Beginning Linux for the Particle Physicist

According to David Black

Most recent version and easy-to-navigate html version (under construction) at my website <http://bballdave025.kodingen.com>

Before you run away because of the long length of this article, stop and read this. It's supposed to be a comprehensive guide to installing Linux on VirtualBox and learning beginning coding, including ROOT. However, there are many solutions to small little problems and bugs that could be very helpful for you. Go ahead and search through the text if a search for a solution has brought you here. Anyway, now onto the information:

I've written this guide hoping to keep other people from spending all the time I did trying to figure out how this [insert cuss word of choice] thing works. I write it from the perspective of a physics major/Ph.D. candidate as well as from the perspective of a computer science minor. I am at heart a Windows user, and I didn't want to have to do a different boot to go from Linux to Windows. I looked for a solution that allowed me to use the programming tools provided by Linux, while still having my native Windows for Mathematica, Matlab, Word, Excel, etc. The solution I found was virtualization, but we'll get to that later.

What I'm presenting here is a way to set up Fedora Linux using Oracle VM VirtualBox. This basically has the whole Linux operating system as a window that you can minimize, put off to the side, do whatever with. After getting that set up and making it usable, I'll go on to show you a little of how to do programming from the terminal (command line) as well as with some development tools (like gedit and KDevelop). Some things might have changed since I made this tutorial in June 2011. If a screen comes up that's not in the tutorial, the default will usually work. Basically, don't freak out if everything isn't exactly the same. Trust the installation instructions.

After that, I'm going to show you an example program that is basically a very simple particle simulator, but that will allow you to see some pretty cool things, e.g. example tracks from the decay of an Ω particle that will be similar to those seen when it was discovered, all in 3D. Finally, I'll show you how to set up ROOT, which is obviously very important to the particle physicist. ~~As I plan to go into heavy ion physics myself, I also hope to learn how to get PYTHIA going, and show you some examples from that.~~

I ran into a bunch of problems as I tried to set this up, and I'll hopefully be able to share some insights on problem solving. My basic philosophy, and my plan of attack for solving the problems I ran into, can best be summarized by an xkcd comic <http://xkcd.com/627/>. Basically, type in your error into Google and look for the solution. At times, this took me 4-5 hours, but I could usually find a solution that worked even for the beginning Linixist that I was/am. (I usually found the solution without having to go deep into CS land.) Good luck!

Note that the most recent version of this tutorial should be at my website <http://bballdave025.kodingen.com>. It should be at a link, along with some other cool stuff. I have all the info about The "ABC" Particle Physics Simulator there as well as the relevant files.

Also, a little disclaimer: by using these instructions, you accept the risk of messing up your computer if you use these instructions. You should be fine, but I'm no expert. Just letting you know I can't be held responsible for problems. I've been fine, as have several of my friends who have tried the same thing.

I'd love to hear what you think and to hear about all the problems/unclear things that I'm sure I have in here. Feel free to let me know if you have any questions as well. I can't promise a swift response--I'm a graduate student in physics (at least while I'm writing this.)

v8 published 2012-06-12 bballdave025[at]yahoo[dot]com

Navigation

This document is pretty big; but don't get scared. A lot of the size comes from all the screenshots I've put in. I think you could get through this whole thing in five hours; and I'd be really surprised if it took you more than 45 minutes to do any one section. My buddy got through all the Linux stuff (up to "Starting C") in about 2 hours, and he said he was really careful and double-checked everything.

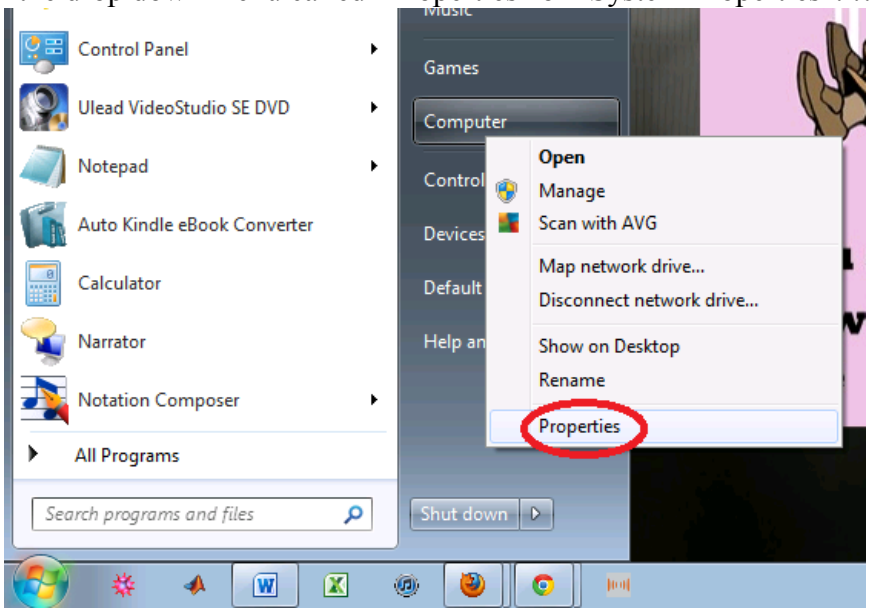
Anyway, I'm putting here some internal hyperlinks that you can use if you're viewing this as a Word document in .DOCX format (and maybe in some other formats, too.) Press and hold either "ctrl" button on your computer and then click on the link to navigate there.

YOU USE THESE INSTRUCTIONS AT YOUR OWN RISK. IT WORKED FOR ME BUT THERE IS NO GUARANTEE THAT, EVEN IF YOU FOLLOW THESE DIRECTIONS EXACTLY, YOU WON'T MESS UP YOUR COMPUTER IN SOME IRREVERSIBLE WAY OR OPEN YOURSELF UP TO ATTACKS FROM HACKERS. YOU ARE WARNED!

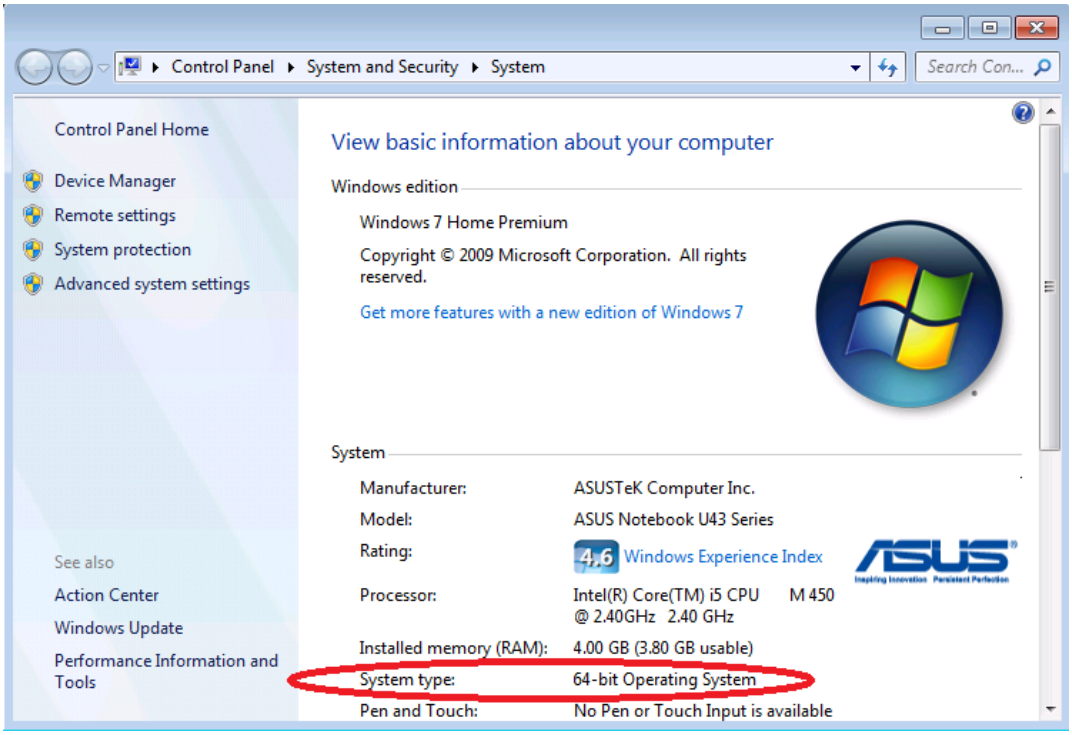
Okay, now that I put up a "just in case" warning, I'll let you know that I've been running my machine like this for 3 years and have had no problems. I've had several friends also follow these instructions with no problems. Quit freaking out.

System goto Navigation

Note that the instructions and screenshots I have here are for installing everything on a Windows 7 system, but other systems work pretty similarly. I've done all the internet stuff on Firefox, but it's not too bad of a cross-over for other browsers. Also, I have a 64-bit system and am installing a 32-bit Linux system. You might need to download different versions of everything if your system is different. If you're on Windows, get to "My Computer" or "Computer", either from the start menu or from its icon on the desktop. Right click it, and choose the option from the drop down menu called "Properties" or "System Properties". …



…That will bring up a window like this:

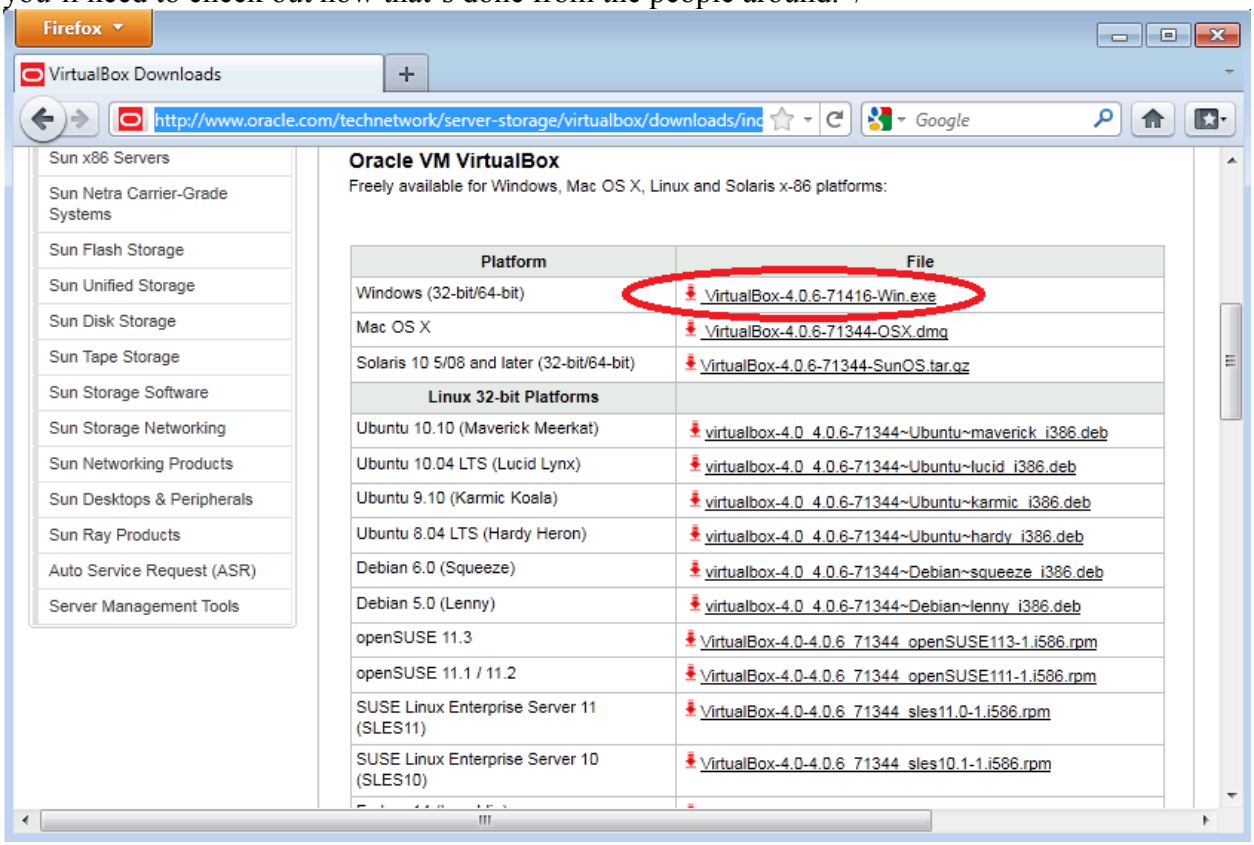


which will tell you what type of operating system you have. Obviously, mine is a 64-bit system. For older versions of Windows, you might need to: Click the "Advanced" tab; Click "Environment Variables"; In the "System variables" list at the bottom of the "Environment Variables" window, look for a variable called "PROCESSOR_ARCHITECTURE". If your PC has a 32-bit processor, this variable will have a value of "x86". If it has a 64-bit processor this variable will have a value of "x64".

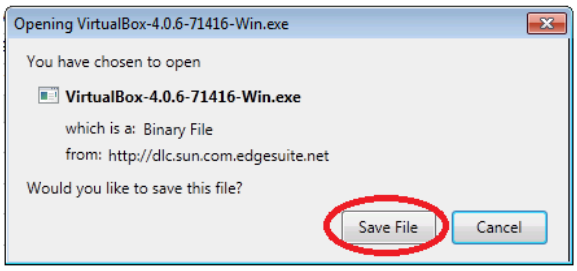
VirtualBox goto Navigation

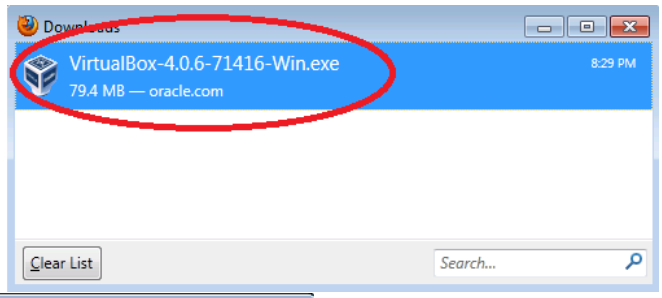
Now that you know this, the first thing you need to do is to install Oracle VM VirtualBox. You can use another program for virtualization, though I would seriously advise against using Windows Virtual PC. I tried this myself, and after way too many hours, I gave up. At the time I'm making this guide, the most current version is 4.0.6, so I'm installing it. Your numbers might be different. Note that since I started writing this tutorial, VirtualBox upgraded to 4.0.8. This version is necessary for Fedora 15, which is what I load here.

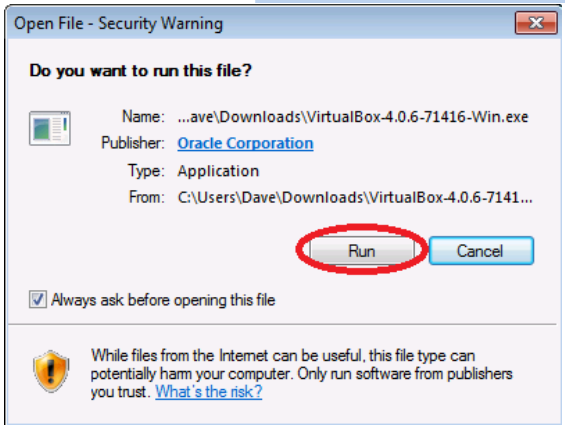
Go to the VirtualBox Downloads website <http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html#vbox> and click on the download that pertains to your system architecture. /*If you're at a place where you need to get the internet through a proxy, you'll need to check out how that's done from the people around.*/



When it asks if you want to save the file, click on “Save File”. When the “Downloads” window comes up, double click on the “VirtualBox-4.0.6-71416-Win.exe” link and click on “Run” when it asks if you want to run it.

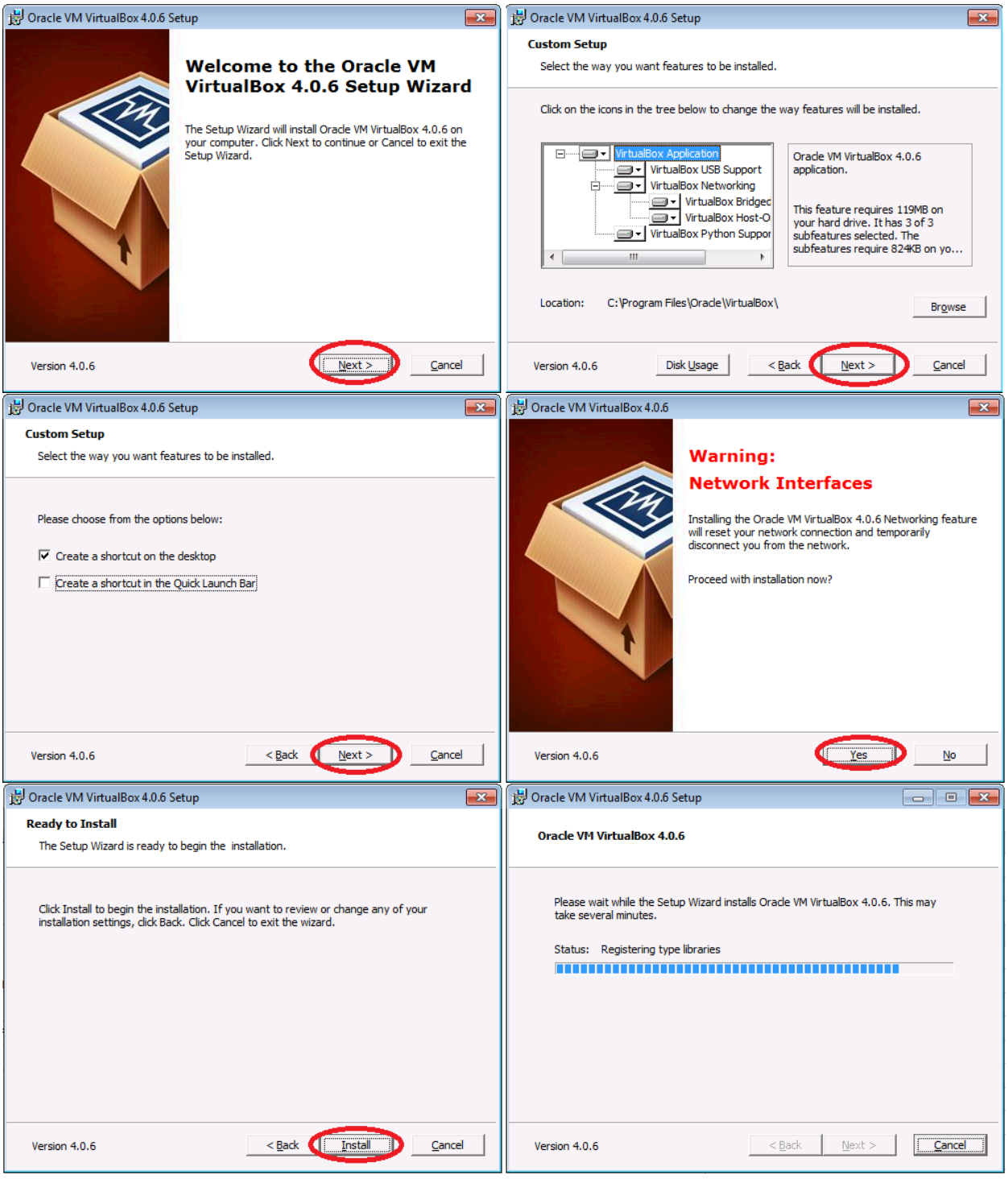




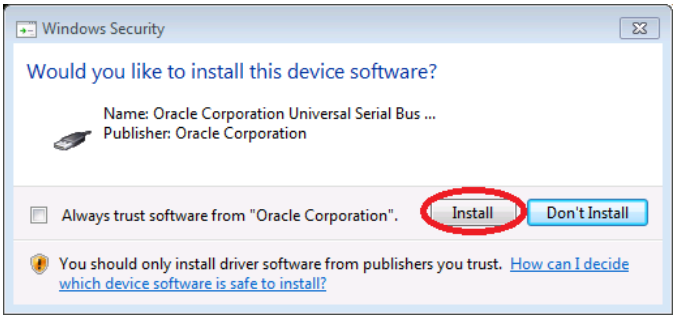


Now just let yourself be guided through the installation:

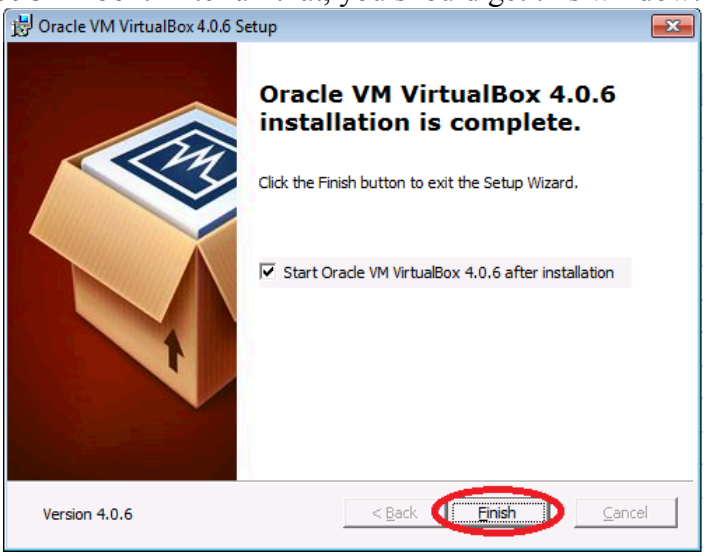
These screenshots show the progress of the installation. When it welcomes you, click "Next". The next screen will ask you what kind of install you want. I just let it do the default by clicking "Next". It gives a warning about networks, but you don't need to worry about that, just click "Yes". Then it decides to check and make sure you really want to install it. Click "Install". It will then show the status as it installs everything.



Note that Windows might try to make sure you really want to install certain things with a window like this:

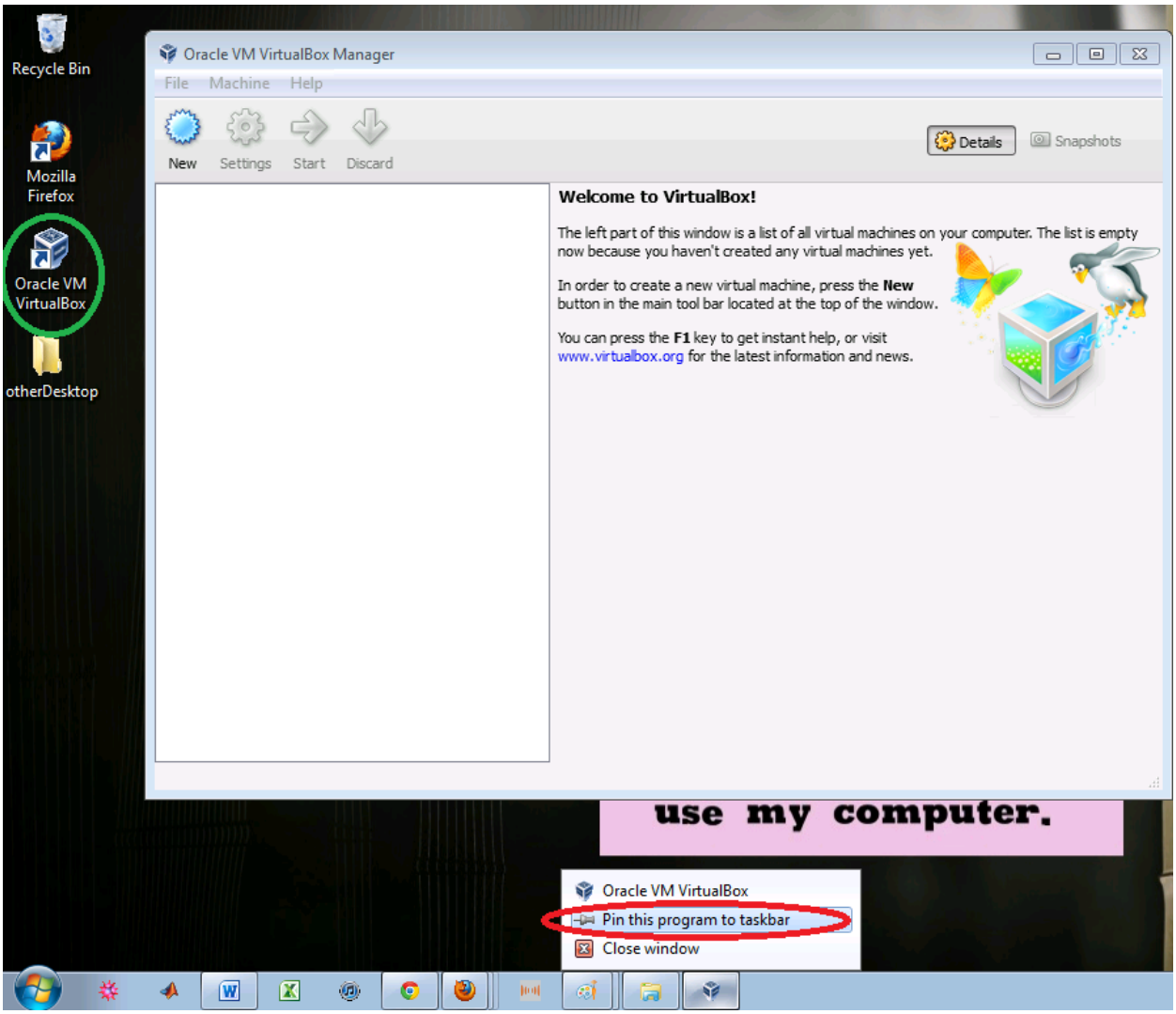


Just click “Install”. If you want to save yourself some time, check the “Always trust software from "Oracle Corporation"” box. After all that, you should get this window:



If you want to start right now, just click “Finish”. If not, uncheck the box and you’ll just have the program installed. The next screenshot is a little busy. It’s basically just the window that should come up when you click “Finish”. I use the program quite a bit, so I pin it to the taskbar by right clicking on the icon that is on the toolbar and clicking “Pin this program to taskbar”.

You’ll also note that there’s a new shortcut icon that I’ve circled in green on the next screenshot. When you want to come back to the program, just double click this icon. If you decided not to make a shortcut, you’ll need to find the program in the “Program Files” or wherever you put it. Go to the start menu (the “Start” or the little blue button with a Windows logo inside of it), click on “All Programs”, go until you find the folder named “Oracle VM VirtualBox”, click on it, go to the folder named “ and click on the “VirtualBox” icon. If you don’t see it, do a file search for “VirtualBox.exe”, or look under the C:\ drive to see if you have to “Program Files” folders, and look in both of them. Go to “My Computer” (or “Computer”) -> “OS (C:)” and see if there are two folders, one named “Program Files” and one named “Program Files x86”. If you did the default installation, it should be in one of these. If you installed the program somewhere else, you’ll have to go wherever you installed it.



Sorry this section was so painfully detailed. I wanted this to be a guide for someone going "from the ground up;" a really good step-by-step guide to getting this going. In the future, I'll just say things like, "Install the program." I'll only be so detailed if it's the first time I'm going over a certain thing.

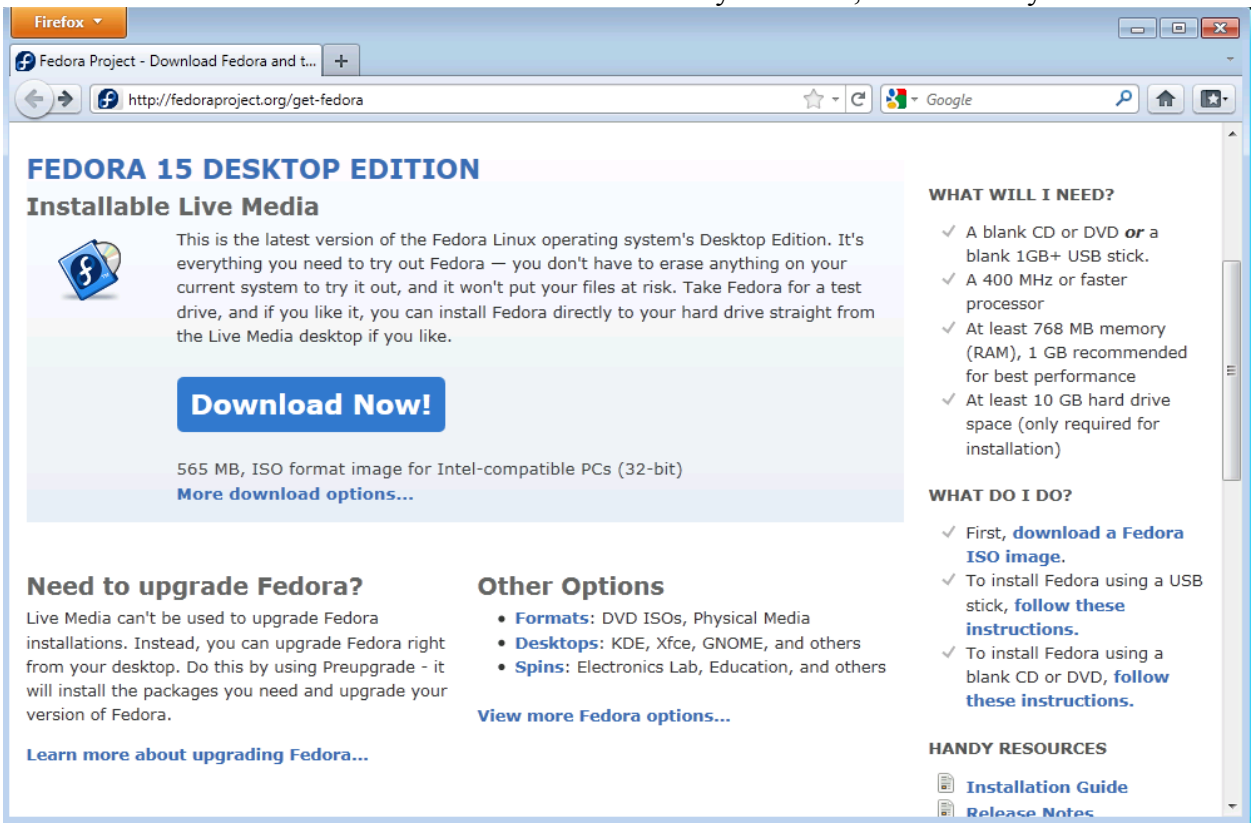
Fedora (Linux) goto Navigation

Now that you have VirtualBox installed, we want to put a Linux operating system into a "virtual machine" on the program. A virtual machine (VM) is a "completely isolated operating system installation within your normal operating system". (Thanks Wikipedia.) What this means is that part of your hard drive is "set apart" to run the other operating system. Often, if you want to run multiple operating systems (e.g. Windows, Mac OS, Linux…), you have partition your hard drive and pick different boot options when you start your computer. With virtualization, you don't. Here's an even more simple explanation of virtualization: It lets you run Linux in a window while still running Windows.

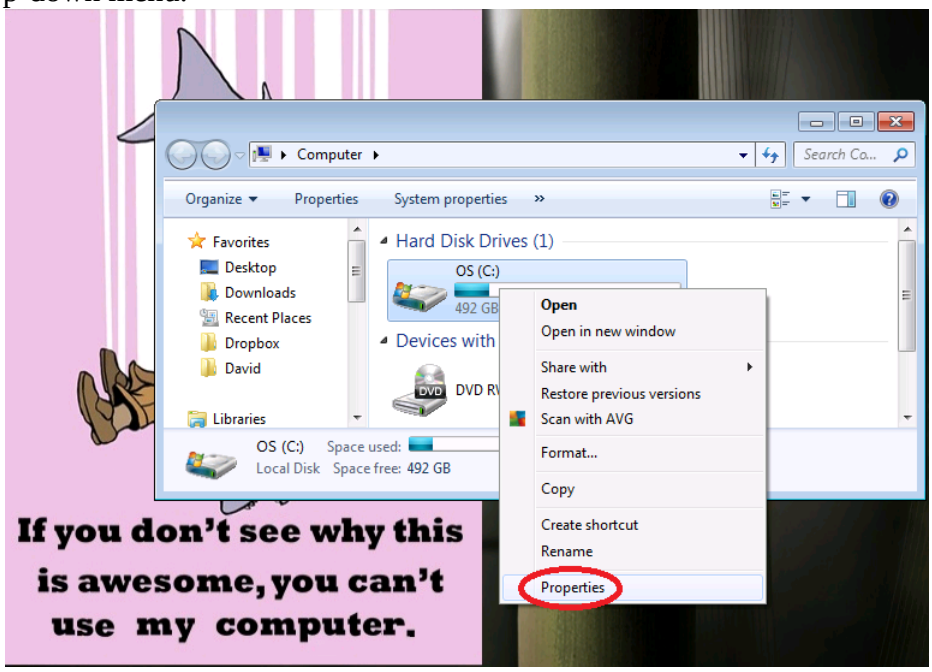
The version of Linux we'll be installing is called Fedora. As I'm writing this, the most current fully stable version of Linux is Fedora 15, so these instructions are given for that. The installation instructions for the different releases are pretty much the same.

Note that Fedora 15 is new (at least while I'm writing this in May/June 2011), and so there are some bugs. One of the major ones is with its new desktop. I've put in a fix for that. If you don't want to mess with it, you can download Fedora 14 instead. Instructions are below.

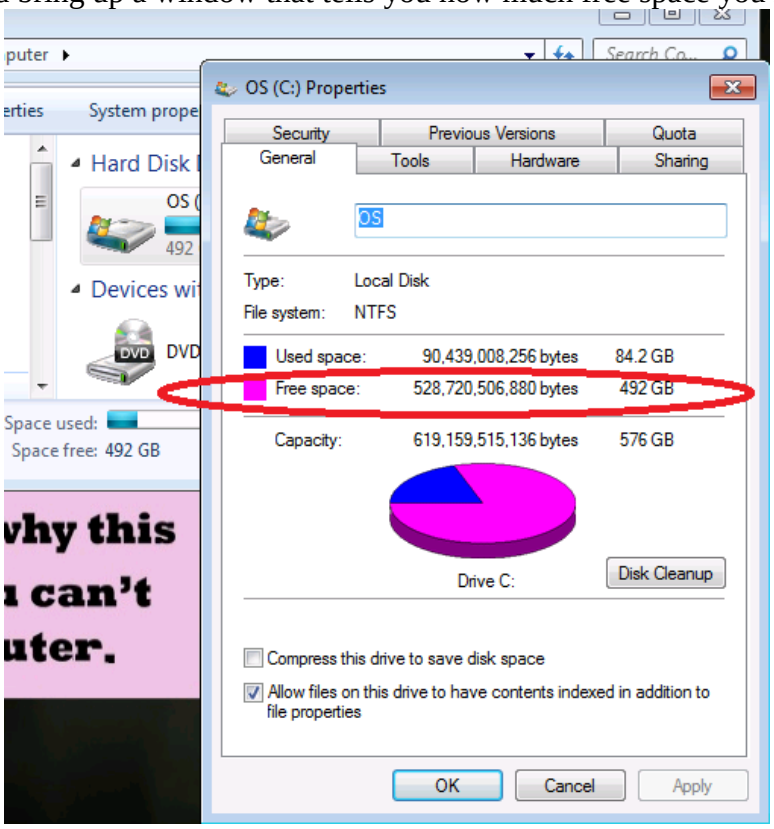
Start by downloading the file that will "install" the operating system. Go to the Get Fedora website <http://fedoraproject.org/get-fedora>. DON'T PUSH THE BIG BUTTON YET! Make sure you have the minimum requirements (look at the section after this screenshot). Note the nice big "Download Now!" button, but don't press it. This screen says you need a blank CD or DVD or a blank 1GB+ USB stick. You won't actually need one, as I'll show you below.



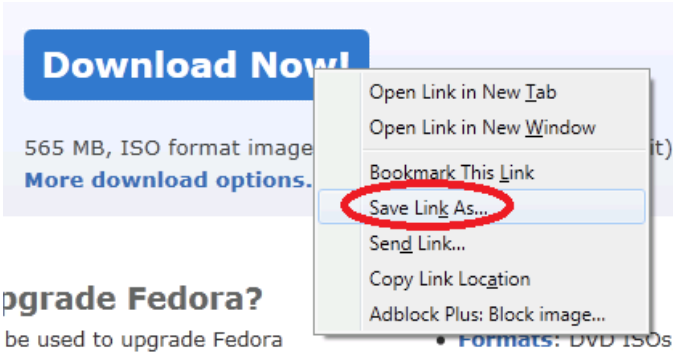
Note that it says you need a blank CD or DVD or a blank 1GB+ USB stick. Actually, you can just install it somewhere on your computer, which is what I'll do. Note that you have to have at least 512 MB memory (RAM), 1 GB recommended and at least 10 GB hard drive space (for this version.) You can find out how much RAM you have by looking at the system properties, as described at the beginning of the section on installing the VM. To find out how much hard drive space you have, go to "My Computer". It should just tell you how much hard drive space you have (underneath the OS (C:) part.) If not, right click the "OS (C:)" icon and select "Properties" from the drop-down menu.



