

Programming the LEGO™ Mindstorms Robot Using C++, Code::Blocks, BrickOS and the BrickEMU

CPlusVEBot: C++ Virtual Environment for Programming Robots

Amy Delman
Lawrence Goetz
Yedidyah Langsam
Theodore Raphan

Brooklyn College
City University of New York

Version 1.0.5
Copyright © 2009

Table of Contents

Section 1:	Introduction.....	3
Section 2:	Basic Concepts.....	4
Section 3:	Programming Environment.....	7
Section 4:	Real Brick and its Emulator	8
Section 5:	The Lego Robot and the RCX Brick.....	9
Section 6:	Installation.....	12
6.1:	System requirements	12
6.2:	Installation – Step a Copying the installation files to your disk (on a PC).....	12
6.2:	Installation – Step a Copying the installation files to your disk (on a Mac).....	15
6.2:	Installation – Step b Installing VirtualBox (on a PC).....	17
6.2:	Installation – Step b Installing VirtualBox (on a Mac)	26
6.2:	Installation – Step c Installing the Ubuntu Client	33
6.3:	How to Uninstall the Software.....	47
Section 7:	First Use	48
Section 8:	The Ubuntu Linux Desktop	52
Section 9:	How to Shutdown the Ubuntu Client and the VirtualBox Console	55
Section 10:	First BrickOS Program	57
Section 11:	Using the Lego RCX Brick.....	64
11.1	Installing the USB Tower	64
11.2	How to install the firmware	71
11.3	How to transfer the compiled program to the LEGO Brick.....	72
11.4	Running your program.....	72
Section 12:	Advanced Topics	73
12.1:	Setting-up the Printer	73
12.2:	How to Setup Shared Folders	73
12.3:	Using the DDD Debugger	79
Appendix I:	References.....	80
Appendix II	Installing the BrickOS Firmware into the Emulator.....	81
Appendix III	Installing VirtualBox Guest Additions	83

Programming the LEGO™ Mindstorms Robot¹ Using C++, Code::Blocks², BrickOS³ and the BrickEMU⁴

Section 1: Introduction

The Lego Mindstorm Robot is a wheeled robot that can be assembled in different configurations and can be made to move and sense things through touch and light sensors as it moves. The purpose of this tutorial is to give you some understanding of how to program the robot. You will be introduced to the commands that will control wheel motion and sensing through touch and light. This type of programming is especially important for science and engineering students because programming does not just involve writing code to perform calculations and output results on a computer screen, but also involves process control. Many computer applications require the writing of programs to sense the environment and perform a task based on the information that is sensed. These issues come up in multimedia applications, including game playing. By learning to program a robot, you will be introduced to the concept of how programs can be written to interact with the environment.

As this is the first course in programming, the programs that you will write will be simple, but will introduce you to basic mechanisms of sensing and control using C++.

¹ <http://www.lego.com/>

² <http://www.codeblocks.org/>

³ <http://brickos.sourceforge.net/>

⁴ <http://hoenicke.ath.cx/rcx/brickemu.html>

Section 2: Basic Concepts

A computer *program* is a set of instructions that can be followed by the computer to perform a given task. These instructions may be written in any number of languages, such as C++, Java, and so on. These languages are known as high-level languages and are a convenient way for the programmer to express the algorithm used to solve the program. A program written in a high-level language is known as the *source code*. Unfortunately, the computer can only understand a difficult to follow low-level language, consisting of zeros and ones, known as *machine code*. A *compiler* translates the source code into an intermediate form known as *object code*, which is then combined with various pre-written routines contained in libraries to form the executable file (machine code) which can be *executed* (or *run*) by the computer. In order to distinguish the various files from one another, they are assigned extensions. Thus source code written in C++ may have the *.cpp* extension, the output of the compiler the *.obj* extension and the executable may be assigned the *.exe* extension. Under the BrickOS system which we will be using to program the Lego Mindstorms Robot, the compiler translates the C++ source code into a form understood by the robot, which is saved in a file having the *.lx* extension

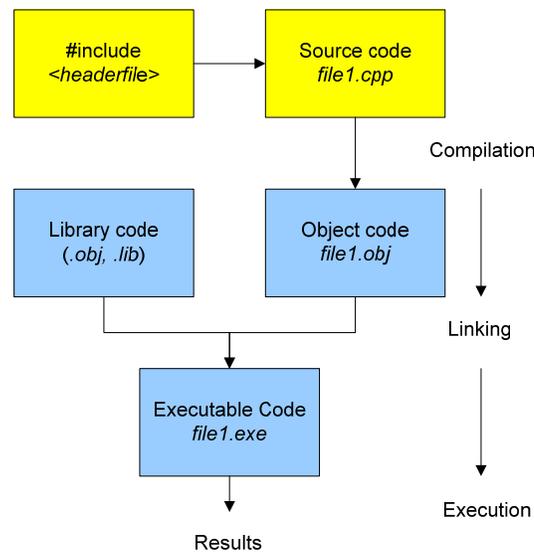


Figure 1

All computers must follow a set of basic instructions which starts up the computer, allow for communication between the devices in your computer (such as your hard drive and graphics card) and the rest of the system hardware. This software is known as the *BIOS*, or Basic Input/Output System and is stored in a semi-permanent location known as the *CMOS* (Complimentary Metal-Oxide Semiconductor) memory. The term *firmware* is used to describe the software that implements instruction sets that can run any device and is generally stored in the memory of a device. In the computer that runs a robot, the firmware is the program residing in the memory of the robot that interprets and implements the program that instructs the robot to behave in a certain fashion. Below, we

will describe the firmware, *BrickOS*, which will be used to run the C/C++ programs that will control the Lego robot.

An *operating system* (OS) provides the interface between the user, the application programs and the computer's hardware, Figure 2.

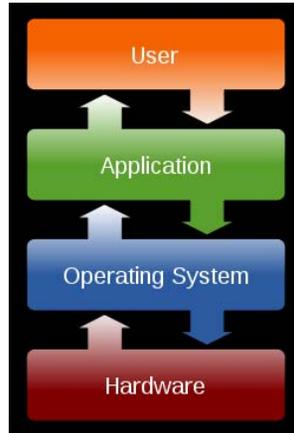


Figure 2

All programs that you use or write are managed by the operating system that is installed on your computer. Operating systems with which you may be familiar include: Windows XP, Windows Vista, Mac Os and Ubuntu Linux.

Most users select a single operating system, often pre-installed on their computer at the time of purchase. Thus IBM compatible – Intel based computers often have Windows Vista or Windows XP pre-installed, while Apple- Intel based computers will have Mac OS X as their native operating system. (It is interesting to note, that Mac OS X is actually based upon a Linux variant). In order to take advantage of features and software programs that may be available only on a specific operating system, advanced users often find it useful to be able to use more than a single operating system. One way to accomplish this is by permanently installing two different operating systems using what is known as a dual-boot system. However, a more modern (and more flexible) alternative is known as platform virtualization or as the *virtual machine*. A virtual environment is created by software (such as Sun's VirtualBox or Microsoft's VirtualPC) that is installed on a *host machine*, which is running a particular host operating system (This could be any of the operating systems described above). This software creates a *virtual machine* on which other operating systems can be installed. These are known as *guest* operating systems. The guest operating system, runs just as if it were installed on a stand-alone hardware platform, that has the characteristics of the virtual machine. Typically, many virtual machines can be simulated on a single physical machine, their number limited by the host's hardware resources. Figure 3 is an example of Windows XP hosting two virtual machines: a virtual machine running a Windows Vista guest as well as an independent virtual machine running Ubuntu Linux as a guest, using VirtualBox as the virtualization platform.

