Becoming Supercalifragilistic

Coding in a Linux Terminal

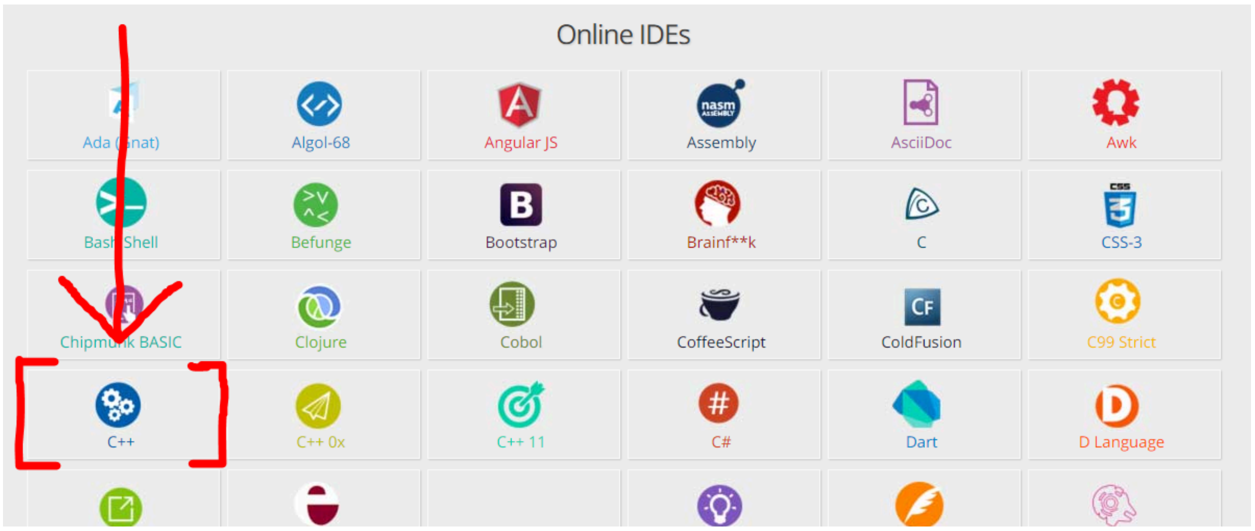
This document details two (2) ways to gain access to a Linux environment. The first method, through “CodingGround”, is the quickest to start up but does not offer much customizability and will be slower to code, compile, and test in. The second method, installing a virtual machine, allows greater customizability and speed.

CodingGround

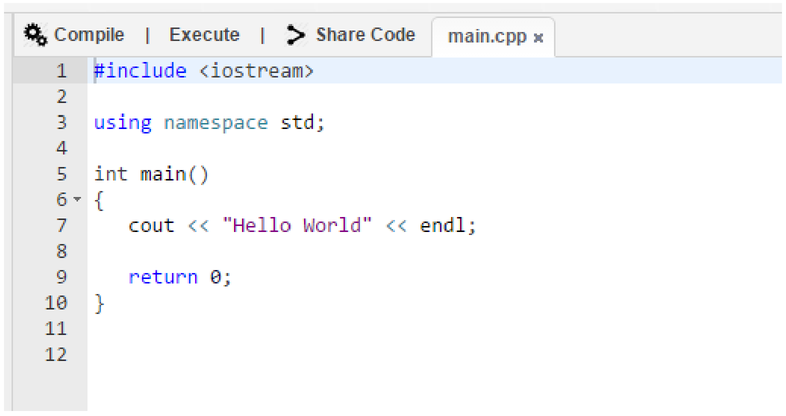
(Good)

Here is the link to CodingGround: http://www.tutorialspoint.com/codingground.htm

Look for the C++ IDE. It might be in a different position based on your screen size.



You will find yourself looking at an Online IDE for C++. Below is an image of the editor where you will type your source code.



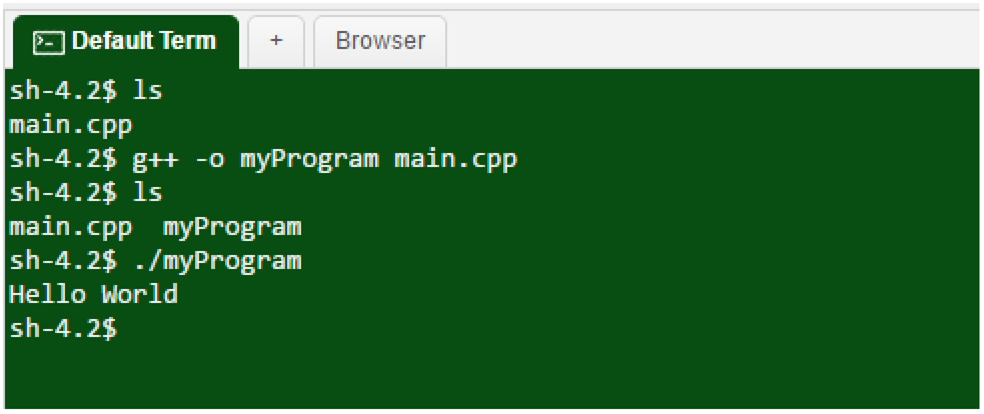
```
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello World" << endl;

   return 0;
}
```

At the bottom of the screen is a *terminal.* The terminal will allow you to compile and run your code. You can use the editor's Compile and Execute buttons if you want, but I suggest you gain some familiarity with using the terminal. The terminal looks like this:



```
sh-4.2$ ls
main.cpp
sh-4.2$ g++ -o myProgram main.cpp
sh-4.2$ ls
main.cpp  myProgram
sh-4.2$ ./myProgram
Hello World
sh-4.2$
```

Anything you see after the **$** symbol is text I entered as commands. Let's take a second to understand these commands in order.

