Virtual Machines via Vagrant and VirtualBox

As previously discussed, the second half of the Wednesday January 30th lecture will be run as a lab. The lab is compulsory and very important since it will help you get ready for all your assignments. Here are the requirements for the lab to be successful:

- Bring your laptop to class
- Your laptop will need to have been set-up for the lab in advance (Note: this set up is necessary for your assignments as well so it is a good idea to do it early and let Evan (the TA) and me know ahead of time if there are any problems).
- It would be useful for you to try the installation ASAP and go to Evan's office hour on Thursday (3pm-5pm in the DMTI Kitchen area) to get some help on it. Evan will also hold my regular office hour on Monday (2:30pm-3:30pm in the DMTI Kitchen area), and I will hold my regular office hour on Wednesday Jan 30 (1pm-2pm in DMTI 112-B).
- Here are the details of the set up:
    - Your laptop needs to be able to run a virtual machine (i.e., a software simulation of a real computer)
    - Your virtual machine should be able to run the Linux Operating System
- Prior to the Jan 30th lab, I will post a file containing C and Shell programs on Blackboard. These files will need to be saved in a directory in your Linux environment on your virtual machine.
- I will also post a set of instructions. You will be able to view this set of instructions in class.

There are several choices of **hypervisors** or **virtual machine managers** available on the internet. A hypervisor is a piece of software that creates and runs virtual machines. I selected **VirtualBox** which is free and reliable. Similarly, there are several choices of Linux Operating Systems. I selected **Linux Mint** Version 18.1, which was recommended to me and seems reliable as well. Please make the same choices so that I do not have issues testing your programs, once you hand in your assignments.

Here are the instructions to follow to prepare your laptop (these were prepared jointly by Evan and myself):

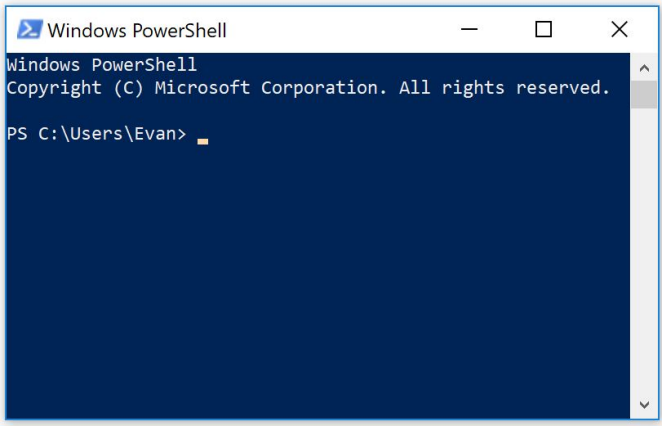
Step 1: Download and Install Vagrant

Vagrant is basically a configuration layer which sits on top of your virtual machine(s). You can download the appropriate version for your host OS here. To verify which version of Vagrant to download (e.g., Windows 32-bit / 64 bit), you need to verify which operating system or which operating system version your host machine is running. To do so for Windows, for example, click Start and type "system" in the Start Search Box. Then look at "System Type" and the version running will be clear. In my case, for example, I see "x64-based PC", which tells me to download the 64-bit version. Once Vagrant is downloaded, click to install it. If it comes up, say "Yes" to the question "Do you allow this software to make changes to your system?"

For the next part of this tutorial, we will need to use our system **command line interface** or **shell**. This is simply a text based interface for interacting directory with our operating system. Each operating system

may have several different shells. Windows users should use **Powershell**, and MacOS/Linux (i.e. *nix) users should use the default **Bash** shell.[1] Powershell shares many familiar Bash commands, such as `ls`, `pwd`, `cd`, `mkdir`, and `mv`, so both Powershell and Bash users should have no trouble following along.

To access Powershell, Windows users can simply type `Powershell` at the start search window. This will bring up a window known as a **terminal emulator**. Below, we show the default Windows terminal emulator.[2]



The text `PS C:\Users\Evan>` is known as a **prompt**. The `PS` at the beginning of the prompt indicates it is indeed a powershell prompt. `C:\Users\Evan` is the current directory. A yellow underscore indicates the cursor position. For brevity, we will abbreviate the prompt `PS C:\Users\Evan>` with `>`. E.g., `PS C:\Users\Evan> echo "hello world"` becomes `> echo "hello world"`. We will abbreviate a Bash style prompt as `$`. This should make it easier to see whether commands are being entered in the host shell (prompt: `>`), or the guest VM shell (prompt: `$`). By the way, when you copy/paste code, do not include the prompt!