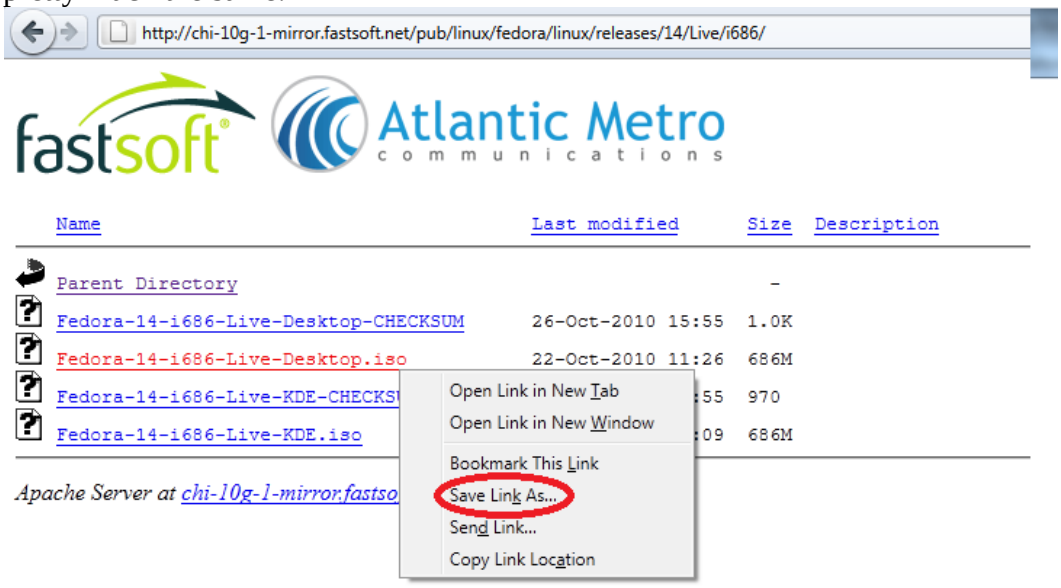
This should bring up a window that tells you how much free space you have, like so:



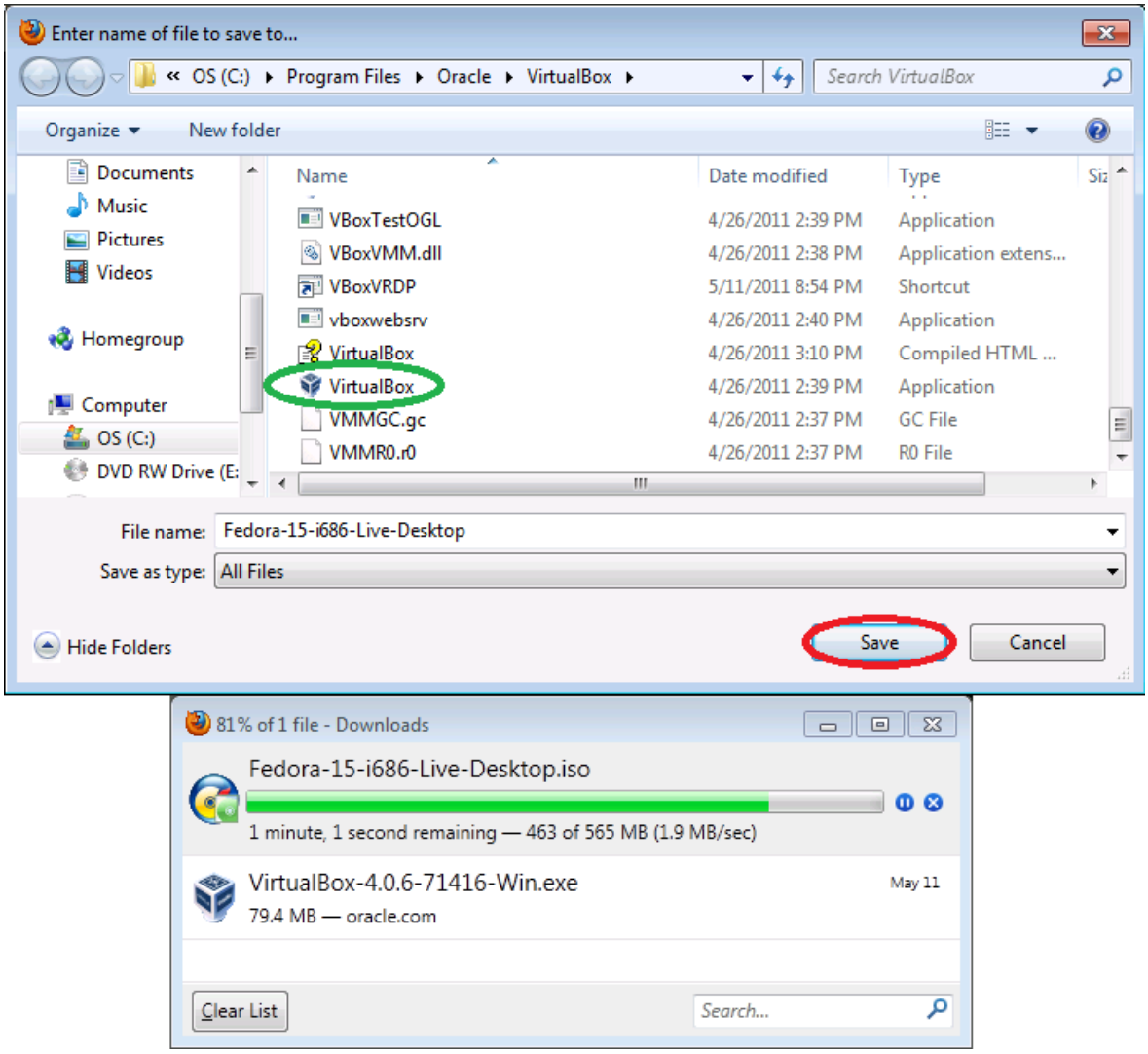
Okay, now that you've checked that out right-click the big "Download Now" button and click "Save link as…".



If you want to use Fedora 14 (I would suggest using whatever the most recent release is rather than using Fedora 14), go to this link <http://chi-10g-1-mirror.fastsoft.net/pub/linux/fedora/linux/releases/14/Live/i686/>, and right-click the "Fedora-14-i686-Live-Desktop.iso" and save it using the same steps. Then everything will be pretty much the same.

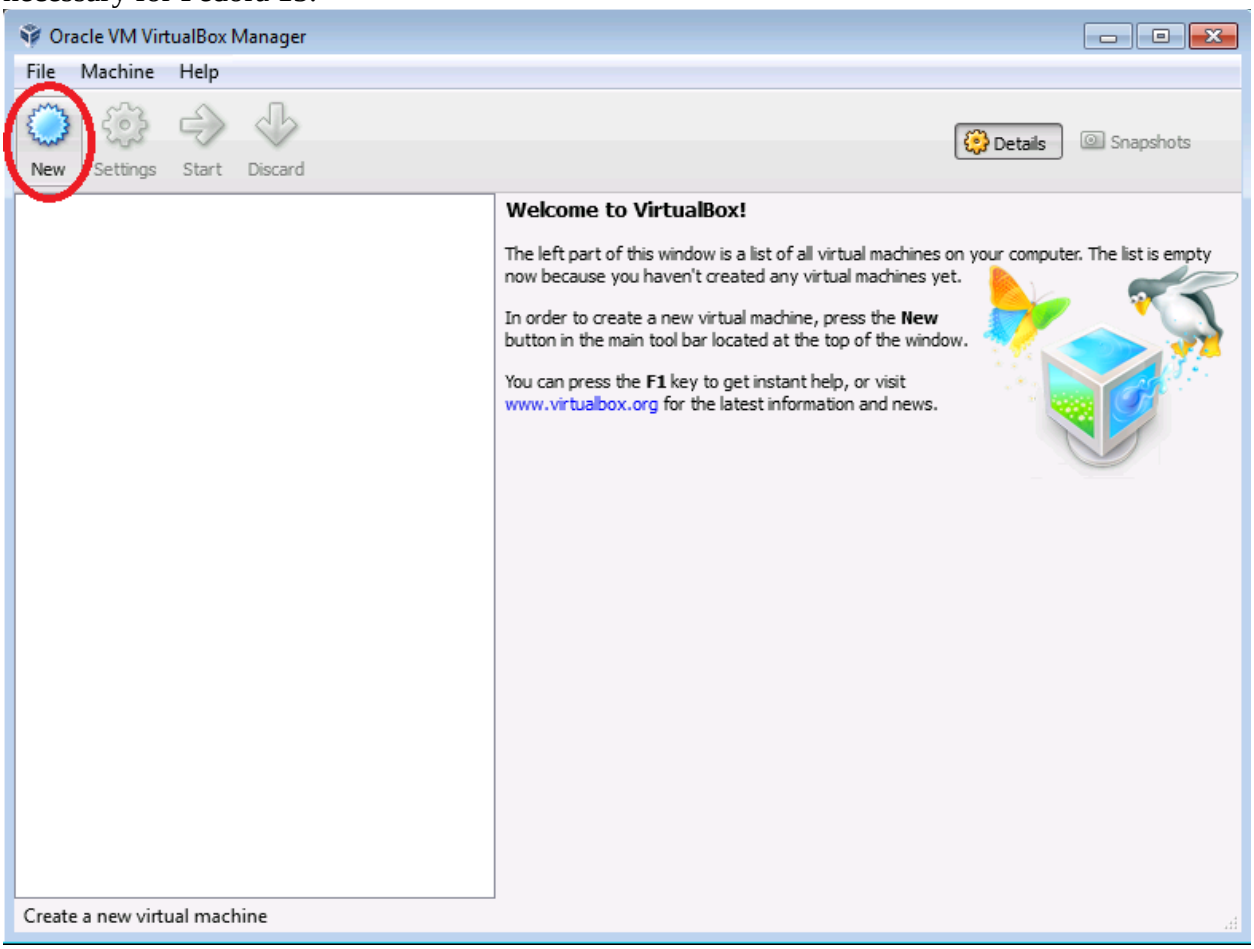


You can save this wherever you want. I'm going to put it in the same folder as the VirtualBox program. If you did the default installation, follow the file path "C:" -> "Program Files" -> "Oracle" -> "VirtualBox", and click "Save".

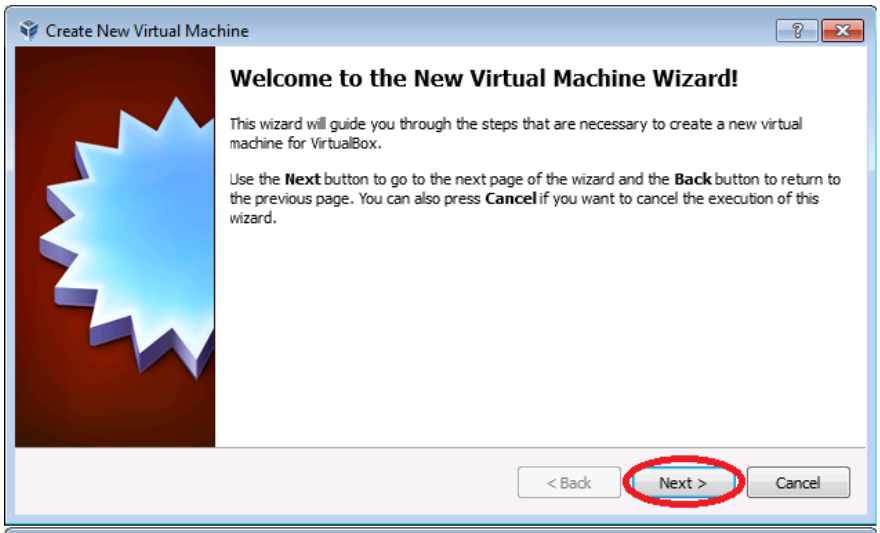


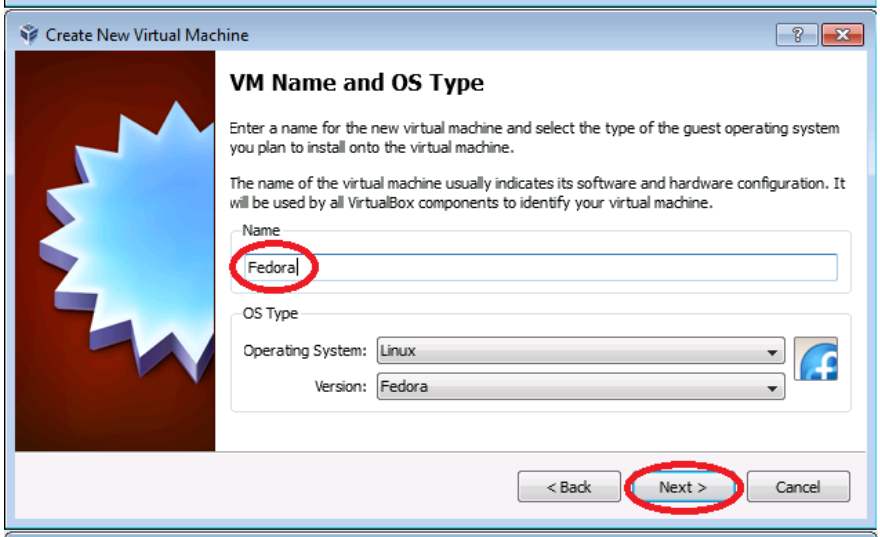
Now you'll need to wait for it to load. This could take a little bit. I was on my school's wireless with a 1-2 MB/sec download rate, and it took about 10 minutes. Go play Minesweeper or something. Notice also the VirtualBox icon that I've circled in green. You can click on this to start VirtualBox.

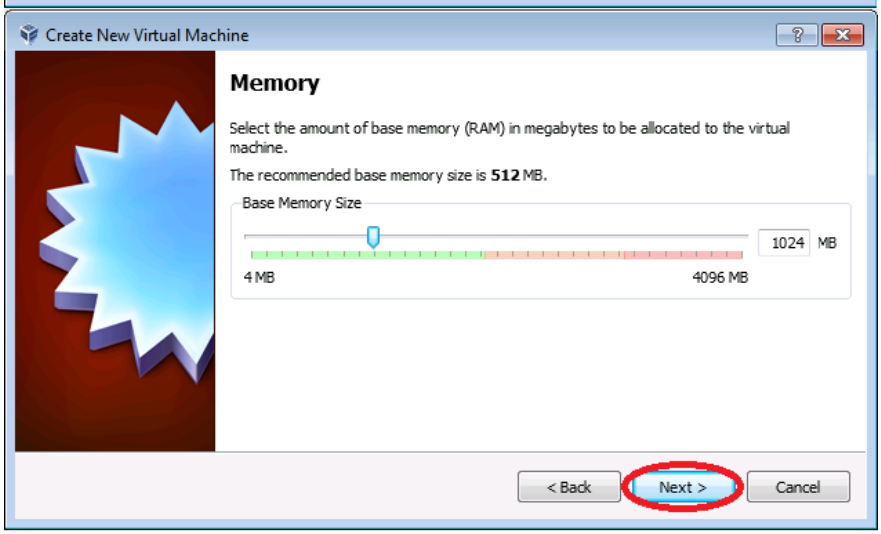
All right, now it's time to get the Linux Operating System (OS) accessible from VirtualBox. (The ISO file you downloaded, which is called a live CD, is what you'll use to start installing the OS.) Head back to the VirtualBox program. I just clicked on the icon I have on my desktop. Note that since I started this tutorial, VirtualBox updated to 4.0.8. This version is necessary for Fedora 15.



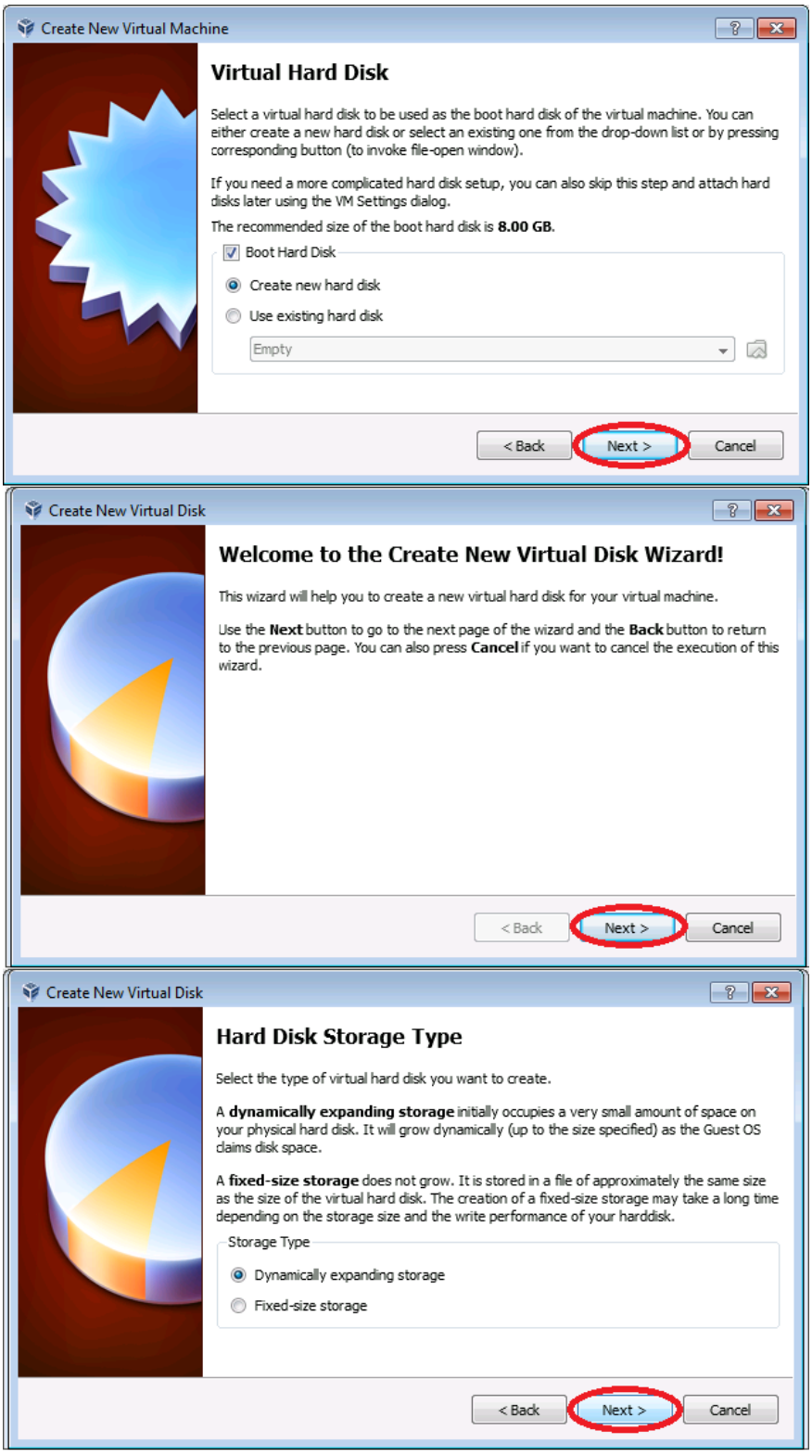
Now click on the "New" icon in the upper-left corner and allow yourself to be led through the installation process. When it welcomes you, press "Next". On the "VM Name and OS Type" type in "Fedora" in the "Name" box. This will automatically select the Operating System as Linux and the Version as Fedora. Press "Next" again. Now you'll need to allocate a certain amount of RAM to the VM. The recommended base memory is 512 MB. I gave it 1 GB of RAM out of the 4 GB my computer has. (Note 1 GB = 1024 MB, which is why I have 1024) I might go back and change this later in the "Settings" part of VirtualBox, but that's good start. Note that selecting too much to give to the VM will make your host OS (Windows) run more slowly while you are running the VM.


Create New Virtual Machine
Welcome to the New Virtual Machine Wizard!
This wizard will guide you through the steps that are necessary to create a new virtual machine for VirtualBox.
Use the Next button to go to the next page of the wizard and the Back button to return to the previous page. You can also press Cancel if you want to cancel the execution of this wizard.
< Back
Next >
Cancel

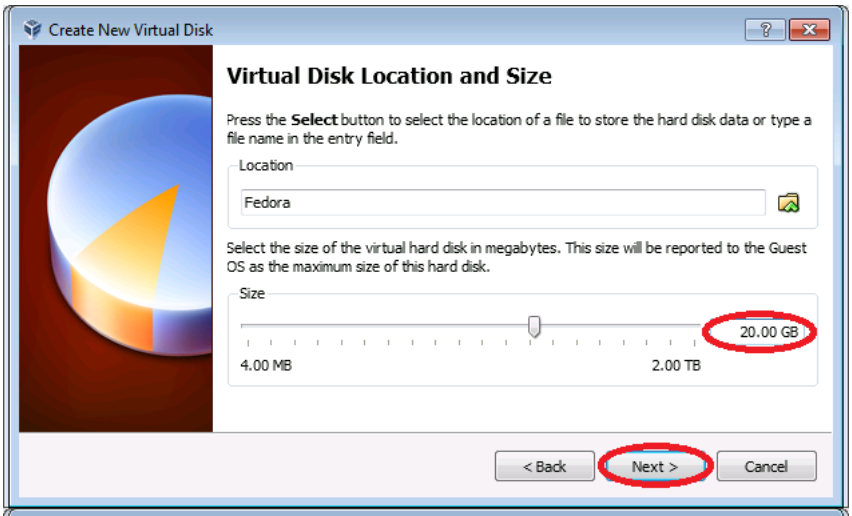

Create New Virtual Machine
VM Name and OS Type
Enter a name for the new virtual machine and select the type of the guest operating system you plan to install onto the virtual machine.
The name of the virtual machine usually indicates its software and hardware configuration. It will be used by all VirtualBox components to identify your virtual machine.
Name
Fedora
OS Type
Operating System:
Linux
Version:
Fedora
< Back
Next >
Cancel

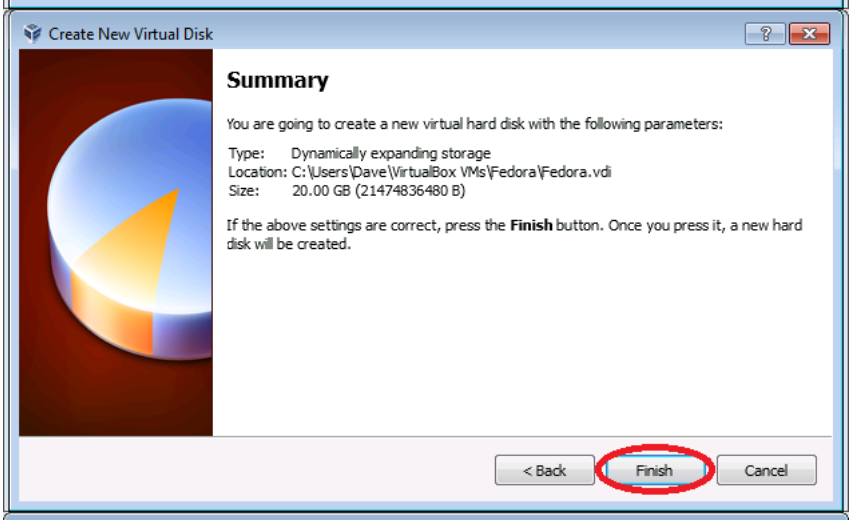

Create New Virtual Machine
Memory
Select the amount of base memory (RAM) in megabytes to be allocated to the virtual machine.
The recommended base memory size is 512 MB.
Base Memory Size
1024
MB
4 MB
4096 MB
< Back
Next >
Cancel

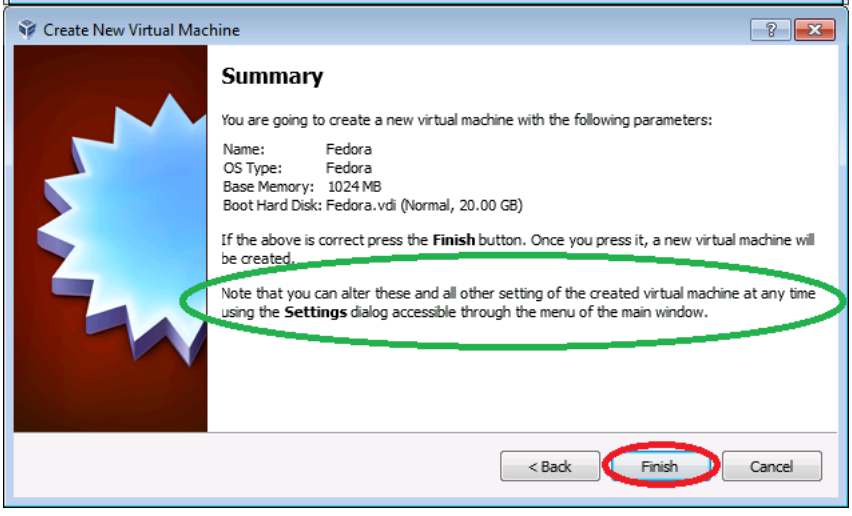
On the “Virtual Hard Disk” screen, press “Next”. This just tells it you want to make a new virtual hard disk. It will welcome you to another wizard, press “Next”, and ask you what kind of storage you’d like. Another screen might come up asking what kind of disk you want. The option “VDI” should be there. Choose that. “Dynamically expanding storage” will just change the size of the hard drive as you have more stuff on it. This is ideal. It will give you the option to state a max size.



It will come up with a “Location and Size” window. The base size is 8 GB. I plan on using this quite a bit, so I chose 20.00 GB. This will be the max size of the virtual hard drive. You’ll get two summary windows. Press “Finish” on both of them. Notice what I’ve circled in green, which says you can change the settings. This should make you worry less.

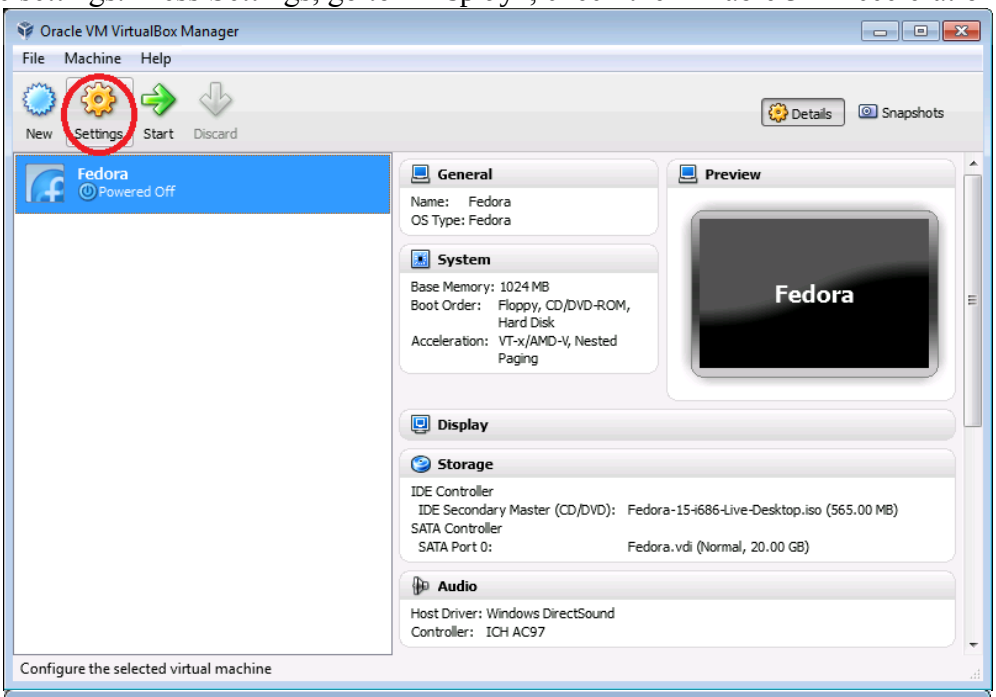

Create New Virtual Disk
Virtual Disk Location and Size
Press the Select button to select the location of a file to store the hard disk data or type a file name in the entry field.
Location
Fedora
Select the size of the virtual hard disk in megabytes. This size will be reported to the Guest OS as the maximum size of this hard disk.
Size
20.00 GB
4.00 MB
2.00 TB
< Back
Next >
Cancel

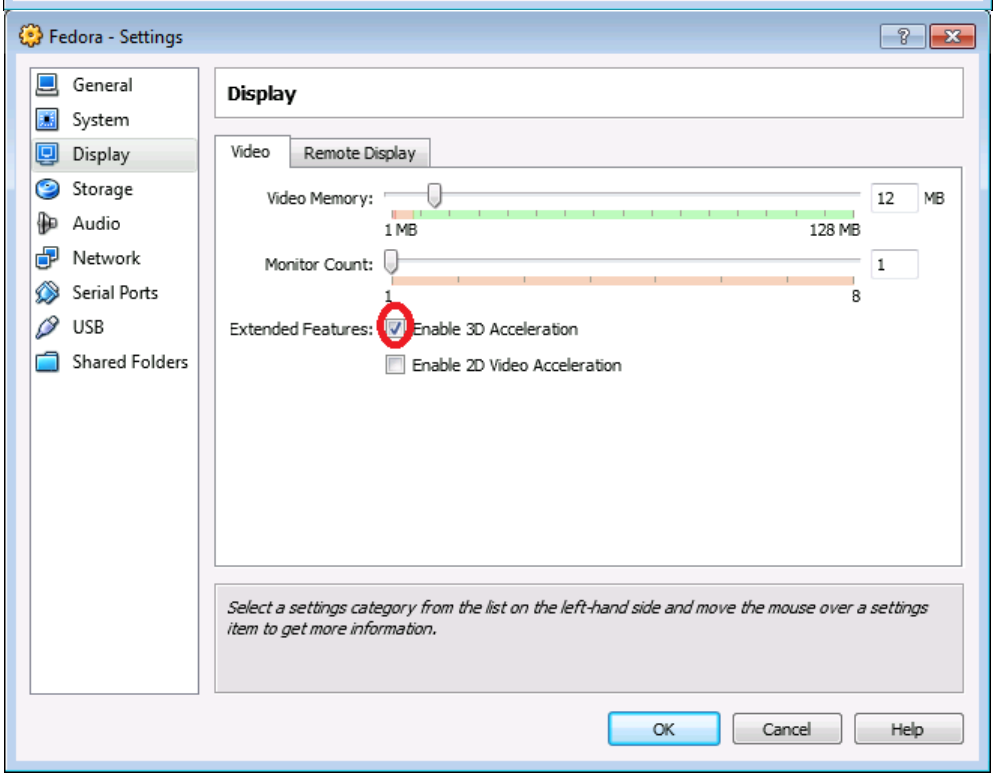

Create New Virtual Disk
Summary
You are going to create a new virtual hard disk with the following parameters:
Type: Dynamically expanding storage
Location: C:\Users\Dave\VirtualBox VMs\Fedora\Fedora.vdi
Size: 20.00 GB (21474836480 B)
If the above settings are correct, press the Finish button. Once you press it, a new hard disk will be created.
< Back
Finish
Cancel


Create New Virtual Machine
Summary
You are going to create a new virtual machine with the following parameters:
Name: Fedora
OS Type: Fedora
Base Memory: 1024 MB
Boot Hard Disk: Fedora.vdi (Normal, 20.00 GB)
If the above is correct press the Finish button. Once you press it, a new virtual machine will be created.
Note that you can alter these and all other setting of the created virtual machine at any time using the Settings dialog accessible through the menu of the main window.
< Back
Finish
Cancel

Running Linux (Live CD) goto Navigation

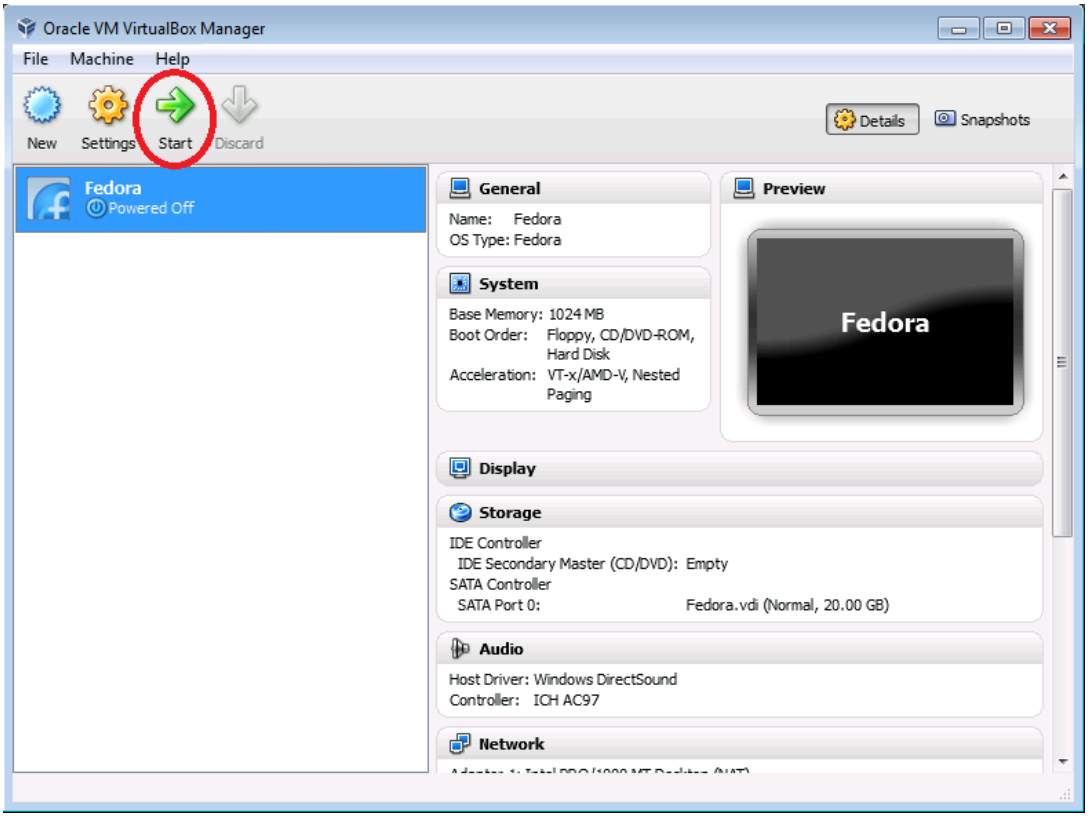
Now you'll have a virtual hard disk and we can run Linux off the live CD. We're not through yet, though. The result of your last action should be the next screenshot. You'll need to change the settings. Press Settings, go to "Display", check the "Enable 3D Acceleration" box.



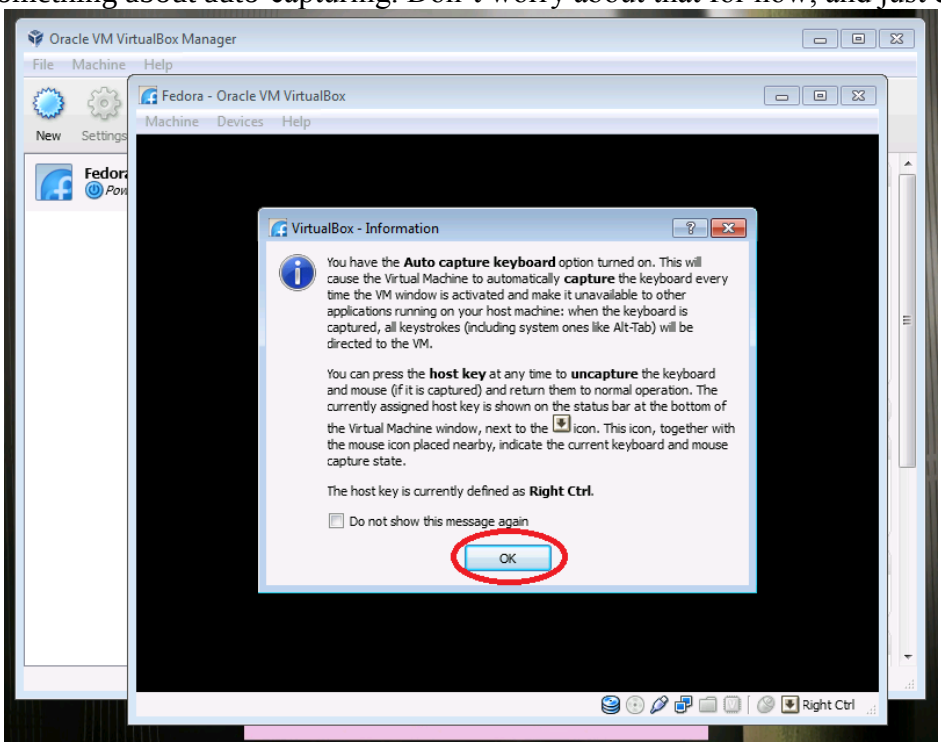


Note that the "Enable 3D Acceleration" box should be checked. Press "OK".

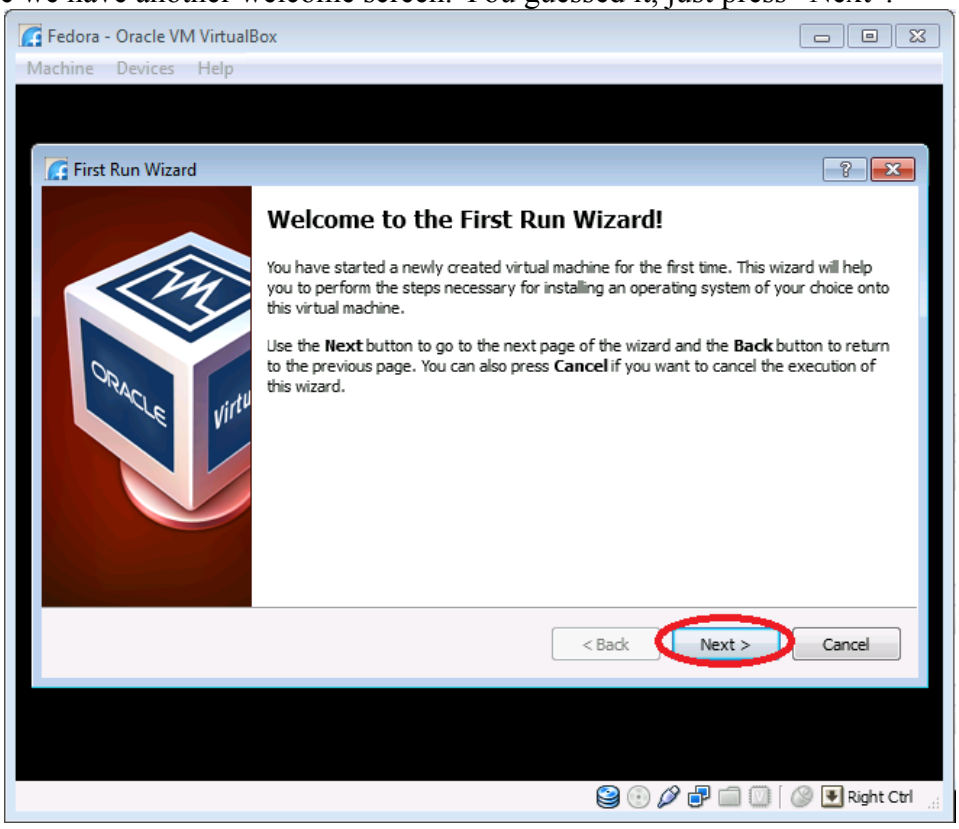
Go ahead and press "Start".



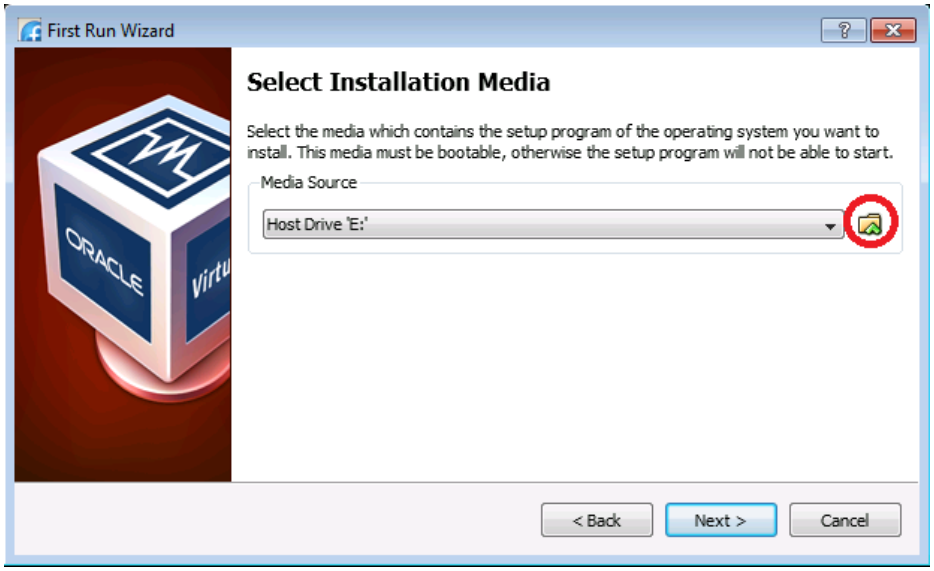
It will say something about auto-capturing. Don't worry about that for now, and just click "OK".