`$ ls`

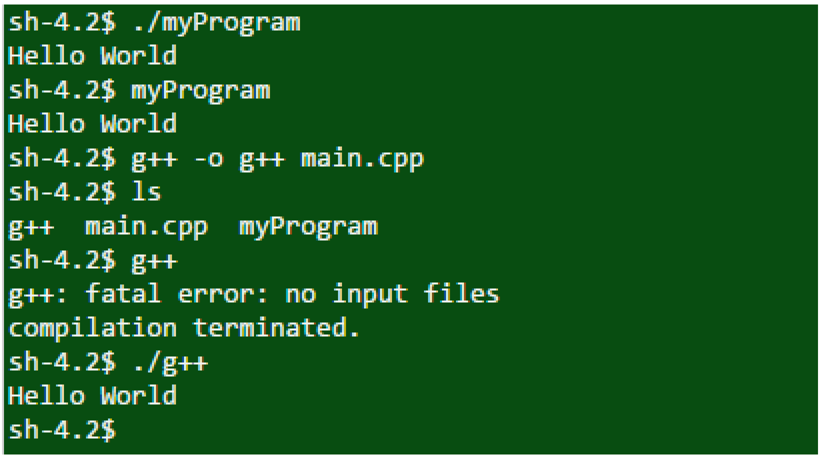
This is the list command. It lists the files in the current folder/directory.

`$ g++ -o myProgram main.cpp`

g++ is a compiler for c++ code. We use the -o *flag* here to tell g++ that we want our output to be called myProgram. Then we tell g++ to compile the main.cpp source code file. After this command completes, I ran another `ls` to show the new myProgram executable listed next to the source code file in our directory.

`$ ./myProgram`

This command runs the executable in the terminal. We then see the output, "Hello World", written right to the terminal. You could simply type **`$ myProgram`** to run the executable, but we use the `./` to specify that we want to run the myProgram that is in our current directory. Below is an image demonstrating this process.



```
sh-4.2$ ./myProgram
Hello World
sh-4.2$ myProgram
Hello World
sh-4.2$ g++ -o g++ main.cpp
sh-4.2$ ls
g++  main.cpp  myProgram
sh-4.2$ g++
g++: fatal error: no input files
compilation terminated.
sh-4.2$ ./g++
Hello World
sh-4.2$
```

Virtual Machines

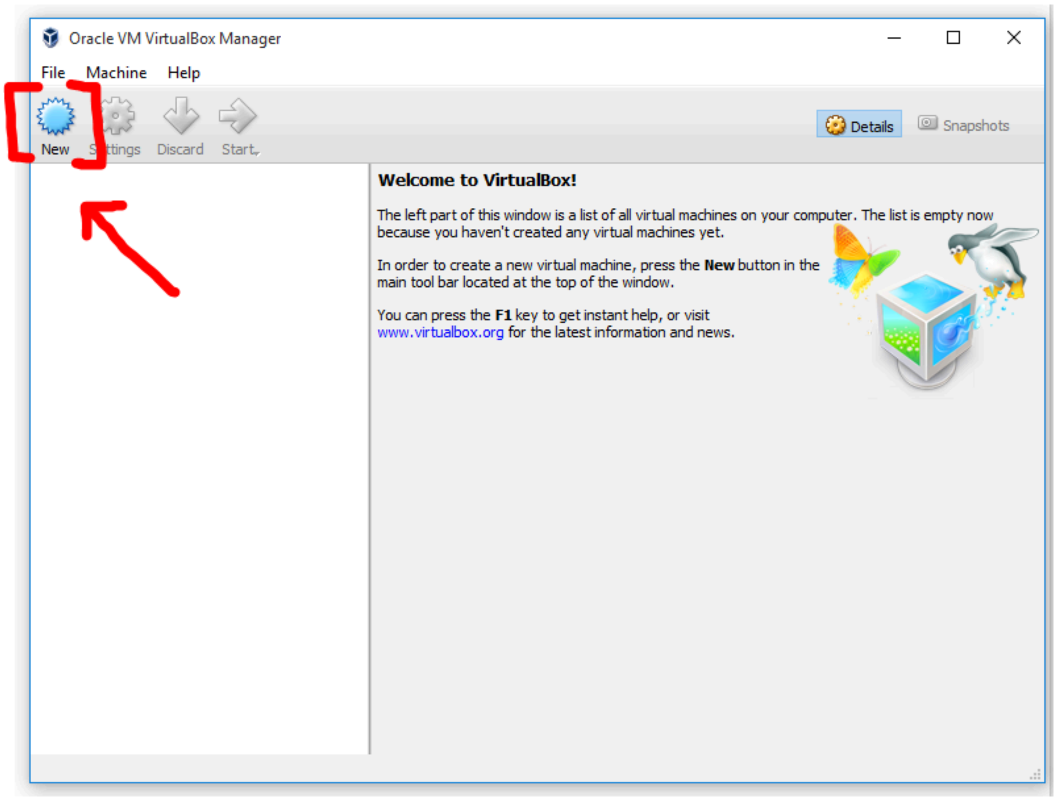
(Better)

First, we need to download and install VirtualBox. Go to this address and click the Windows Hosts link to download VirtualBox: https://www.virtualbox.org/wiki/Downloads
Run the installer accepting all the default options.

