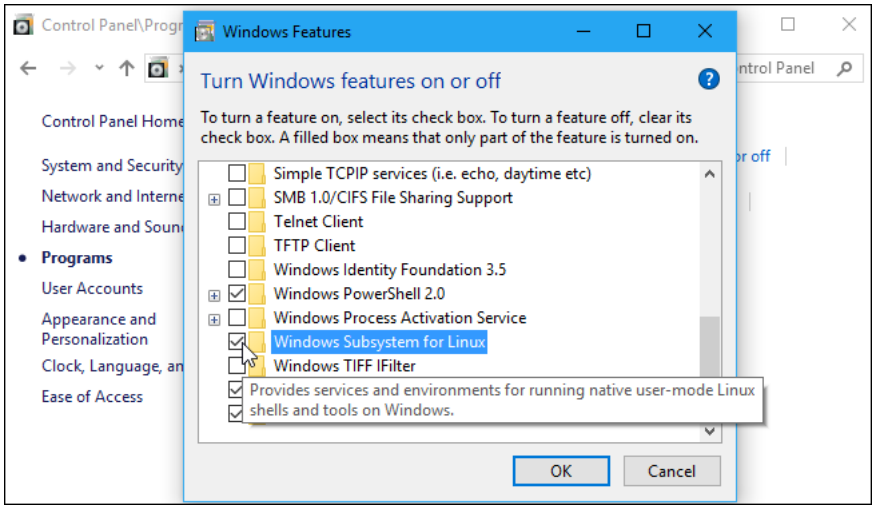

Figure 2: Enabling the Windows Subsystem for Linux

3. **Install Ubuntu** Go to the Microsoft Store and search for "Ununtu". You will see several results, one of which is Ununtu 18.04 LTS. Please install that one, following all the on-screen instructions and prompts. Once finished, you will be asked to Launch the application. The first time you do this, the

system will install some additional components. You will see a terminal window, with a message: "Installing, this may take a few minutes...".

In a little while, you will be prompted to enter an new UNIX username. Please input the username you wish to use for your Ubuntu subsystem. It does not need to be the same as your Windows username. Also, it needs to be all lowercase and contain no spaces. Alphanumeric characters only. You will then be asked to choose a new UNIX password. Again, it does not need to be the same as your Windows password. Please choose a password that you will remember, because you will need to use it a lot. Also note that when you type in the password, there will be no characters appearing in the terminal window, so for those of you who have not experienced this, it will look like you are not typing at all. Do not worry, as this is perfectly normal in Linux. Just type the password and press enter. Then you will be asked to retype the password, so do that and press enter again. You will then be given a message that the installation has successfully completed.

4. **Installing Dependencies** We will assume that you have already installed the Visual Studio Code text editor. If you have not, please install it by downloading from Microsoft's website, and following their instructions.

   In addition to VS Code, we need to install some compilers and other development tools. We will be installing these from within your Ubuntu shell. There is a way to install all these in one step, so in the Ubuntu terminal type in `sudo apt update`, and press enter. You will be asked for your password, which you should type in and press enter. After a short update process you will be ready to install the developer tools. To do that, type

   ```
   sudo apt install build-essential
   ```

   After displaying some information on the terminal, the program will ask you if you want to continue, with a message `Do you want to continue? [Y/n]`. Just press enter there, as the option for "Yes" is the default option, since the letter `Y` is capitalized. This will take some time but it will show you its progress so just wait for it to complete.

   The next thing we need to install is the boost library, which is a collection of algorithms for C++ that lets us do many things for which there is no built-in support. To install the boost library type:

   ```
   sudo apt install libboost-all-dev
   ```

   As before, press enter when asked if you want to continue, and just wait for it to complete. At some point, you may be asked if the installation script can "restart services during package upgrades without asking", you should say "Yes" to that.

   In order to be able to launch Visual Studio from the Ubuntu terminal, just type `code`. The first time you do it, it will install some additional components and will launch Visual Studio Code in Windows. When it opens, make sure its terminal is enabled. If there is no terminal on the bottom right of Visual Studio Code, press Ctrl+`(the key immediately to the left of 1), to toggle the terminal view. Otherwise, use the main menu View → Terminal. The first time you do this, you may have defaulted to the Windows powershell. This is not where we need to be, as we need to use bash. To fix this issue, just type `bash` and press enter. You will switch to bash. To make the change permanent, click on the drop-down which says "1: bash", and click on the option "Select Default Shell". A list appears at the top of the window, and one of the options is "WSL Bash", which is the one you need to select.

   That's all folks.

2.2 macOS

The following will work on any recent version of macOS. It has been tested as far back as High Sierra v10.13.

1. **Install Xcode** You can download and install Xcode from the Mac App Store for free. The process is quite long it installs all the compilers and tools that you will need. Once the installation is complete, open a terminal and type the following:

   ```
   xcode-select --install
   ```

   This will give you all the command line tools you need.

2. **Get Homebrew** In Ubuntu, we have the `apt` utility, which makes it easy to install packages. macOS does not have this utility out of the box but it can easily be installed. Go to the Homebrew website: `https://brew.sh` and follow the installation instructions. It will involve copy-pasting a `curl` command in the terminal and following the prompts.

3. **Installing Dependencies** The only dependency is the boost library, so open a terminal and type:

   ```
   brew install boost
   ```

   We also assume that you have Visual Studio Code, which you downloaded from Microsoft's website. Make sure the Visual Studio Code.app is in your Applications folder, and you are not running it from within the .dmg image that it came in. This is important, so make sure it's done correctly.

2.3 Ubuntu

On Ubuntu the hardest part is actually installing Visual Studio Code. The rest of the stuff is a piece of cake. If you are running Ubuntu (or some other distro) natively on your laptop, then you probably know all this but here it is anyway.

1. **Installing Visual Studio Code** Open a terminal type the following commands:

   ```
   curl https://packages.microsoft.com/keys/microsoft.asc |
      gpg --dearmor > packages.microsoft.gpg
   sudo install -o root -g root -m 644 packages.microsoft.gpg /usr/share/keyrings/
   sudo sh -c 'echo "deb [arch=amd64 signed-by=/usr/share/keyrings/packages.microsoft.gpg]
      https://packages.microsoft.com/repos/vscode stable main" >
      /etc/apt/sources.list.d/vscode.list'
   ```

   The commands above basically add a repository to the apt client on your computer so you can just use the commands below to install:

   ```
   sudo apt-get install apt-transport-https
   sudo apt-get update
   sudo apt-get install code
   ```

   There may be other ways to install, so if you're not having luck with this, you can look for alternatives.

2. **Installing Dependencies**

```
sudo apt install build-essential
```

After displaying some information on the terminal, the program will ask you if you want to continue, with a message `Do you want to continue? [Y/n]`. Just press enter there, as the option for "Yes" is the default option, since the letter `Y` is capitalized. This will take some time but it will show you its progress so just wait for it to complete.

The next thing we need to install is the boost library, which is a collection of algorithms for C++ that lets us do many things for which there is no built-in support. To install the boost library type:

```
sudo apt install libboost-all-dev
```

As before, press enter when asked if you want to continue, and just wait for it to complete. At some point, you may be asked if the installation script can "restart services during package upgrades without asking", you should say "Yes" to that.