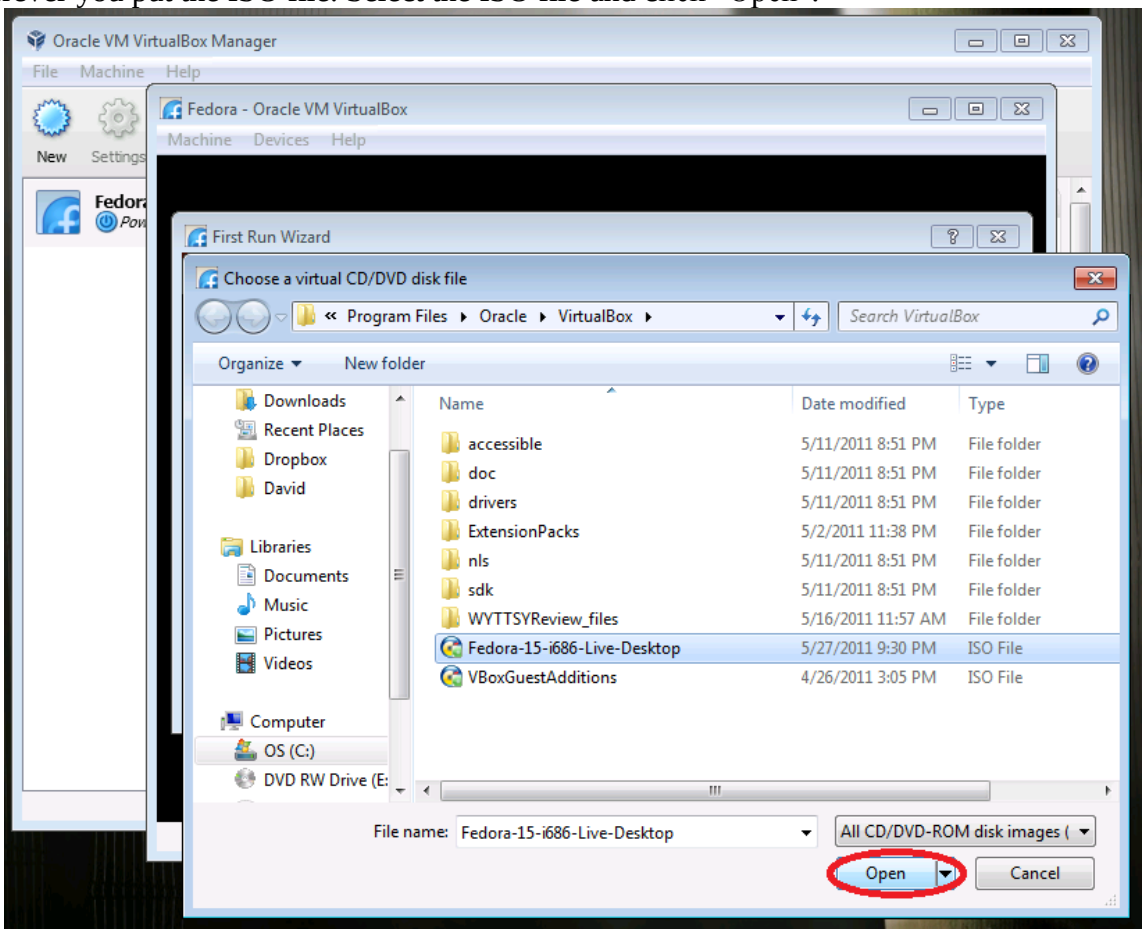
Here we have another welcome screen. You guessed it, just press “Next”.



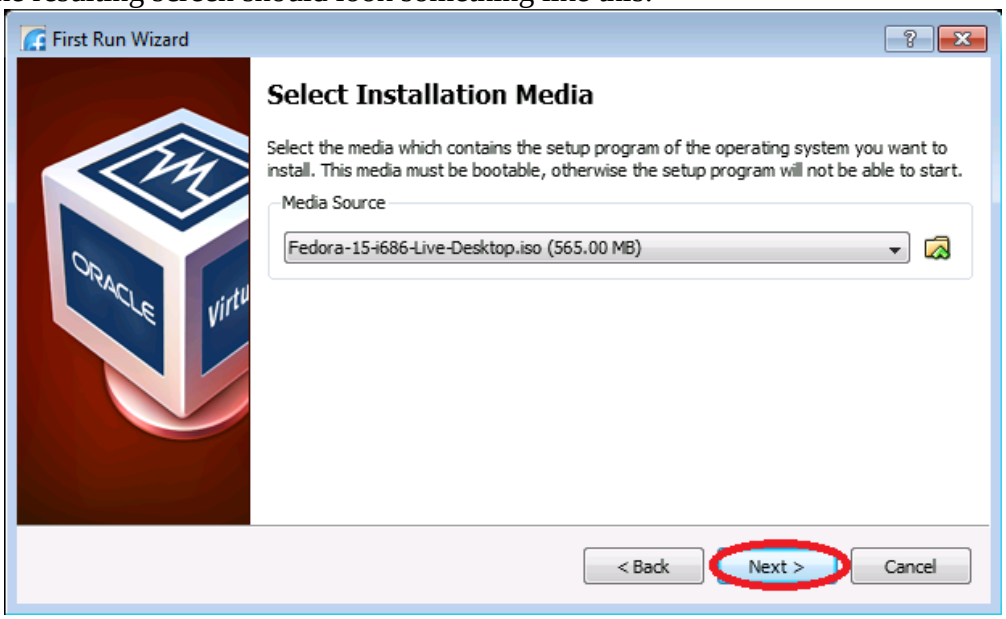
Now we’ll need to select the installation media. Click on the “Browse” icon—the folder with the green arrow on it.

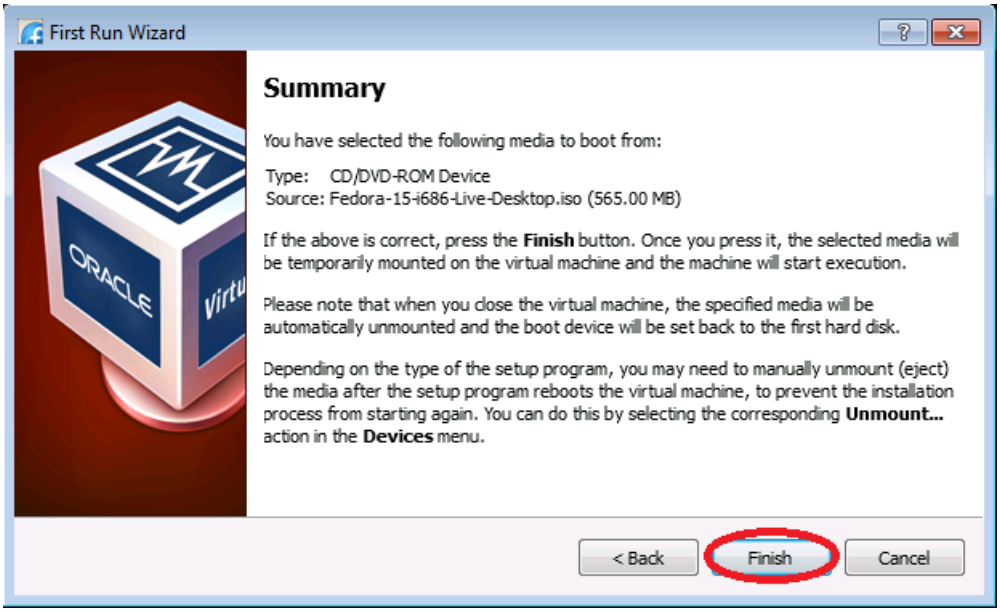


Remember that we put this under C:/ProgramFiles/Oracle/VirtualBox , so navigate there or wherever you put the ISO file. Select the ISO file and click “Open”.

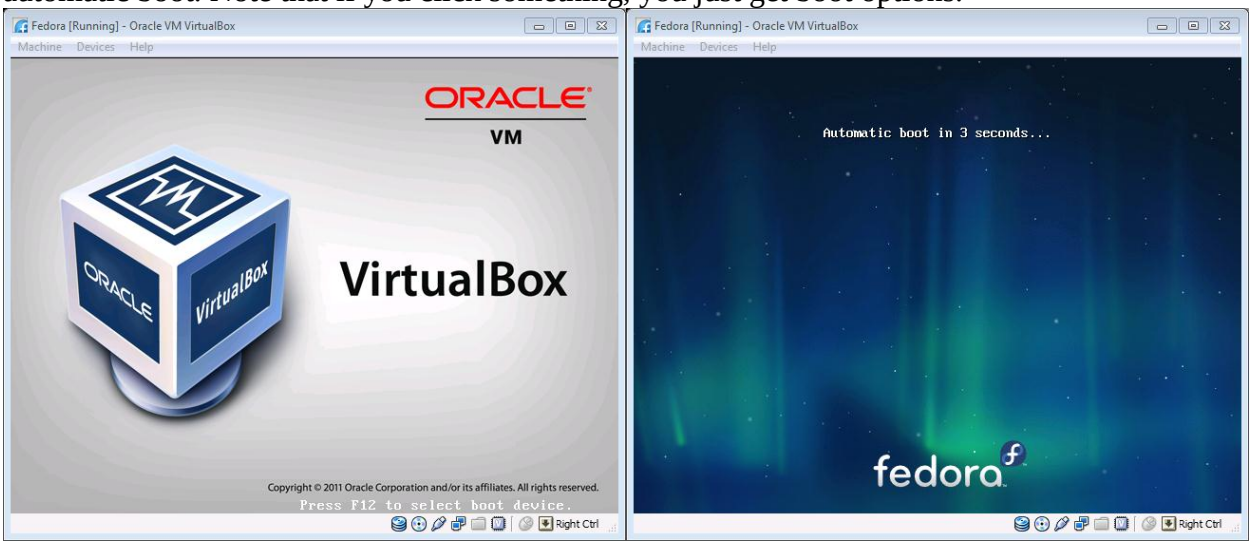


The resulting screen should look something like this:

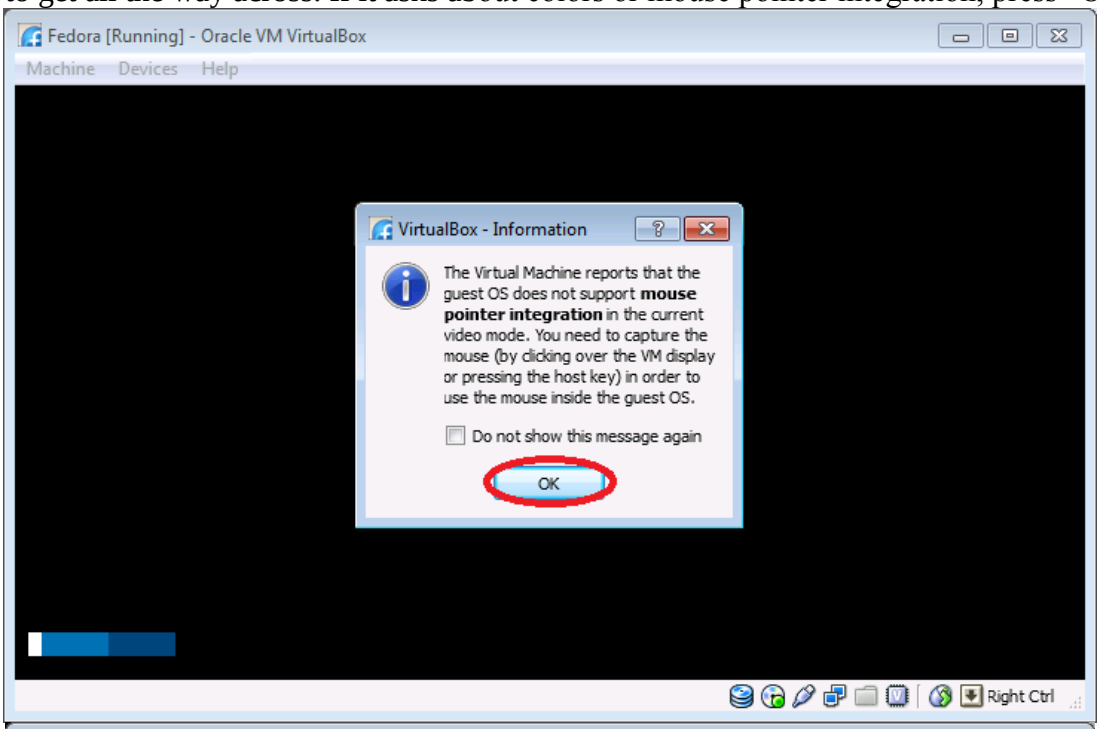


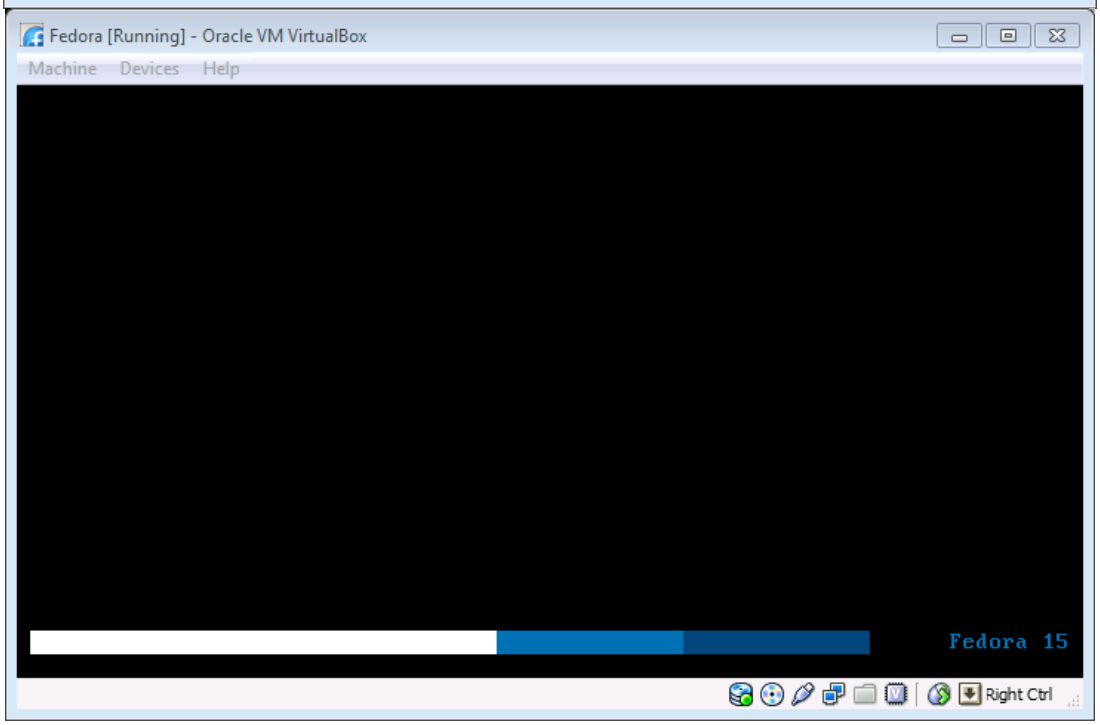


Select “Next” and then select “Finish” on the “Summary” window. The VM window will come up and it will say that it’s going to do an automatic boot in a certain number of seconds. If it says something about color options or the mouse pointer integration, just press “OK”. Just let it do its automatic boot. Note that if you click something, you just get boot options.

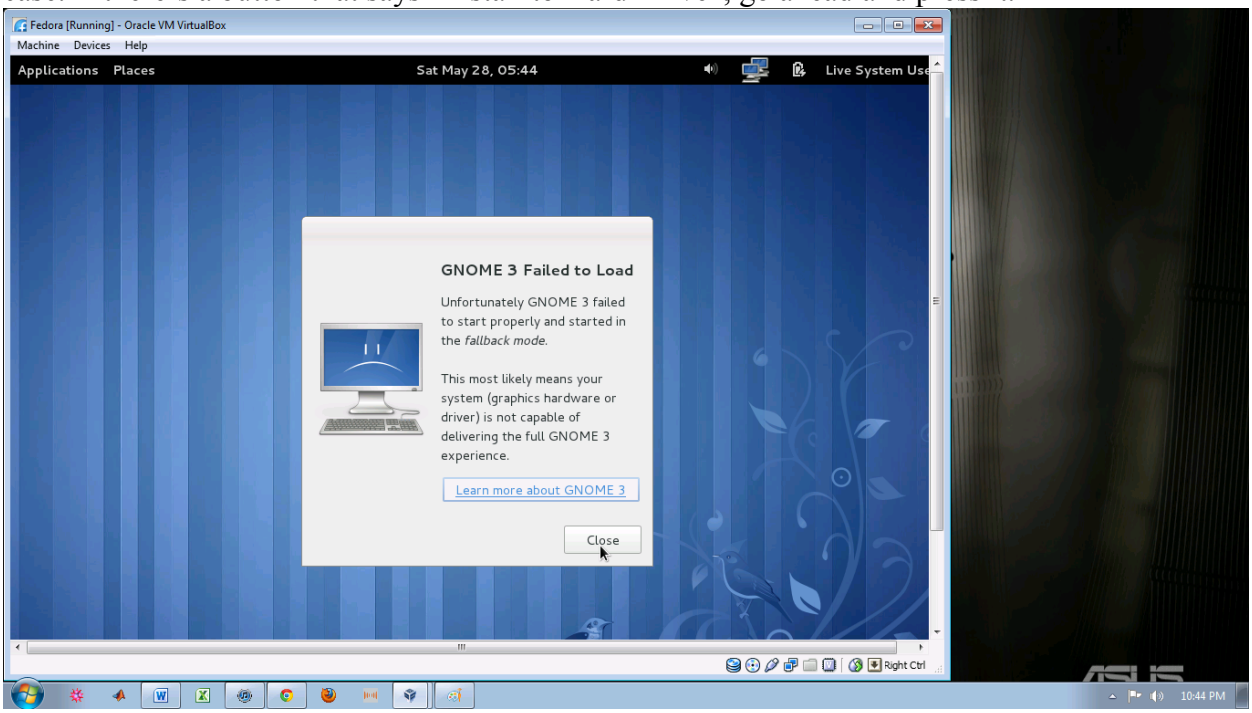


Now you should have a Fedora loading screen. Wait for the two shades of blue and the white to get all the way across. If it asks about colors or mouse pointer integration, press “OK”.





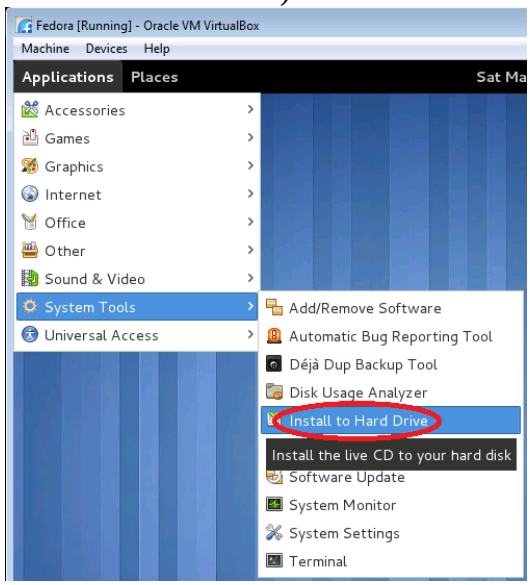
Now the VM screen should get bigger, and in a couple of seconds, the following screen will come up (don't worry if there's a warning.) With some newer systems (like Fedora 16 or 17) it might come up with an error that asks if you want to log out. Go ahead and log out if this is the case. If there's a button that says "Install to Hard Drive", go ahead and press it.



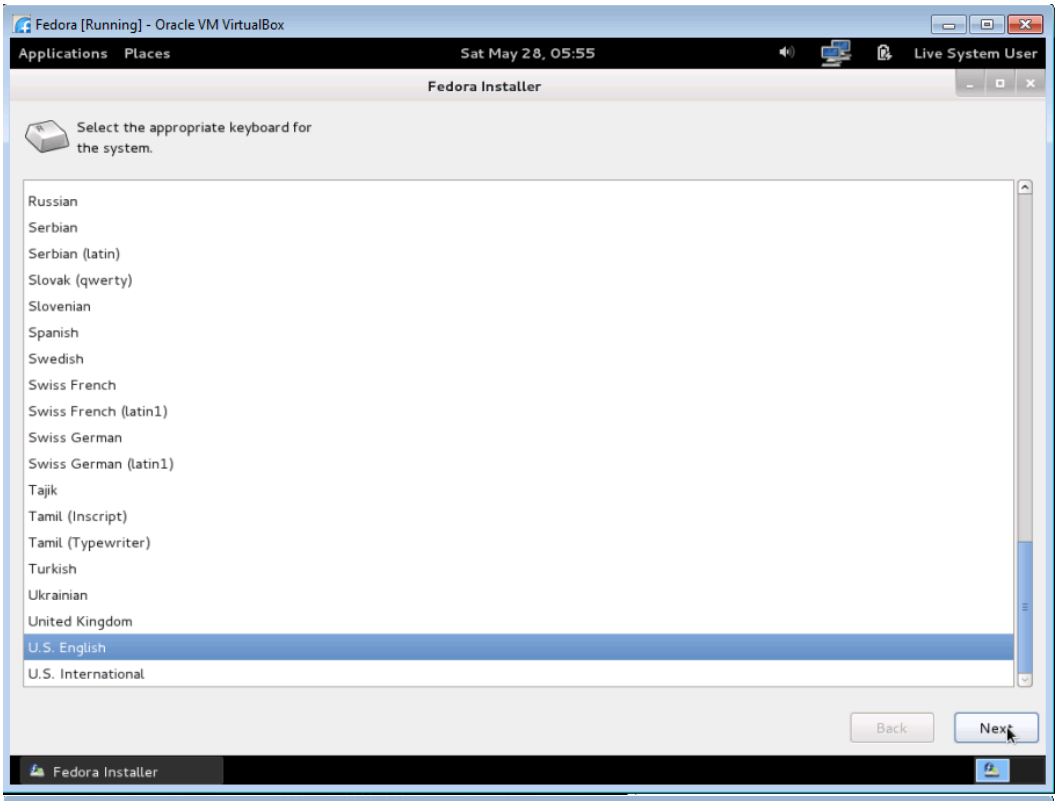
There's your first view of Linux running in Windows. Look at all its majesties! Don't worry about GNOME 3 right now, we'll fix that later. Just press "Close". (No, it's not circled, but you can actually see the mouse pointer now, so I'm not worrying about it now.)

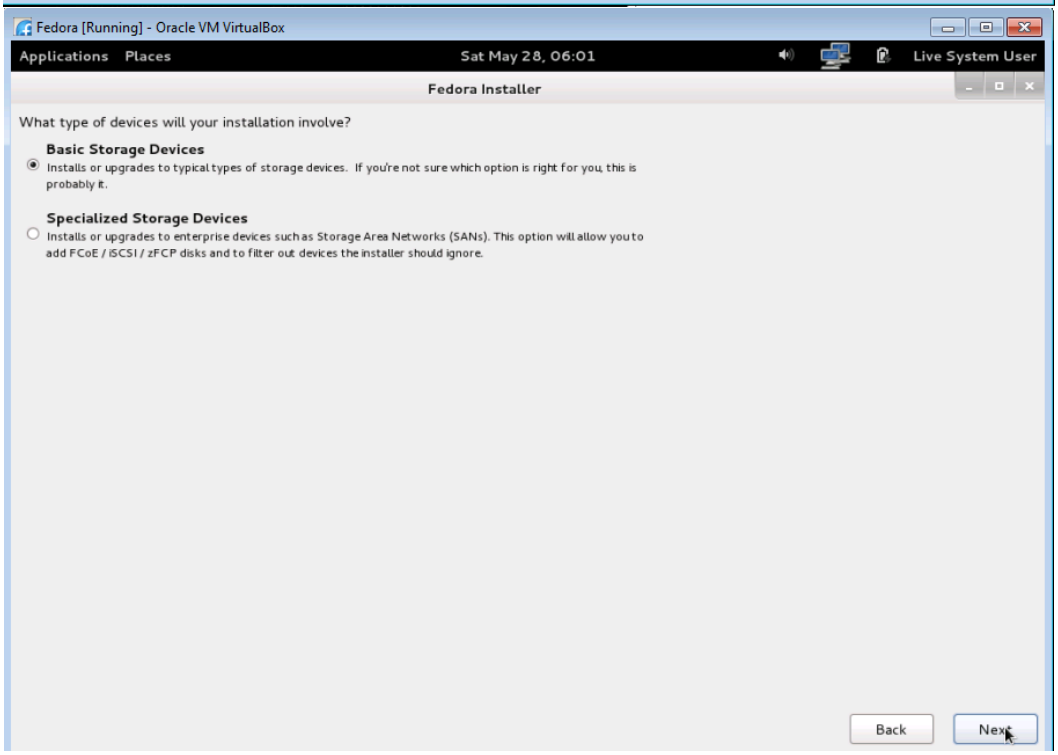
Running Linux (Hard Drive) goto Navigation

If you haven't already clicked something that installs Linux to your hard drive, follow the instructions in the following paragraph. From your VM, go to "Applications" then "System Tools" and finally click "Install to Hard Drive". We're running Linux off of the live CD right now. We could do stuff with it like this, but when we shut down the VM, we'd lose everything we did. In the following steps, I've used "Machine"-> "Switch to Scale Mode" to make everything visually better. If you do this and can't find out how to get back, it's "Host + C", where in this case "Host" means the right control button, the "ctrl" to the right of the spacebar. In some ways, it's easier to just scroll than to use scaled mode. You do what you want. (My roommate will make fun of me for that statement.)

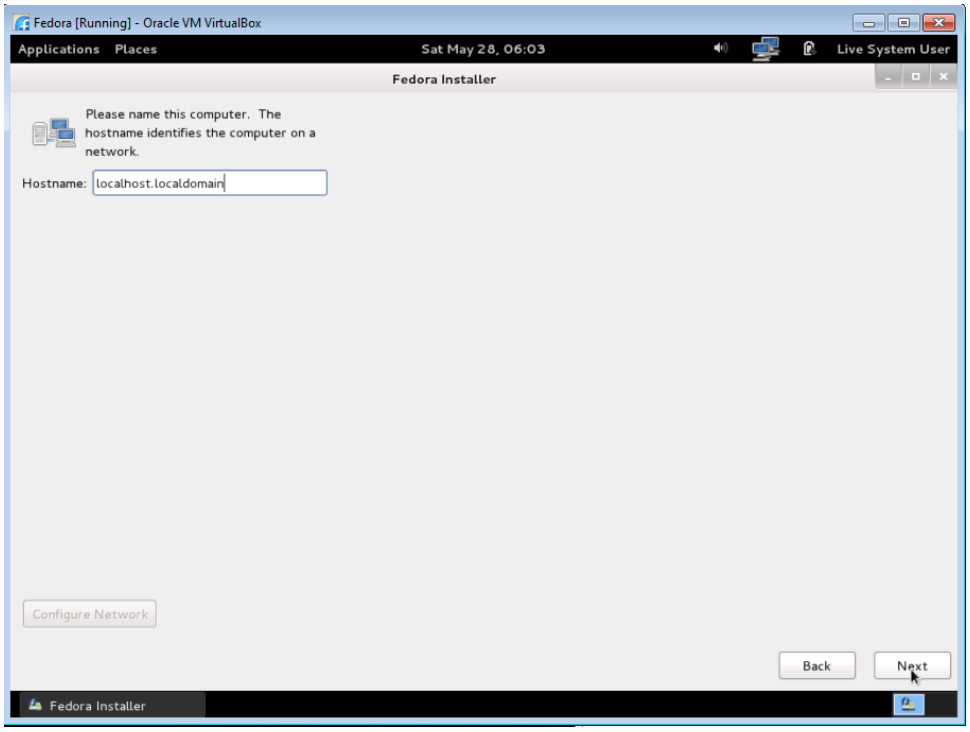


Select the language for the keyboard and press “Next”. Press “Next” for the “Basic Storage Devices”.

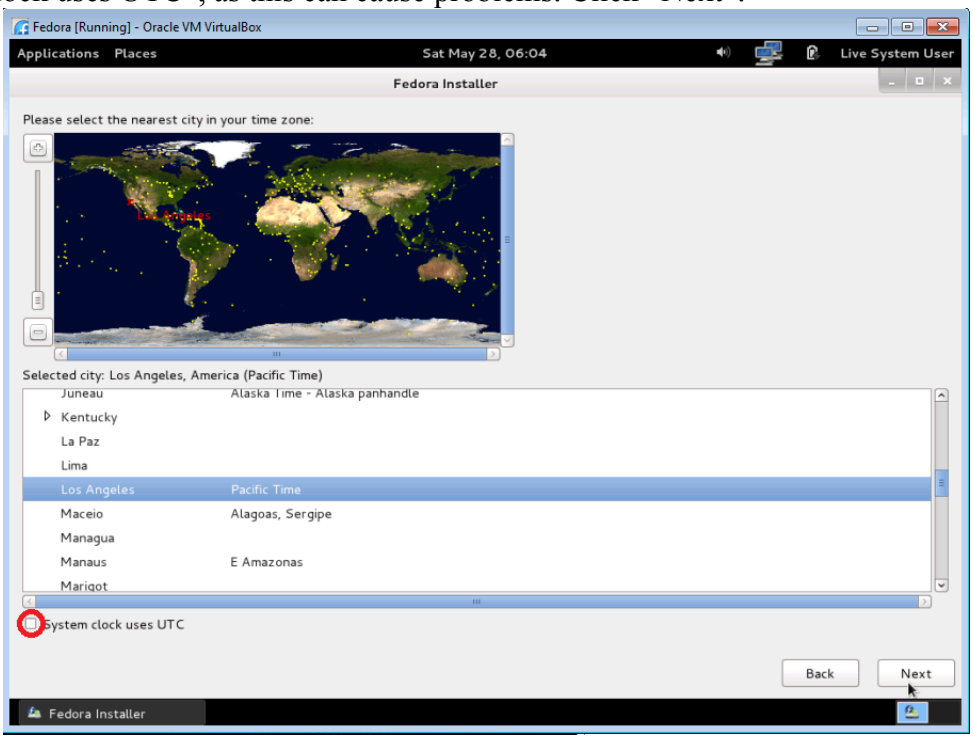




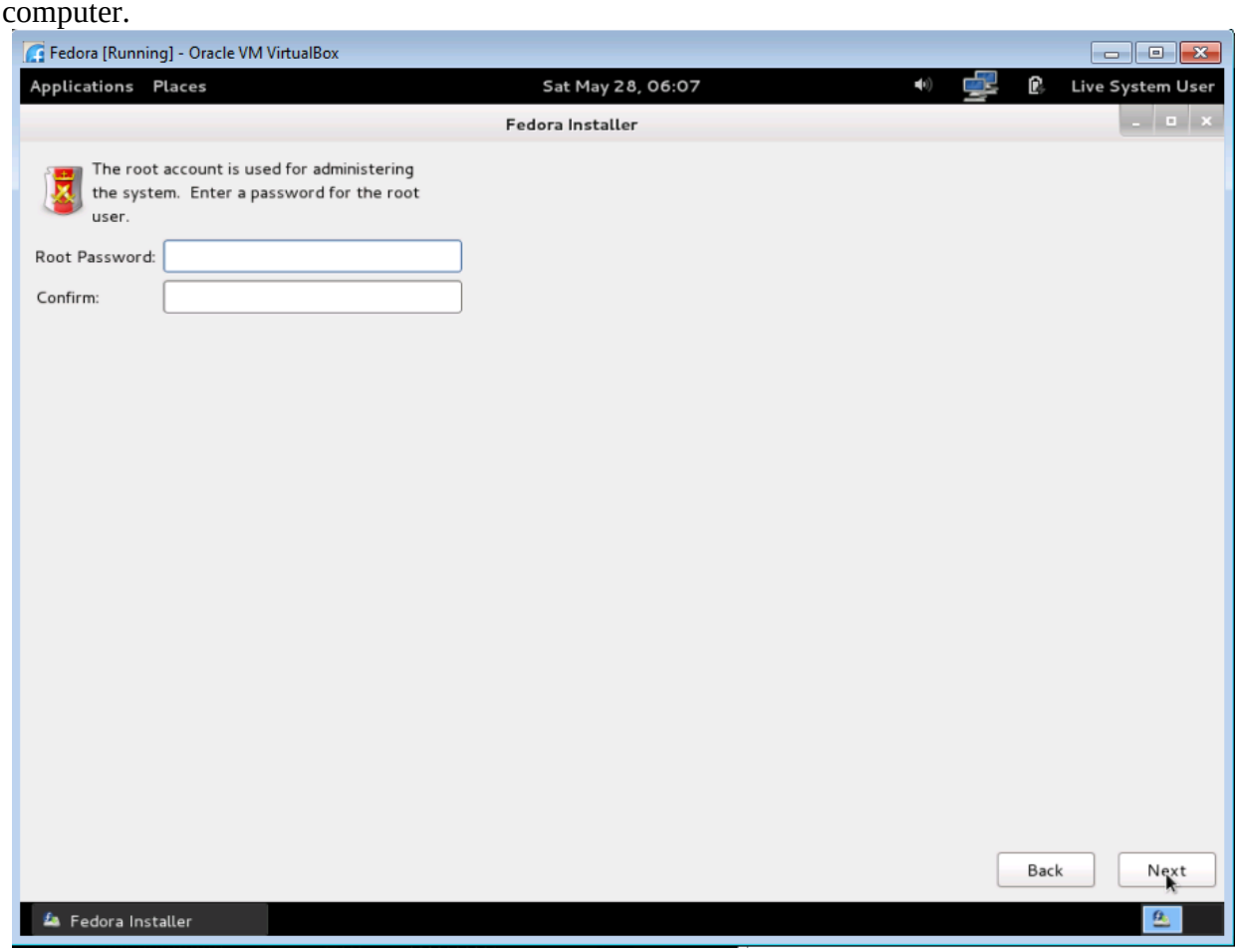
It's possible that a "Storage Device Warning" could come up, but not likely. If it does, and this is your first time installing a VM, select "Yes, discard any data". This likely won't come up. It will next ask for the name of the VM. Something like "yourname.yourdomain" will be good. Press "Next".



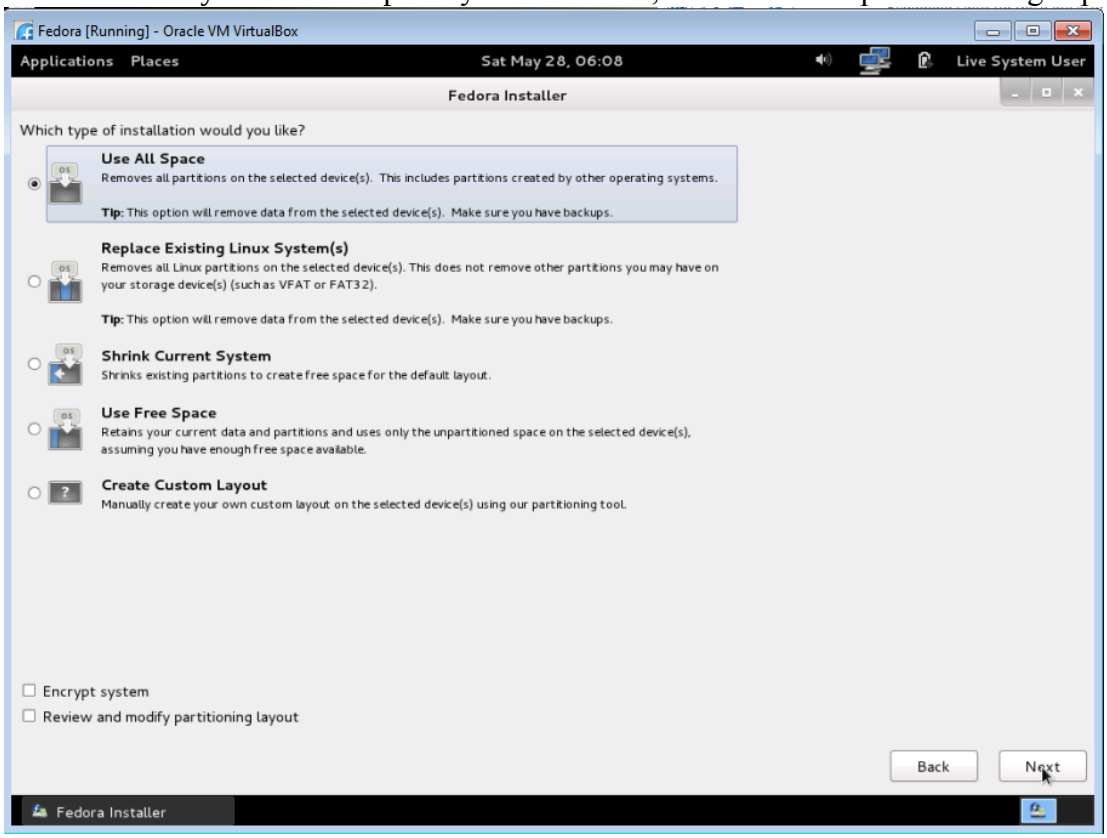
It will next ask for your time zone. New York is the default. Make sure to uncheck "System clock uses UTC", as this can cause problems. Click "Next".



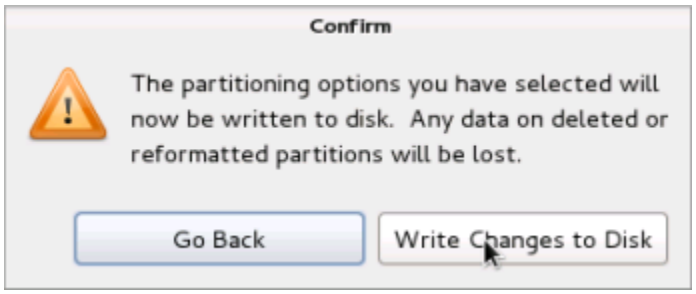
Choose a password for the root user. Note that the “root user” have nothing to do with the ROOT program that we’re going to install later. This is the user that can make all kinds of changes on the Linux computer, so you want to make it a strong password. When you’re logged in as root, it’s possible to mess up your VM really quickly. That’s another perk of virtualization—anything you mess up will just be messed up in the VM, not on the normal computer.