Now we will need an Operating System to run inside VirtualBox. I recommend Ubuntu which can be found here: https://www.ubuntu.com/download/desktop

Ubuntu an easy entry into the Linux world and is one of the most popular distributions. If you want to look at other distributions (or “distros”) of Linux, check out this list at DistroWatch: https://distrowatch.com/dwres.php?resource=major

Once you’ve downloaded an OS image, launch VirtualBox. You should see something similar to the image below. Click New.



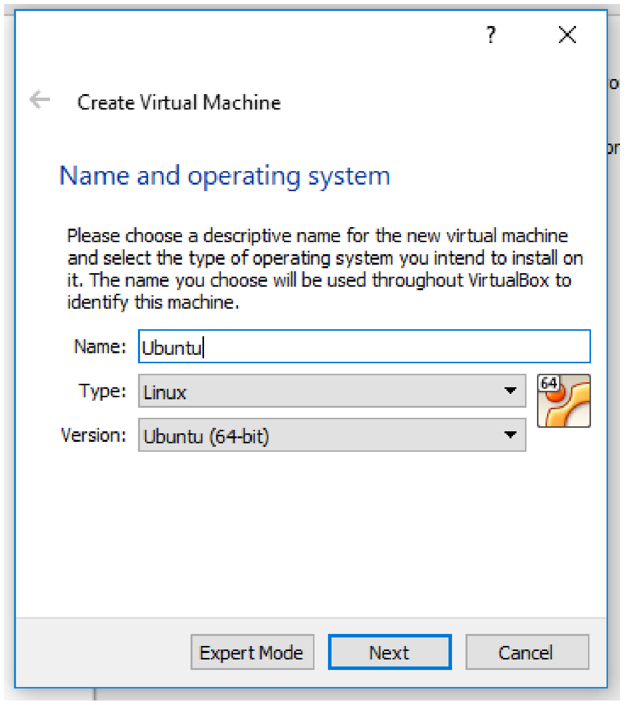
Welcome to VirtualBox!

The left part of this window is a list of all virtual machines on your computer. The list is empty now because you haven't created any virtual machines yet.

In order to create a new virtual machine, press the **New** button in the main tool bar located at the top of the window.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for the latest information and news.

Now, let’s name our virtual machine and set it to Ubuntu (64-bit).



Name and operating system

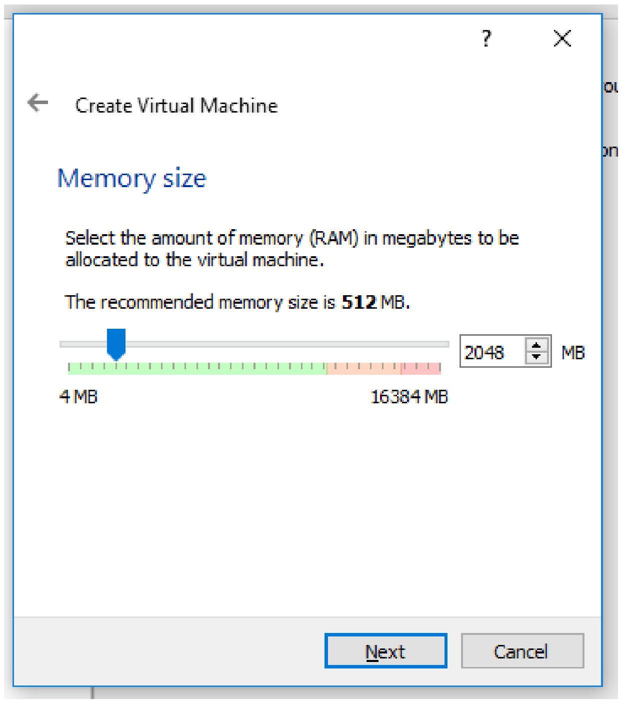
Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name: Ubuntu

Type: Linux

Version: Ubuntu (64-bit)

Then, let’s set the amount of RAM for our virtual machine. You can go with the value VirtualBox suggest, but I always give my machines a bit more RAM than they need for reasons I haven’t yet identified.