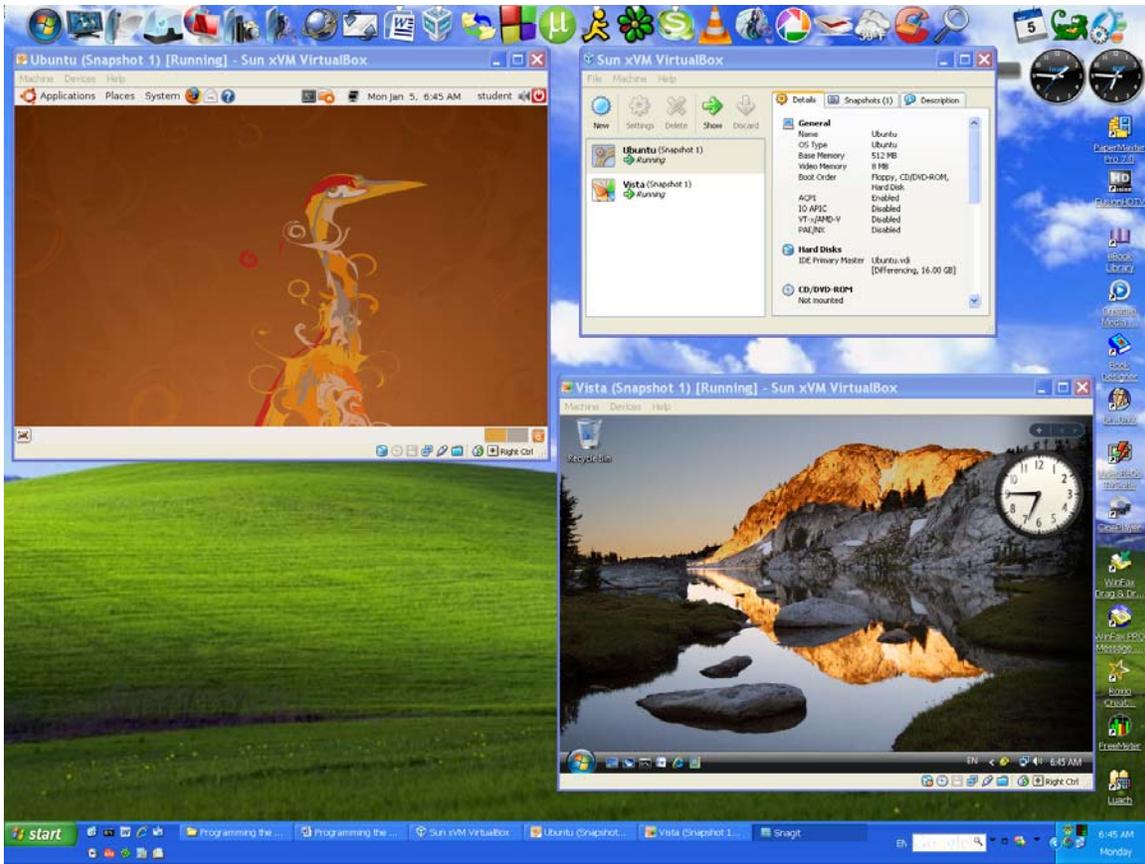


Figure 3 - Windows XP hosting Vista and Ubuntu Linux

In this example, Windows XP is the host OS while Ubuntu Linux and Windows Vista are the guests. Each of the guests may be considered to be running on independent virtual machines, so that the user has the luxury of owning three computers at the price of one.

Open source software (OSS) refers to software that is developed, tested, or improved through public collaboration and distributed with the idea that the must be shared with others, ensuring an open future collaboration.⁵ In contrast to proprietary software, OSS may be freely downloaded and used (as long as they include the source code for others to modify and change). All the software that we will be using in this manual is of the OSS variety. Because of the nature of the public collaboration, discussion forums are available for the user (and developer) who need further information. Students are urged to follow the links given in the Appendix for more information.

⁵ http://searchenterpriselinux.techtarget.com/sDefinition/0,,sid39_gci214343,00.html

Section 3: Programming Environment

Programs in this course will be developed using an *Integrated Development Environment* (IDE) called *Code::Blocks*, whose use and structure is described in the *Code::Blocks Student Manual*, by Goetz, Langsam and Raphan, available here: <http://www.sci.brooklyn.cuny.edu/~goetz/codeblocks/codeblocks-instructions.pdf>. Briefly, Code::Blocks is an environment that allows you to *build* (*compile and link*) programs that can be run. For running C++ programs under Code::Blocks, a *compiler* will be used to translate the C++ programs which you will write, into an intermediate form of machine language which can be understood by the computer. A *linker* combines your program with other modules that are in a library that are necessary to print, read data and do the calculations that make the program work. To *build* your program is to compile and link the intermediate code and pre-written modules into the final form which can be executed on the computer. In more advanced programs, the linker combines different modules to form one major program. For normal programming, a compiler is used that works for the host system. When working on a PC under Microsoft Windows, the compiler will translate the program to work with the processor that is in the computer as well as generating the appropriate code to interact with the Windows environment. On a Mac, it will translate the program to the processor on the computer and interact with MacOS X. Similar translations will occur for computers running other operating systems such as Linux.

To control the robot, it will be necessary to translate the program into a language the robot understands. This will require a different type of compiler, which is called a *cross-compiler*. This *cross-compiler* has been incorporated into the Code::Blocks IDE environment. Thus, Code::Blocks can be used to cross-compile robot programs and translate them into the language that the robot microprocessor can run and make the robot perform.

Once the robot program has been compiled, it will then be necessary to download the program into the robot. This is done through a wireless (infrared) connection from the host computer, using a USB *Tower*, to the robot computer, which is known as the *Brick*. Before a program can be downloaded to the Brick, however, the Lego BrickOS firmware (which will reside in the memory of the Brick), must be downloaded. So, running a robot program requires five steps:

1. Download the BrickOS firmware to the Brick over the wireless connection from the host computer to the Brick.
2. Write a program using the Code::Blocks IDE.
3. Build the program (cross-compile and link) using the Code::Blocks IDE.
4. Download the built program to the Brick
5. Push the *run* button on the Brick and watch the robot do its thing.

As mentioned in the introduction, the different stages of compilation and creating executable code generate files that have different extensions.

Extension	Purpose
.coff	<i>brickOS.coff</i> – firmware that runs the C++ code downloaded to the Brick
.lx	<i>filename.lx</i> – compiled BrickOS C++ code
.c	<i>filename.c</i> – Source code written in C
.cpp	<i>filename.cpp</i> – Source code written in C++
.o or .obj	<i>filename.o</i> – Object code produced by the C++ compiler
.exe	<i>filename.exe</i> – Executable file produced by the linker which may be run on the computer

Table 1

Section 4: Real Brick and its Emulator

Since not everyone has the resources to buy and build a real Lego Robot, an open source emulator has been developed, which is called *BrickEMU*. This emulator is a system that runs on a computer and executes the instructions of the real Brick, but on your computer. Using the BrickEMU emulator you can download firmware to the virtual Brick, write programs for the Brick, download them to the Brick have them run just as if you had the real Brick. This helps debug programs that you will write so that you can then have them run on the real robot.

The emulator has also been incorporated into the Code::Blocks Integrated Development Environment. Using the Code::Blocks IDE you can download firmware, write and test programs, and run them in emulation just as if you had the real Brick all within the same environment.

The BrickOS Emulator has been designed to function from within a Linux environment. In order to provide a common platform to all students, whether they are using a Windows PC with the XP or Vista operating system, or a Mac PC using OS X, all the programs necessary to program the real or emulated Brick have been incorporated within an Ubuntu Linux client hosted on a VirtualBox virtualized PC. This will provide a common environment for the entire class.

Section 5: The Lego Robot and the RCX Brick

One of the many robots that can be built using the Lego RCX Robot system is shown in Figure 4.



Figure 4 - A Lego Roverbot with two touch sensors, one light sensor, and two motors

The heart of the Lego Robot is the RCX Brick (Figure 5) which contains a Hitachi 8600 microprocessor, three sensor ports (Sensor 1, Sensor 2, and Sensor 3) and three motor ports (A, B, C). The motors and sensors can be independently controlled by the program. It also has a LED display so that messages can be sent to the robot and displayed. The Lego USB Tower by which programs developed on the PC (.lx files) can be downloaded to the Brick is shown in Figure 6.



Figure 5 - Lego RCX Brick



Figure 6 - Lego USB Tower

The BrickEMU is designed to mimic the actual RCX Brick and is shown in Figure 7.