MacOS users can use the built-in **Terminal** application, which by default runs the same Bash shell that we will use in Linux. Terminal can be found using the Spotlight search (Command + Space bar).

Once Vagrant is downloaded, both Windows and *nix users can verify that the installation was successful by typing `vagrant -v` in your system shell. If Vagrant is installed correctly, it will show its version number, as shown below.

```
> vagrant -v
Vagrant 2.1.5
```

---

[1] By default Windows includes two system shells; **Powershell** is the default command line for Windows 10, and is much newer, more powerful, and intuitive than the old command prompt **CMD**. Given the choice, Windows users should use powershell.

[2] The default Windows terminal emulator leaves much to be desired. Interested students may wish to consider Hyper, a free cross-platform terminal emulator. Hyper has nice features such as window splitting, which allows one to open several Powershell terminals in one Hyper Window.

Step 2: Download and Install VirtualBox

VirtualBox can be downloaded here. We will use it to download a "base image" or "box" from which virtual machines are built. Download VirtualBox 6.02 (or VirtualBox 5.2 if your computer runs a 32-bit Operating System). Once VirtualBox was downloaded, click to install it. Again, if prompted, say "Yes" to the question "Do you allow this software to make changes to your system?"

Step 3: Create a Directory to Store Your Virtual Machines

You need somewhere to store the virtual machine which will be created from the box. For our purposes, we'll create a directory named `OS-VM` (short "for operating systems virtual machine") as a subdirectory of your home directory. Your home directory is abbreviated by `~`. E.g, for me, `~` abbreviates `C:\Users\japkowic`. You then want to change into this directory. This can be done at the system command line [which you get to as above] by:

```
PS C:\Users\japkowic> mkdir ~/OS-VM
```

```
PS C:\Users\japkowic> cd ~/OS-VM
```

After performing these operations, your prompt will change to `PS C:\Users\japkowic\OS-VM>` to reflect the fact that the working directory has changed to `C:\Users\japkowic\OS-VM`.

Step 4: Download the Vagrantfile from Blackboard

The vagrant configuration file is called a Vagrantfile, and is literally named `Vagrantfile.` Download it from Blackboard from the folder called "Virtual Machine Set Up" in the Content area.

Step 5: Move the Vagrantfile to the Directory `~\OS-VM`

Move the Vagrantfile you downloaded to `~\OS-VM`. Do not change the file extension of the Vagrantfile (by default it has no extension, this is correct. E.g., do not rename it Vagrantfile.txt). To do so in Windows, you can open two directories: "`~\Downloads`" and "`~\OS-VM`" and drag the Vagrantfile from the Download directory to the OS-VM directory. If you want to do this from the command line, do:

```
PS C:\Users\japkowic> mv ~/Downloads/Vagrantfile ~/OS-VM/Vagrantfile
```

Step 6: Start the Virtual Machine

From the Command Line, go to the directory `~/OS-VM`, by typing, from the command prompt,

```
PS C:\Users\japkowic> cd ~/OS-VM
```

From the command line, run `vagrant up`. The first time you run this, vagrant will download the `artem-sidorenko/mint-18.1-cinnamon` box, from which the virtual machine is built. The file is

quite large (~4GB), so this may take a while. It should look something like this (Note: this time the user is Evan rather than “japkowic”. On your machine, your user name will show up). In terms of delay, please note that it may take a minute or so before you see the following. After you see that, expect something in the order of 10 minutes as the program is very large. There will be a line updating you on percent progress and estimated time remaining.

```
PS C:\Users\Evan\OS-VM> vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'artem-sidorenko/mint-18.1-cinnamon' could not be found. Attempting to find and install...
    default: Box Provider: virtualbox
    default: Box Version: >= 0
==> default: Loading metadata for box 'artem-sidorenko/mint-18.1-cinnamon'
    default: URL: https://vagrantcloud.com/artem-sidorenko/mint-18.1-cinnamon
==> default: Adding box 'artem-sidorenko/mint-18.1-cinnamon' (v0.0.2) for provider: virtualbox
    default: Downloading: https://vagrantcloud.com/artem-sidorenko/boxes/mint-18.1-cinnamon/versions/0.0.2/providers/virtual
box.box
```

Step 7: SSH into Virtual Machine

In the directory `~/OS-VM`, run `vagrant ssh`

```
PS C:\Users\Evan\OS-VM> vagrant ssh
Welcome to Linux Mint 18.1 Serena (GNU/Linux 4.4.0-53-generic x86_64)

 * Documentation:  https://www.linuxmint.com
Last login: Tue Nov  1 21:50:11 2016 from 10.0.2.2
vagrant@localhost ~ $ echo "welcome to linux"
welcome to linux
```

Note that the Windows style prompt (`>`) has changed to a Linux style prompt(`$`), confirming you are indeed connected to the VM. At this point, you have a fully functioning virtual machine running Linux Mint 18.1.