Memory size

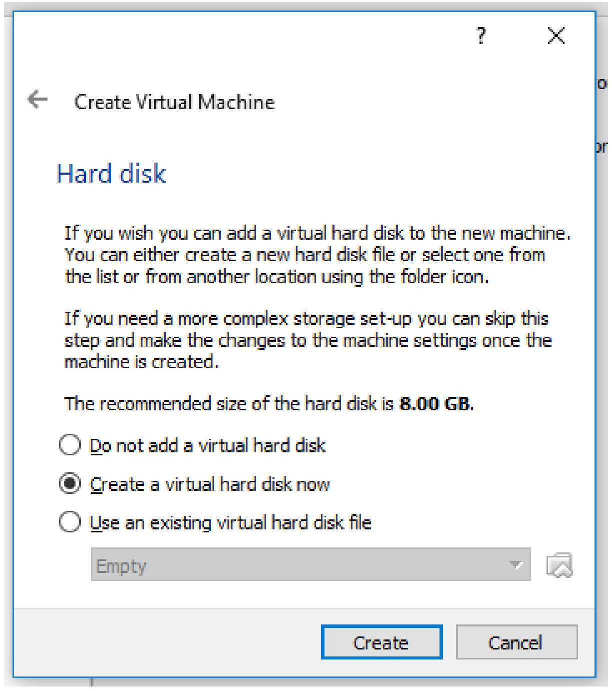
Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **512** MB.

2048 MB

Looks good to me!

Now we create a virtual hard disk for our machine to save data to. Again, you can go with the default values given by VirtualBox here. If you are going to be using this machine a lot and will need to save a lot of files and programs, you might consider allocating more space. The decision between fixed size or dynamic is up to you. I went with fixed size here.



Hard disk

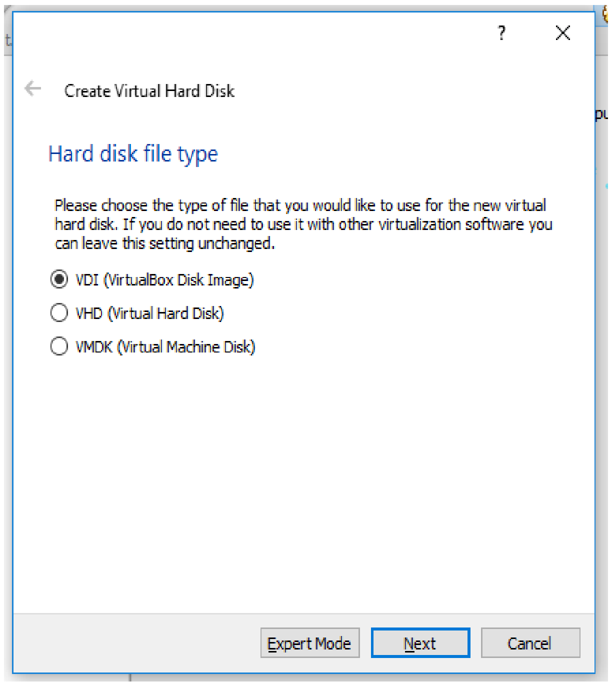
If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select one from the list or from another location using the folder icon.

If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.

The recommended size of the hard disk is **8.00 GB.**

- ( ) Do not add a virtual hard disk
- (●) Create a virtual hard disk now
- ( ) Use an existing virtual hard disk file

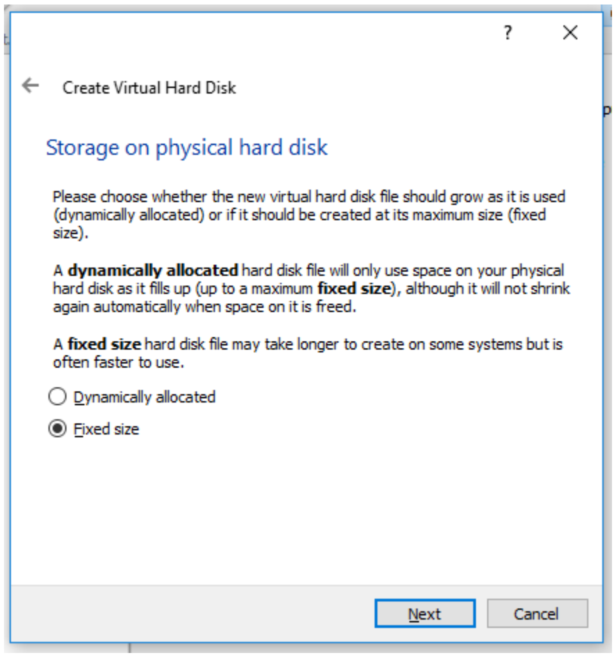
Empty



Hard disk file type

Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.

- (●) VDI (VirtualBox Disk Image)
- ( ) VHD (Virtual Hard Disk)
- ( ) VMDK (Virtual Machine Disk)



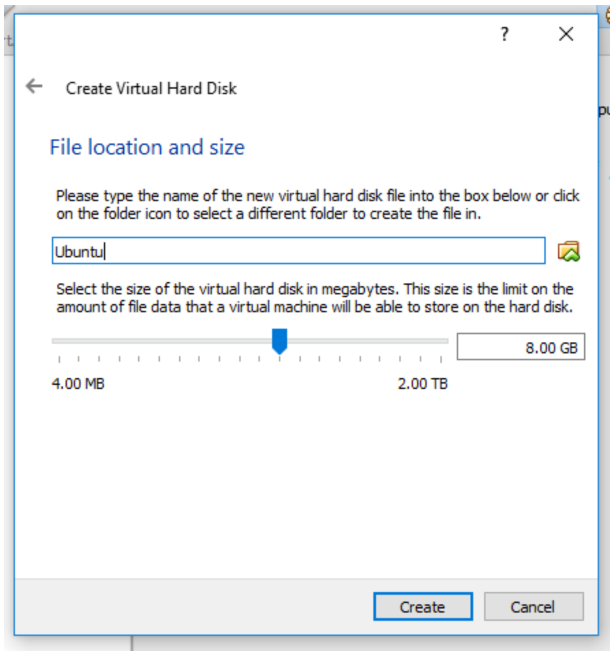
Storage on physical hard disk

Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).