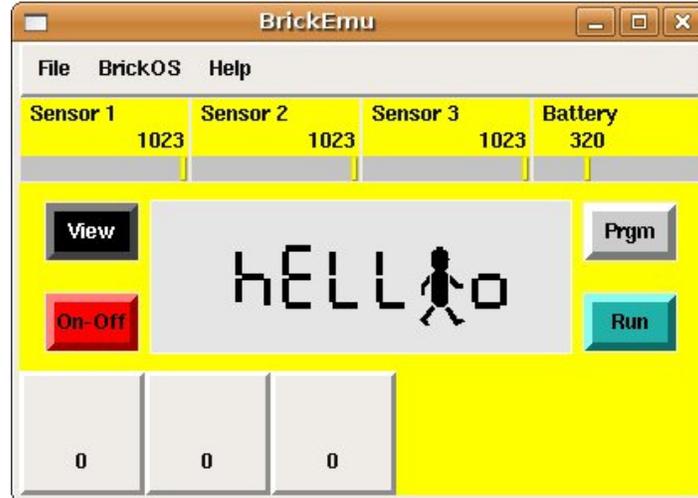


Figure 7 - BrickEMU

The values sent to the motor ports are displayed in the boxes at the bottom of the display (currently displaying a value of 0 for all three motor ports). The sensor ports as well as the battery level may be modified by sliding the tiny yellow bar under each sensor. You will need to change the value by about 50% before the sensor will react. In a similar fashion you may simulate the battery level by adjusting the yellow bar in the “Battery” section of the display. Each of the four buttons (*View*, *Prgm*, *On-Off*, and *Run*) emulate the buttons on the actual brick. And of course, the center panel emulates the brick’s LCD.

We will have more to say about the emulator later on.

Section 6: Installation

6.1. System requirements⁶

In order to run VirtualBox on your machine, you need:

- **Reasonably powerful x86 hardware.** Any recent Intel or AMD processor should do. (Note: You can not run VirtualBox on a PowerPC based Mac.)
- **Memory.** You will need at least 512 MB of RAM (but probably more, and the more the better). Basically, you will need whatever your host operating system needs to run comfortably, plus the amount that the guest operating system needs. So you probably won't enjoy the experience much with less than 1 GB of RAM. Two or more GB of RAM will be more than adequate on an XP host, while 3-GB of RAM will be more than adequate for a Vista Host.
- **Hard disk space.** While VirtualBox itself is very lean (a typical installation will only need about 30 MB of hard disk space), the virtual machines will require fairly huge files on disk to represent their own hard disk storage. So, to install Windows XP, for example, you will need a file that will easily grow to several GB in size. For our system you will need a minimum of 6 GB free space. The DVI file (containing the virtual disk) is over 4GB. Technically the DVI file has a max of 16 GB so that a total of 20GB of free space would be more than adequate for future growth.

6.2. Installation

The installation of the system while lengthy is quite easy to do. It contains three steps:

- a. Copying the installation files to your disk
- b. Installing VirtualBox
- c. Installing Ubuntu Linux

Step a: Copying the installation files to your disk (Windows based PC)

Insert the Installation Disk (available from your instructor) into your computer.

After a short period of time, on a Windows XP/Vista PC you will see a screen similar to the following.

If you are using a Mac please go to page 15.

⁶ http://www.virtualbox.org:80/wiki/End-user_documentation



Figure 8

Select **Run Menu.bat**. The screen on the next page will appear.

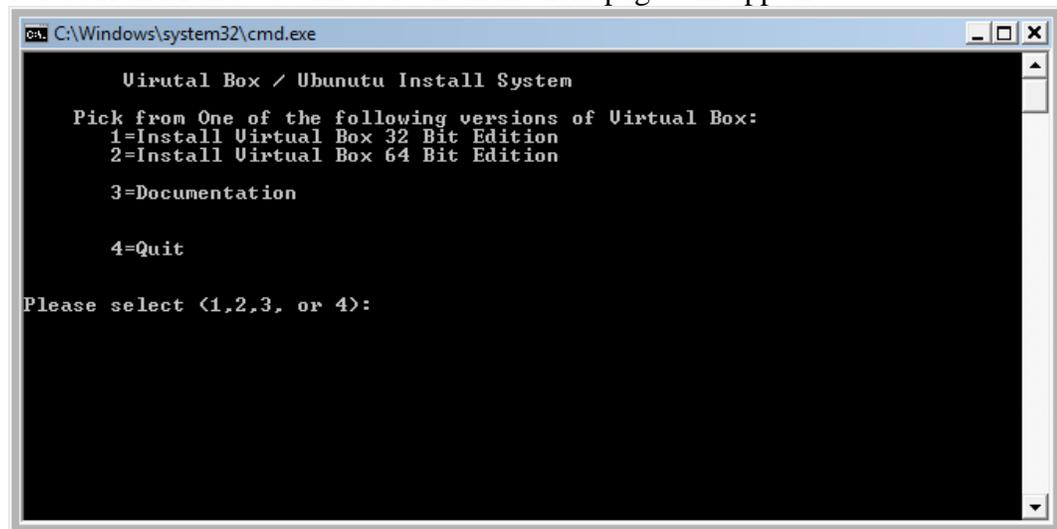


Figure 9

Select *choice 1* in order to install the 32 bit edition of VirtualBox or *choice 2* in order to install the 64 bit edition of VirtualBox. (Be sure to match the version of your host operating system). Press *Enter* after making your selection.

After the install is done, the system will copy Ubuntu to your computer. Please wait while this takes place, as it will take several minutes. Proceed to Step 2 of the installation procedure (page 12).

To read the documentation, select *choice 3* then *Enter*.

To quit, select *choice 4* and then *Enter*.

The DVD contains some useful documentation for you to read. Upon selecting *choice 3* the following Documentation Screen appears:

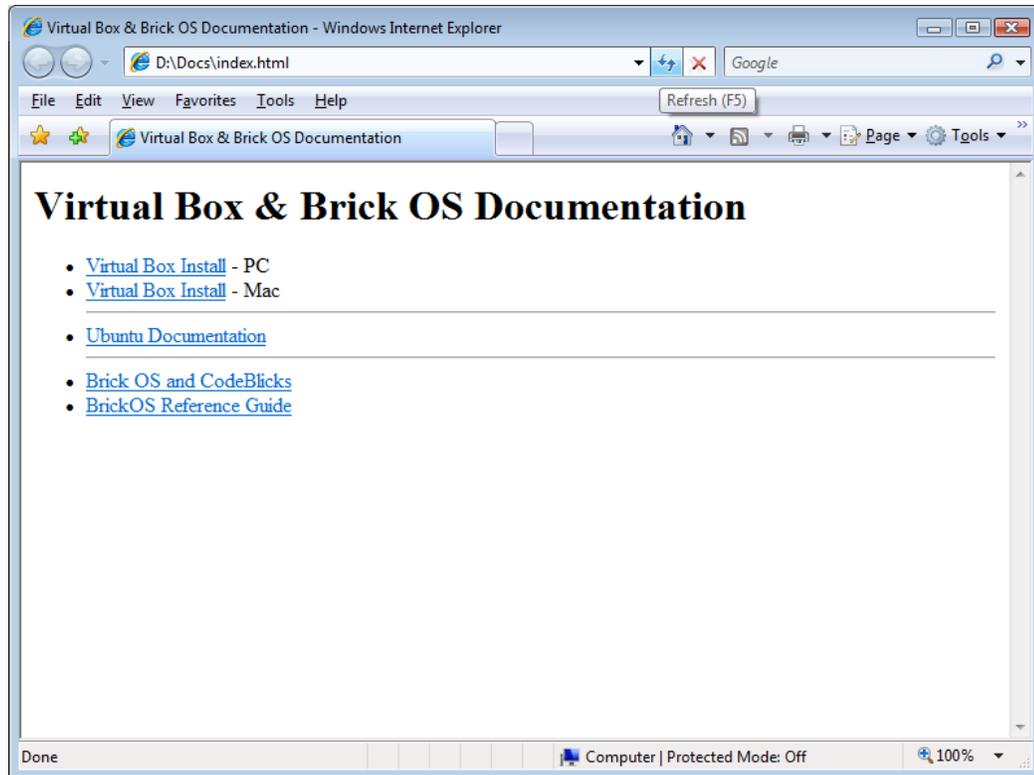


Figure 10

Step a: Copying the installation files to your disk (MAC OS X)

Insert the Installation Disk (available from your instructor) into your computer.

After a short period of time, on a MAC OS X you will see a screen similar to the following.

If you are using a Windows based PC please go to page 12.



Figure 11

Click on the DVD icon.