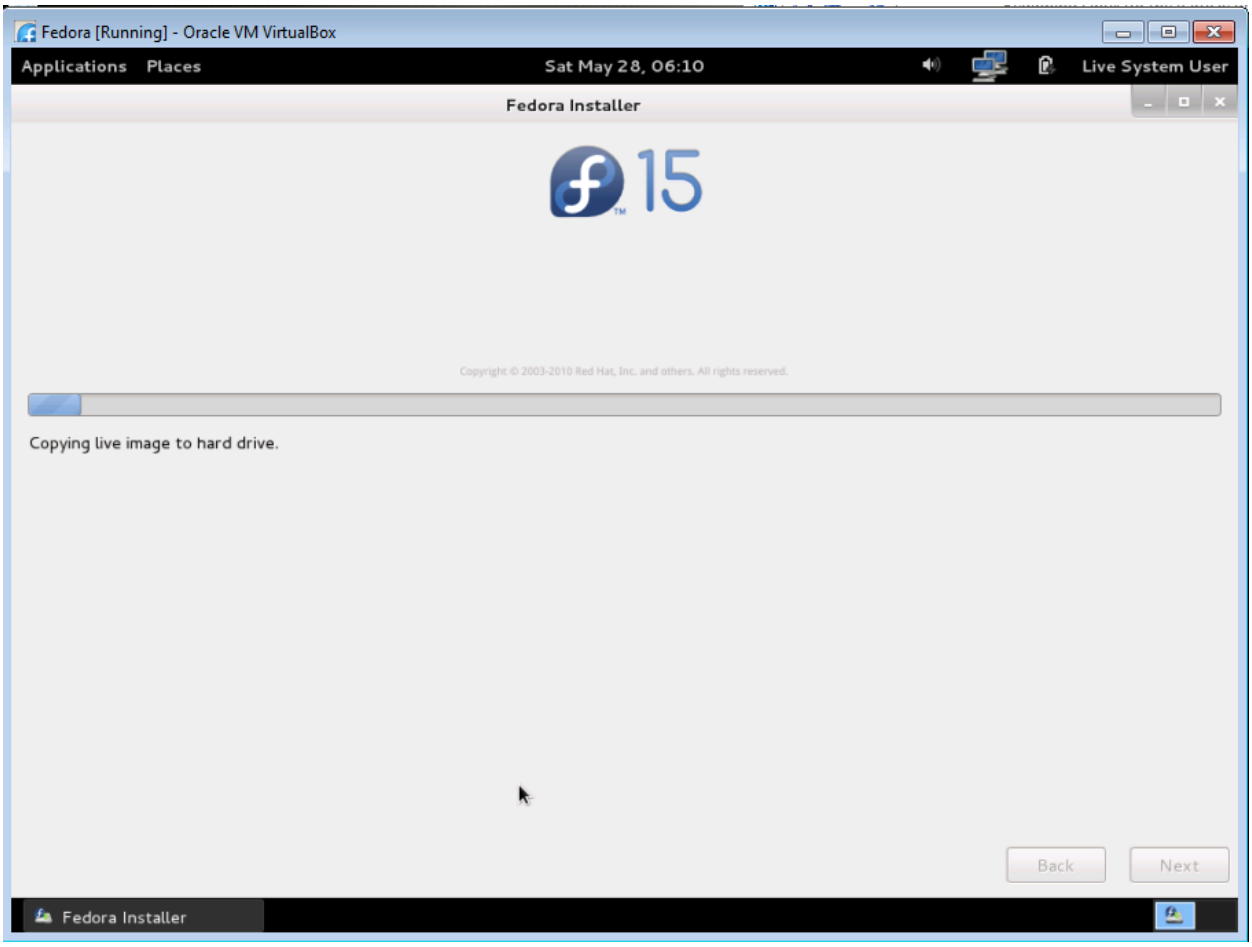
Choose the “Use All Space” option (as long as you don’t have another OS running in the same VM. If you don’t know what that means, select the “Use All Space” option. If you do know what that means and you want to replace your older VM, choose the “Replace Existing” option.)



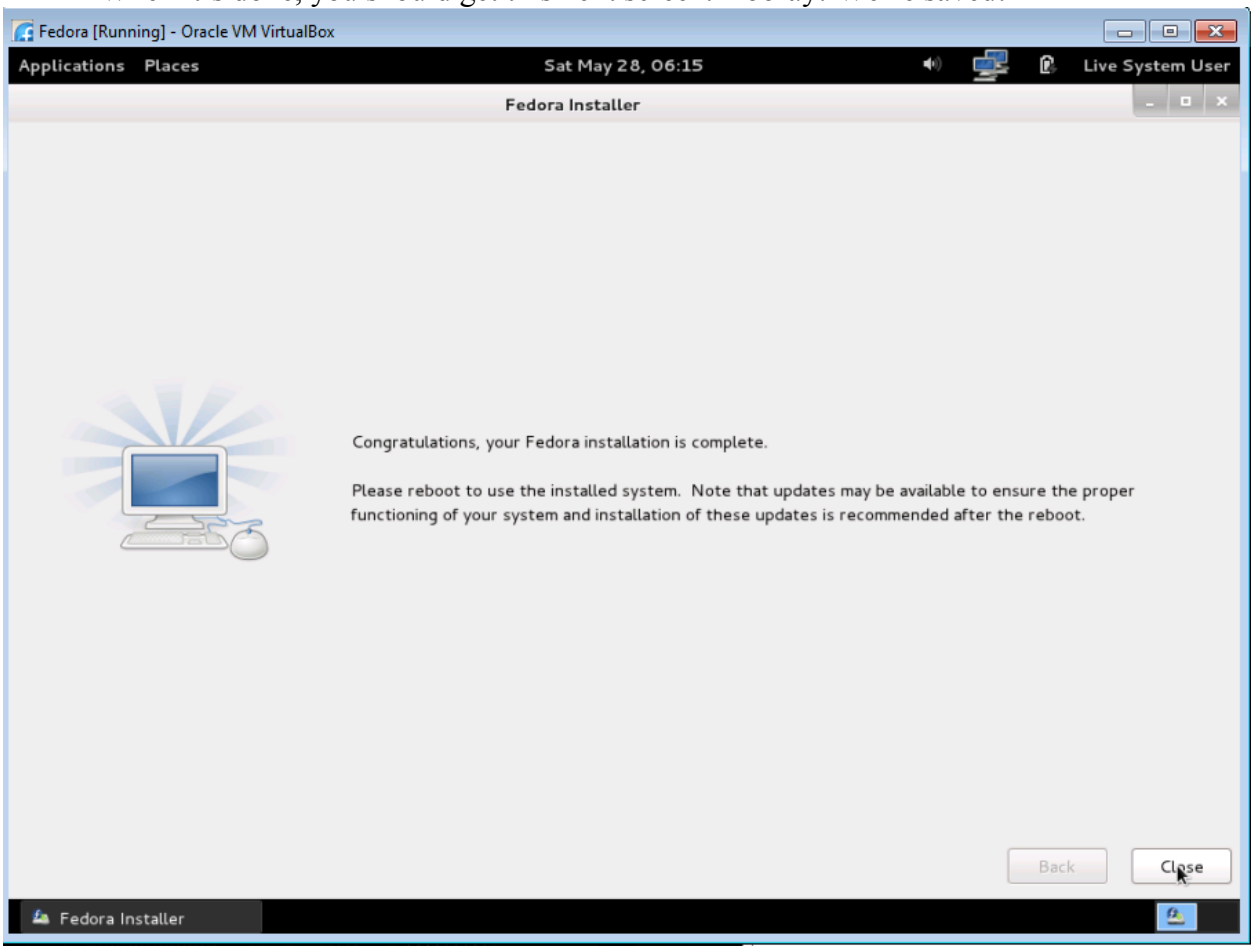
It’ll now ask you if you’re sure you want to do this. Go ahead and click “Write Changes to Disk”.



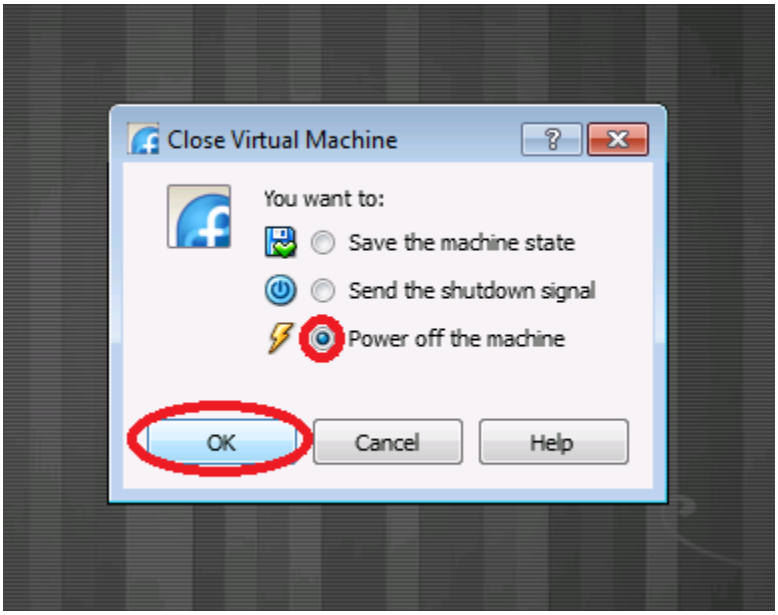
It will probably check a few things, and then it will go to a “Copying live image to hard drive” screen. You’ll need to wait a few minutes for this to load--I’d say 7-10. Also, it will go through “Performing post-installation filesystem changes” saying it might take a few minutes. It probably will. It will also check some things, do things with a bootloader, and generally just load.


Fedora [Running] - Oracle VM VirtualBox
Applications
Places
Sat May 28, 06:10
Live System User
Fedora Installer
15
Copyright © 2003-2010 Red Hat, Inc. and others. All rights reserved.
Copying live image to hard drive.
Back
Next
Fedora Installer

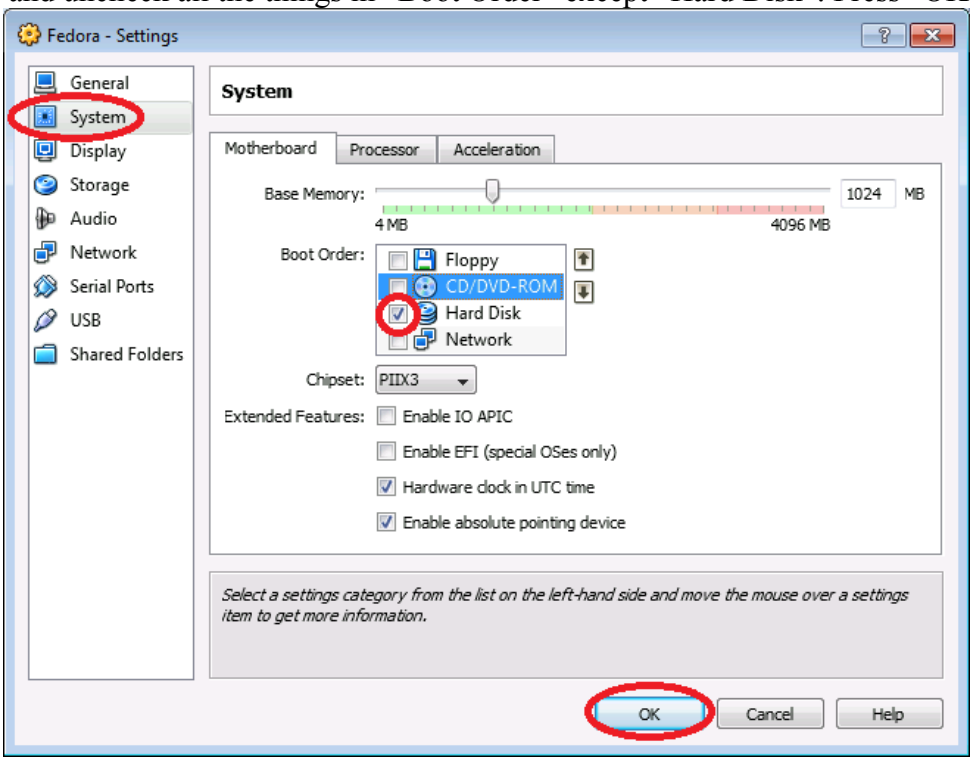
When it's done, you should get this next screen. Hooray! We're saved!



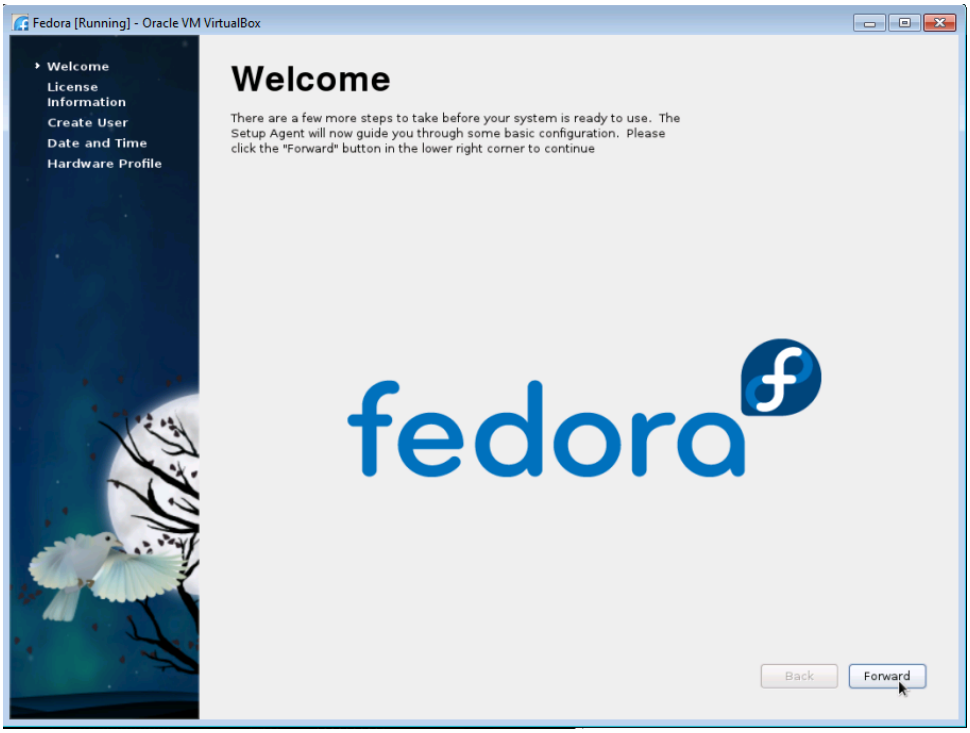
Click on the red "X" in the upper right corner, select "Power off the machine", and press "OK".



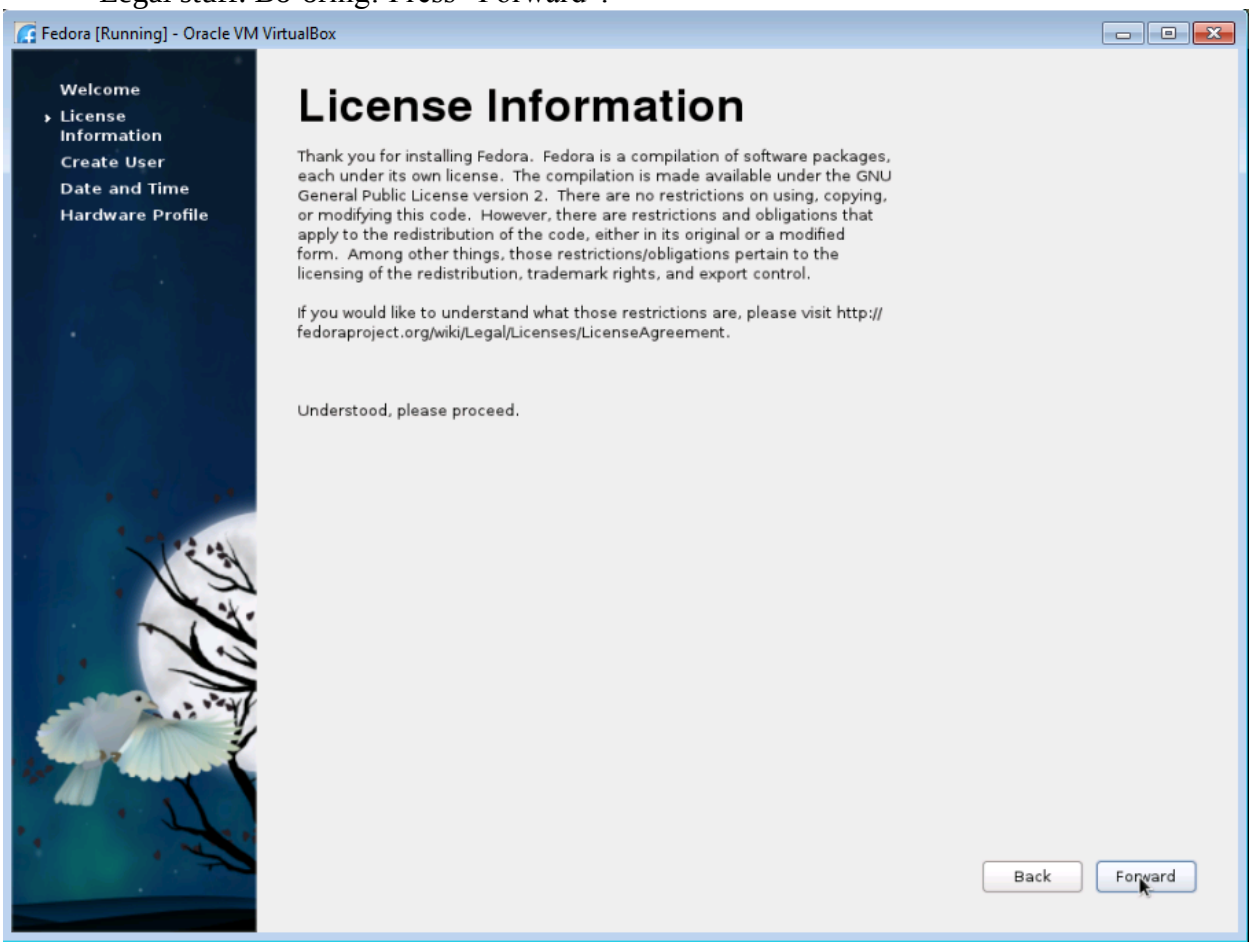
Now, from the VirtualBox Manager, go to “Settings” for Fedora, go to the “System” tab on the left, and uncheck all the things in “Boot Order” except “Hard Disk”. Press “OK”.



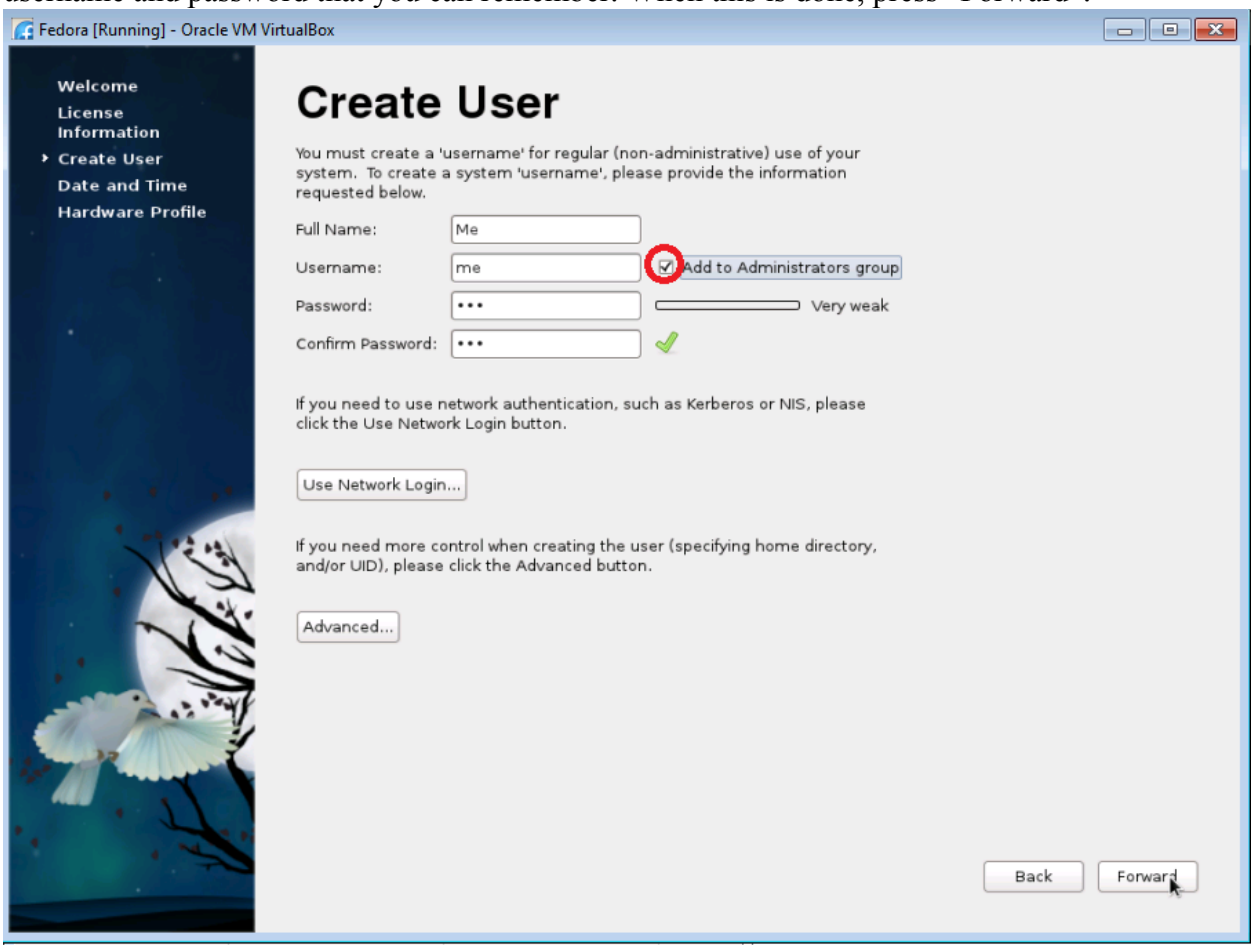
Now select the Fedora machine from VirtualBox and press “Start”. Don’t you feel welcome? Press “Forward”.



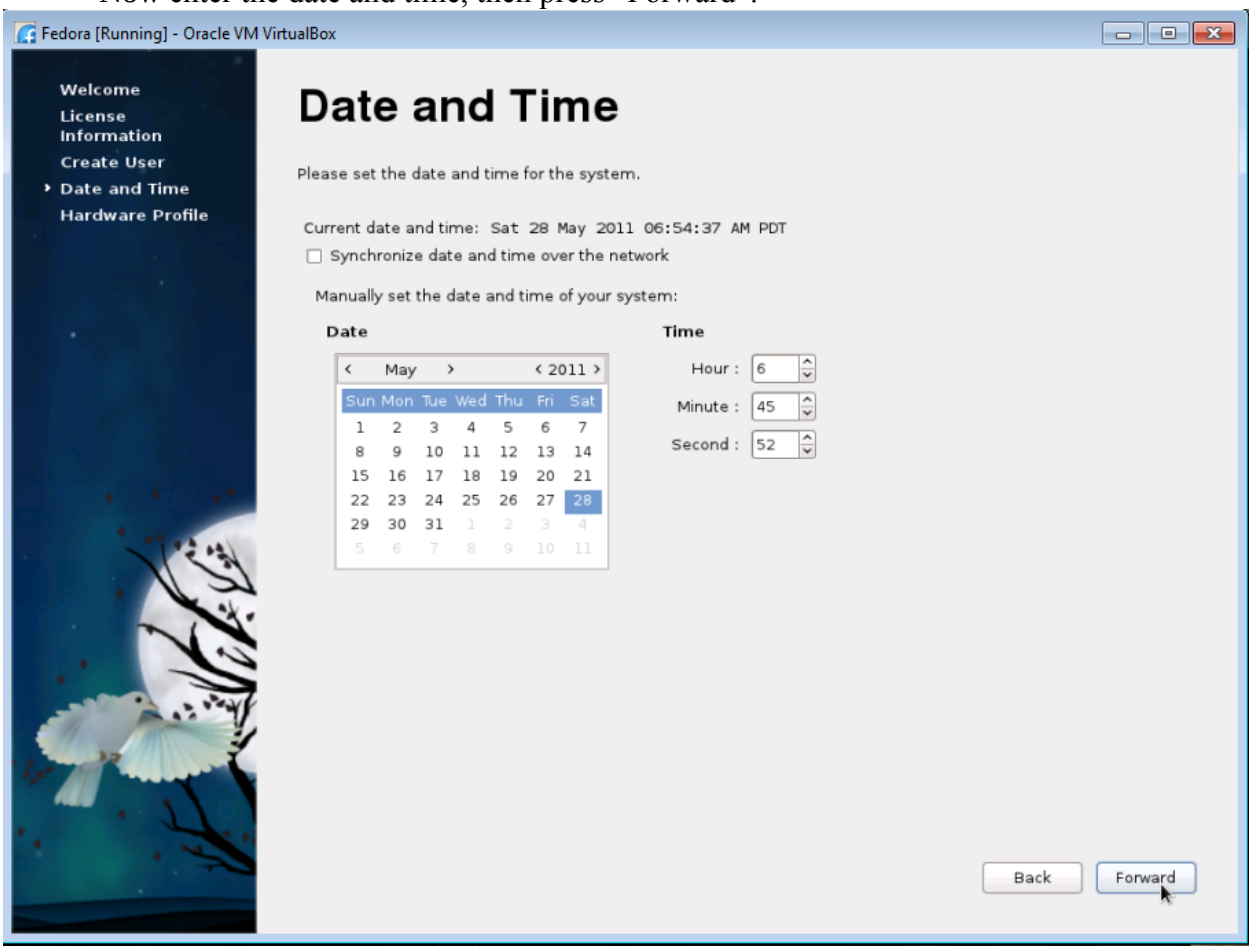
Legal stuff. Bo-oring! Press “Forward”.



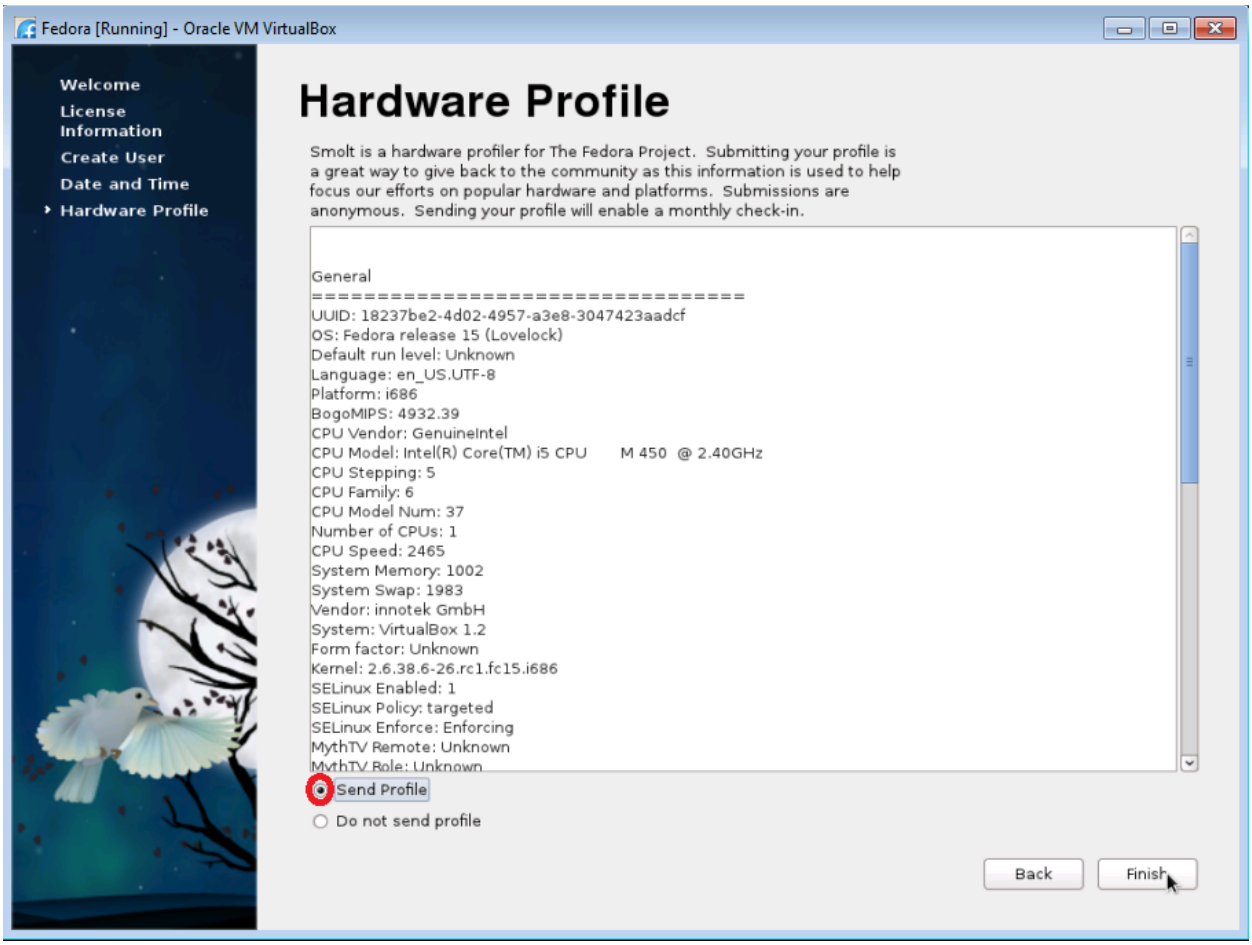
Now comes some important stuff. You'll set up a user. I would strongly suggest against using "Me" as a "Full Name" or "Username", and against using a three-letter password, but this is just an example. Check "Add to Administrators group" if this will be the main user. This will allow you to install stuff later. Do not use the same password as you used for the root user. Pick a username and password that you can remember. When this is done, press "Forward".



Now enter the date and time, then press “Forward”.



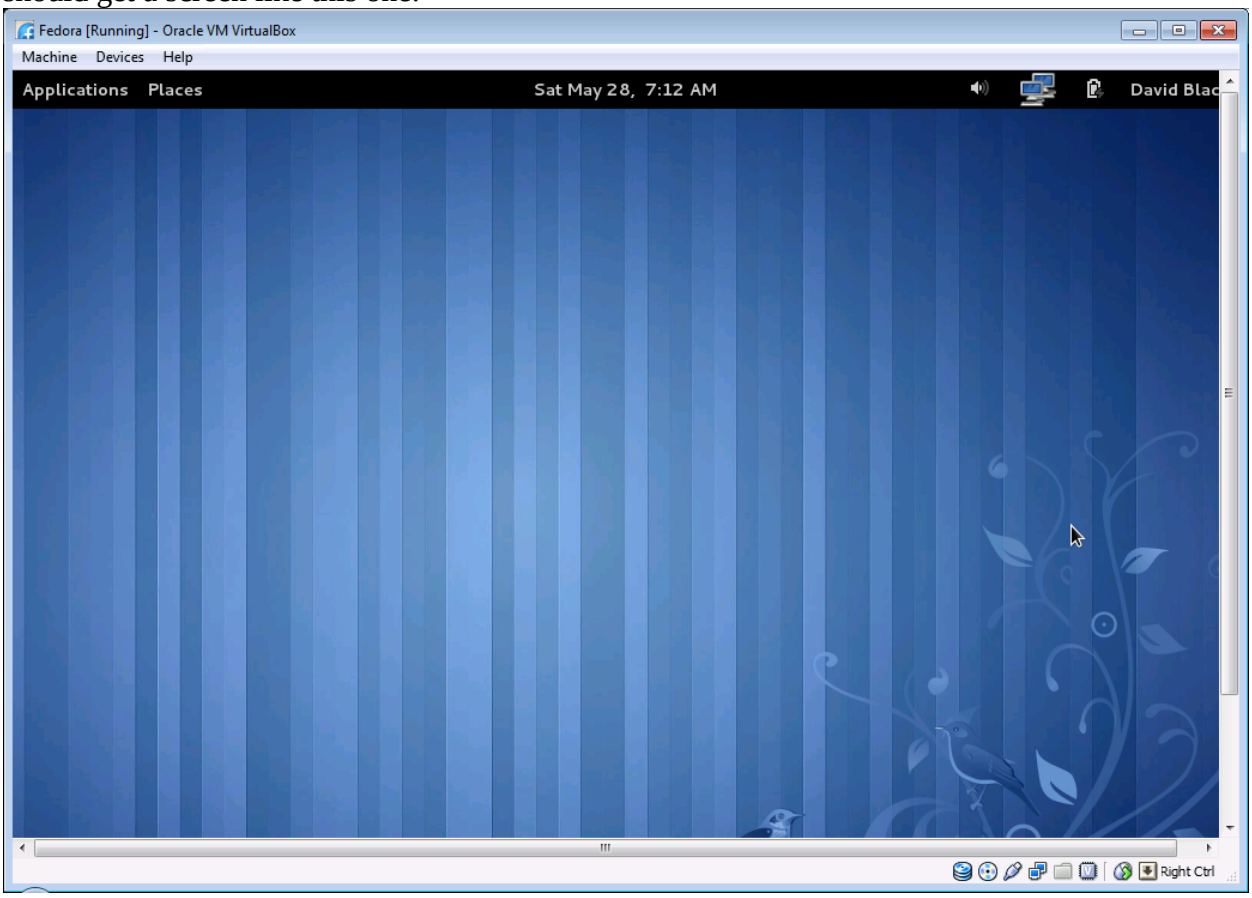
You will be asked if you want to send your hardware profile. This is actually a good thing to do, since if your install fails for some reason, the profile can be used to locate the error, or any bugs unique to your profile can be fixed. It also helps Fedora to fix things. If you don't want to send it, don't; it's your right. Press "Finish". This is the last button to push for the installation of Fedora Linux.



A log-in screen will come up. Click on your name (the username you just made) and enter your password. Now you're in Linux. Note that if your mouse gets "stuck" inside the Linux window, this can usually be fixed by pushing the Host button (right_ctrl).

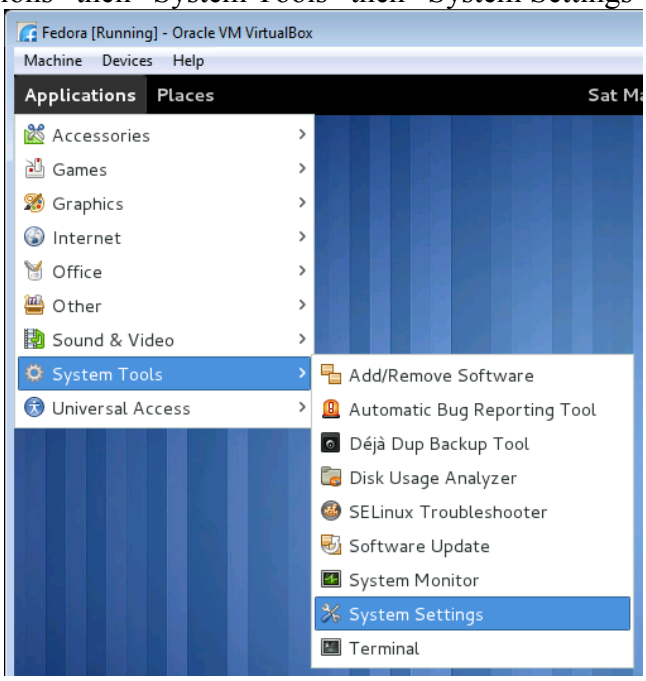
VirtualBox Guest Additions goto Navigation

Once you've done this, the same screen talking about how "GNOME 3 Failed" will come up. Click "Close". We'll fix that now by installing what's called "Guest Additions" and forcing fallback mode. This will also fix those annoying scroll buttons without Scale Mode. It also will allow you to copy and paste things from Windows to Linux, work in a seamless mode, and lots of other cool stuff. If you've been using Scale Mode, push Host+C (right_ctr + C) and you should get a screen like this one:

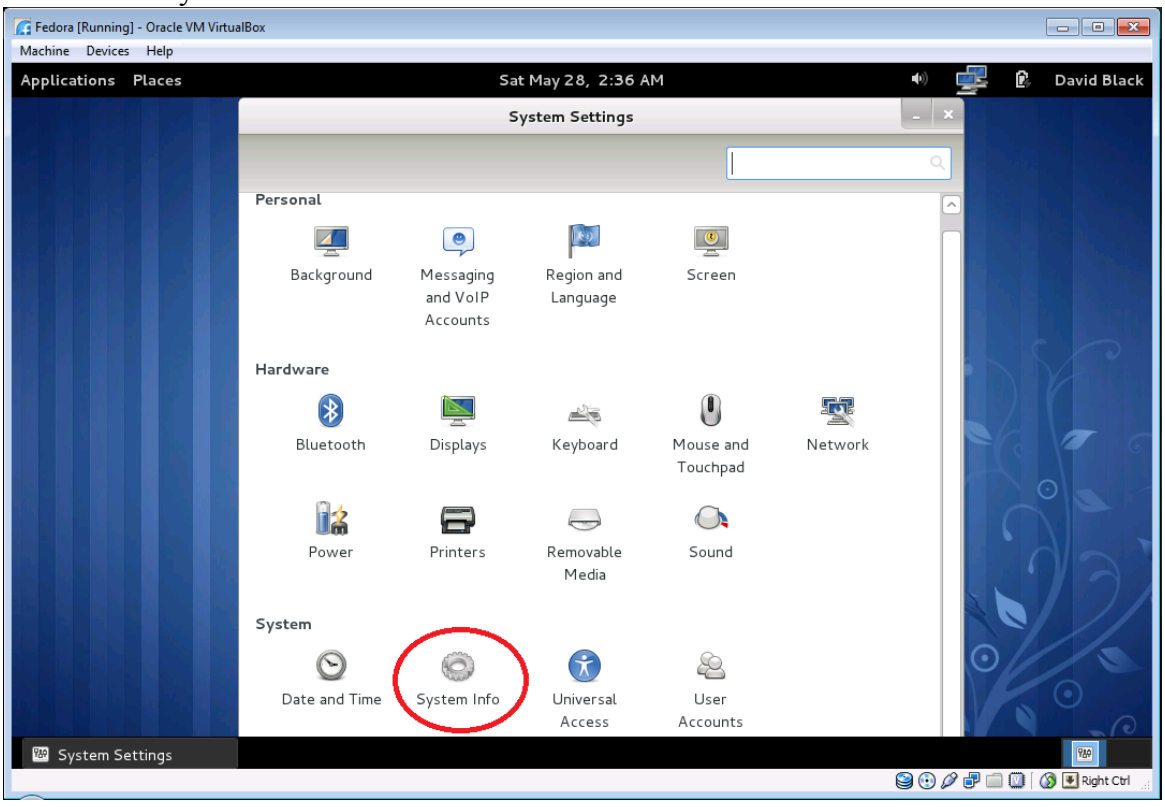


Part of the reason Linux is so nice is that it is easy to develop (write code) on it. With this is mind, I suggest you use the "fallback mode" by forcing VirtualBox to go there. Besides making it easier for development, this fixed some problems I had with resizing the window. You can try going into the new GNOME 3 desktop, but I ran into a bunch of problems. If you want the GNOME 3 desktop, you'll have to turn up the video memory in the "Settings"->"Display" part of VirtualBox. To use the settings, you'll need to turn off the VM. Note that Fedora 15 is very new while I'm writing this, so these bugs will likely get ironed out. If you installed Fedora 14, you can avoid these problems. Power off the machine (using the big, red X) and restart it (the Start button)

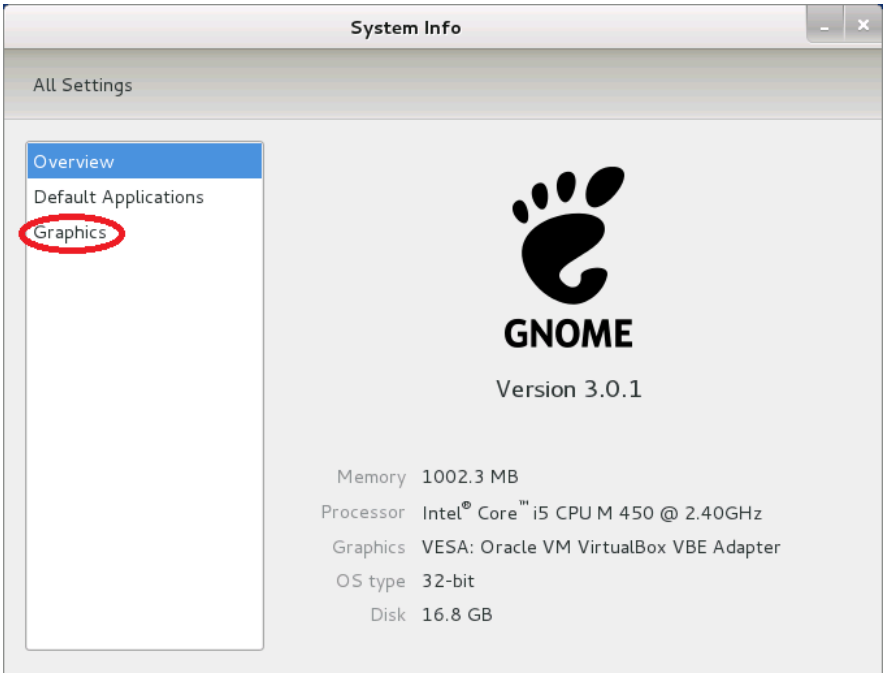
Go to “Applications” then “System Tools” then “System Settings”.