A **dynamically allocated** hard disk file will only use space on your physical hard disk as it fills up (up to a maximum **fixed size**), although it will not shrink again automatically when space on it is freed.

A **fixed size** hard disk file may take longer to create on some systems but is often faster to use.

- ( ) Dynamically allocated
- (●) Fixed size



File location and size

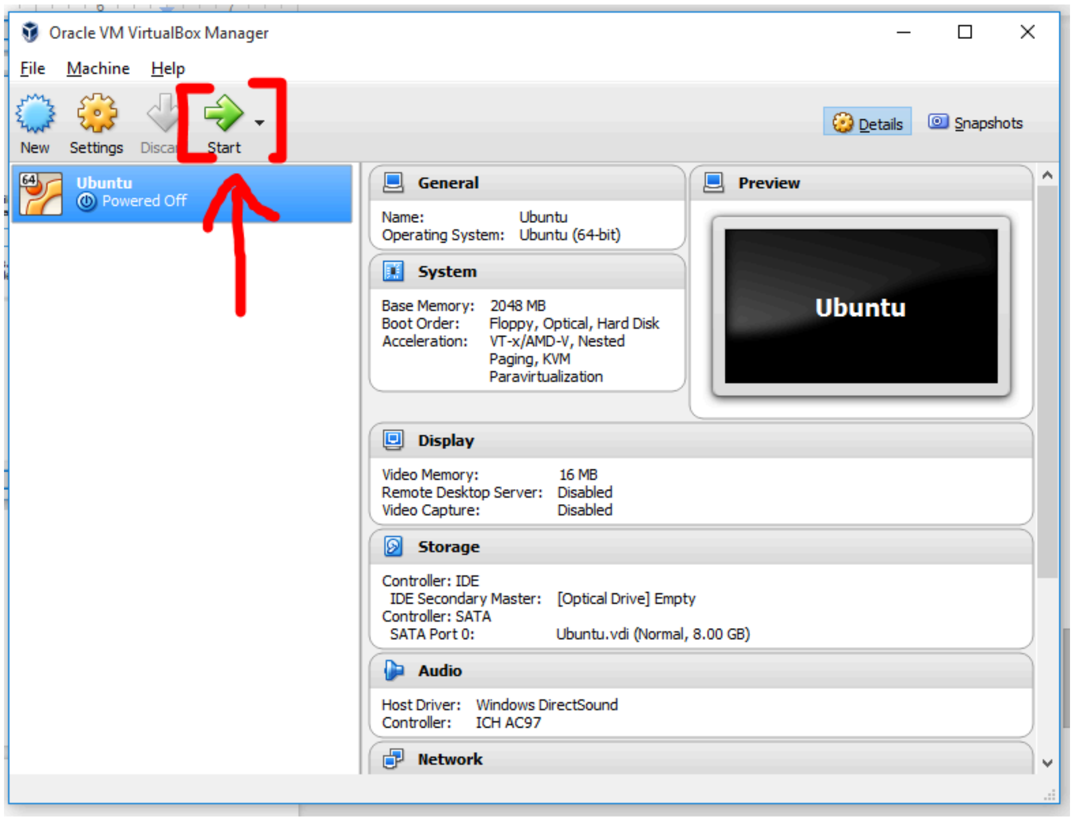
Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.

Ubuntu

Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.

8.00 GB

Great. Now our machine has been created. Let's boot 'er up!



General
Name: Ubuntu
Operating System: Ubuntu (64-bit)

System
Base Memory: 2048 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Preview
Ubuntu

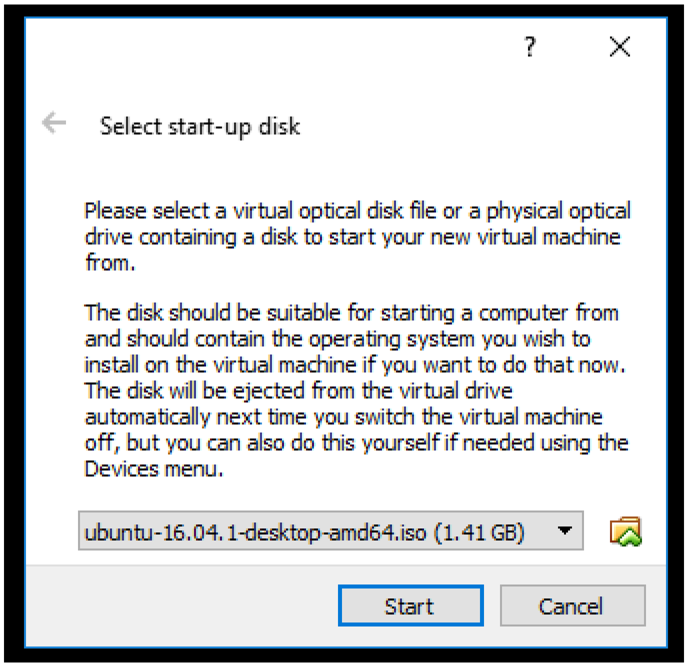
Display
Video Memory: 16 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Storage
Controller: IDE
IDE Secondary Master: [Optical Drive] Empty
Controller: SATA
SATA Port 0: Ubuntu.vdi (Normal, 8.00 GB)