You will then see the following screen:

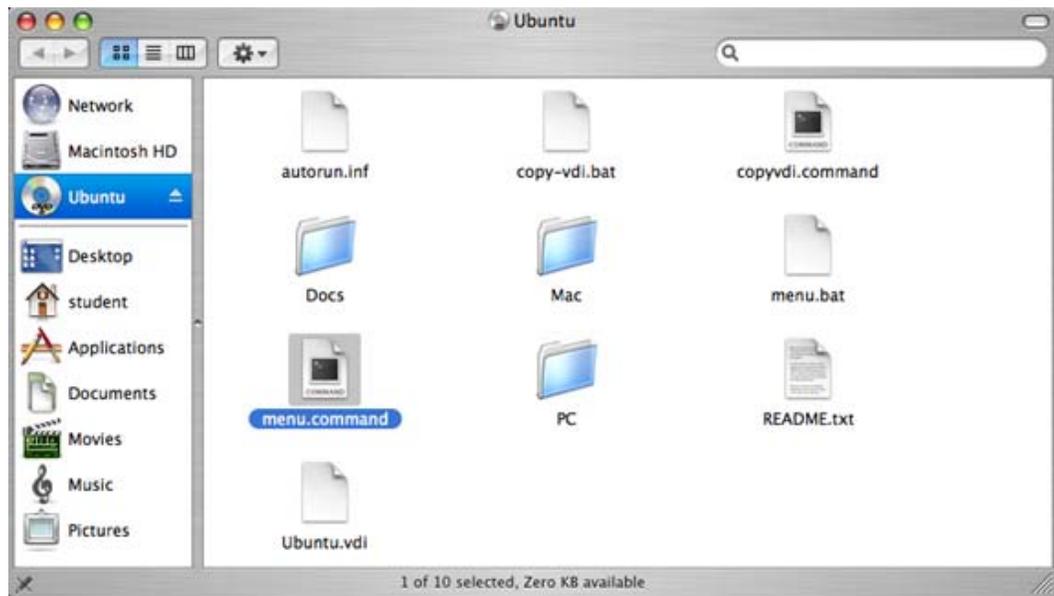


Figure 12

Double click on the file called *menu.command* and you will see the screen below



Figure 13

Select menu option 1 to install VirtualBox and proceed to *Step b: Installing VirtualBox (on a Mac)* on page 26.

Step b: Installing VirtualBox (on a PC)

If you are installing VirtualBox on an Apple Mac please skip to page 26.

You will see the following screen. Follow the steps of the installation wizard as shown below:

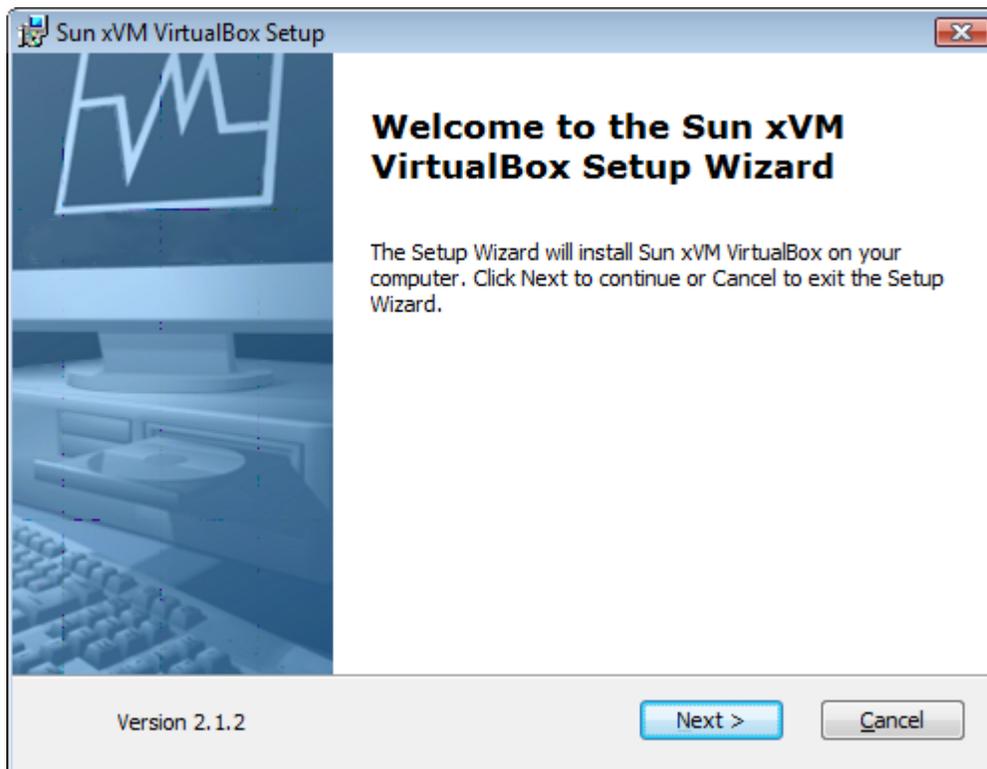


Figure 14

Click **Next** to continue.

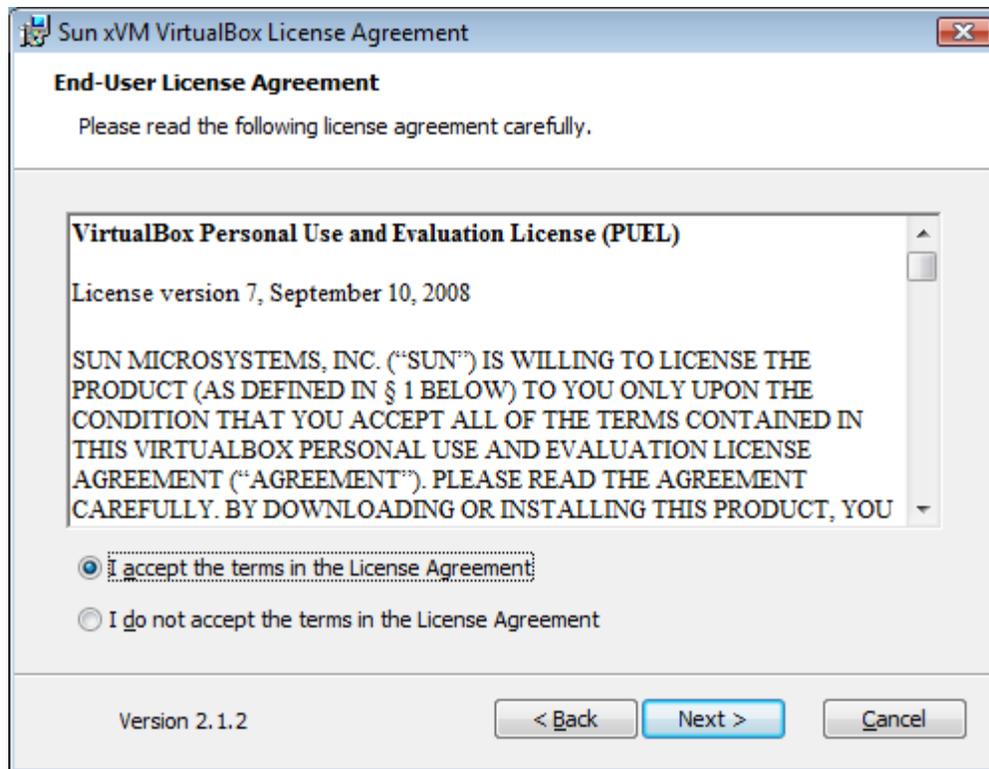


Figure 15

Accept the agreement and select **Next**.

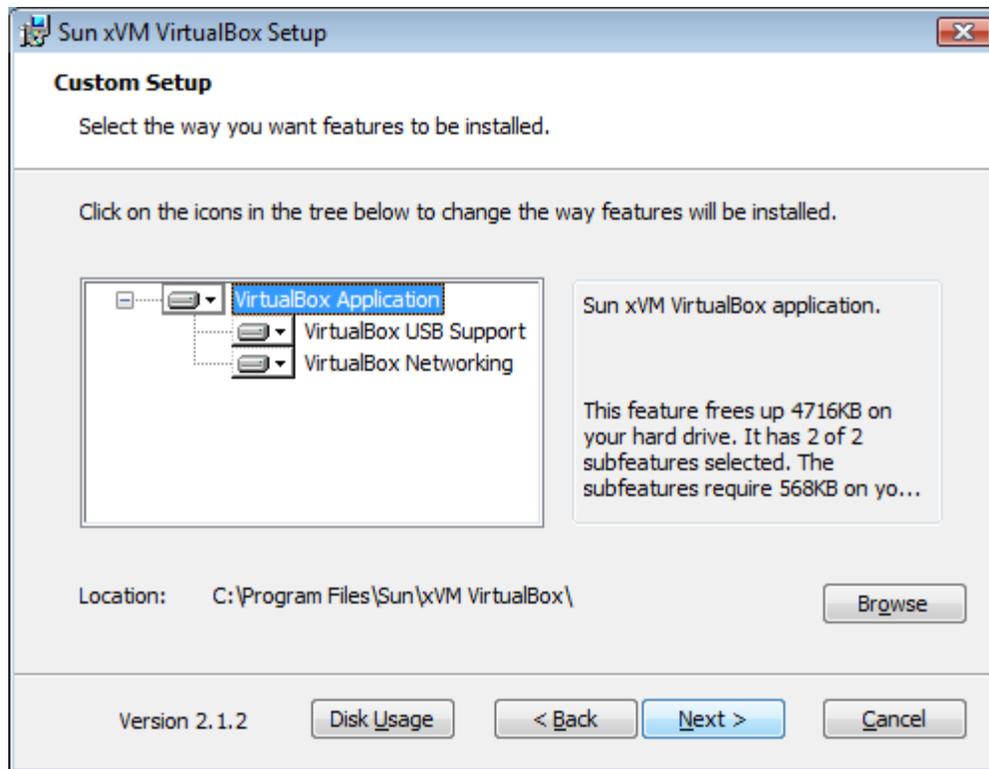


Figure 16

Leave the defaults set and click **Next**.