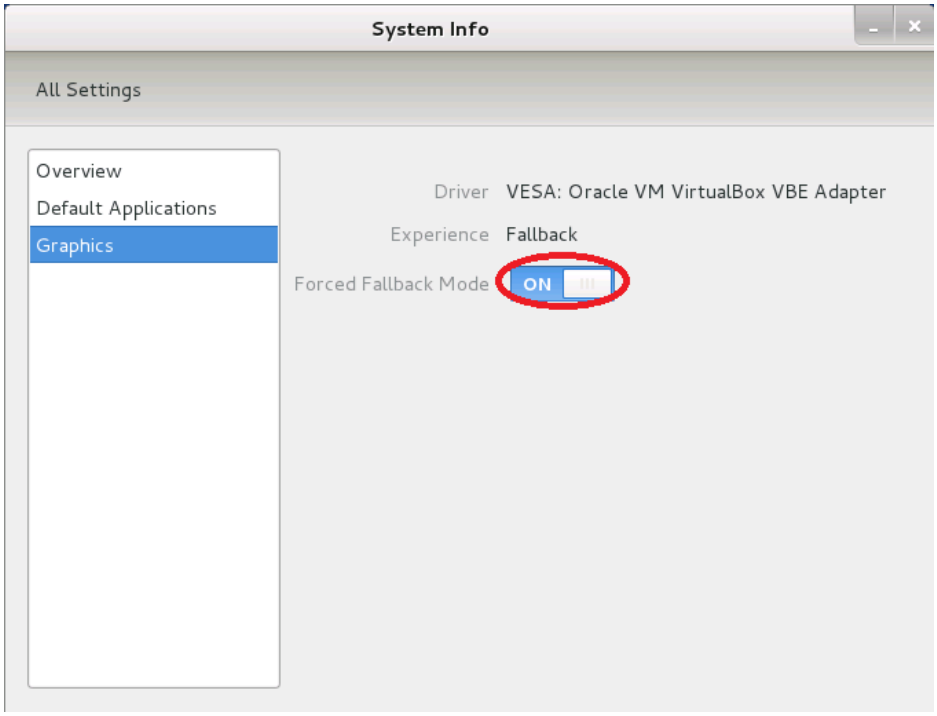
Now click the “System Info” icon.



Next, click on the Graphics option located on the left of the “System Info” window. I like the foot.

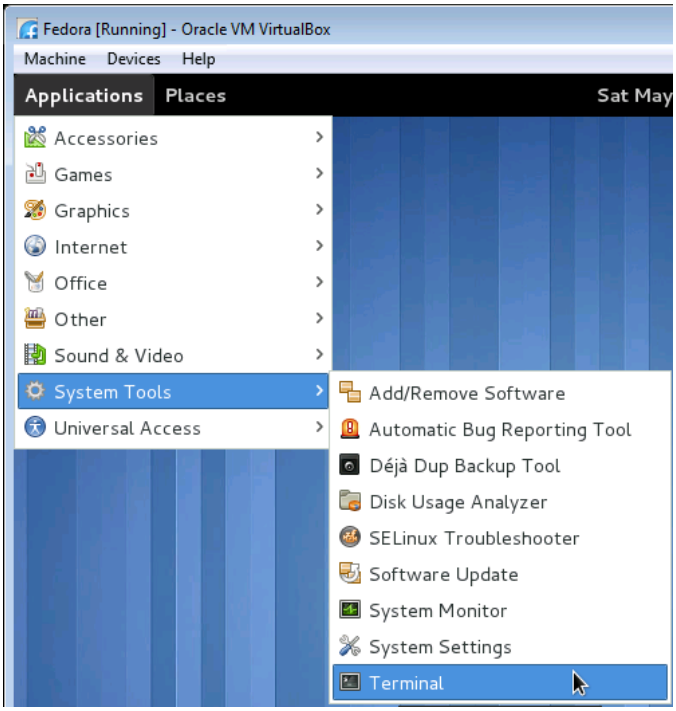


Finally, slide the switch next to “Forced Fallback Mode” to “ON”.

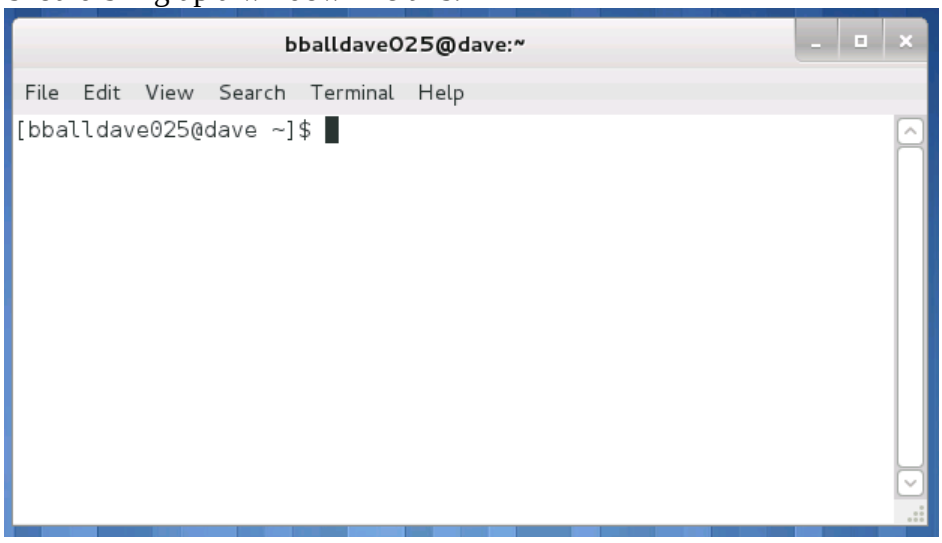


Exit out of the “System Info” window.

Now you get to learn how to install packages. Go to “Applications” then “System Tools” then “Terminal”. Welcome to the Linux terminal. You’ll use it a lot.

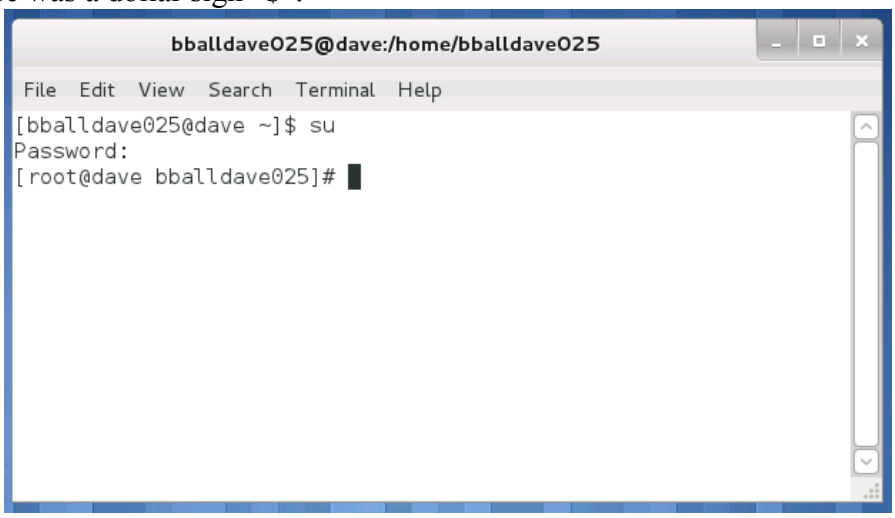


This should bring up a window like this.

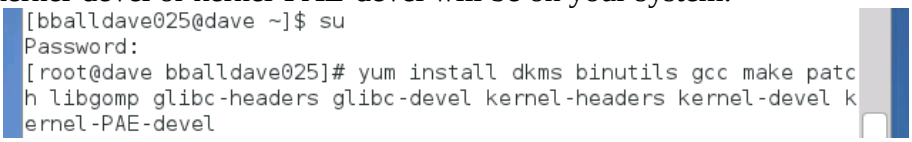


Before we get going, I want to share with you the wonders of “ctrl+C” (hold down either “ctrl” button and press the “C” key on your keyboard--caps don’t matter). Whenever anything is acting weird, this is the magic way to abort everything. When you’re in the terminal is when “ctrl+C” becomes your friend. Anything goes screwy, that will abort everything.

You need to install stuff as the root user. (Once again, this is the Linux user that can change things on the computer, not the particle physics program, ROOT, that we'll install later.) To log in as the root user, type "su" (for super-user) at the terminal prompt. It will ask you for your password. Use the password you made for root. The terminal should now have a pound sign (#) where there was a dollar sign "$".

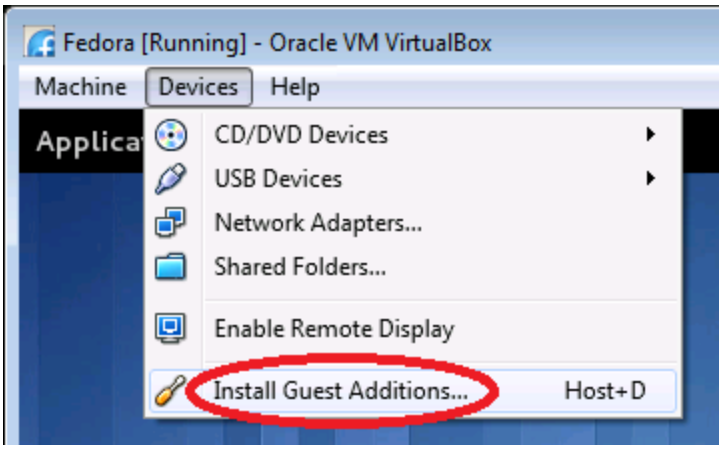


Now you get to meet yum, which is an installer. /*If you're at a place where there's a proxy, you'll need to set up Linux, the terminal, and yum to go through this proxy. I might put more stuff in about this later. I had to do it while re-installing at BNL.*/ Type "yum install dkms binutils gcc make patch libgomp glibc-headers glibc-devel kernel-headers kernel-devel kernel-PAE-devel". It would be really nice to copy and paste this. Unfortunately, you can't copy and paste between host and guest (Windows and Linux) until Guest Additions is installed. Sorry. Note that either kernel-devel or kernel-PAE-devel will be on your system.

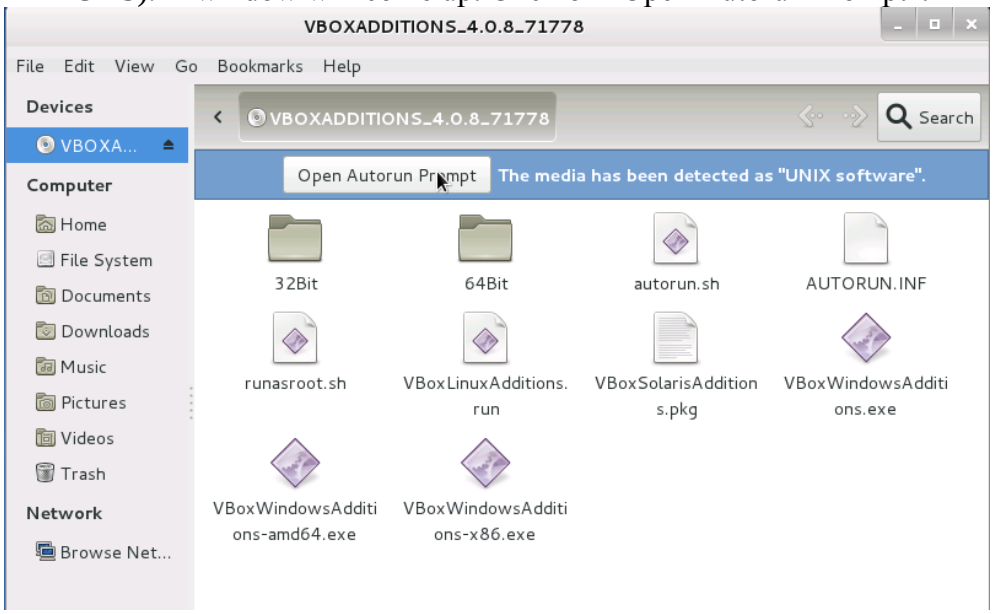


It will now install a bunch of stuff. From time to time, it will ask, "Is this ok [y/N]:". Type "y" and press enter. When it comes up with "Complete!", you're ready to move on. Enter "exit" at the command prompt to log out from root. /* This is a note for myself--something you don't have to do, and which could take 10-20 minutes. I type "yum update" at this point, just to make sure everything's current.*/

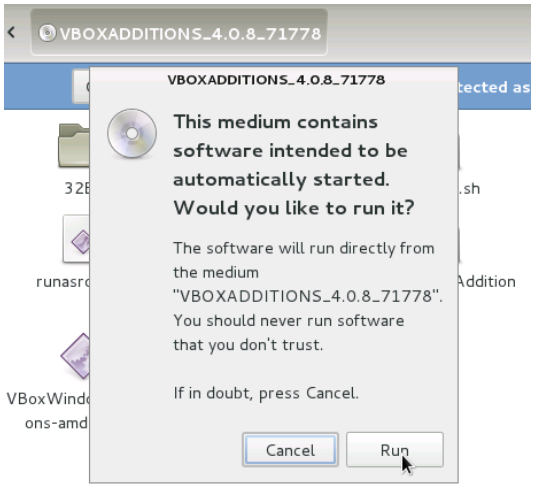
Click on "Devices" then "Install Guest Additions…". Note that nothing will come up, but it will install what you need to get it going. Just click where I've circled and then go on to the next step.



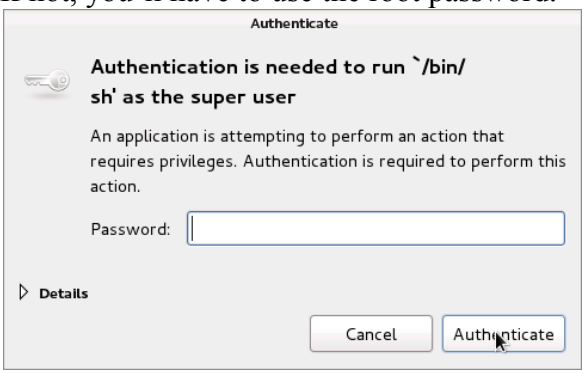
Now go to “Places” and then “VBOXADDITIONS”… (there will be some numbers after VBOXADDITIONS). A window will come up. Click on “Open Autorun Prompt”.



A warning will come up. Click “Run”.



You'll need to authenticate. If you made your user an administrator, you can just enter the username password here. If not, you'll have to use the root password.



This might take a while (~3-5 minutes), especially the "Installing graphics libraries and desktop services component" part. Press Return when tells you to. When I ran it, I got the following error:

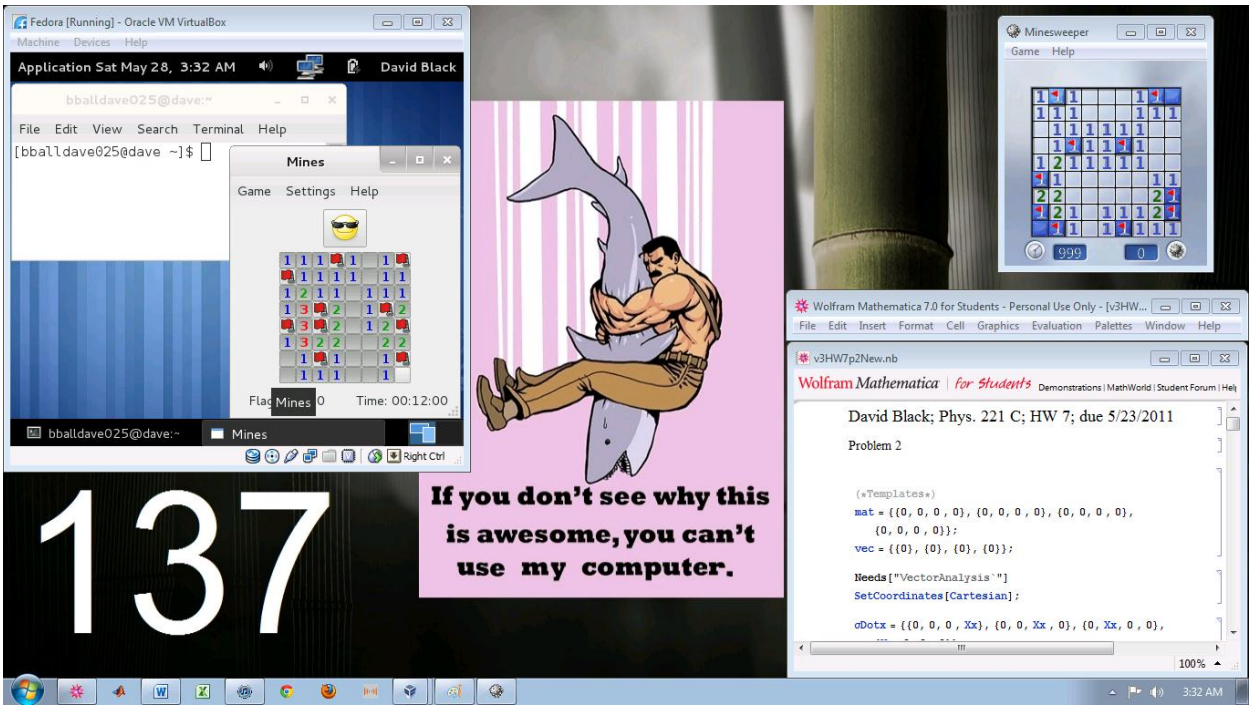
```
Building the VirtualBox Guest Additions kernel modules
The headers for the current running kernel were not found. If the following
module compilation fails then this could be the reason.
The missing package can be probably installed with
yum install kernel-devel-2.6.38.6-26.rc1.fc15.i686
                                                              [FAILED]
Your system does not seem to be set up to build kernel modules.
Look at /var/log/vboxadd-install.log to find out what went wrong.
Once you have corrected it, you can run

  /etc/init.d/vboxadd setup

to build them.
```

This is easy enough to fix; just do what it tells you to. Wait until that installation finishes, then go to the terminal and log in as root (enter "su") and type "yum install kernel-devel-2.6.38.6-26.rc1.fc15.i686". Remember to exit root when you're done by typing "exit". Press "y" when it asks you.

```
[bballdave025@dave ~]$ su
Password:
[root@dave bballdave025]# yum install kernel-devel-2.6.38.6-26.rc1.fc15.i686
```

Go back and redo the installation of the VirtualBox Guest Additions from the "Places"->"VBOXADDITIONS" part. You'll need to restart the VM for Guest Additions to work. I'll show you how to do this below, after the screenshot of my Windows desktop. Note that, on my machine, the "Send the shutdown signal" option isn't working.

Now you can restart the VM and you'll be able to run Linux while running other things in Windows! Oh, and if you're a little confused about this shot of my desktop I had to squeeze in, you're either a really beginning particle physicist or something else. Check this link out < http://en.wikipedia.org/wiki/Fine-structure_constant#Numerological_explanations>.

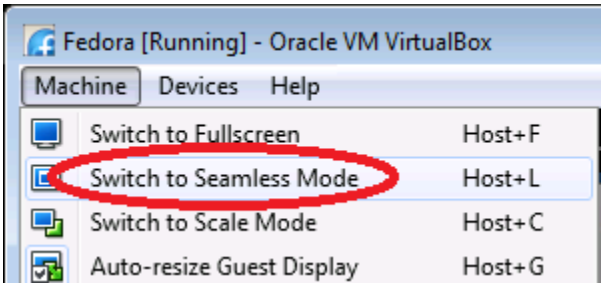


And, of course, the first thing you know when you start a program is how to stop that program. The best way is to log out by clicking on your name in the upper-right corner and then clicking "Log out…". When this is done, click on the red "X" in the upper-right corner of the VirtualBox window, select "Power off the machine", and click "OK", as shown in the screenshot. These instructions are what I want you to do when I say, "Turn off the machine," or "Power off the machine."

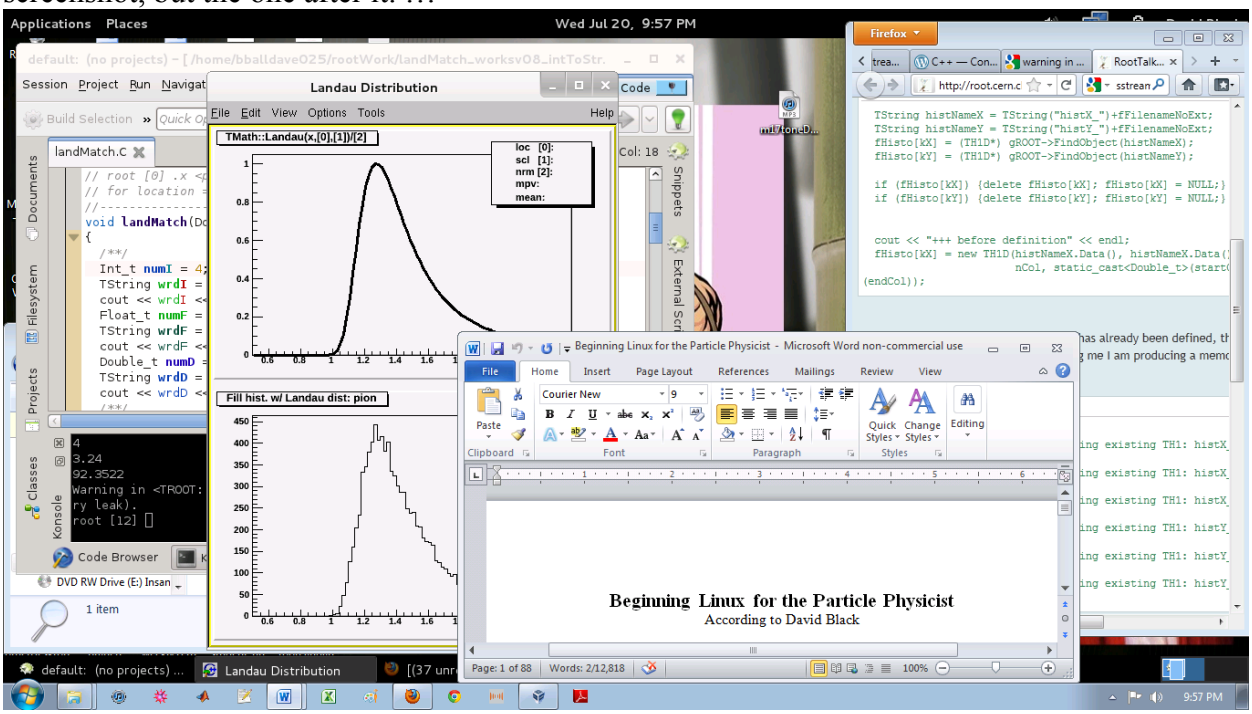
You won't want to do this every time you exit out of the VM, though. Most times you won't log out and you'll just select "Save the machine state" from the "Close Virtual Machine" window that comes up when you click the big red "X" in the upper-right corner.

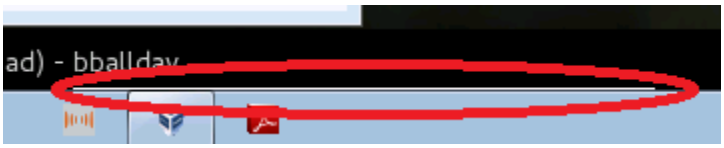
When I first wrote this tutorial, I thought that what I was using--the Auto-Resize option, from the Fedora window: "Machine"->"Auto-Resize Guest Display"--was seamless mode.

Seamless mode is, I think, a lot better. Once again, from the Fedora window: “Machine”->”Switch to Seamless Mode”.

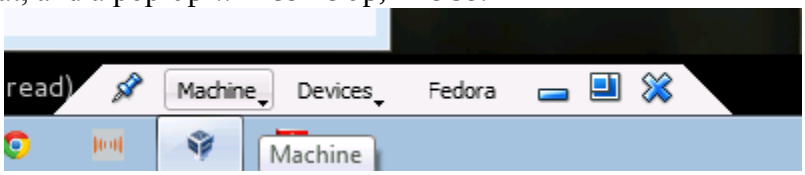


Now, your desktop can look like this, with the Linux toolbar stuff sitting on top of the Windows toolbar stuff, and not even having to switch between windows. That means you can take any window, be it in Linux or Windows, make it full screen, go back and forth much more easily; life is good. To exit seamless mode (for example, if you want to turn off the VM,) you’ll need to press the right ctrl button and the “L” key (“right-ctrl+L”), or you can do it from the VirtualBox icon on your Windows toolbar (right-click it and select “Close Window”, at which point it will ask you about saving the machine state, sending the shutdown signal, or powering off the machine). There’s another way, a little bar that I’ve circled in red, not in the very next screenshot, but the one after it. …





… Hover over that, and a pop-up will come up, like so:

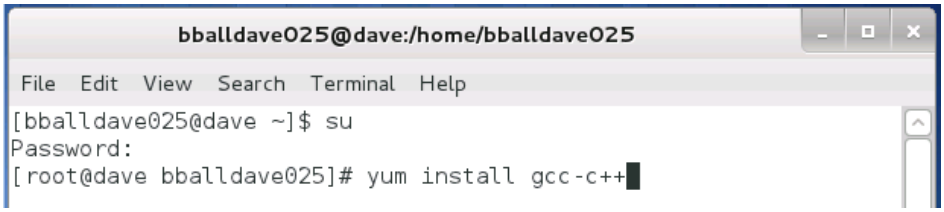


You can do whatever you need to from there.

Starting C goto Navigation

Now it's time to do a little bit of programming. Don't worry, we'll get to the physics soon enough. Go ahead and open up your terminal ("Applications" -> "System Tools" -> "Terminal"). We want to start by installing a C++ compiler. A compiler takes your program and makes it so that the computer can run it.

Log in as root by typing "su", after which you will enter your root password. Now type "yum install gcc-c++". Type "y" when it asks. If you get anything about trying another mirror, don't worry about it. It will just keep installing. After it's complete, type "exit" to log out from root.

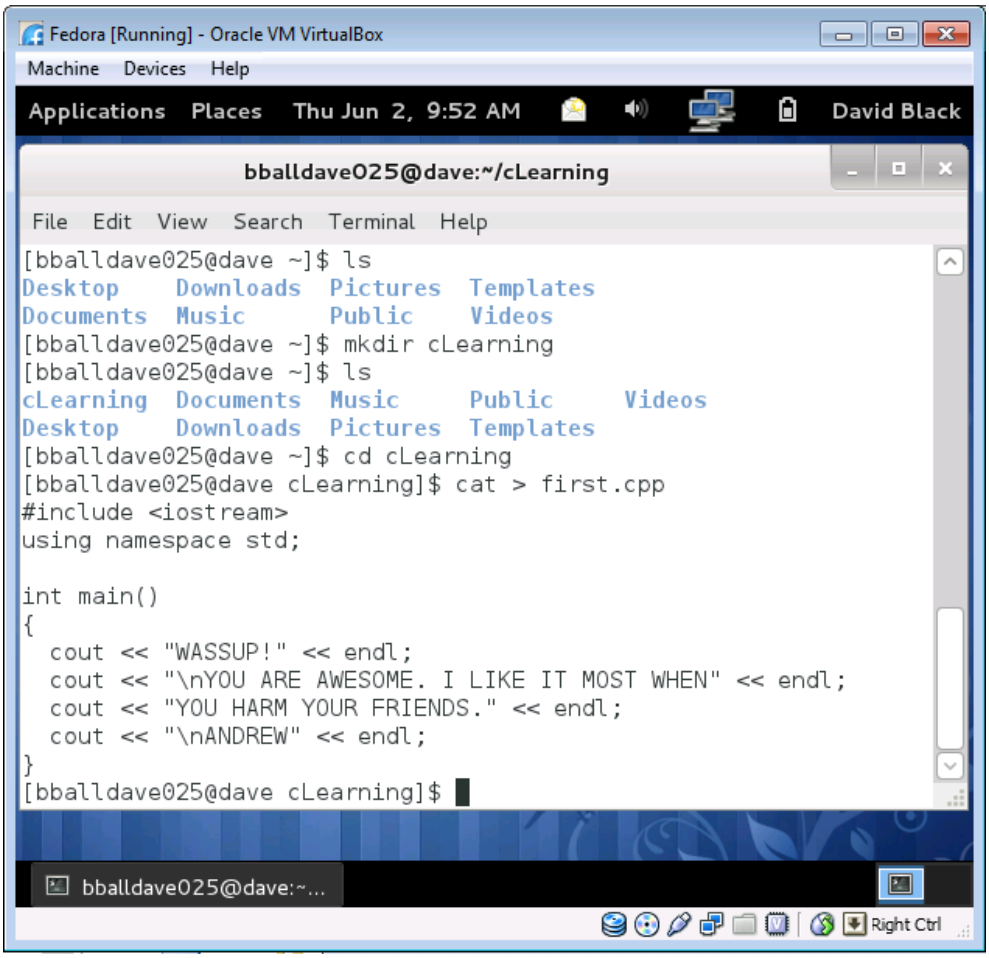


Now we're going to write a quick program. First, we'll just do it on a terminal. After opening the terminal, type "ls" (for "**ls**t"). This will list all of the directories (folders) that are available from your home folder (in my case, from "bballdave025"). It's got some cool stuff. Now we're going to make a new directory by typing "mkdir cLearning". "mkdir" stands for "**m**a**k**e **dir**ectory". We'll call it cLearning, because, well, we're learning C. DO NOT use spaces while naming your directories. Now when you type "ls", you'll see the directory there. By the way, if this seems a little unclear, I have a screenshot with everything you need to do a little further down.

Now we'll go into that directory. (Remember that "directory" is just another fancy word for "folder".) Type "cd cLearning" at the terminal—"cd" stands for "**c**hange **d**irectory. Now we'll use a tool called cat to write a program right in the terminal. Type what I have indented just after this paragraph. [Enter] means hit the enter key. If you make a mistake press "ctrl+C" (while holding down the "ctrl" button, press the "C" button on your keyboard). You *could* just copy and paste this to the terminal (though you have to right-click and select "Paste" on the terminal--"ctrl+V" won't work,) but typing it will help you learn programming.

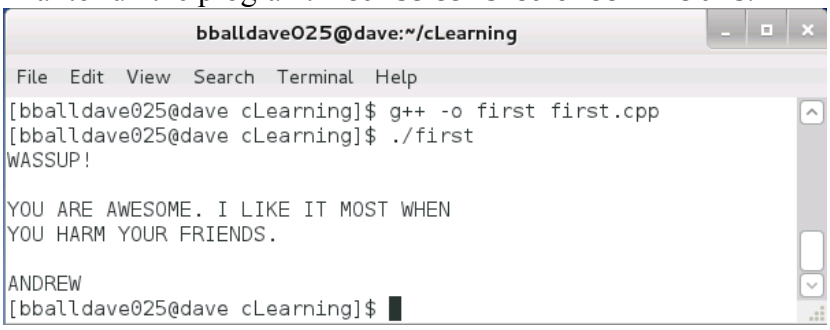
```
cat > first.cpp[Enter]
#include <iostream>[Enter]
using namespace std; [Enter]
[Enter]
int main()[Enter]
{[Enter]
 cout << "WASSUP!" << endl; [Enter]
 cout << "\nYOU ARE AWESOME. I LIKE IT MOST WHEN" << endl; [Enter]
 cout << "YOU HARM YOUR FRIENDS." << endl; [Enter]
 cout << "\nANDREW" << endl; [Enter]
}[Enter]
```

When you've typed all this, press "ctrl-"D (for **D**one). You've just written a C++ program.



This isn't supposed to be a tutorial on programming, so I won't go into details about what everything means. Just a quick note that a "<" or a ">" means that something is getting directed in whatever direction the sign is pointing and that "\n" means a new line.

Now we want to compile the program. This is done by typing "g++ -o first first.cpp". "g++" is the compiler, the "-o" makes an object (runnable) file, "first" is the name of the object file, and "first.cpp", of course, is the file we just wrote. To run the program type "./first". The "./" tells the terminal to run the program. Your screen should look like this:

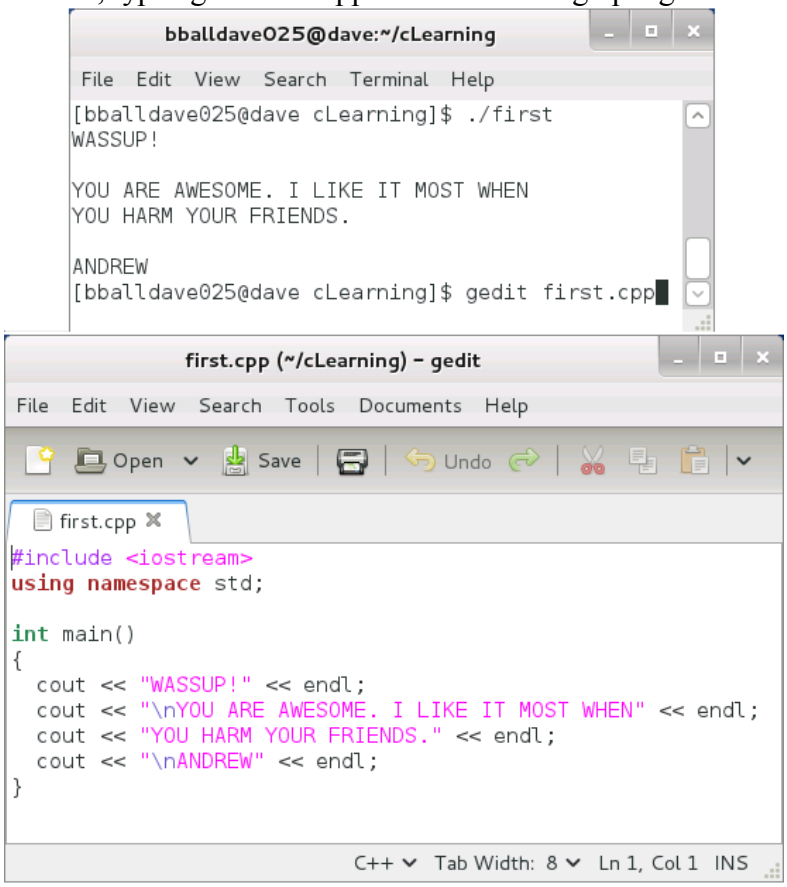


This is the best beginning program I've ever put in a tutorial. [Your friends](<http://www.homestarrunner.com/sbemail46.html>) might be able to explain why. You could put

in something boring for the computer to output like a [salutation to the planet](http://en.wikipedia.org/wiki/Hello_world), but that's not my style.

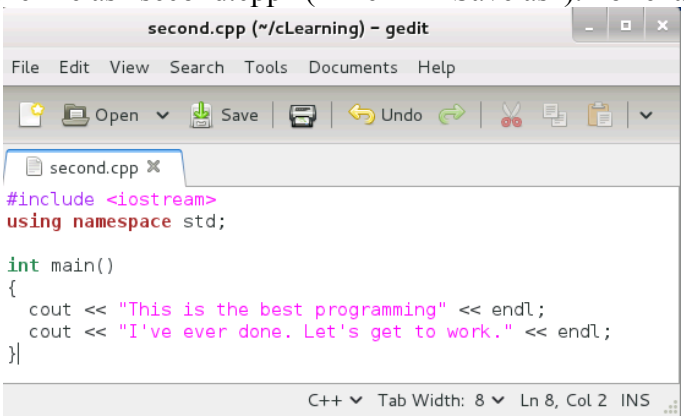
You might have noticed when you were entering things in that it can be a pain when you make a mistake. A text editor makes things nice. I'm going to show you the "gedit" editor that comes with Linux, and then a KDevelop that is nice for more advanced programming. Once again, this isn't a programming tutorial, so I'm just going to go over them briefly. Some people will use "vim" or "gvim" or "vi" or "emacs". Any text editor will do.

From the terminal, type "gedit first.cpp". This will bring up a gedit window.

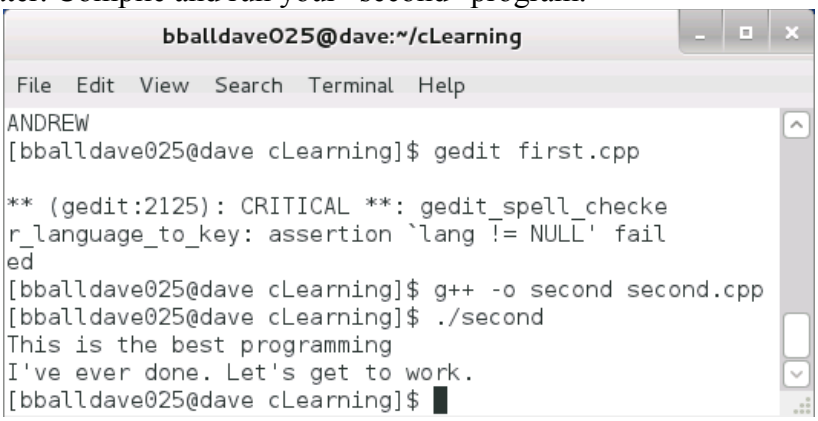


The program will be a lot easier to edit here. It also gives cool colors that make programming easier. Go ahead and delete and/or add lines like those that begin with "cout", and replace the things in quotes with whatever you want. This will make the computer output different things.

Save the file as “second.cpp” (“File” -> “Save as”). For example:



Go ahead and return to the terminal by exiting out of gedit. (Use the “X” at the upper right corner.) My terminal came up with an error, but that’s no big deal right now. I might have to update gedit later. Compile and run your “second” program.