Audio
Host Driver: Windows DirectSound
Controller: ICH AC97

Network

And then tell the machine where your .iso file is.



Select start-up disk

Please select a virtual optical disk file or a physical optical drive containing a disk to start your new virtual machine from.

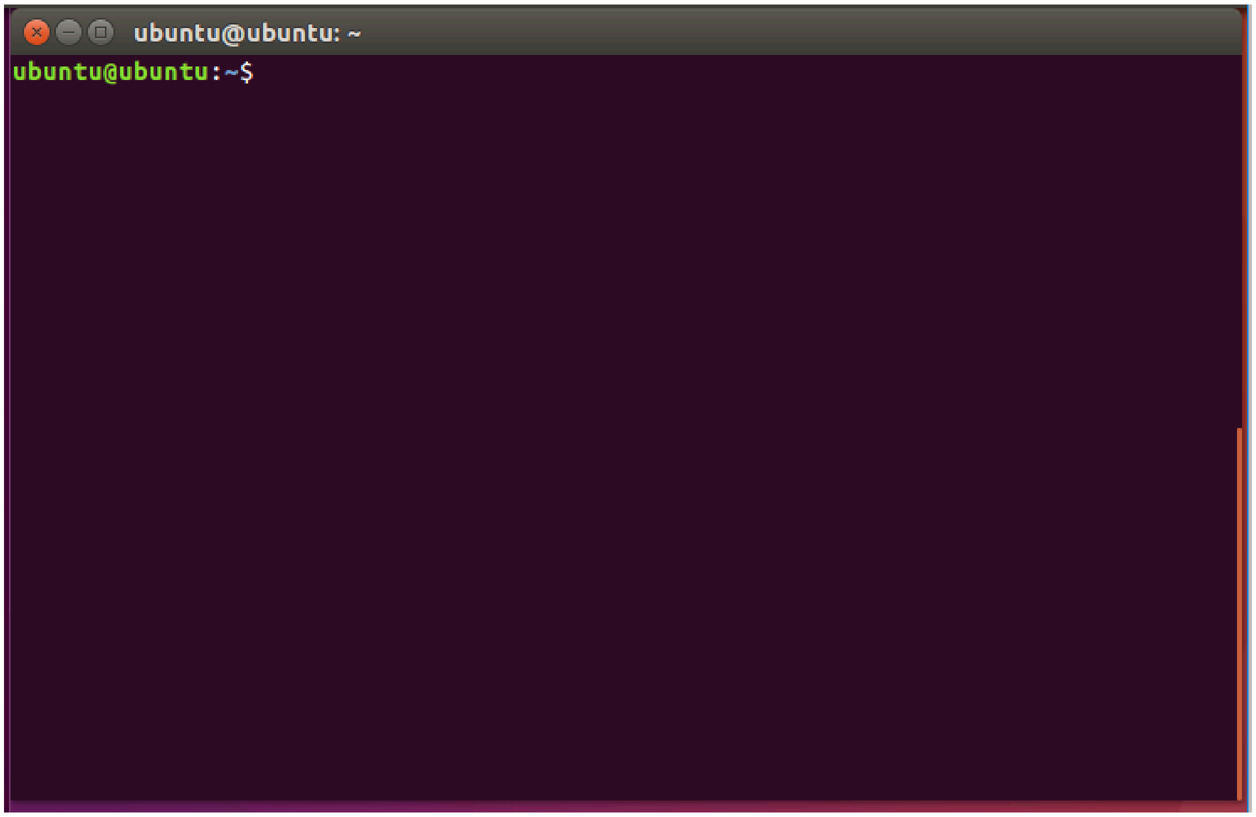
The disk should be suitable for starting a computer from and should contain the operating system you wish to install on the virtual machine if you want to do that now. The disk will be ejected from the virtual drive automatically next time you switch the virtual machine off, but you can also do this yourself if needed using the Devices menu.

ubuntu-16.04.1-desktop-amd64.iso (1.41 GB)

After a minute, you'll be presented with a menu. For now, let's "Try Ubuntu without installing." You should now be inside a "live" version of your OS. This is running right off the .iso file. If you want, you can choose to install your OS onto your new virtual hard disk, but that process is not covered in this document.