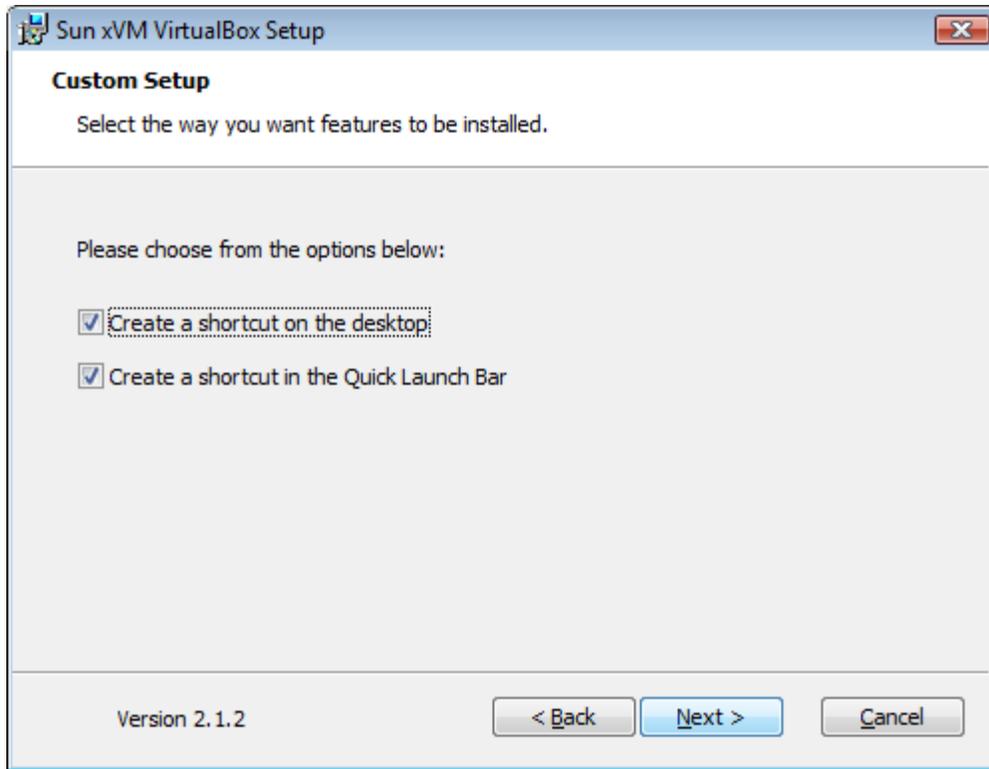


Figure 17

Leave the defaults set and click **Next**.

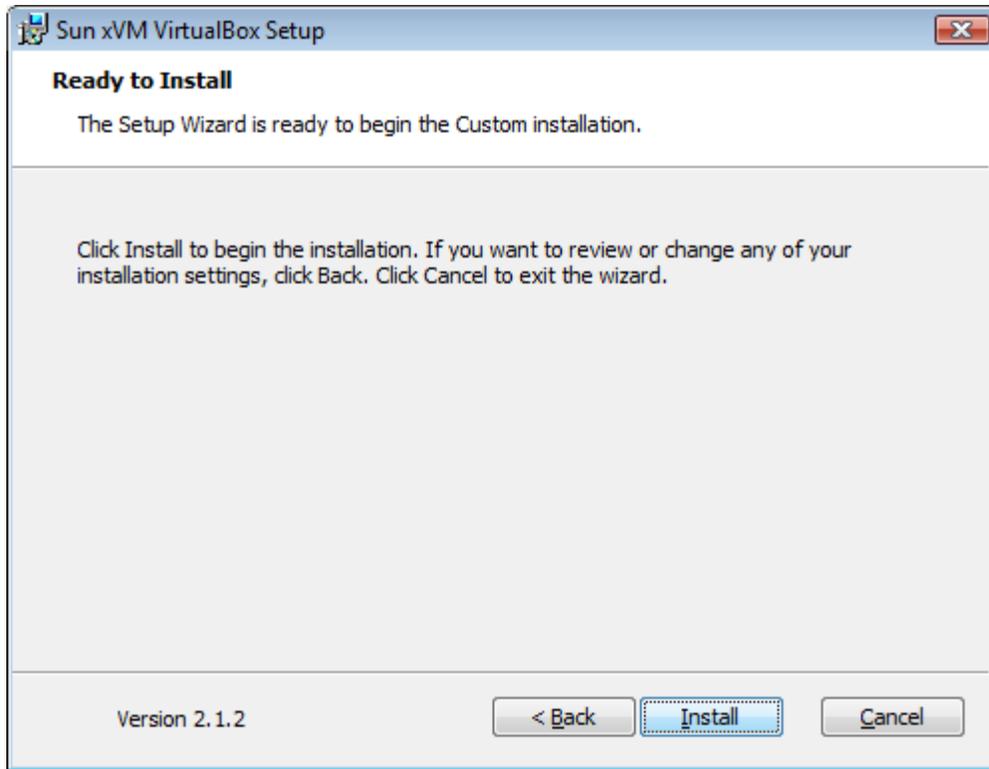


Figure 18

Click **Install** to install the program.

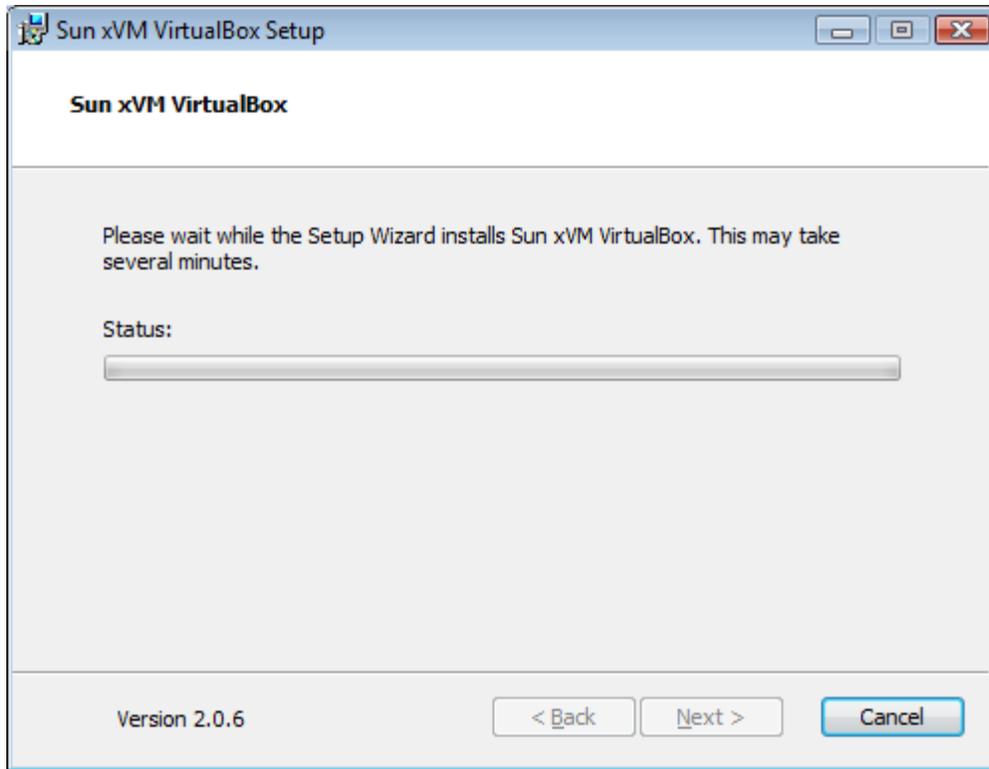


Figure 19

Please wait.