KDevelop IDE goto Navigation

Good work! If you plan on doing more serious programming (which I assume you do), you'll probably want a more sophisticated editor. I'll show you how to use KDevelop. Note that there are other such editors, such as gvim ("yum install vim*") and emacs "yum install emacs" that you can use. A similar one to KDevelop is Oracle. Log in as root and type "yum install kdevelop". Remember to type "exit" after you're done. This installation could take a little bit of time, since this is a really good editor. I'd say 5-10 minutes. Actually, it's called an Integrated Development Environment (IDE), which is a fancy name for a fancy text editor.

```
[bballdave025@dave ~]$ su
Password:
[root@dave bballdave025]# yum install kdevelop
```

I'll be illustrating this IDE with a simple program called `particleInfo.cpp` . The ".cpp" is for a C++ program. I've put the whole program in an appendix. It looks long, but that's because I've documented it (put in comments so other people, [and me, when I'm looking at it in six months,] know what's going on.) Without the comments, even with good line-indentation practices, there are only 162 lines (okay, I messed with it and added stuff, so the number's bigger now, but the idea's the same): of which 18 are white space, 22 are lines with just a brace "{" or "{", and several are multi-line strings, which could be made into one line. Basically, it could be written and compiled with about 100 lines. I strongly suggest against writing programs like that, though. Good programming practices like that save you tons of time.

If you're at BNL and just need to get ROOT running quickly, you can skip down to the ROOT link. However, this will be helpful for your learning, so I suggest doing it.

Anyway, you'll eventually copy and paste the whole program into Linux. (That's right, with VirtualBox, you can copy and paste text. Files are a little more difficult, but we'll get to that when we talk about Shared Folders.) Then I'll show you how to compile and run it. You'll also learn something about linking to libraries, which will be very helpful if you want to write any kind of useful program. A library is a group of functions (lists of programming commands) which have already been written. That means you don't have to write them. There are a bunch that you can get from standard compilers—called standard libraries—to which you don't have to link. However, in order to make the program do a cool scrolling thing, we'll use a library called ncurses. Yeah, you may want to swear sometimes when you're programming, but these are just tools for the cursor.

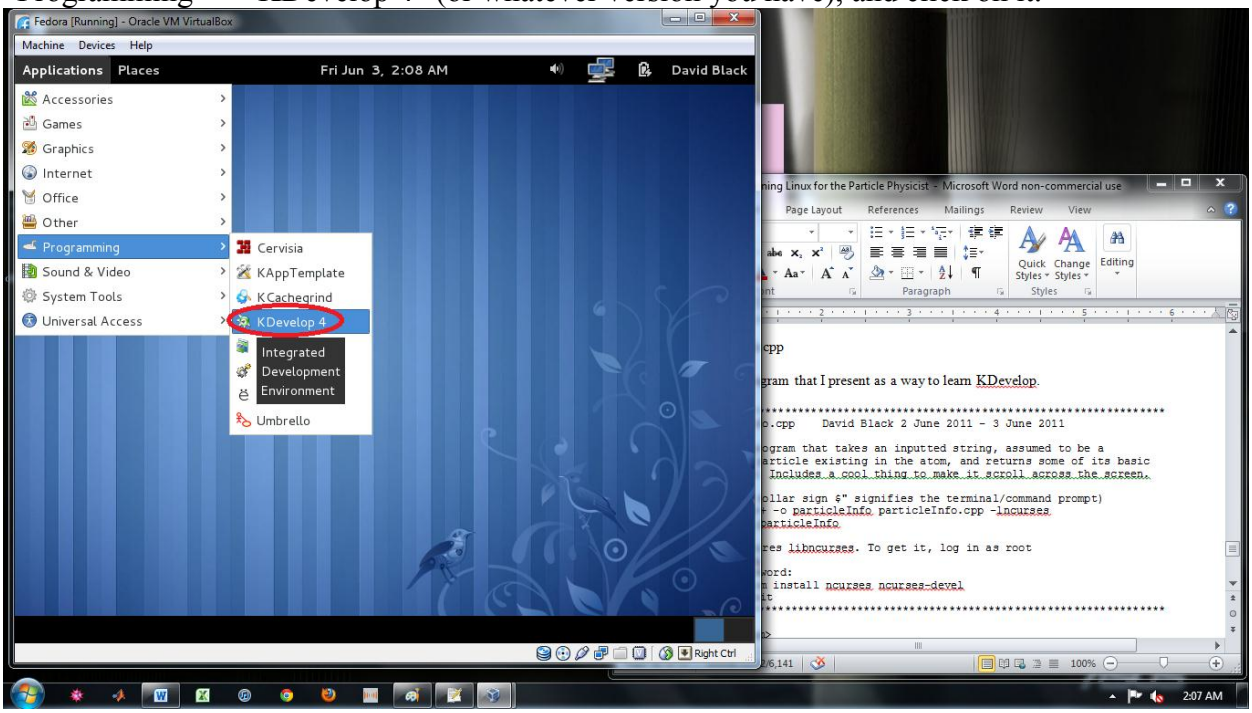
So, hopefully KDevelop has installed by now. Now we'll log in as root and install "ncurses" and "ncurses-devel". The "devel" just has more programming tools. Do this (type "yum install ncurses ncurses-devel"):

```
[bballdave025@dave ~]$ su
Password:
[root@dave bballdave025]# yum install ncurses ncurses-devel
```

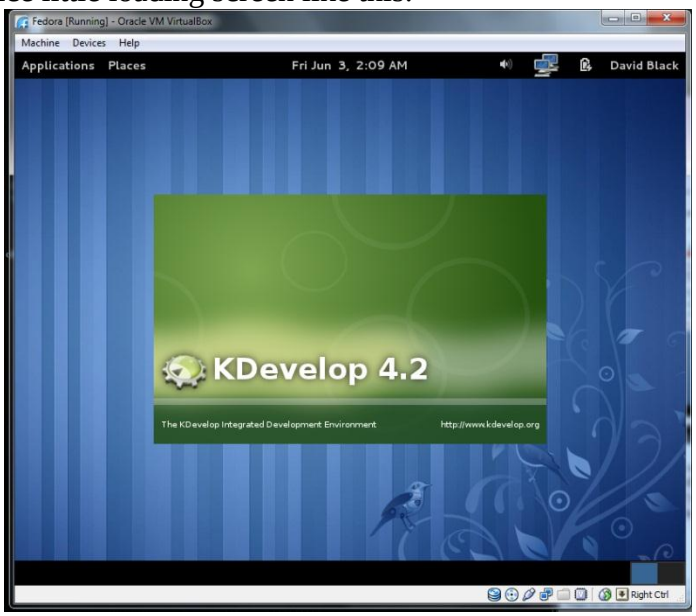
By the way, if there's something you need to install but you don't know the exact name, you can type "yum search <whatever>". For example, you could type "yum search ncurses" and find the things we just installed.

Okay, after all that jabbering, we're finally ready for (what I consider) a really cool program. Go ahead and go down near the end of the document to Appendix A. Highlight it all—from the beginning "`//*******`" all the way down to the "`}//endof void delay(int)`"—and copy it either by right-clicking the mouse and selecting copy or by pressing "ctrl+V" on your

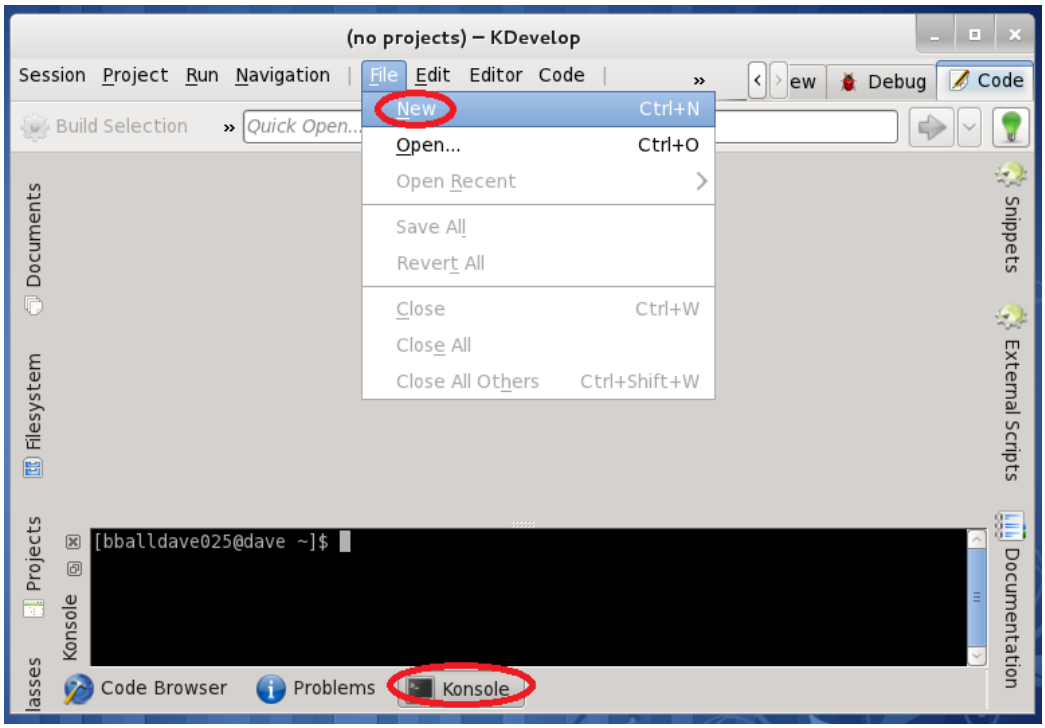
keyboard. Now go to your Linux Virtual Machine window. Go to “Applications” -> “Programming”-> “KDevelop 4” (or whatever version you have), and click on it.



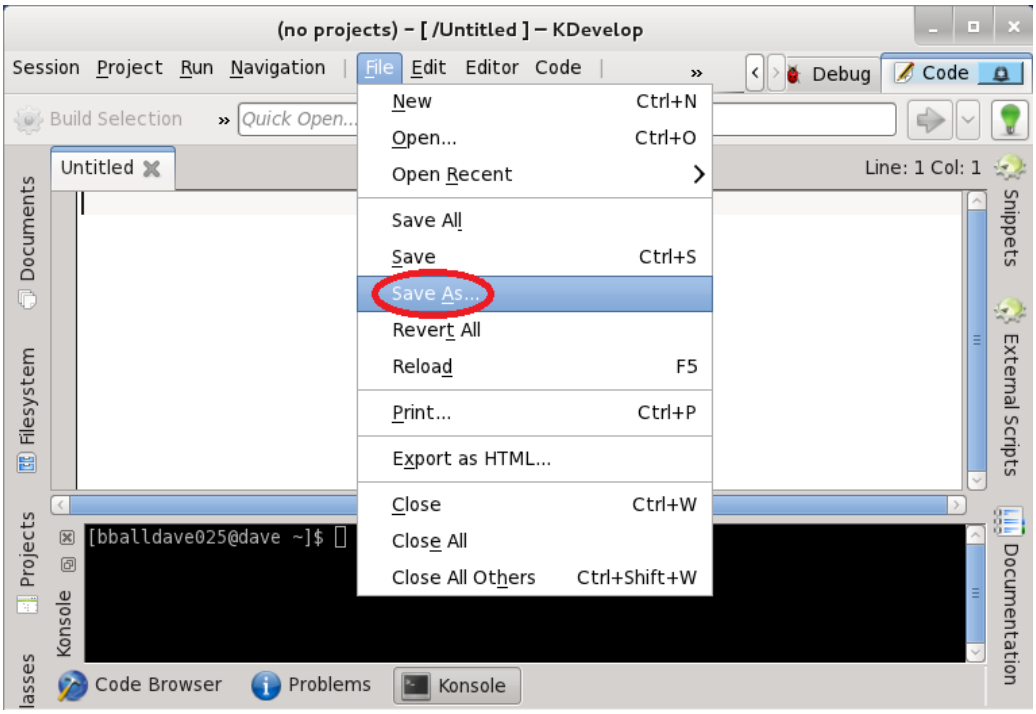
You’ll get a nice little loading screen like this:



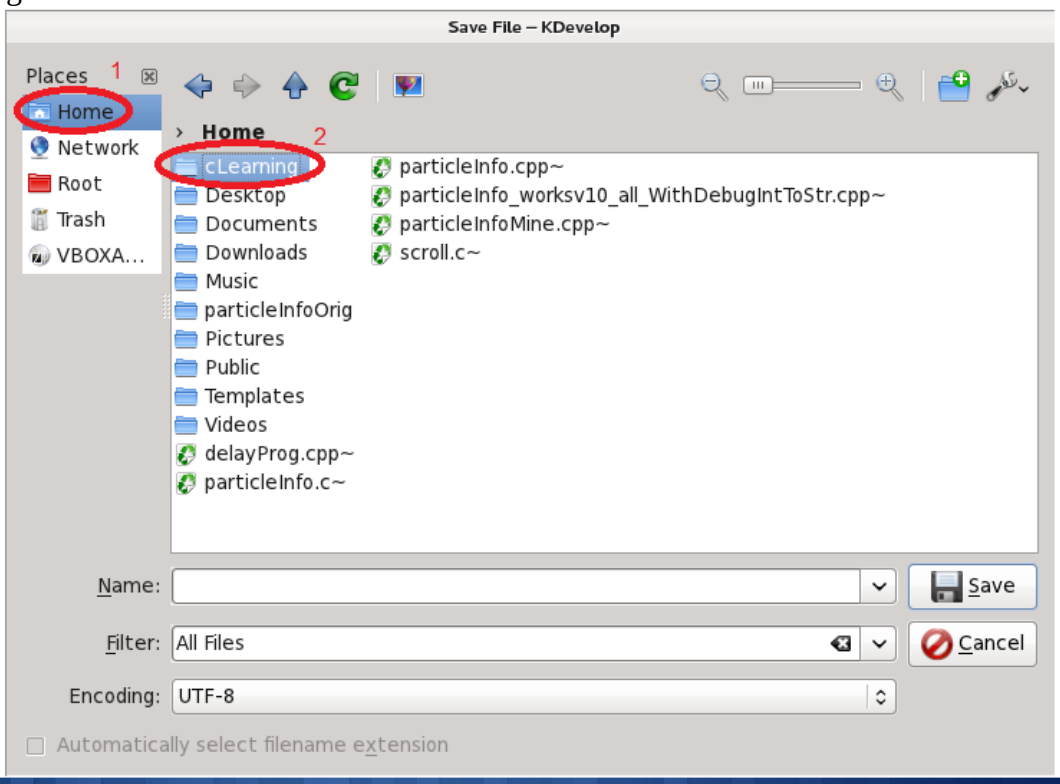
Now a window will come up. Resize it how you'd like. Click on the "Konsole" button at the bottom-center of the window. This is a little console (terminal) inside the IDE. Nice! I resized mine so I could have some more lines of terminal showing. You can change it by dragging the border between the Konsole and the other window up and down. If you're working on coding (writing the program), have a smaller Konsole. If you're running the program, have a bigger Konsole.

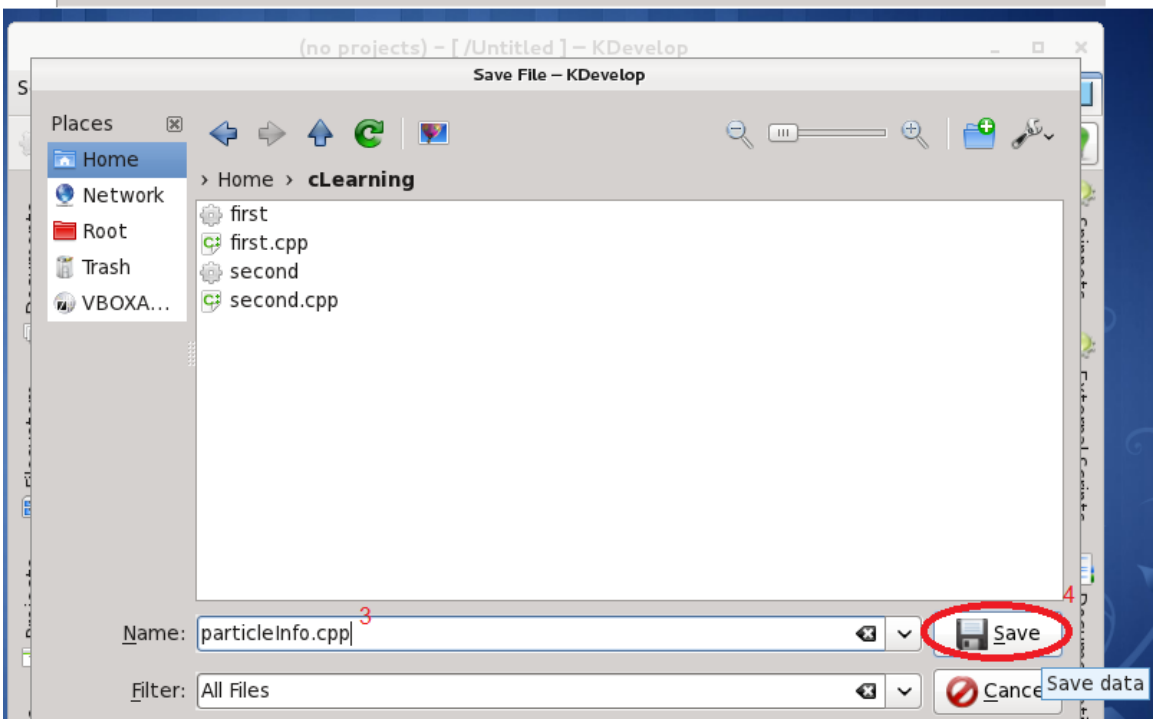


Go to "File" -> "Save As"

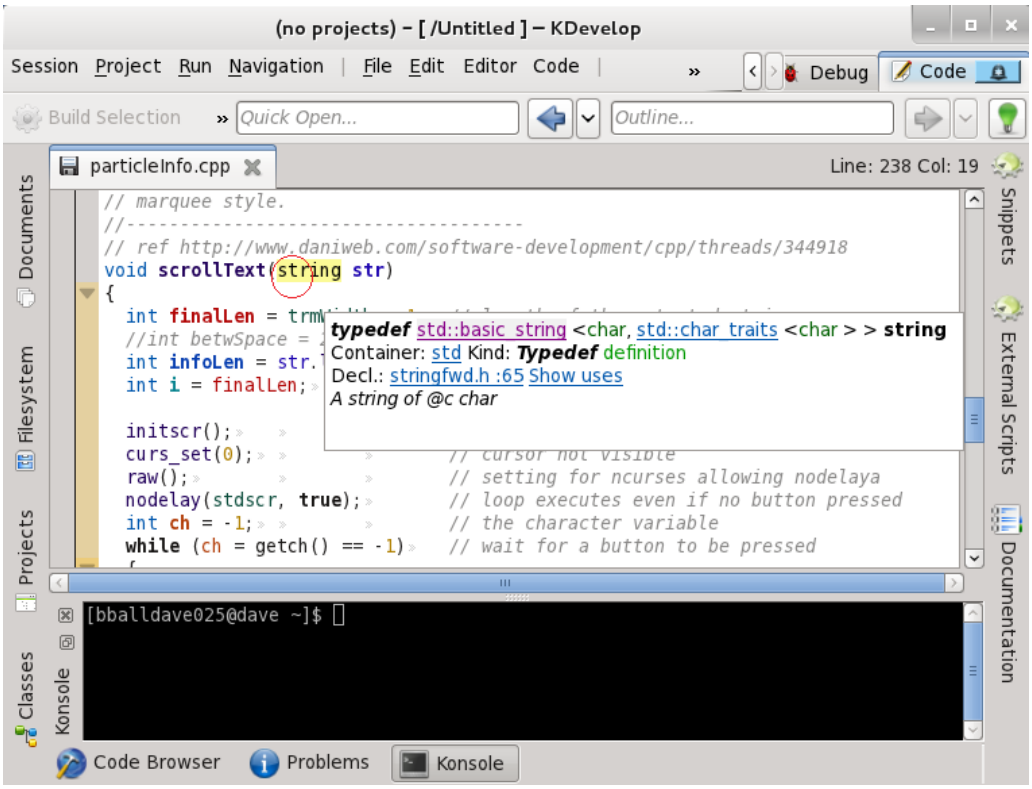


Save the file as “particleInfo.cpp” in your cLearning folder. You can get there by double-clicking on “Home” under the “Places” title bar on the left. Enter the name and click on “Save” at the lower-right.





Now click inside the coding window on your VM and paste in the code that you copied from this document, either by right-clicking and selecting “Paste” or by typing “ctrl+V” on your keyboard.

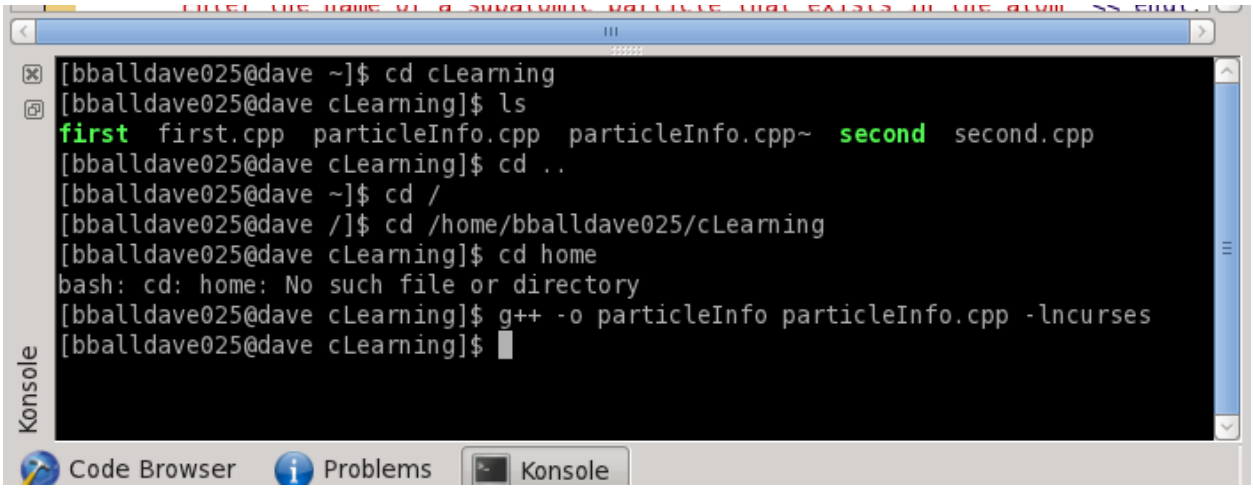


You should have a bunch of text in the coding part of the window. You can see how nice KDevelop is by seeing that when I hovered my mouse pointer (indicated by the red circle) over a word—in this case “`string`”—it gives a description of it. It will do things like give you valid words that you can finish, letting you know about some errors with the little red jagged “misspelling” lines, give you quick access to documentation, and all sorts of good stuff.

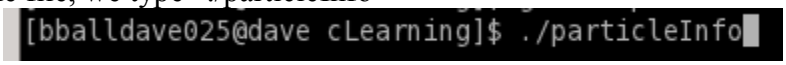
Anyway, you should save the file by going to “File” -> “Save” or pressing “ctrl+S” on your keyboard. You’ll know it’s saved because the picture of a floppy disk to the left of the “particleInfo.cpp” on the tab will go away (along with all the prepositional phrases in that last sentence). If there’s a picture of a disk there, it means you’ve changed the file since the last time you saved it. Now click on the Konsole part of the window (the black). You can use this to do anything you did on the terminal.

We can see this by changingthe directory to where we are. The thing that says “[yourusername@yourcomputername ~]$ means that we’re in the “Home” directory. We can type “cd cLearning” to get in the directory where we have the file. On the next screenshot, I’ve made the console bigger and done some stuff that’s useful to know with “cd”. “cd ..” means go up a directory. On the shot, we went from /home/bballdave025/cLearning to /home/bballdave025 (the second one being the home folder) . “cd /” takes us to the “/” directory, which is the base directory, kind of like “C:/” in Windows. I specified the full directory name in the last change directory command to take us back where the file is. I also tried to do “cd home”, but it turns out this doesn’t work, and the Konsole tells me so and ends up leaving me in the same directory. This will all probably be useful to you at some point.

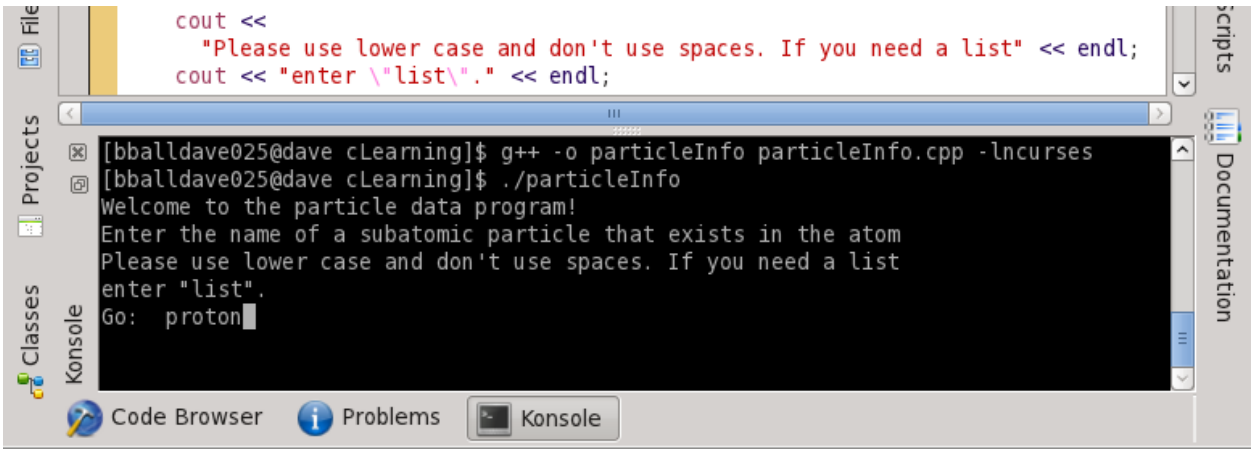
Now you're going to compile and run the program. Remember how I said we're going to link to an external library called "libncurses"? To do this, we use the same syntax as we did to run our previous programs, but we add a "-lncurses" to the end of the g++ command, like so: "g++ -o particleInfo particleInfo.cpp -lncurses"



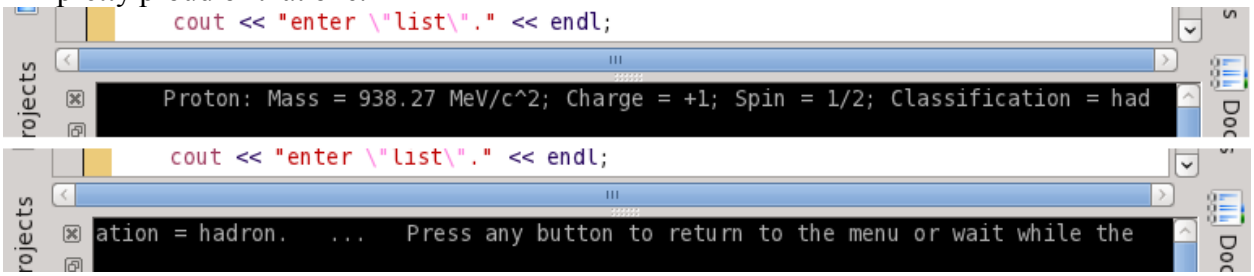
Now, to run the file, we type "./particleInfo"



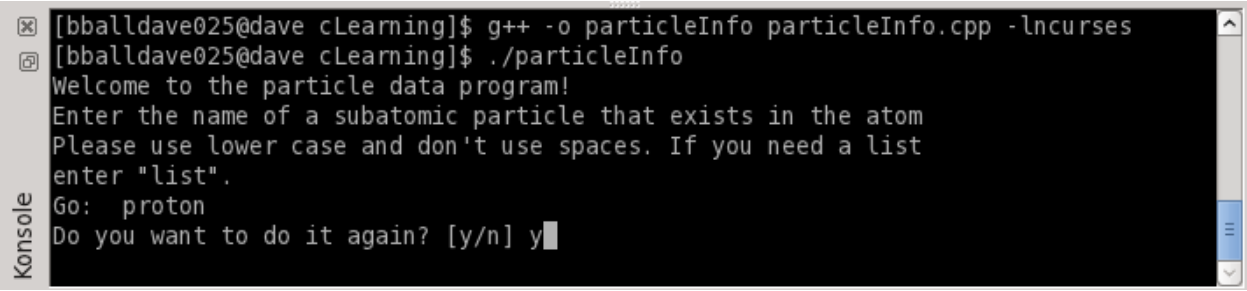
Go ahead and type "proton" and see what happens.



Hooray! We get some important characteristics of the proton, and it scrolls in a cool way. I'm pretty proud of that one.

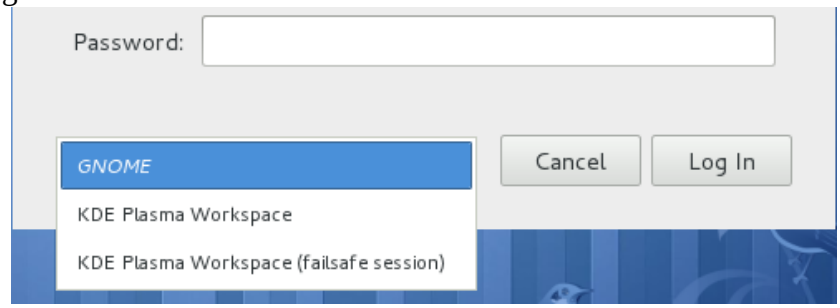


After you press any button, it asks you if you want to do it again. Go ahead and try some other things in the atom, like “upquark” or “electron”, or you can type “list” to get a list of particles you can type in. If you type something that’s not a subatomic particle that exists in the atom, the program will take care of that, too. Just so you know, there’s an Easter egg in this program. Good luck finding it. Okay, it’s really not that hard. All you have to do is look in the code.



When you’re done, press “n” when it asks if you want to do it again, and the program will be sad…I mean will exit. You can also run this program from the normal terminal. Go ahead and try it. (Just open the terminal, get to the right directory, and type “./particleInfo”.) There you go. You’re coding right along there. You probably don’t want to run this for too long, as the routine that handles how fast the message scrolls will take a lot of processing cycles and do everything that comes along with that, such as heating up your computer and making it slower. (I hope) it won’t hurt to run it for a while. I let mine run for quite a while, and it didn’t slow down, which I assume means I’ve kept memory leaks out. If you don’t know what that means, no worries.

Just a note after that KDevelop stuff: If you log out then press the red “X” in the corner and then “Power off the machine”. That will restart the computer. If you do that and start the VM again, or maybe if you just log out (I’m not sure, since I just noticed this,) you will see another option on the log-in screen.



This gives you the option of using a different Workspace, which is basically just a different desktop. However, there are possible problems with the graphics, similar to those that happened with GNOME 3. I would suggest just using GNOME, which basically means don’t change anything. I just thought you might like to know why your login screen is different.

The "ABC" Particle Physics Simulator goto Navigation

We have one more fun type of program before we get on to ROOT and PYTHIA—not that those aren't fun. This was made by a guy named David Ambrose while he was at the University of Texas (in 1997 for his PHY 381C term project—just in case you wanted all the details.) I just sent an email to his utexas address to contact him and ask him for permission to use it. I don't know if he uses that address anymore, (I doubt it,) but I hope he's okay with it. He did all the grunt work. I took his code and tweaked some things to make it runnable on a general Linux system with a general C-compiler, as well as to make the program meet the current C-standard. (I guess it's changed a little since 1997.) I did so, and the program works great. I just want to emphasize that it wasn't me, but David Ambrose who was the brain behind this sweet-action program. You can find the site for his term project here <http://wwwrel.ph.utexas.edu/Public/Students/ambrose/welcome.html>. You can also find his descriptions of it, which are enlightening. They are in PostScript format, so you'll need something like psfile and GhostView to see them.

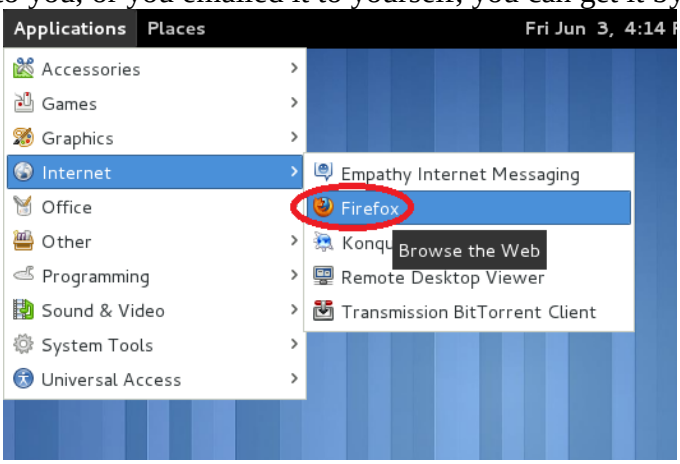
This is another section you can skip if you're in a hurray to get ROOT working at BNL. To get ROOT working, follow the ROOT link.

I've put PDF versions of the papers at my web site <http://bballdave025.kodingen.com/physicsCoding/abcSimulator/> as well as at my Scribd document page <http://www.scribd.com/david_black_23>. Note that the most recent version of this tutorial should also be at this page, along with some other cool stuff. ~~I plan on getting all of the files for the updated "ABC" Particle Physics Simulator there so people besides those to whom I email the tutorial with the files can use it.~~ I have finally uploaded the documents needed to run this simulator! They are at my web page <http://bballdave025.kodingen.com/physicsCoding/abcSimulator/>. For now, they're just that site with "abc-curr"+whatever_document appended at the end. The document name can be found at the list a page or two down. Hopefully the other Dave will get back to me so I know that it's okay to do this.

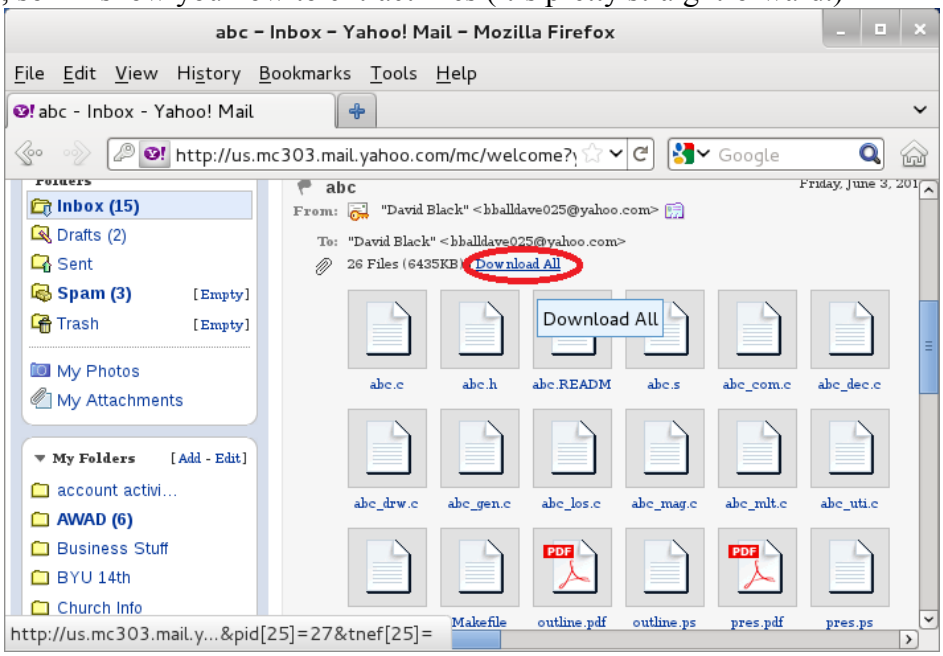
Note that you most likely will not be able to run the code from Dave Ambrose's website, since (at the time of my writing this tutorial) the code is 14 years old and the C-standard (the rules that C-compilers use) has changed. Also, his original program was written to work on the utexas network of computers, which probably had specific properties in how it looked for certain parts of the program. (Don't worry about details of that now—it's just linker stuff that you probably don't need to worry about now.)

Now you're ready to go. You'll need a bunch of files in an abc directory that you'll make. You can copy and paste them when I get them to my Scribd account, making sure to save them with the appropriate file extensions (.c, .h, .egg, .muffin, or whatever). If I email them to you (and you're welcome to email me asking for them. My email address is at the very bottom of the document,) you can download them in Linux. You can also download them in Windows and email them to yourself. There's also the option of Shared folders, which I might get to discussing in this tutorial.

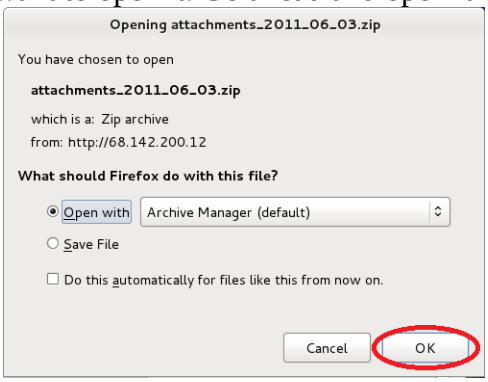
If I emailed it to you, or you emailed it to yourself, you can get it by going into Firefox.



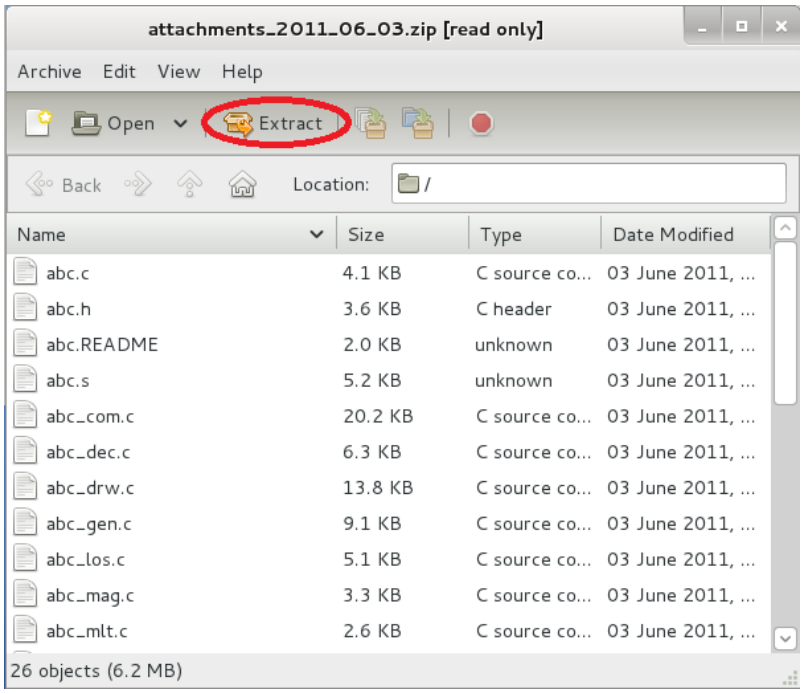
Download all the files from your email. When I downloaded it from Yahoo! mail, it came in a .zip file, so I'll show you how to extract files (it's pretty straightforward.)