Note: whenever you want to work on your assignments for the course, you will have to do the following:

- Move to the OS-VM by typing: `cd ~/OS-VM`run
- Start the virtual machine by typing `vagrant up` (this will be way faster than the first time you do it)
- SSH into the the Virtual Machine by typing `vagrant ssh`
- Once that’s done, you will be able to write programs using a convenient editor (see below), compile your programs using the C and C++ compiler gcc (Note: your program’s extension will have to be .c) and run your programs by typing `./a.out`.

Theoretical Discussion

We now discuss how to intelligently use the VM. This discussion was included to explain to you what is happening between the host OS (e.g., Windows) and the guest OS (Linux). After this discussion, we will get back to more practical guidelines.

VMs have several important advantages for programmers:

- everyone develops on exactly the same system, regardless of the host OS. Thus “works on my machine” bugs are eliminated.
- The VM can be damaged or even destroyed without any consequences to the host OS. This is very useful for beginning OS programmers!
- The VM can be quickly rebuilt from the box.

Below, we illustrate two workflows. In the first workflow [which we will not use here, unless you already know this approach and are comfortable using it], illustrated in figure 0.0.1, everything is done on the VM. Files are stored, edited, compiled and run - all on the VM. This has several downsides

- Since VMs are run “headless” i.e. with no GUI, we can’t use modern text editors like V.S. Code or Sublime Text this way. We must use text-based editors over the terminal like vi and Emacs, both of which date to the mid 1970s, and have large learning curves.
- Since the files are stored on the VM, if the VM is deleted (or say accidentally destroyed), so are your files

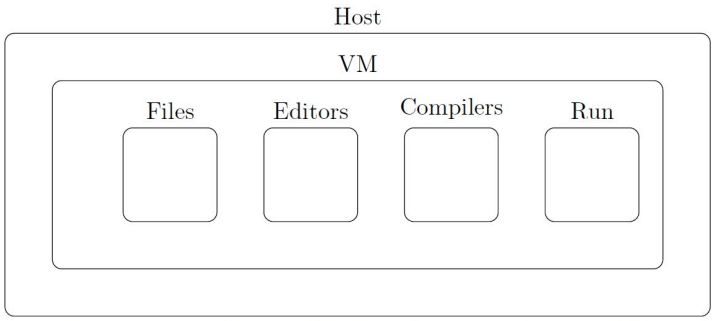


FIGURE 0.0.1. Naive VM Use - Everything Done on VM

The smarter way of using a VM [which we will use] is shown in figure 0.0.2. In this workflow, files are saved in a shared folder, edited using modern GUI editors on the host OS, and compiled and run on the guest OS (i.e. the VM). The key difference is the use of shared folders. Shared folders are useful since files in them can be:

- accessed from both the host OS (e.g., Windows) and the guest OS (in our case, Linux Mint 18.1)
- preserved even if the VM is destroyed, keeping your work safe.
- edited using a modern editor of choice on the host machine (e.g., Windows) such as Word, though, for programming, Evan recommends V.S. Code (See below for an introduction to V.S. Code) .