If any windows ask for your permission to continue, please allow them to install.

If you get a window like these, select to **Install** this driver software anyway.

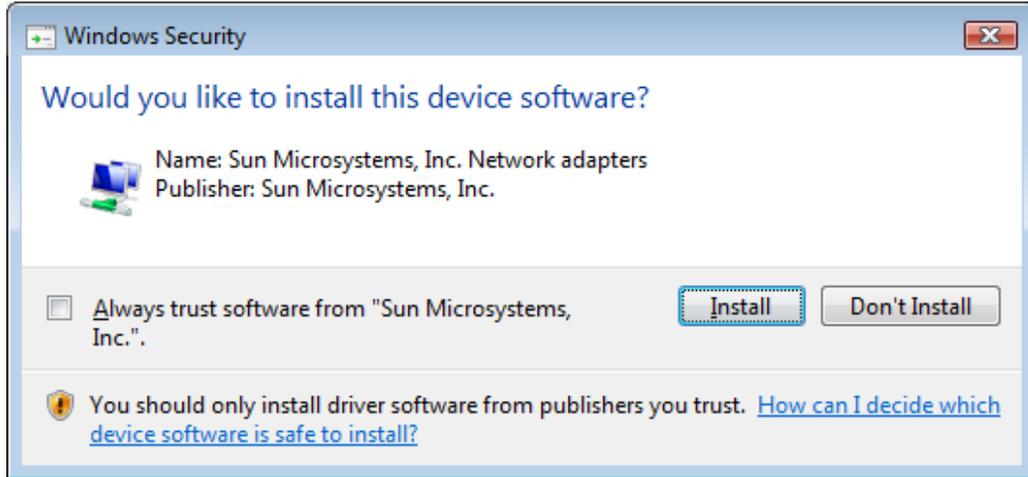


Figure 20

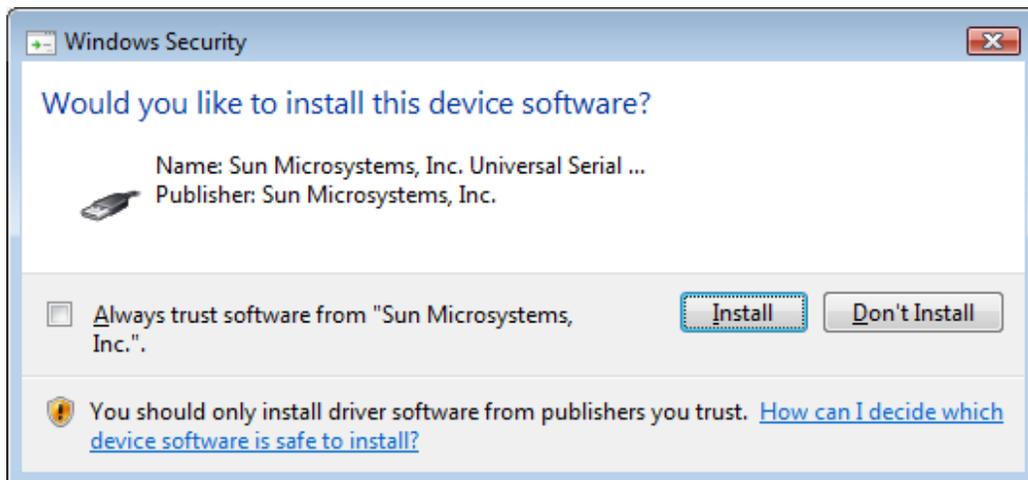


Figure 21

In Windows XP, please select to **Continue Anyway** when questioned.



Figure 22

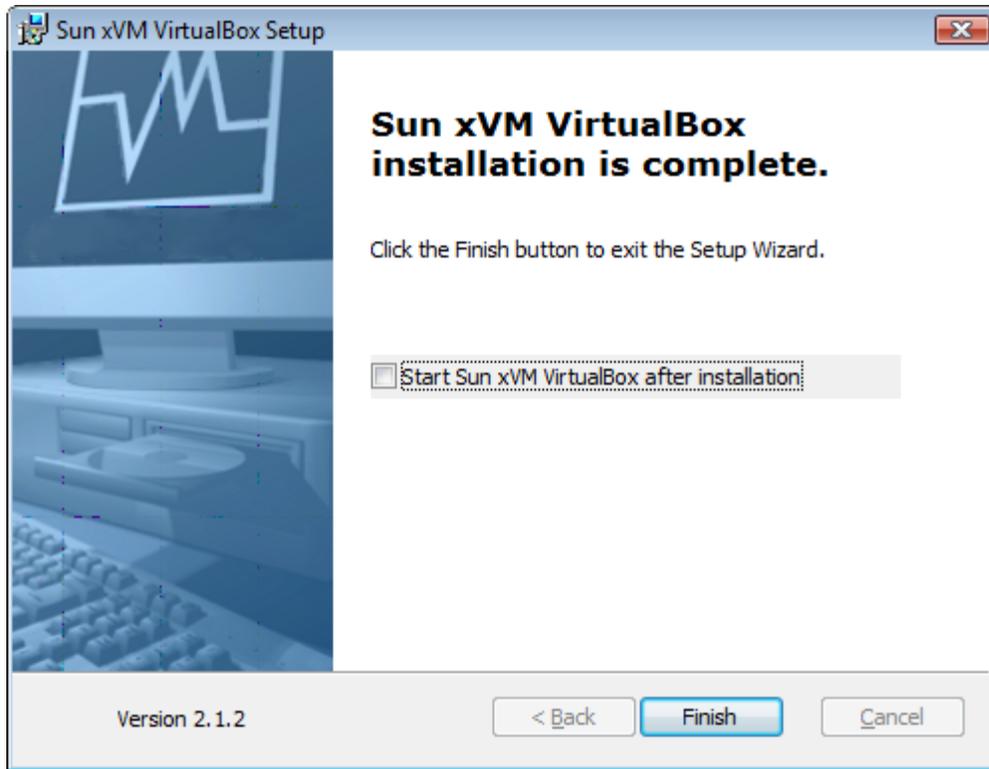


Figure 23

Uncheck 'Start *Sun xVM VirtualBox* after installation' and press **Finish** to exit.

Virtual Box has been installed!

Proceed to Step c 'Installing the Ubuntu Client' on page 33.

Step b: Installing VirtualBox (on a Mac)

If you are installing VirtualBox on a PC please go to page 17.

You will see the following screen. Follow the steps of the installation wizard as shown below:

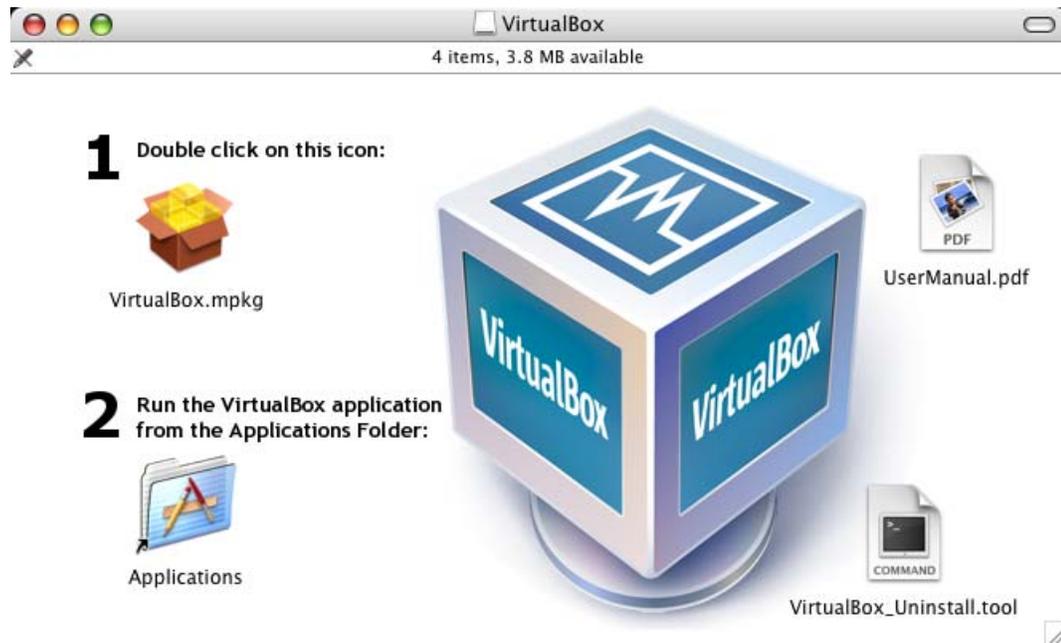


Figure 24

Double click the **box icon** from step 1.