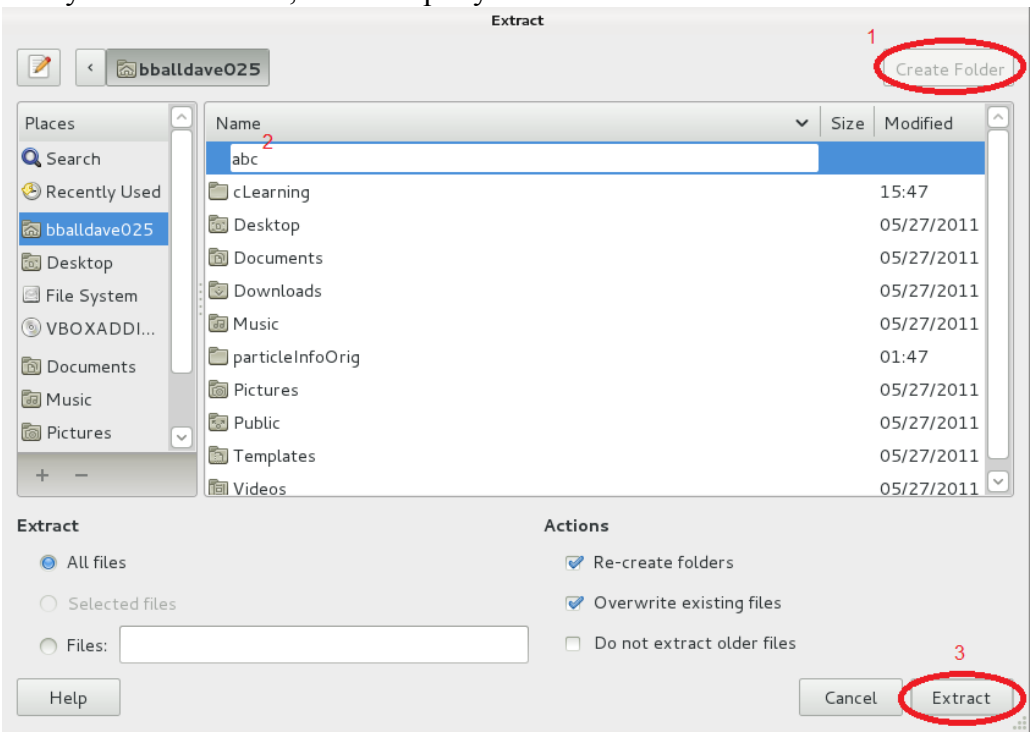
Push OK when it asks if you want to open it. Go ahead and open it with Archive Manager.



Click on “Extract”.



Go ahead and click on “Create Folder” in the upper-right corner and name your new folder “abc”. When you click “Enter”, it should put you in that folder. Click “Extract



The next page shows all the files that come with the program.

Files for The “ABC” Particle Accelerator (see the abc.README file)
Absolutely necessary:

abc.c - Main program
abc.h - Header file
abc.s - Script file (default)
abc_com.c - Command routine
abc_dec.c - Particle decay
abc_drw.c - Display routine (drawing package)
abc_gen.c - Generator routine
abc_los.c - Energy loss
abc_mag.c - Magnetic field routine
abc_mlt.c - Multiple scattering
abc_uti.c - Utility routines
example.s - Script file for default
Makefile - the Makefile; NO FILE EXTENSION

All files:

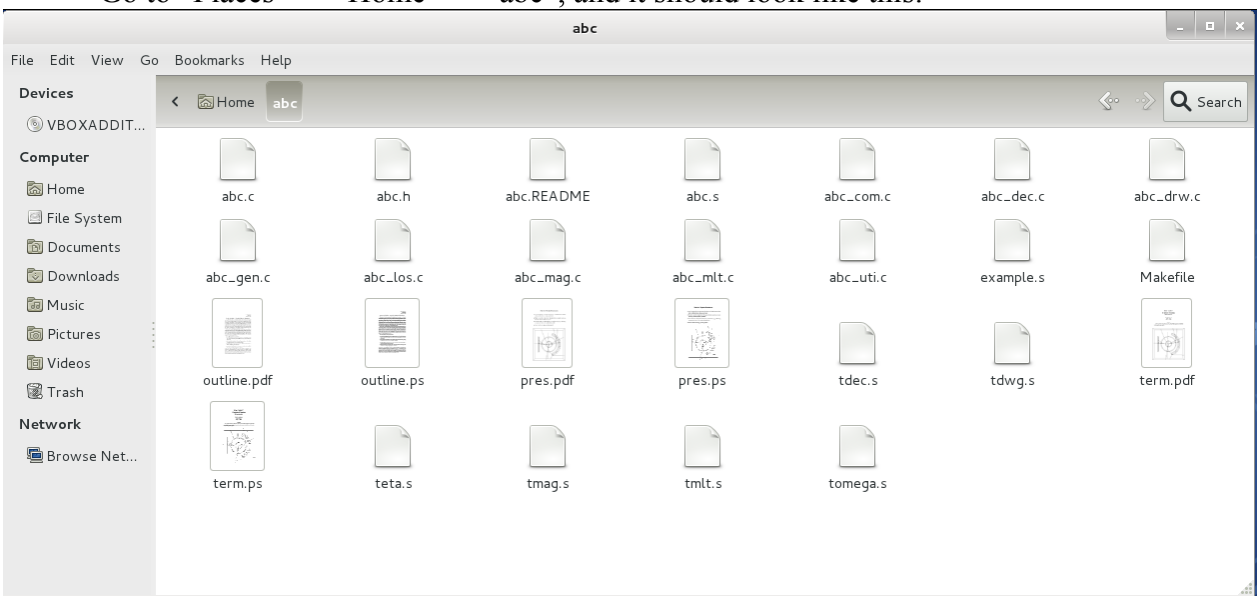
abc.c
abc.h
abc.README [not technically necessary, and the installation instruction aren't exactly right for our purposes, but still useful and informative]
abc.s
abc_com.c
abc_dec.c
abc_drw.c
abc_gen.c
abc_los.c
abc_mag.c
abc_mult.c
abc_uti.c
example.s
Makefile [Note there is NO FILE EXTENSION. This is correct.]
tdec.s
tdwg.s
teta.s
tmag.s
tmlt.s
tomega.s

And I'm going to include the informative files, because I think that the abc directory is a good place to put therm. They aren't necessary for the program.

outline.pdf
outline.ps
pres.pdf
pres.ps
term.pdf
term.ps

I added an abcOrigScript.s, which you can use to restore abc.s to its original working) state.

Go to “Places” -> “Home” -> “abc”, and it should look like this:



You’ll need to install the OpenGL, which is an **O**pen **G**raphics **L**ibrary that will allow us to make some 3D plots and do all types of cool stuff. I’m going to show you a different way to install, using “sudo”. Note that you can do all of this installation by logging in as root (root user, not ROOT the program). Type “sudo yum install freeglut-devel”. Read the cool Spiderman quote and enter you *username* password. This works because you set yourself as an administrator.

```
[bballdave025@dave ~]$ sudo yum install freeglut-devel

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for bballdave025:
```

There’s still one little bug we need to fix before it can work. If you try to type “make” now (I’ll describe what “make” is later), it will give you an error like the one in the next screen shot. Note that I wouldn’t type “make” right now, I would wait until you’ve installed the things and made the links I’m going to show you. I just want to show you some of the errors you might run into and how to solve them.

Basically what happens with this error is that the linker looks for something called "libXmu.so", but on the system, there is only a libXmu.so.1 . For most .so files (don't worry about exactly what they are--it's just a certain type of library,) this is taken care of. The solution in our case is to make what's called a soft link. (Make sure to do this next part.) /****Do as I didn't and go to a better directory with "cd /usr/lib/". Type the following two commands into the terminal: "ln -s /usr/lib/libGL.so.1 libGl.so" and "ln -s /usr/lib/libXmu.so.6 libXmu.so". The "-s" option makes it a soft link and "ln" is for **lin**k.

```
[bballdave025@dave ~]$ cd abc
[bballdave025@dave abc]$ make
gcc -o abc -L/usr/lib -lglut \
 -lGLU -lGL -lXmu -lXext -lX11 -lm  *.c
/usr/bin/ld: cannot find -lXmu
collect2: ld returned 1 exit status
make: [abc] Error 1 (ignored)
[bballdave025@dave abc]$ ln -s /usr/lib/libGL.so.1 libGL.so
[bballdave025@dave abc]$ ln -s /usr/lib/libXmu.so.6 libXmu.so
[bballdave025@dave abc]$
```

PS, don't need to libGL.so anymore. They fixed that.

Typing "make" again still gives us an error. The reason is that we are programming (developing,) so we need the development tools that come with libXmu (NB libXmu is a graphics library that lets you make a window.) To solve this, I'm going to show you how to use yum search. (Make sure to do this part.) Log in as root and type "yum search libxmu". It comes up with the name of the package we need. Go ahead and type "yum install libXmu-devel". Type "y" when it asks you to.

```
[bballdave025@dave abc]$ make
gcc -o abc -L/usr/lib -lglut \
 -lGLU -lGL -lXmu -lXext -lX11 -lm  *.c
/usr/bin/ld: cannot find -lXmu
collect2: ld returned 1 exit status
make: [abc] Error 1 (ignored)
[bballdave025@dave abc]$ su
Password:
[root@dave abc]# yum search libxmu
Loaded plugins: langpacks, presto, refresh-packagekit
============================ N/S Matched: libxmu ============================
libXmu.i686 : X.Org X11 libXmu/libXmuu runtime libraries
libXmu-devel.i686 : X.Org X11 libXmu development package

  Name and summary matches only, use "search all" for everything.
[root@dave abc]# yum install libXmu-devel
```

Log out from root by typing “exit”. Now you’re ready to run The “ABC” Particle Physics Simulator. Go to the terminal, make sure you’re in the right directory, and then type in “make”. This will call the Makefile. Don’t worry about the details of that right now—basically a Makefile makes your life as a programmer easier, saves you a lot of time compiling when you edit stuff, and makes the code very portable. Now there are no errors! Run the program by typing “LIGBL_ALWAYS_INDERECT=1 ./abc”. I’m honestly not sure why it only works with that first statement. It has something to do with the way the VM looks for the graphics folders. Anyway, that’s the only way I can get the thing to work. If you find a solution let me know.

```
[bballdave025@dave abc]$ make
gcc -o abc -L/usr/lib -lglut \
 -lGLU -lGL -lXmu -lXext -lX11 -lm  *.c
[bballdave025@dave abc]$ ./abc
```

<NB edit> LIBGL_ALWAYS_REDIRECT=1

If you’re comparing this with his README, you’ll want to note that the external libraries must be linked to with the following statement:

“-o abc -L/usr/lib -lglut -lGLU -lGL -lXmu -lXext -lX11 -lm”

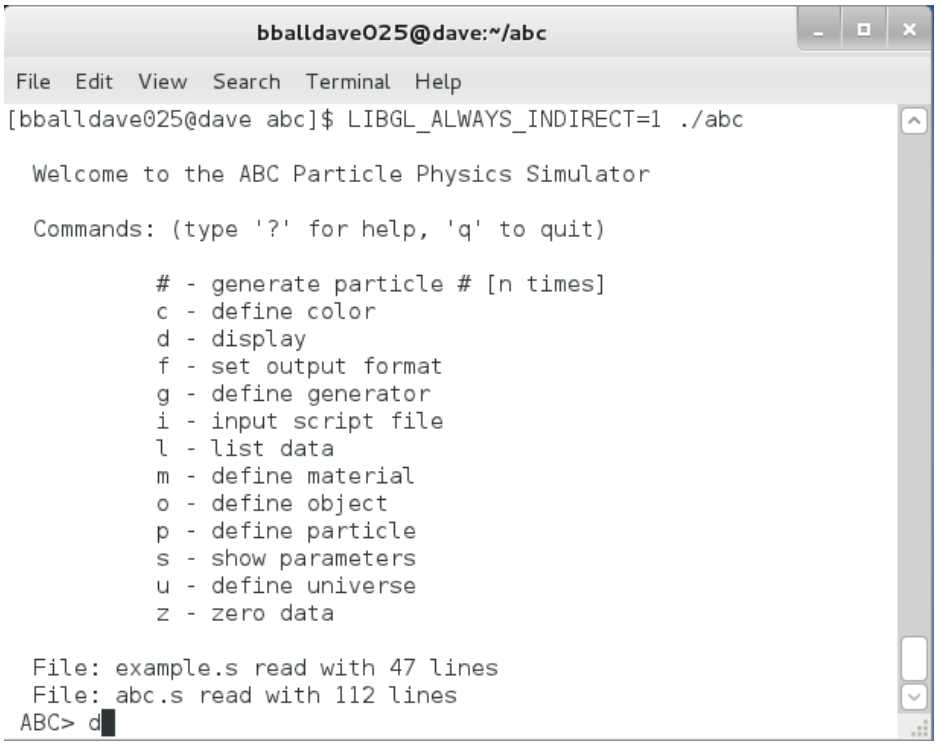
The differences come because he was using the utexas server (he calls it the einstein server) which has different linkage, and because his server has some connection to a win32 Windows OS whereas our OS is completely Linux.

In his Makefile, Dave Amrose also notes:

> Run the program by typing "abc". By default, the script file abc.s will load the example script (example.s). This call should be commented out (using a "#" at the last line of abc.s) if other scripts are desired. Unfortunately, I have only tested the program on einstein.ph.utexas.edu. Also, when running ABC on the einstein terminal, the mouse function (for zooming in views) does not work correctly, but views can be adjusted by scaling and moving.

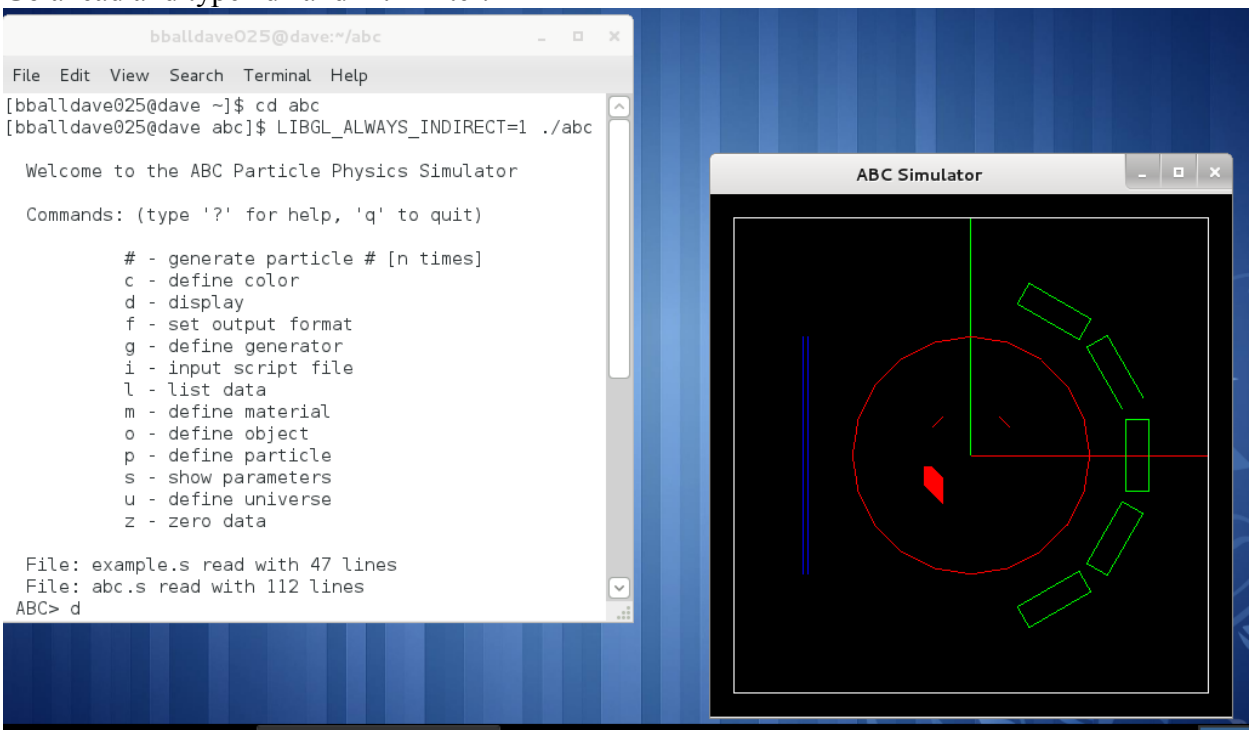
This much is also true in our case, except you type “./abc” instead of “abc”. I’ll go over those details later

After you have made the executable (typed “make” in the terminal) and run it, you will get something like this. You’ll soon find out this is 3D. Pretty awesome!

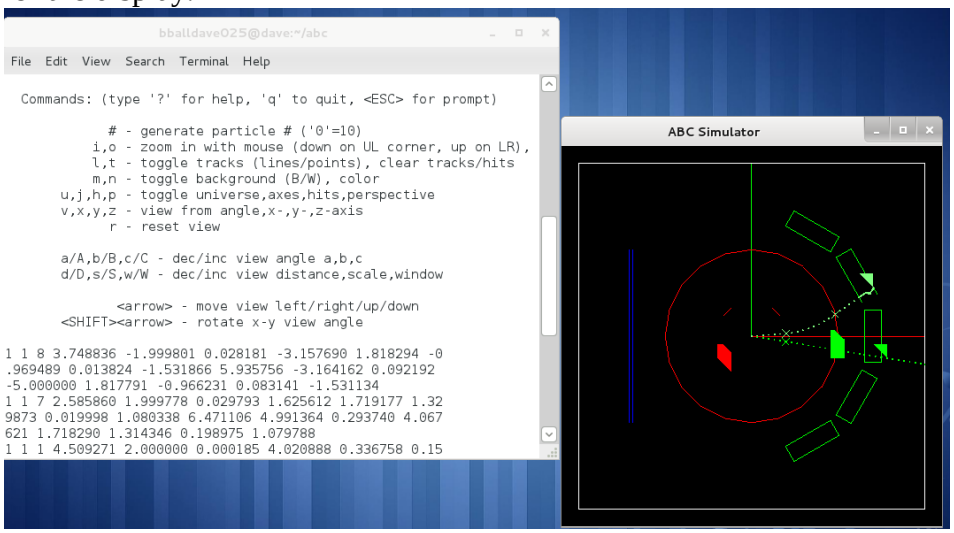


The greater-than sign is there to show that what you type will be directed to ABC, but you can just think of it as your new command prompt. ROOT and PYTHIA have similar things. Let's play with this a little bit, and then we'll get on to the other two programs.

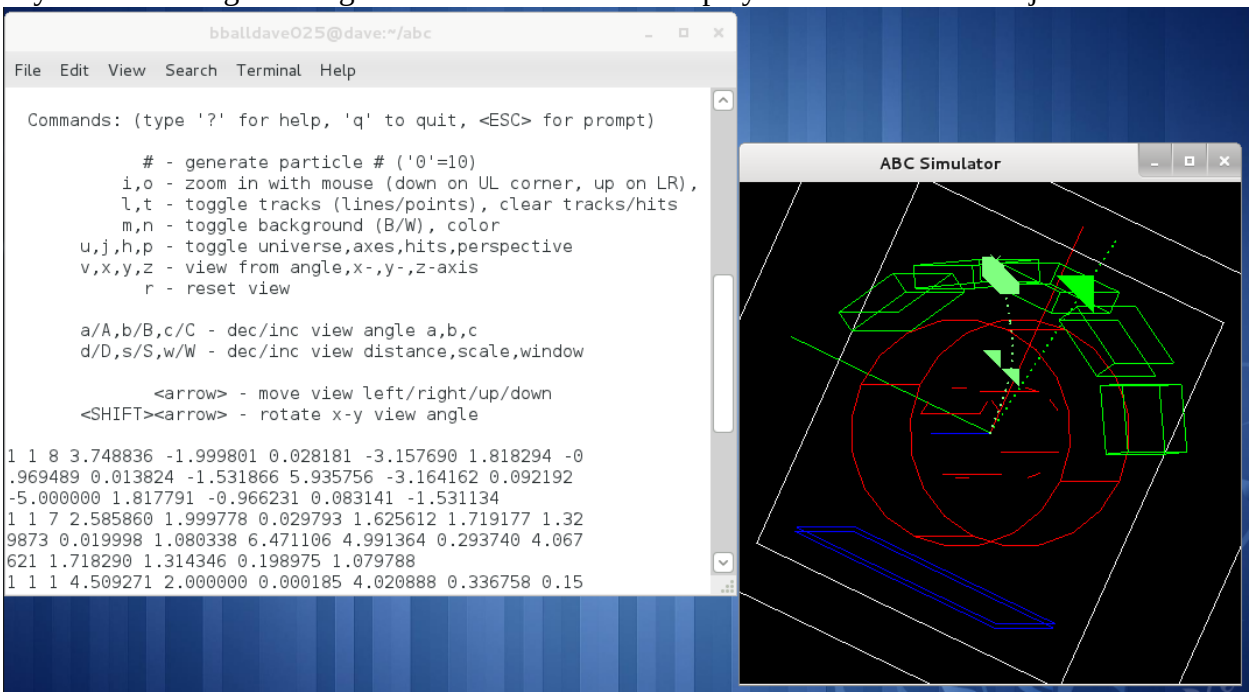
Go ahead and type "d" and hit "Enter."



There's your first view of the simulator. When I installed this on Fedora 14 on my old VirtualBox program, I didn't get those weird extra/incomplete shapes. I'll try to figure out how to fix that. This is supposed to be a hollow cylindrical detector with rectangular detectors (blue and green) around it and a **B** field coming out of the screen. The straight red and green lines (there'll be a blue one when you rotate) are the axes. While the display window is highlighted, type "9". This will give you a π^- decay (see the documentation or read the "example.s" file to see what particles there are). Note that you'll need to scroll down on the terminal window to see the options for the display.



Your will probably look different than mine, since it's a Monte Carlo simulator, there's an element of chance, just like reality. Hold down "shift" and press the arrow keys on your keyboard to change the angle of the view in the 3D display. Check out all its majesties.



You might have figured out that the two dashed tracks are the μ^- and the $\bar{\nu}_\mu$ that come from a positive pion decay, with the straight track being that of the muon anti-neutrino. Also, you'll note that, in the terminal window, there's some data. This can be sent to other plotting programs (like ROOT.) Here's a table for the other particles (and numbers you need to enter for them) that you can use in this program "right out of the box".

5 K^- decay 2
6 K^+ decay 3
7 K^L decay 6
8 μ^- decay
9 π^- decay
0 moving p^+

By the way, you can get rid of some of those extra shapes by pushing the "h" button while the display window is highlighted. This toggles hits. If you want to go back to the terminal and try something else, press "esc". You can also type "?" for help and "q" to quit. Those commands can also be typed from the prompt if you want to get the same options there.

I won't go into too much detail about how all of this works. ***(6/3/2011) Eventually, I plan to come back and expand this tutorial to show the options for making it do an Ω decay and some other cool stuff. You're welcome to look through his documentation, from which you might be able to figure it out. For now, though, it's on to ROOT.***

ROOT

[Description of ROOT and disambiguation between the program and the user]

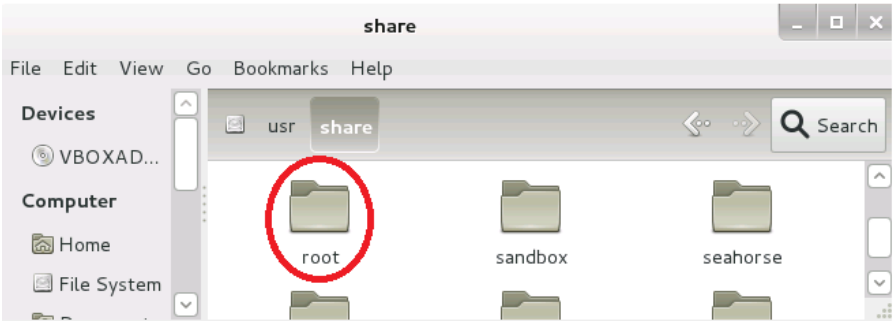
Since I first made this tutorial, it has come to my attention that ROOT can be installed from the yum package manager. This takes a LOT less time and sets all the environment variables for you. I'm leaving in the tutorial for how to do it the old way, in case you ever need to learn how to install a program that isn't in the yum repository. I'm also not sure if there is a similar installation for package managers in others distros of Linux (like apt-get for Ubuntu), so I'm leaving it in just in case users of these other Linux flavors want to install it. This is found below in dark blue (like this) text. I would just skip it for now and follow the instructions below.

Anyway, all you need to do is type "yum install root". (Remember to log in as the root user before doing this by typing "su" and entering your password.) After you press enter, it should only take 5-7 minutes to install (compared to the other way, which took 45 minutes - 1 hour.) Once again, press "y" when it asks you to do so.

```
[bballdave025@daveFedora ~]$ su
Password:
[root@daveFedora bballdave025]# yum install root
```

When it's done, skip all of the blue stuff and go run ROOT. Note that your "root" folder (where you might need to go later to find files you'll have saved) is now located at /usr/share/root

```
[bballdave025@daveFedora root]$ ls
class.rules       icons   plugins     system.rootauthrc    system.rootrc
gdb-backtrace.sh  macros  root.mimes  system.rootdaemonrc
[bballdave025@daveFedora root]$
```



/**********Beginning of more complicated way**********/

We'll start by downloading some of the packages we need. We'll start with the absolutely necessary ones. Some we have already installed. If you want to see the complete list (for example if you're just following this part of the tutorial,) go to the ROOT Build Prerequisites page <http://root.cern.ch/drupal/content/build-prerequisites>. Log in as root and type "yum install subversion libXpm-devel libXft-devel". Remember that you can just copy and paste if you want to. Note that to paste in the terminal, you have to right click and select "Paste".

```
[bballdave025@dave ~]$ su
Password:
[root@dave bballdave025]# yum install subversion libXpm-devel libXft-devel
```

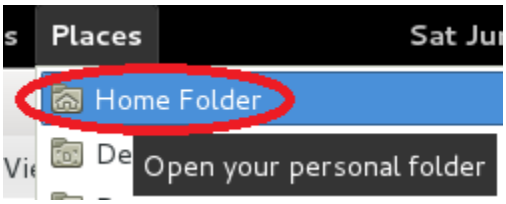
Now I'll have you install the recommended files, as described on the prerequisites page. Type "yum install gcc-gfortran pcre-devel glew-devel ftgl-devel mysql-devel fftw-devel cfitsio-devel graphviz-devel avahi-compat-libdns_sd-devel python-devel libxml2-devel gsl gsl-devel root-mathmore". You'll most likely want to use copy-paste for this one. Remember to right-click and select "Paste" in the terminal. Note that the site suggests "libldap-dev" and "gsl-static", but a yum search didn't turn up anything for those packages. I did add on "gsl", "gsl-devel", and "root-mathmore", as these came up when I did a yum search for "gsl". This could take a little while (~3-5 minutes.)

```
[root@dave bballdave025]# yum install gcc-gfortran pcre-devel glew-devel ftgl-
devel mysql-devel fftw-devel cfitsio-devel graphviz-devel avahi-compat-libdns_
sd-devel python-devel libxml2-devel gsl gsl-devel root-mathmore
```

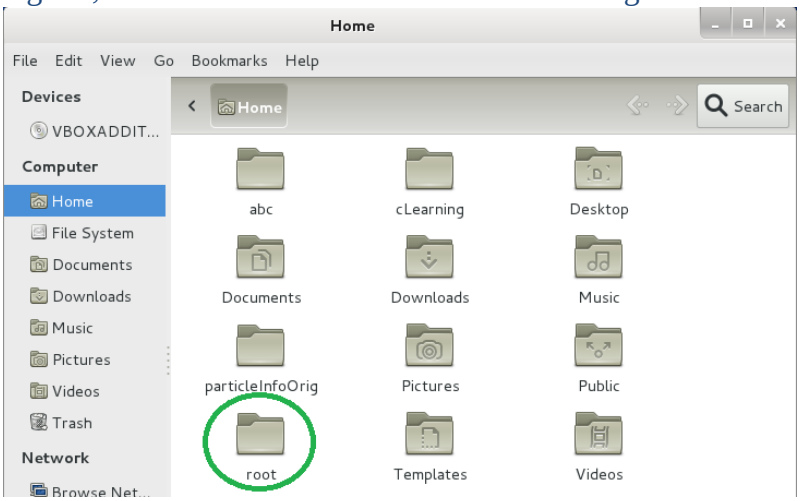
Now we'll get all the files. It's possible to go to the ROOT site (which resides on the CERN site) and download them, but there's an easier way. Make sure you're in your Home folder (there should be a tilde, "~", just before the bracket and dollar sign at your terminal prompt). Just type "svn co http://root.cern.ch/svn/root/trunk root". This could take a while--I'd say 3-5 minutes.

```
[bballdave025@dave ~]$ svn co http://root.cern.ch/svn/root/trunk root
```

When you go to "Places" -> "Home Folder" …



…you should see a "root" folder. This folder, even though it's in lower-case, is for the ROOT particle physics program, not for the root user who can make changes on the Linux system.



Sweet! Now we're ready to make it runnable. Type "export ROOTSYS=/home/<username>/root" Notice that there's only one "root" in that. This basically changes an environment variable to give the directory of where you want ROOT installed. Don't worry if you don't know what that means; just type it.

```
[bballdave025@dave root]$ export ROOTSYS=/home/bballdave025/root
```

Now we'll install it with make. Type "cd root", then "./configure --help", then "./configure $ARCH", then "gmake", and finally "gmake install". This last step will take a while, and not just the "little while" we've had before. We're talking 30 minutes to an hour. Here again we can see why it's nice to have a virtual machine. You can do other stuff on your computer, like emailing me and telling me about the errors you found while trying to do the other things in this tutorial (bballdave025[at]yahoo[dot]com), while it installs without slowing down the installation,. Note that I've put a blue background on the screenshot. That's because the terminal outputs stuff between the commands.

```
[bballdave025@dave ~]$ cd root
[bballdave025@dave root]$ ./configure --help
[bballdave025@dave root]$ ./configure $ARCH
[bballdave025@dave root]$ gmake
[bballdave025@dave root]$ gmake install
```

Now you wait for quite a while until the terminal prompt returns

```
cp /home/bballdave025/root/cint/reflex/python/genreflex/gccdemangler.py lib/py
thon/genreflex/gccdemangler.py
[bballdave025@dave root]$
```

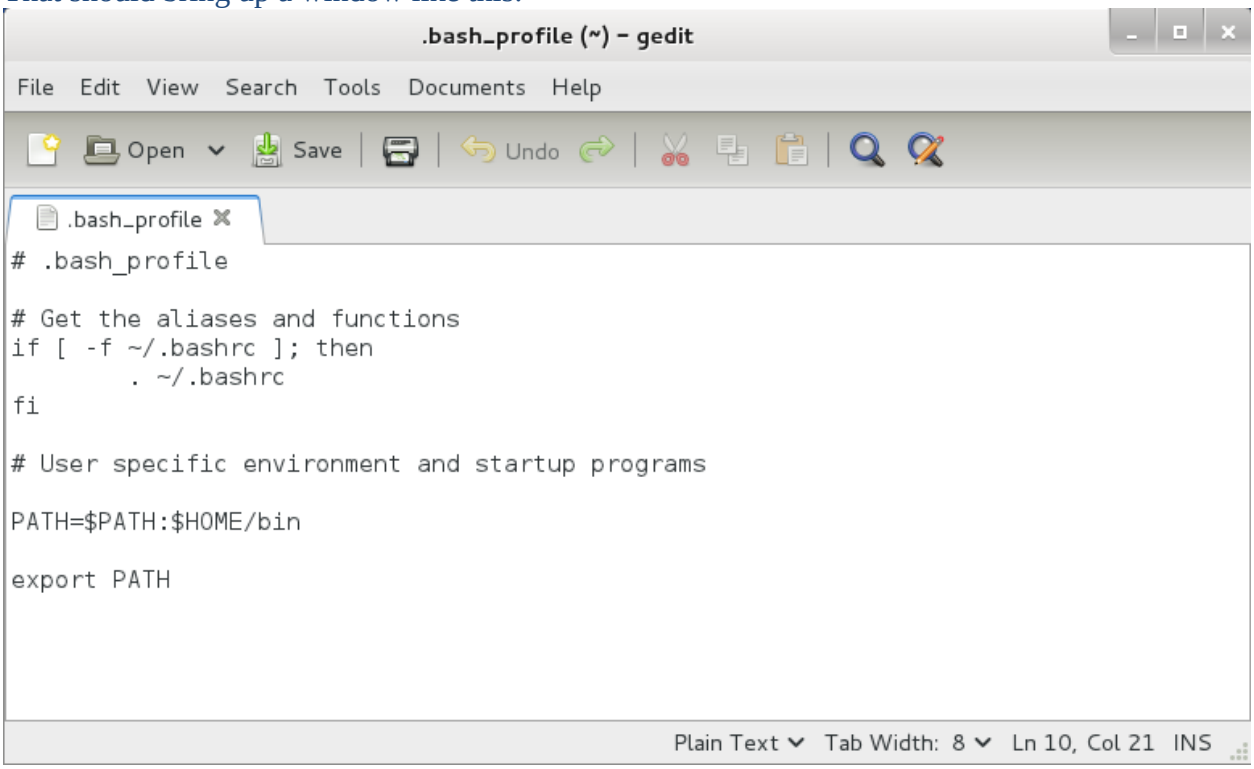
When you see that screen, (or something similar--something with the terminal prompt at the end), be happy. Your long wait for the installation is over. You'll need to do some stuff to make your computer look in the right place for the ROOT files. I'm going to explain this in technical jargon for those who wonder why I'm doing what I'm doing, but put it inside brackets. Feel free to skip to the end of the brackets.

[Don't do anything that I describe in these brackets. It wouldn't hurt anything, but it would be useless. The method to change the path that's shown on the site works, but you would have to edit the environment variables each time you started a new Linux session. They suggest either: (1) typing "export PATH=$ROOTSYS/bin:$PATH" and then "export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH" or (2) having a program called "`thisroot.sh`" do the same thing when you type ". bin/thisroot.sh". The only problem with this method is that it will only modify the path for the current session. Once you log out, any changes to the path will be discarded, so you'd have to do those steps over again every time you log out and then log back in. A better solution is to permanently change the environment variables, which is what I will show you how to do next.]

Go to your Home directory by typing “cd home/<username>” and then type “gedit .bash_profile”. The period (.) before the word “bash” is very important (basically, it signifies that this is a hidden file.)

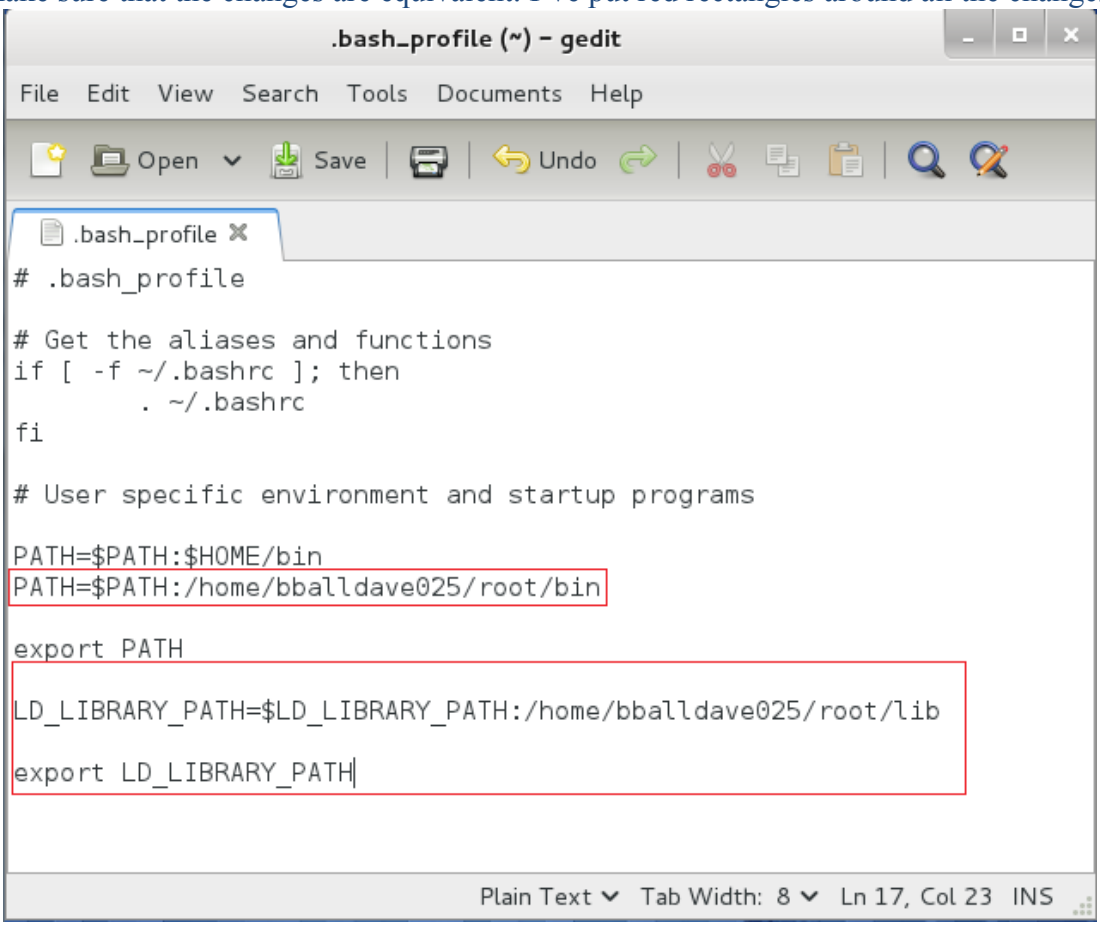
```
[bballdave025@dave README]$ cd /home/bballdave025
[bballdave025@dave ~]$ gedit .bash_profile
```

That should bring up a window like this:



You’ll edit this file in the following ways. Make sure you look at my screen shot before saving this file. First, add a line after the one that says “PATH=$PATH:$HOME/bin”. Add the following text “PATH=$PATH:/home/<username>/root/bin”. Next, after the “export PATH” line, add a blank line and then a line with the text “LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/<username>/root/lib”. Add another blank line and then a line with the text “export LD_LIBRARY_PATH”.

Make sure that the additions to the file look like what I have below. If you mess up the file you're editing, it could mess up your system. If your original file looked different than mine, just make sure that the changes are equivalent. I've put red rectangles around all the changes.