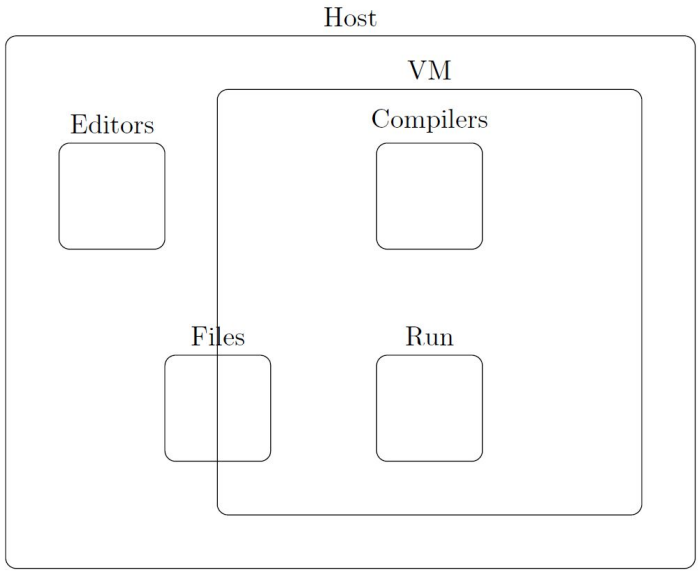


FIGURE 0.0.2. Intelligent VM Use

Once you've finished writing your code in your editor of choice and saved your program in a shared folder, you can then simply SSH into the VM, change into the shared directory, and use excellent linux compilation tools such as `gcc` to compile your programs, which can then be executed on the VM. In the next section, we discuss how to use shared folders in practice and first time setup.

Using Shared Files

By default, Vagrant sets the shared directory on the host to be the location of the Vagrantfile. In our case, this is `~/OS-VM`. This is mapped to `/vagrant` on the VM. Thus any files you put into `~/OS-VM` on the host will instantly be shared to `/vagrant` on the guest and vice versa. We illustrate this below on a Windows host (top) with a linux VM (bottom). Note that, for instance, the "Vagrantfile" document (which in this case is the Vagrantfile) is shared between host and guest machines. Vagrant takes care of the creation of these basic shared folders for us (although we can add more shared folders by manipulating the vagrantfile (line 44)).

```
> cd ~/OS-VM
> ls

    Directory: C:\Users\Evan\OS-VM

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        1/17/2019  10:49 PM                .vagrant
-a----        1/21/2019   3:40 PM           3135 Vagrantfile
```

```
vagrant@localhost ~ $ cd /vagrant
vagrant@localhost /vagrant $ ls
Vagrantfile
```

First Time Setup

After connecting to your VM for the first time via SSH, run the following commands. Be sure to do this prior to trying to compile and programs with `gcc`.

```
$ sudo apt-get update
$ sudo apt-get install build-essential
```

The Visual Studio Code Editor

As previously discussed, an intelligent VM workflow takes advantage of the host GUI. A GUI is particularly useful when editing code, as this enables the use of a mouse, icons, menus, etc. Modern text editors provide some or all of the following features:

- Error checking
- Code navigation
- Code completion
- Code coloring/syntax highlighting
- Version control integration
- Integrated debugging tools
- Extensions/Add-ons/Plugins

For purposes of this class, students may use a text editor of their choice. For students who do not already have a strong preference, we recommend **Visual Studio Code** aka **V.S. Code**. In choosing to recommend a GUI text editor, we follow the wisdom of the crowd. According to the Stack Overflow Developer Survey (2018), **Visual Studio Code** is currently the most popular text editor in the world. V.S. Code is supported by Microsoft, but is free, open source, cross-platform, relatively fast, and extensible. Users can download the correct version for their OS here. Because of its popularity, there are a massive number of self help resources and tutorials. This YouTube channel has good series of tutorial videos, which might be worth a watch.

Summary

1. Install Vagrant from here.
2. Install VirtualBox from here.
3. Create a Directory to Store your VMs and change into it. This is done by:

```
> mkdir ~\OS-VM
> cd ~\OS-VM
```

4. Download the Vagrantfile from Blackboard
5. Move the Vagrantfile to the Directory `~\OS-VM` via

```
> mv ~/Downloads/Vagrantfile ~/OS-VM/Vagrantfile
```

6. Start the Virtual Machine: In the directory `~\OS-VM`, run `> vagrant up`
7. SSH into Virtual Machine: In the directory `~\OS-VM`, run `> vagrant ssh`
8. After SSHing into your machine for the first time only, run the following:

```
$ sudo apt-get update
$ sudo apt-get install build-essential
```

9. Run Programs! Write them using an editor in your host OS, save them to a shared folder, and compile and run them on the guest VM.
10. Logout of the virtual machine by typing `$ logout` at the linux prompt (i.e in your VM). Back at the host OS prompt, you can shutdown the VM with `> vagrant halt`. You can restart it with `> vagrant up.` For additional shutdown options (e.g. destroying the machine) see *here*.

Troubleshooting

If you have any problems setting up, the following websites contain useful information:

http://www.howtogeek.com/196060/beginner-geek-how-to-create-and-use-virtual-machines/

http://www.instructables.com/id/How-to-install-Linux-on-your-Windows/ (press on the steps one by one

For more on Vagrant, see:

Hashimoto, Mitchell. *Vagrant: Up and Running: Create and Manage Virtualized Development Environments*. " O'Reilly Media, Inc.", 2013.