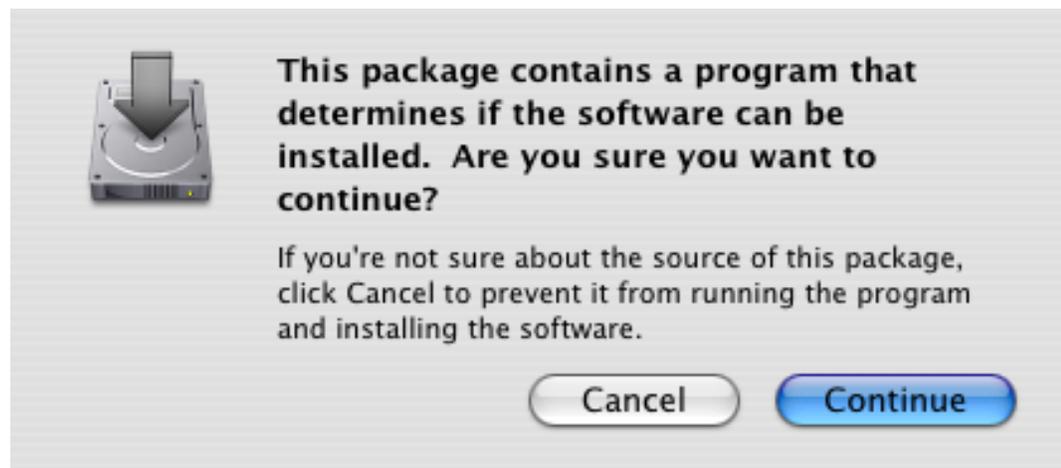


Figure 25

Press **Continue**



Figure 26

Press **Continue**

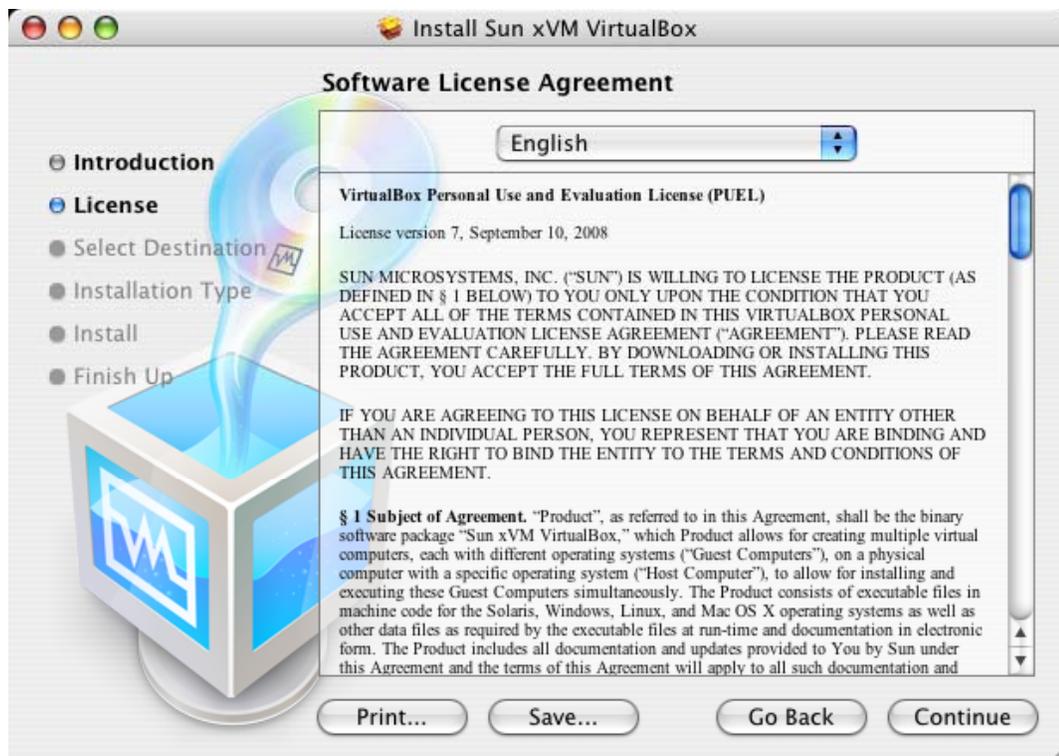


Figure 27

Press **Continue**



Figure 28

Press **Agree**

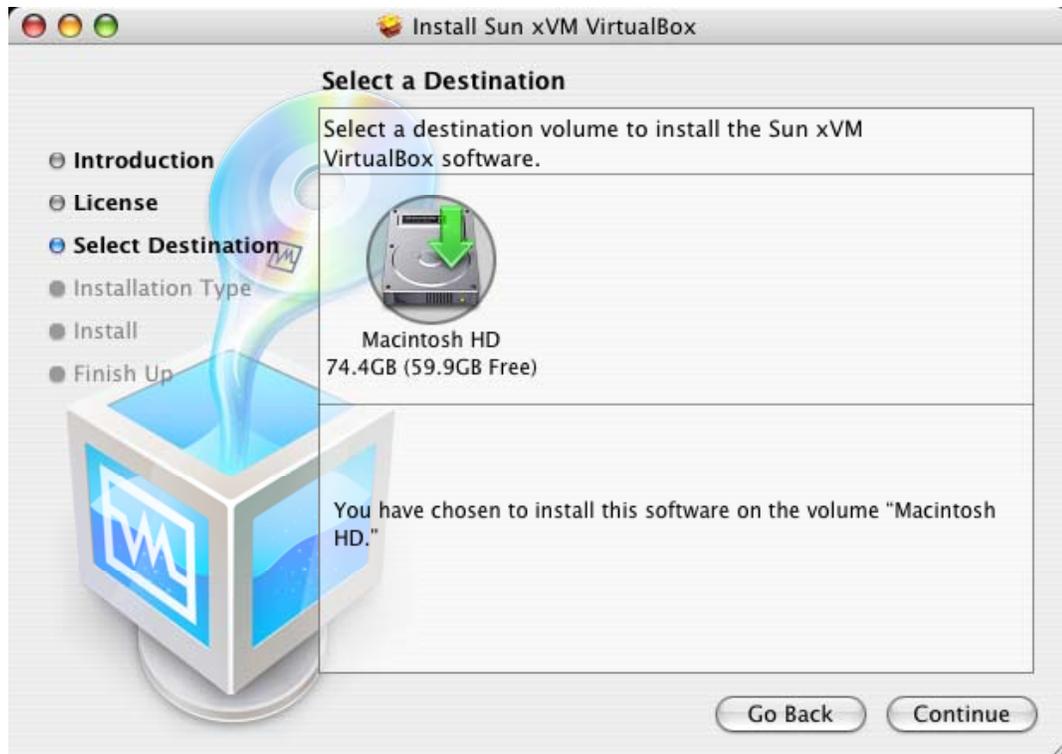


Figure 29

Select your hard drive and press **Continue**

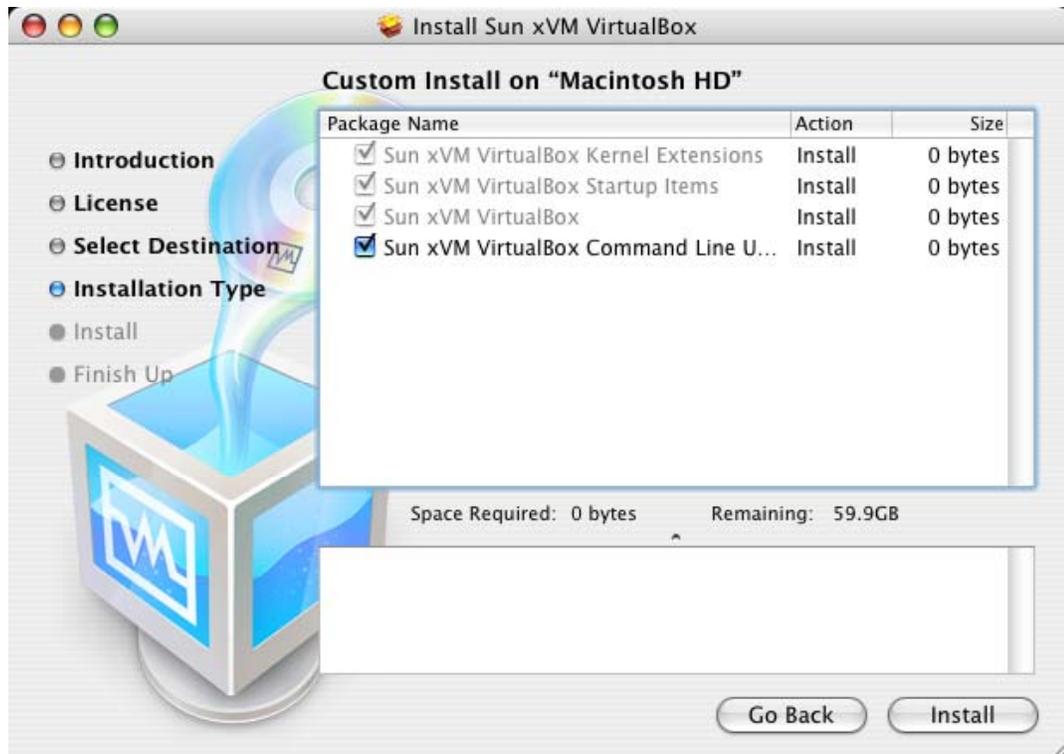


Figure 30

Take the defaults and press **Install**



Figure 31

Enter your administrator's name and password. And then press **OK**.

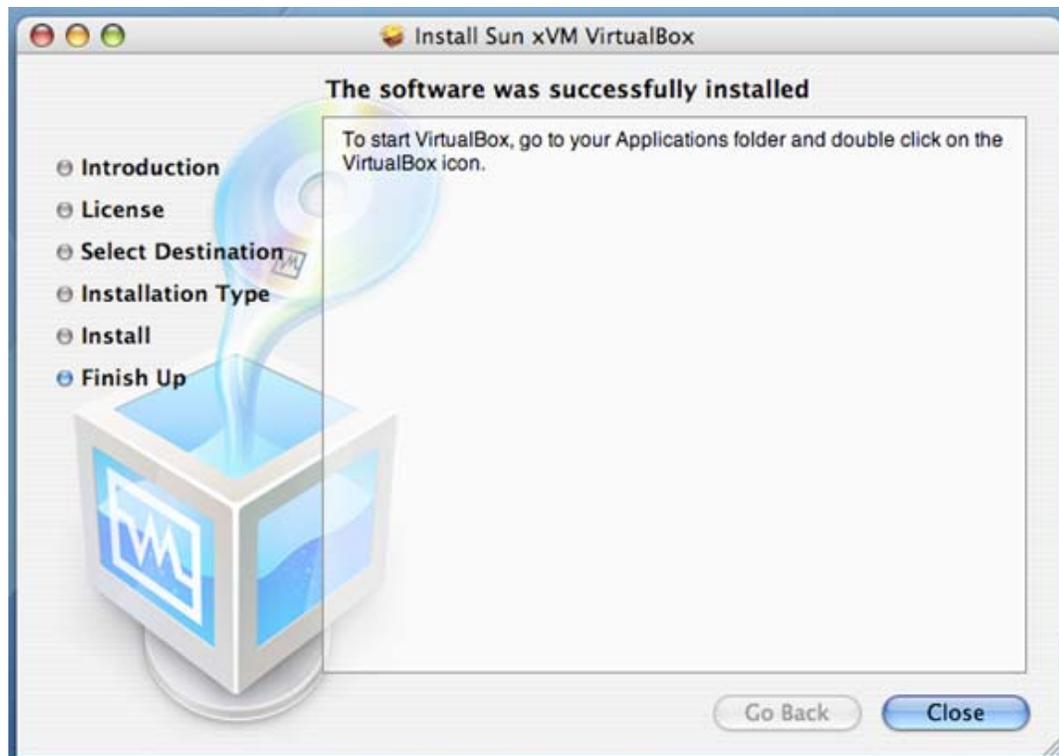


Figure 32

Press **Close** to exit the install.

Return to the installation menu, Figure 13 and select option #2. This will copy the Ubuntu client to your compute. Note: It may take 10 minutes or more for the copy to finish. Please have patience.

When the copy concludes, return to the Applications window, as shown below, and double-click on the Virtual Box icon.



Figure 33

Go to the **Applications** menu to launch VirtualBox and continue with the next section (*Step c 'Installing the Ubuntu Client'*.)

Step c: Installing the Ubuntu Client