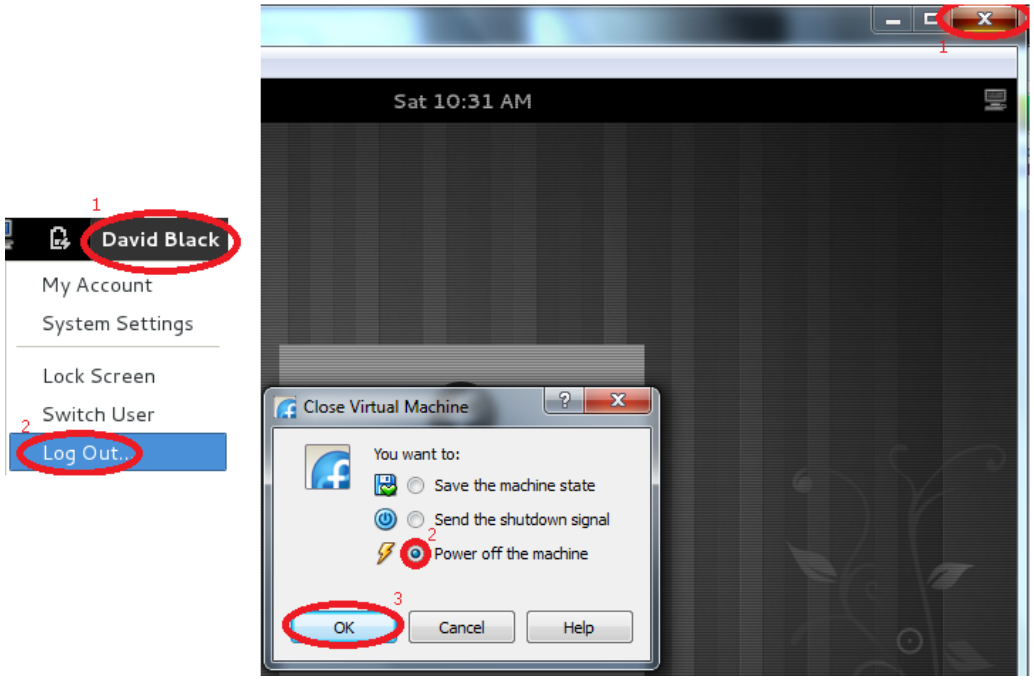


/**********Note that if you're done "yum install root", but still want to change the library path, you do not need to change PATH ($PATH), and you'll need to type:

```
LD_LIBRARY_PATH=/usr/lib/root:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

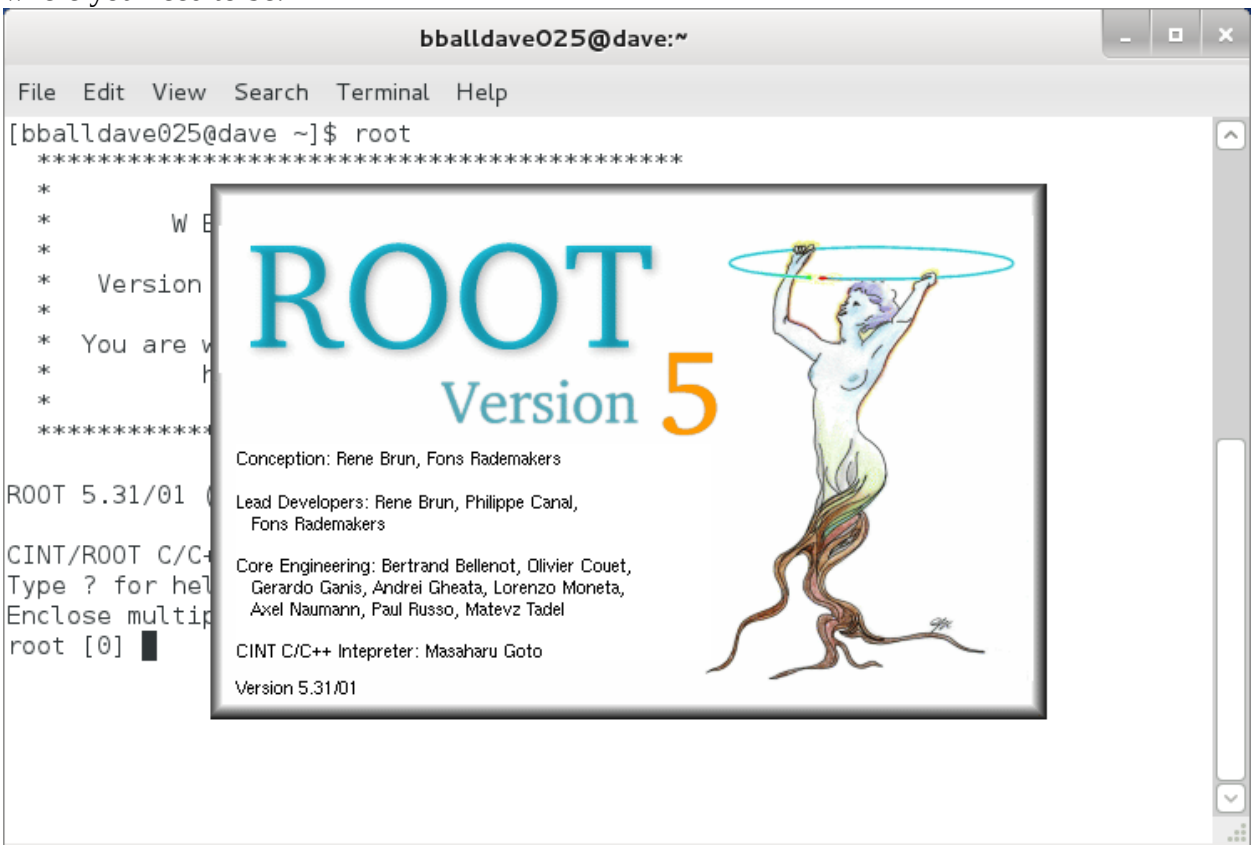
instead of what's written there.**********/

Save the file and exit out of gedit by clicking on the “X” in the upper-right corner. This will return you to the terminal. Whew! We got out of that scary change-the-system stuff all right. Exit out of the terminal and then log out by clicking your name in the upper-right corner, then “Log out…”. Click on the red “X” in the upper-right corner of the VirtualBox window and select “Power off the machine”. You *must* perform this step for the changes you made to take any effect and thus for ROOT to be runnable.



/**********End of more complicated way**********/

Congratulations, your ROOT is currently runnable. From the VirtualBox window, click "Start" and wait for the VM to boot up (if it's not already running.) When your Linux desktop comes up, open the terminal. Type "root" (no "./" needed). If you see the top less tree, you're where you need to be.



Note that you weren't in the folder where we installed ROOT. You can actually type "root" from any directory and get to ROOT. The "root" you're typing is, once again, not the root user, but ROOT the program.

Now you'll have a screen with the promised new command prompt--the "root [0]", which is like the "ABC>" prompt from before. Before we get going, I'll tell you that you type ".q" to quit ROOT. Don't do that yet, though. Let's start with something simple. Type "1+1" and press "Enter".

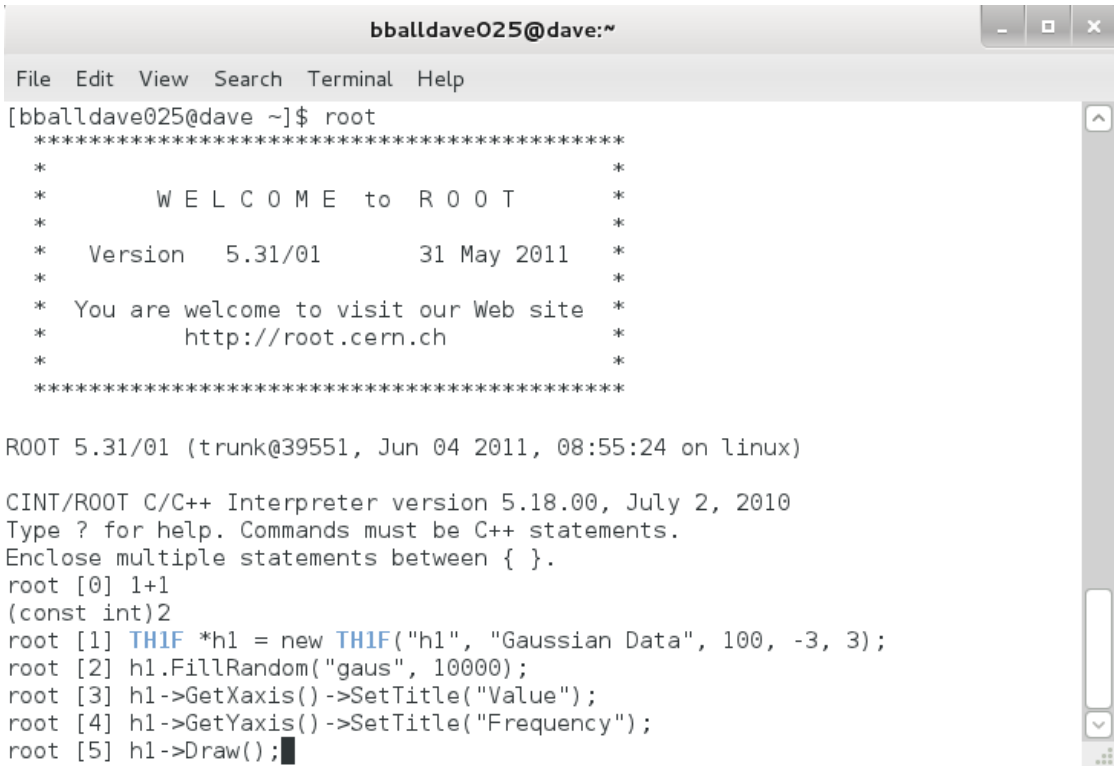
```
ROOT 5.31/01 (trunk@39551, Jun 04 2011, 08:55:24 on linux)

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] 1+1
(const int)2
root [1]
```

Yay! We got the right answer: 2. ROOT also tells us that 2 is a constant integer. You can use ROOT as a calculator like this all day, but I'm assuming that's not why you installed it. Let's make a couple of simple histograms. I'm not going to go over too many details, but I'll put in

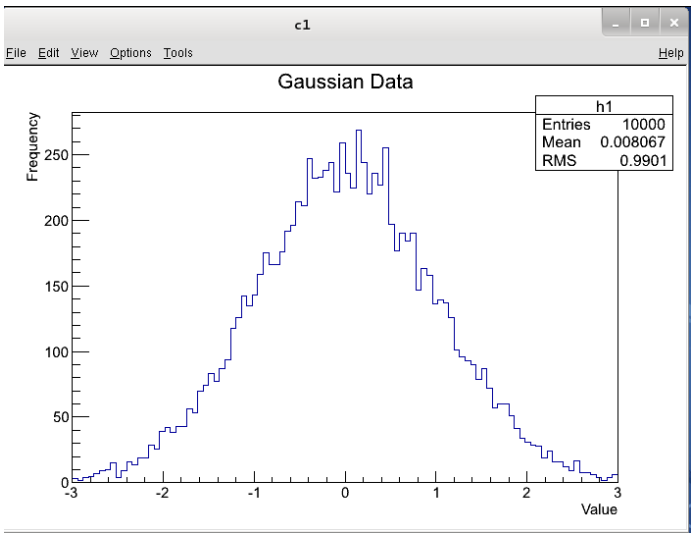
some explanations. My main source was the ROOT website's page about histograms <ftp://root.cern.ch/root/doc/3Histograms.pdf>.

At the prompt type the following lines, being certain to include the semicolons so as not to get bunches of data on the screen (refer to the screenshot):

```
TH1F *h1 = new TH1F("h1", "Gaussian Data", 100, -3, 3);
h1.FillRandom("gaus", 10000);
h1->GetXaxis()->SetTitle("Value");
h1->GetYaxis()->SetTitle("Frequency");
h1->Draw();
```



When you hit enter after that last line, the result should be something like this:

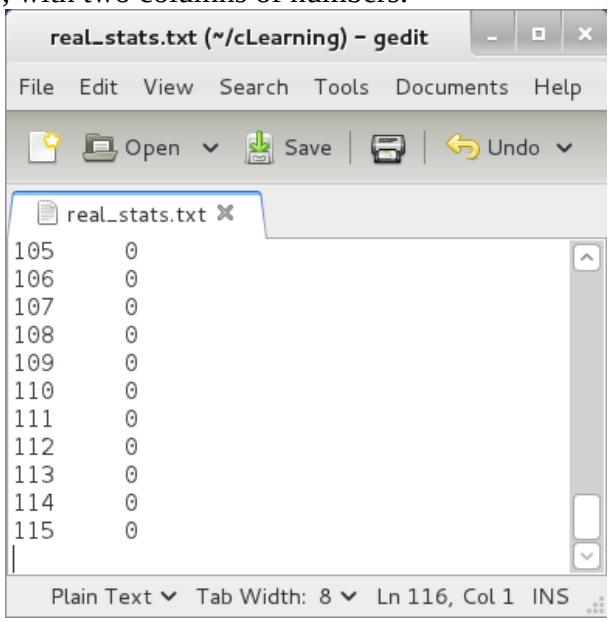


Looks pretty Gaussian to me. Yours will most likely be a little different (it's a random Gaussian generator, after all), but should still follow a Gaussian pattern. Good work! You made a ROOT plot.

Now for a little description A "TH1F" object is a 1-dimensional histogram. It takes as arguments the name of the histogram, the title of the histogram, the number of bins (basically divisions on the x-axis), the x-minimum, and the x-maximum with the following form TH1F("nameOfHistogram", "titleOfHistogram", numberOfBins, xMinimum, xMaximum) Our histogram was named "h1", titled "Gaussian Data", and had 100 bins, and x-min of -3, and an x-max of 3. We filled it with Gaussian data (root[3]), and then labeled the axes (root[4] and root[5]). After that, we made it display by typing "h1->Draw".

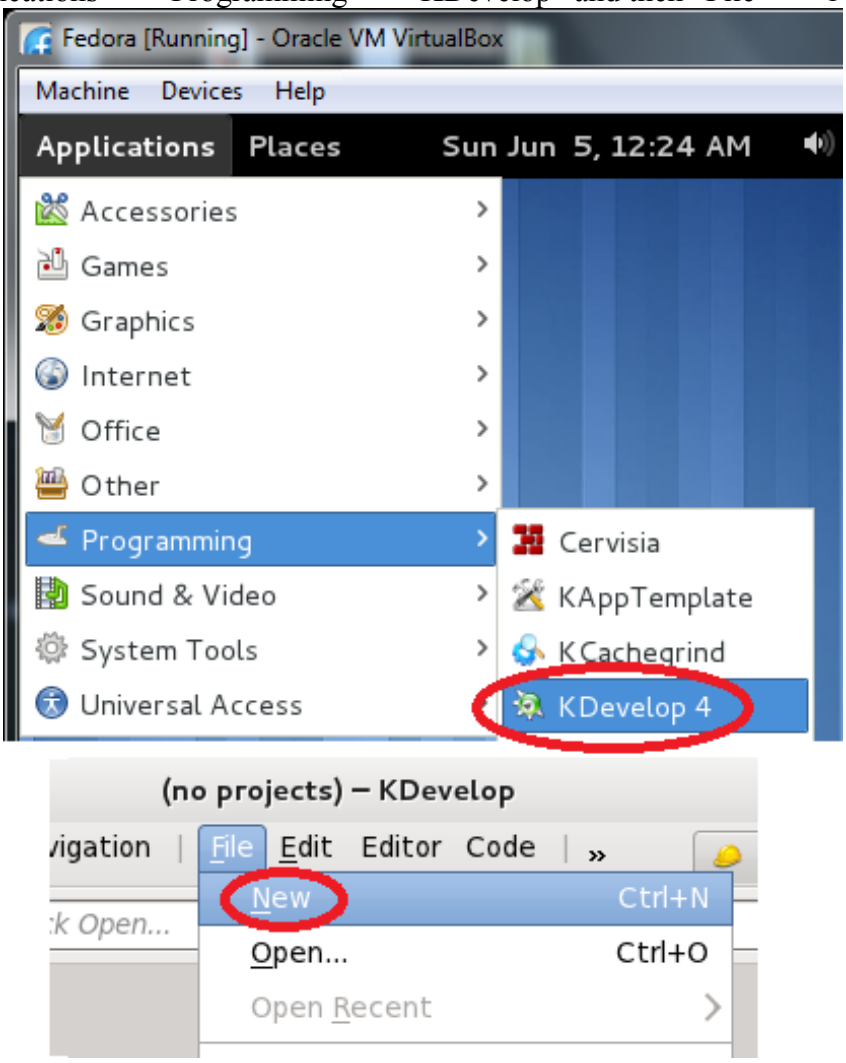
Go ahead and type ".q" to quit ROOT. Remember the period. For a lot of commands, ROOT wants this dot in front of stuff.

Now I'm going to show you how to import data from a file and plot it. In Appendix B, there is some data that I've researched. I'm not going to tell you what it is yet. Highlight all this text (just the numbers) and copy it ("ctrl+C" or right-click -> "Copy"). From your terminal, type "gedit". Paste the data ("ctrl+V or right-click -> "Paste") into the gedit window and save the file as "real_stats.txt". I saved it in my cLearning folder, since that seems like a nice place that won't get in the way of other ROOT stuff. For the program to work, you'll need to save both this text file and the root macro in some location, and I strongly suggest the cLearning directory. **Note that if you're copying from the PDF file into vim, the format will be all wrong.** You'll need to paste what's in the PDF file into Word or some other word processor, then save the file as a text file, then copy and paste the text into gedit or vim or whatever. Just make sure that the text file has the format below, with two columns of numbers.

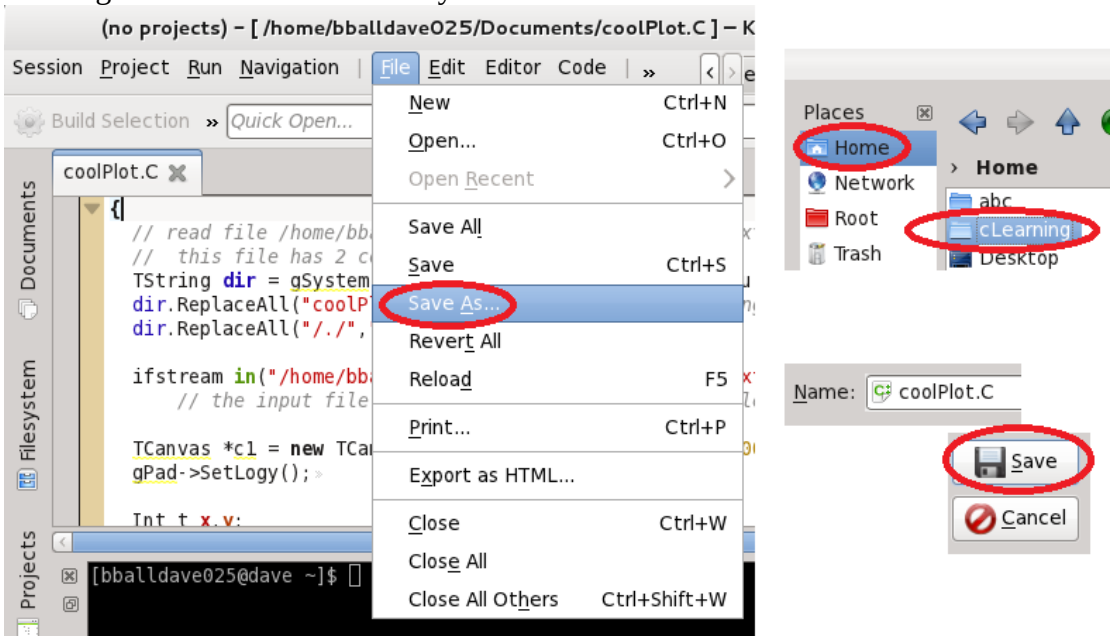


Exit out of gedit (using the "X" in the upper-right corner.)

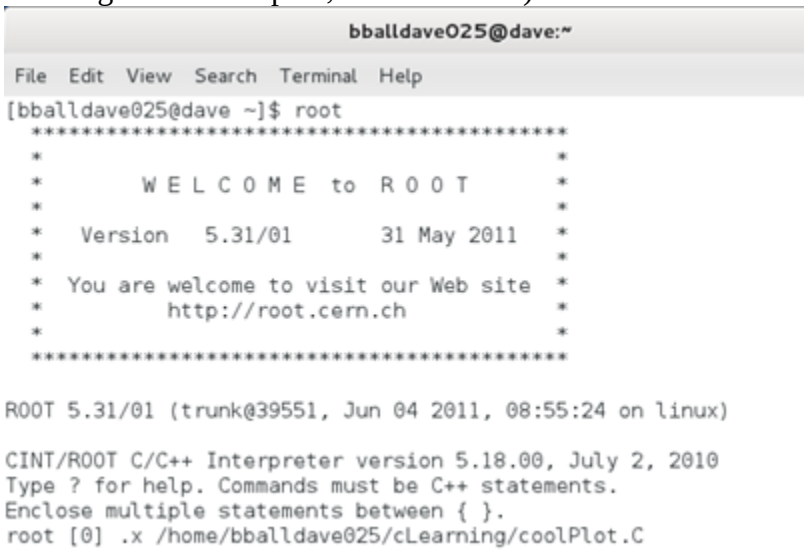
Head on down to Appendix C for another program, which I've called "`coolPlot.C`". I guess you know how I feel about it. Notice the capitol "C". This is the extension for a ROOT macro. It basically just means that it's a list of C++ type commands that ROOT can run. Don't worry; it's nowhere near as long as the last one. Copy and paste it into a KDevelop coding window ("Applications" -> "Programming" -> "KDevelop" and then "File" -> "New").



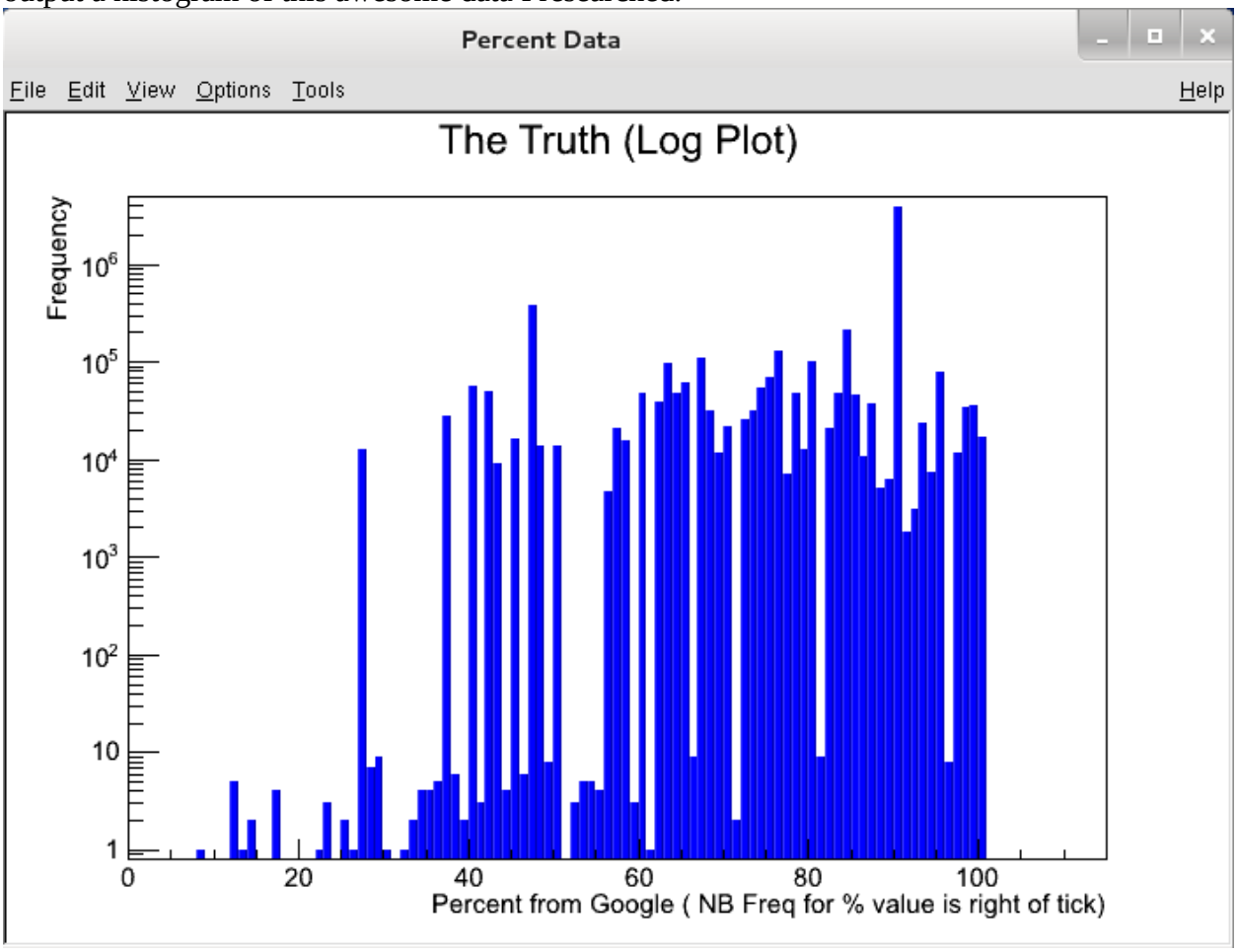
Save the (now empty) file in the “cLearning” directory as “coolPlot.C” with the capitol “C”. You might notice that I had already saved it when I took the screenshot



Go ahead and exit out of KDevelop and then open up your terminal. Make sure you’re in the cLearning directory, or wherever you put the real_stats.txt and coolPlot.C files. Type “root” to start ROOT. You’ll use the “.x” command in ROOT to compile and execute programs. Type “.x coolPlot.C” (not using the full file path, as I did below)



The program will read in the "real_stats.txt" file that you saved before and will then output a histogram of this awesome data I researched.



Can you handle this? What you just plotted are the results of searching Google for "<x>% of * statistics are made up on the spot", where "*" is a wildcard that can stand for any combination of words. Now you know. Apparently, if you're going to make up statistics, shy away from numbers that have remainder 1 when divided by 10. Also, it seems that the space between 100 and 1000 is a barren wasteland in terms of Google results. Thanks to xkcd <http://xkcd.com/715/> for the inspiration and to all the people who use that joke. Oh, and it appears that no one thinks that more than all (or at least between 100% and 115% of) statistics are made up on the spot. By the way, yes, I really did do all that research--it wasn't just made up on the spot.

If you want to know more about file input and plotting options for a TProfile (the kind of histogram we just plotted that has both y and x specified), go ahead and look through the code for the program. It's a good exercise to actually type it all in. This teaches you how to plot y vs. x in ROOT, whereas all the other stuff we did was just counting the frequency of events (something that's done a lot in particle physics.)

(6/4/2011) I'm planning on getting back here and writing about some more of the stuff you can do in ROOT, but this is it for now. I'll include stuff like the user's manual link and maybe read in some data send from The "ABC" Particle Physics Simulator.

General Commentary/Troubleshooting goto Navigation

First of all, it's useful to update your system every once in a while. This is most easily accomplished by going to the terminal, logging in as root, and then typing "yum update".

If something was working and suddenly stops working, you might need to re-install it. Before doing the re-installation, I would update the system as described above. For example, my "Seamless Mode" (the thing that lets you resize the window easily) and my "Clipboard" (what lets you copy/paste between guest, i.e. Linux, and host, i.e. Windows and vice versa) stopped working. These are both things that are provided by Guest Additions. What I did was to log in as root by typing "su" and entering my password, type "yum update" and enter "y" when it asks, and then to go over the same instructions I gave earlier for installing Guest Additions. After that, they both started working again.

```
[bballdave025@daveFedora ~]$ su
Password:
[root@daveFedora bballdave025]# yum update
```

As I mentioned before <http://xkcd.com/627/> , Google can be your best friend.

Remember to not freak out if everything isn't exactly the same as what I have on this tutorial. Usually, the options given in on-screen instructions will get you where you need to go.

v8 published 2012-06-12 bballdave025[at]yahoo[dot]com

Appendix A goto Navigation

particleInfo.cpp

This is the C++ program that I present as a way to learn KDevelop.

```
//************************************************************************
//     particleInfo.cpp    David Black 2 June 2011 - 3 June 2011
//
//     A simple program that takes an inputted string, assumed to be a
//     subatomic particle existing in the atom, and returns some of its basic
//     properties. Includes a cool thing to make it scroll across the screen.
//
//     To run (a dollar sign $" signifies the terminal/command prompt)
//            $ g++ -o particleInfo particleInfo.cpp -lncurses
//            $ ./particleInfo
//
//     Note: requires libncurses. To get it, log in as root
//            $ su
//            Password:
//            # yum install ncurses ncurses-devel
//            # exit
//************************************************************************

#include <iostream>
#include <sstream> // For stringstream (convert int to string)
#include </usr/include/ncurses.h> // For cursor stuff
#include <time.h>
#include <sys/ioctl.h> // For ws_col (terminal width)
#include <errno.h>     // For errors in getting the terminal width
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

using namespace std;

// Function declarations
int getTerminalWidth();
void scrollText(string);
void delay(int);

int trmWidth = getTerminalWidth();
  // the width (in character spaces) of the terminal

int main(int argc, char* argv[])
{
  string outputString;            // string for info about the particle

  bool continueBool = true;
  while (continueBool)
  {
    cout << "Welcome to the particle data program!" << endl;
    cout <<
      "Enter the name of a subatomic particle that exists in the atom" << endl;
    cout <<
      "Please use lower case and don't use spaces. If you need a list" << endl;
    cout << "enter \"list\"." << endl;
    cout << "Go:  ";

    string particleName;
```

```
cin >> particleName; // allows the user to input the particle name

// determine the output string based on the user input
if (particleName == "list")
  outputString =
   "List: proton, neutron, electron, quark, upquark, downquark, gluon, "
   "photon. ";
else if (particleName == "proton")
  outputString =
   "Proton: Mass = 938.27 MeV/c^2; Charge = +1; Spin = 1/2; "
   "Classification = hadron. ";
else if (particleName == "neutron")
  outputString =
   "Neutron: Mass = 939.57 MeV/c^2; Charge = 0; Spin = 1/2; "
   "Classification = hadron. ";
else if (particleName == "electron")
  outputString =
   "Electron: Mass = 0.511 MeV/c^2; Charge = -1; Spin = 1/2; "
   "Classification = lepton. ";
else if (particleName == "quark")
  outputString =
   "Quark: Mass = 2.4(u) OR 4.8(d) MeV/c^2; Charge = +2/3(u) OR -1/3(d); "
   "Spin = 1/2; Classification = quark. ";
else if (particleName == "upquark")
  outputString =
   "Up Quark: Mass = 2.4 MeV/c^2; Charge = +2/3; Spin = 1/2; "
   "Classification = quark. ";
else if (particleName == "downquark")
  outputString =
   "Down Quark: Mass = 4.8 MeV/c^2; Charge = -1/3; Spin = 1/2;"
   " Classification = quark. ";
else if (particleName == "gluon")
  outputString =
   "Gluon: Mass = 0 MeV/c^2; Charge = 0; Spin = 1; "
   "Classification = gauge boson. ";
else if (particleName == "photon")
  outputString =
   "Photon: Mass = 0 MeV/c^2; Charge = 0; Spin = 1; "
   "Classification = gauge boson. ";
else if (particleName == "string")
  outputString =
   "Nice try. ";
else
  outputString =
   "Either you've discovered something new, or you're making stuff up..."
   "er...I mean or what you've entered is not in the atom. ";

outputString += "  ...   Press any button to return to the menu or wait "
             "while the info continues to scroll. ";
scrollText(outputString); // scrolls the info across the screen

bool validInput = false;
while (!validInput)
{
  cout << "Do you want to do it again? [y/n] ";
  string continueStr;
  cin >> continueStr;
  if (continueStr == "y")
  {
   continueBool = true;
   validInput = true;
  }
  else if (continueStr == "n")
```

```
       exit(0);
      else
       cout << "Not valid input." << endl;
    }//endof while (!validInput)
  }//endof while(continueBool)

}// endof int main(int argc, char* argv[])

//-------------------------------------
// Returns the width of the terminal
//-------------------------------------
// ref http://forum.soft32.com/linux/terminal-width-height-ftopict479542.html
int getTerminalWidth()
{
  struct winsize ws;

  if (ioctl(0,TIOCGWINSZ,&ws)!=0)
  {
    fprintf(stderr,"TIOCGWINSZ:%s\n",strerror(errno));
    exit(1);
  }
  return ws.ws_col;
}// endof int getTerminalWidth()

//-------------------------------------
// Scrolls a string across the terminal
// marquee style.
//-------------------------------------
// ref http://www.daniweb.com/software-development/cpp/threads/344918
void scrollText(string str)
{
  int finalLen = trmWidth - 1;    // length of the outputed string
  int infoLen = str.length();     // length of the info string
  int i = finalLen;        // iterator

  initscr();               // get screen ready for ncurses
  curs_set(0);                   // cursor not visible
  raw();                   // setting for ncurses allowing nodelaya
  nodelay(stdscr, true);   // loop executes even if no button pressed
  int ch = -1;                   // the character variable
  while (ch = getch() == -1)     // wait for a button to be pressed
  {
    clear();

    int begSpaceLen, infoSubLen, endSpaceLen;
    int infoStart = 0;

    // For the cases | represents a side of the terminal window and _
    //  represents the info string.
    // So...
    // |              _____________              |
    //   begSpaceLen  infoSubLen  endSpaceLen
    // Case 1 |    _|
    if (i >= 0 && i + infoLen >= finalLen)
    {
      begSpaceLen = i;
      infoSubLen = finalLen - begSpaceLen;
      endSpaceLen = 0;
    }
    // Case 2 | __ |
    else if (i >= 0 && i + infoLen < finalLen)
    {
      begSpaceLen = i;
```

```
      infoSubLen = infoLen;
      endSpaceLen = finalLen - (begSpaceLen + infoSubLen);
    }
    // Case 3 |____|
    else if (i < 0 && i + infoLen >= finalLen)
    {
      begSpaceLen = 0;
      infoSubLen = finalLen;
      infoStart = -i;
      endSpaceLen = 0;
    }
    // Case 4 |_   |
    else if (i < 0 && i + infoLen < finalLen && i + infoLen >= 0)
    {
      begSpaceLen = 0;
      endSpaceLen = finalLen - (i + infoLen);
      infoSubLen = finalLen - endSpaceLen;
      infoStart = -i;
    }
    // Case 5 |    |
    else
    {
      begSpaceLen = 0;
      infoSubLen = 0;
      endSpaceLen = finalLen;
      i = finalLen;
    }
    string begSpaceStr(begSpaceLen, ' ');
    string infoSubStr = str.substr(infoStart, infoSubLen);
    string endSpaceStr(endSpaceLen, ' ');
    string finalStr = begSpaceStr + infoSubStr + endSpaceStr;

    addstr(finalStr.c_str());           // addstr takes a c-type string

    refresh();

    //delay (100000 is ideal)
    delay(100000);
    i--;
  }//endof while

  curs_set(0);                    // cursor visible
  endwin();                // exit from ncurses
}//endof void scrollText(string str)

//------------------------------ -------
// Causes a pause in the system of
// numMillisec milliseconds (ish)
//-------------------------------------
// ref http://www.gidforums.com/t-17762.html
void delay(int numMillisec)
{
  clock_t wait;            // time to wait
  wait = clock() + numMillisec; // get current time and add the delay
  while (wait > clock());
    /*do nothing*/
}//endof void delay(int)
```

Appendix B goto Navigation

`real_stats.txt`

Data I researched for a ROOT plot. Make sure to copy and paste it as is.

```
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	1
9	0
10	0
11	0
12	5
13	1
14	2
15	0
16	0
17	4
18	0
19	0
20	0
21	0
22	1
23	3
24	0
25	2
26	1
27	12400
28	7
29	9
30	1
31	0
32	1
33	2
34	4
35	4
36	5
37	28000
38	6
39	2
40	56800
41	3
42	49200
43	8920
44	4
45	16300
46	6
47	383000
48	13800
49	8
50	13500
51	0
52	3
53	5
54	5
55	4
56	4590
57	21000
```

58	15500
59	3
60	47900
61	1
62	38400
63	97800
64	48400
65	60100
66	9
67	111000
68	32100
69	11500
70	21600
71	2
72	26100
73	31500
74	54100
75	69000
76	128000
77	7120
78	47700
79	12600
80	99200
81	9
82	21100
83	46700
84	210000
85	46600
86	10600
87	37300
88	5130
89	6160
90	3810000
91	1810
92	3030
93	23600
94	7450
95	79900
96	8
97	11600
98	33900
99	35900
100	16700
101	0
102	0
103	0
104	0
105	0
106	0
107	0
108	0
109	0
110	0
111	0
112	0
113	0
114	0
115	0

Appendix C

`coolPlot.C`

This is a program that imports and then plots some data from a .txt file--actually the .txt file in the previous appendix. It's just a way to teach you how to use ROOT macros will also be useful if you ever want to read in data from an ascii file.

```
//**************************************************************************
// coolPlot.C       By David Black            4 June 2011 - 4 June 2011
// ref http://root.cern.ch/root/html/tutorials/tree/basic.C.html
// ref http://superk.physics.sunysb.edu/~mcgrew/phy310/
//                  lectures/phy310-lecture-02-2007.pdf for the TGraph try
// ref http://root.cern.ch/root/html404/TProfile.html (the Class Description)
// ref http://root.cern.ch/root/html/THistPainter.html
//
// Reads in a .txt file with two columns of data and creates a TProfile
// histogram.
//**************************************************************************
// good thing to search: <whatever> root -user cern

#include "Riostream.h"
void coolPlot()
{
  // read file /home/bballdave025/cLearning/real_stats.txt
  //  this file has 2 columns of int data
  TString dir = gSystem->UnixPathName(gInterpreter->GetCurrentMacroName());
  dir.ReplaceAll("coolPlot.C","");             // getting it to accept the
  dir.ReplaceAll("/./","/");                   // macro

  ifstream in("real_stats.txt", std::ios::in);
      // the input file stream that reads in the text file with the data

  TCanvas *c1 = new TCanvas("c1","Percent Data",200,10,700,500); // window
  gPad->SetLogy();         // log plot

  Int_t x,y;
  Int_t nlines = 0;

  TProfile *hprof  = new TProfile("hprof","Percent Data",
                          115,0,115,0,4000000); // the histogram

  while (1)
  {
    in >> x >> y;
    if (!in.good()) break;
    hprof->Fill(x,y,1);
    nlines++;
  }//endof while (1)

  //printf(" found %d points\n",nlines);

  in.close();

  //hprof->SetLineColor(kBlack);
  hprof->SetFillColor(kBlue);
  hprof->SetStats(0);            // don't display statistics

  hprof->SetTitle("Percent Data (Log Plot)");
  hprof->SetMaximum(5e6);
  hprof->SetMinimum(8e-1);
  hprof->GetXaxis()->SetTitle("Percent from Google ( NB Freq for % value is "
                                    "right of tick)");
```

```
  hprof->GetYaxis()->SetTitle("Frequency");
  hprof->Draw("bar2");     // makes visible lines down each bar of the histogram

  // attempt with TGraph--works, but not as good visually; good for reference
  /*TGraph* g = new TGraph("/home/bballdave025/cLearning/real_stats.txt");
  g->SetTitle("Data");
  g->SetMaximum(5e6);
  g->SetMinimum(8e-1);
  g->GetHistogram()->SetXTitle("Percent from Google");
  g->GetHistogram()->SetYTitle("Frequency");
  g->Draw("la");    // 'l' for lego (linked lines), 'a' for axis*/
}//endof void coolPlot()
```

v8 published 2012-06-12 bballdave025[at]yahoo[dot]com