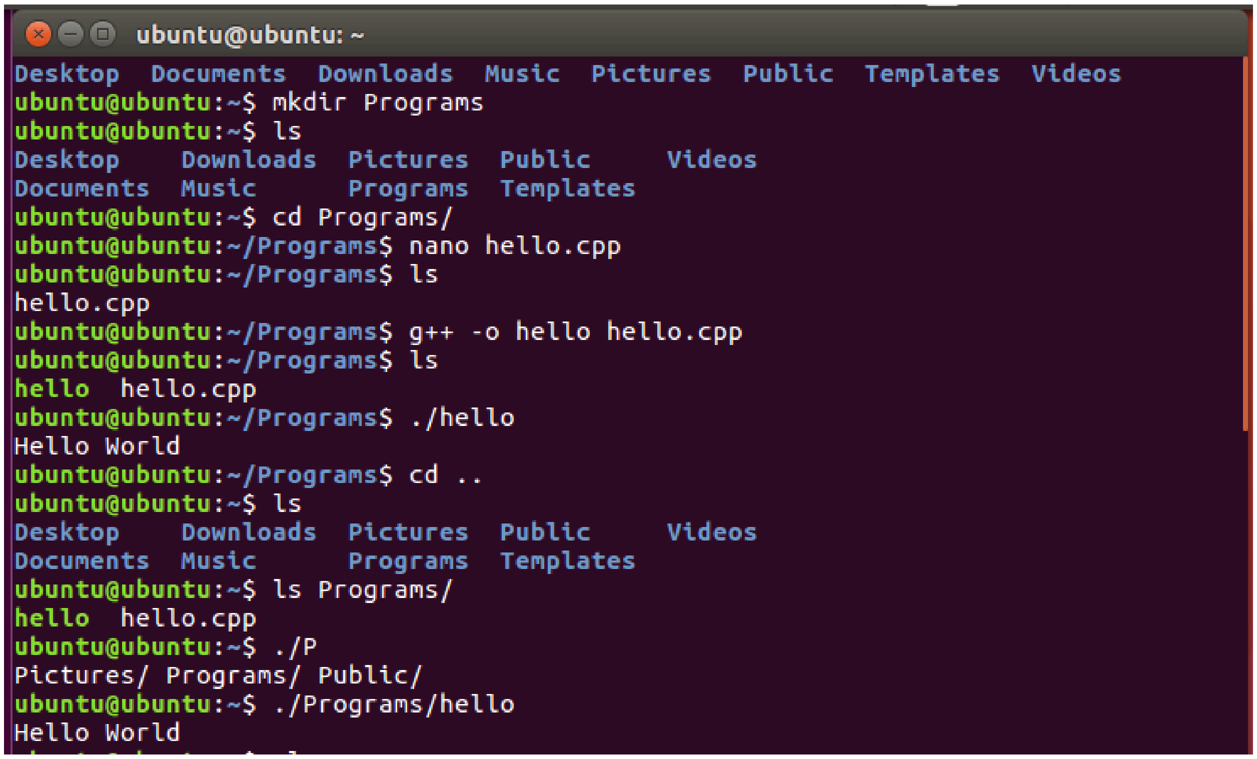
To login, enter "ubuntu" as your username (without the quotes). There is no password, so hit enter on the password prompt. You should now be logged in! Congrats. You've got a linux virtual machine.

Let's open a terminal! Click inside the virtual machine and then press Ctrl+Alt+T together to open a new terminal. You should see something like this pop up:

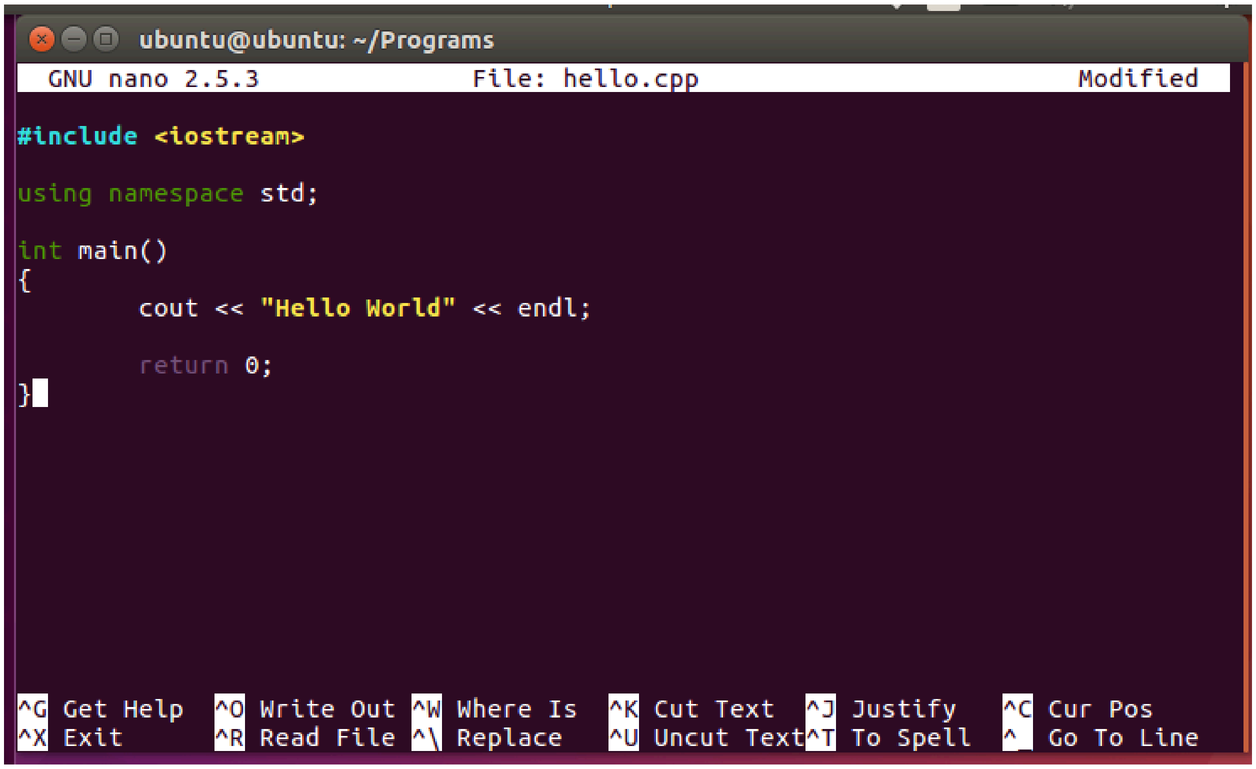


```
ubuntu@ubuntu:~$
```

The following page shows two images. The first image is a set of commands run inside the terminal. The second image shows the nano file editor running within the terminal.



```
ubuntu@ubuntu: ~
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
ubuntu@ubuntu:~$ mkdir Programs
ubuntu@ubuntu:~$ ls
Desktop    Downloads  Pictures  Public     Videos
Documents  Music      Programs  Templates
ubuntu@ubuntu:~$ cd Programs/
ubuntu@ubuntu:~/Programs$ nano hello.cpp
ubuntu@ubuntu:~/Programs$ ls
hello.cpp
ubuntu@ubuntu:~/Programs$ g++ -o hello hello.cpp
ubuntu@ubuntu:~/Programs$ ls
hello  hello.cpp
ubuntu@ubuntu:~/Programs$ ./hello
Hello World
ubuntu@ubuntu:~/Programs$ cd ..
ubuntu@ubuntu:~$ ls
Desktop    Downloads  Pictures  Public     Videos
Documents  Music      Programs  Templates
ubuntu@ubuntu:~$ ls Programs/
hello  hello.cpp
ubuntu@ubuntu:~$ ./P
Pictures/ Programs/ Public/
ubuntu@ubuntu:~$ ./Programs/hello
Hello World
```



```
ubuntu@ubuntu: ~/Programs
  GNU nano 2.5.3              File: hello.cpp                          Modified

#include <iostream>

using namespace std;

int main()
{
        cout << "Hello World" << endl;

        return 0;
}










^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

So, what’s going on here? We start in the root directory. Let’s see what’s here.

```
$ ls
```

This lists the files and folders in the current directory.

```
$ mkdir Programs
```

Now, we **m**a**k**e and new **dir**ectory called Programs

```
$ ls
```

Running ls again, we see the new Programs folder

```
$ cd Programs
```

Change **D**irectory to Programs

```
$ nano hello.cpp
```

Here’s the fun stuff. Nano is a text editor that runs within the terminal. It is very minimal and lame, but works for this demonstration. For a better terminal editor with more features and awesomeness, I recommend installing vim (or emacs I guess).
Inside nano, I created the same HelloWorld program as in the CodingGround example. After the code is typed, pressing Ctrl+O lets us “write” the file (save it to the disk). We have the option here to rename the file. Once written, Ctrl+X exits nano.

```
$ ls
```

We now see the new hello.cpp file ready for us to compile.

```
$ g++ -o hello hello.cpp
```

Again, we use the -o flag to name the output “hello” and then name the source code file to compile.

```
$ ./hello
```

./ Runs the program and we see the output.

```
$ cd ..
```

Changing directory again. The “..” changes to the directory *Above* where we currently are. In this case, back up to the root directory.

```
$ ls
```

Listing the files/folders, we see our new Programs folder, but what’s in it?

```
$ ls Programs/
```

This lists what’s in the specified folder, Programs. Wow!

```
$ ./P [tab]
```

Here, I typed ./P then pressed tab. Tab will normally auto-complete our command. In this case, there were several things that start with P, so the terminal listed them for us.

```
$ ./Programs/hello
```

This runs the file /Programs/hello.

Tada! Not so bad, right?

Dual Booting
(Best)

Dual Booting partitions sections of your hard drive for multiple operating systems so that they may coexist on your computer. When you first boot up a dual-boot system, a menu will ask you which operating system you want to use. This is what I do--though to be honest, I never boot into my Windows OS (except to create this file and take screenshots). If you're serious about learning Linux (which you should be), dual booting is the way to go. You're never without access to Windows, but you still have the power of a full Linux OS operating directly on your hardware, not limited through a virtual machine.

Setting up a system for Dual Booting is time consuming and not covered in this document due to its more advanced technical nature. If you'd like help setting up a dual boot or have **any other questions** please feel free to email me at:
mymailthatis@gmail.com
Or text me (Ryan) at (240) 344-1202

Hope this helps!