You are now ready to start the VirtualBox Console and install the Ubuntu Linux client on your virtual machine. Although we have used Windows Vista screenshots to illustrate the installation, the steps are the same on Windows XP and Mac OS X.

On a Windows PC, start VirtualBox by clicking on its icon.

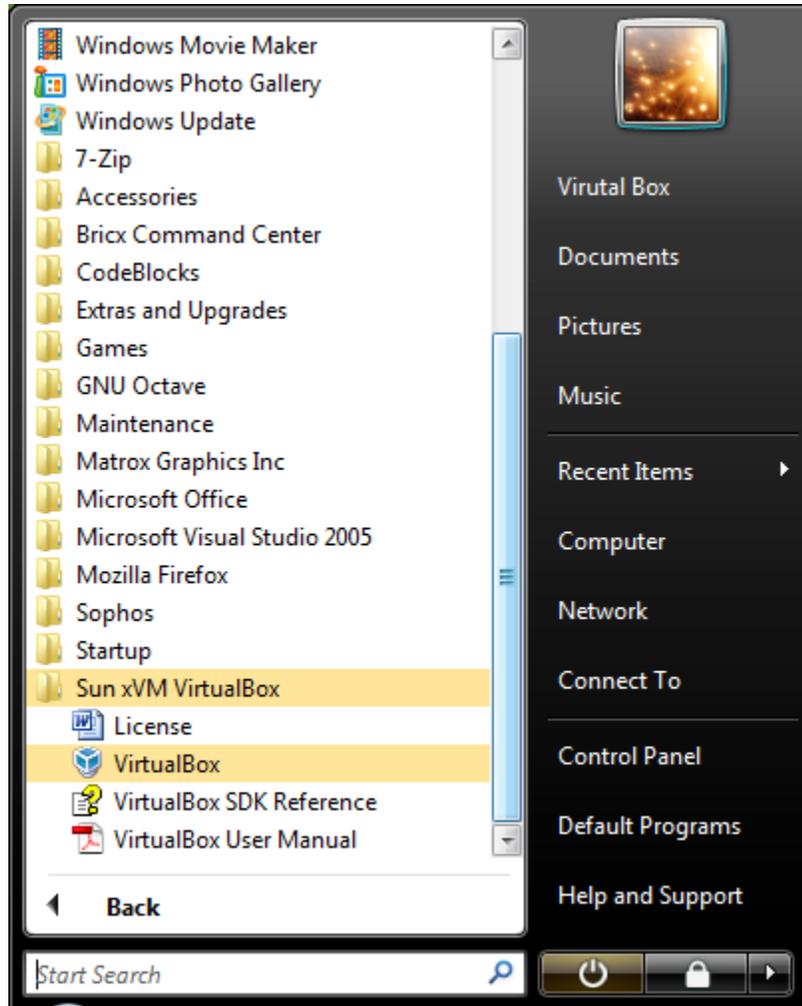


Figure 34

On a Mac OS X double-click on the VirtualBox icon in the Applications window (Figure 33.)

Follow the steps on the next page.



Figure 35

You may register VirtualBox if you so desire. If you prefer not to, you may click the **red X** to close the VirtualBox registration dialog window.

You need to add a Virtual Disk to VirtualBox, so the system you will be building has a disk with existing data on it. The installation DVD contains a virtual disk containing the Ubuntu Operating System with support for CodeBlocks and BrickOS. After you installed VirtualBox, the Ubuntu disk was copied to your system.

If you are on the Mac, make sure that you have run the *copy vdi* option from the *menu.command* file.

We now wish to setup Virtual Box to use the virtual disk which you have copied from the DVD to your computer.

Select *File/Virtual Disk Manager* from the VirtualBox console, as shown below.

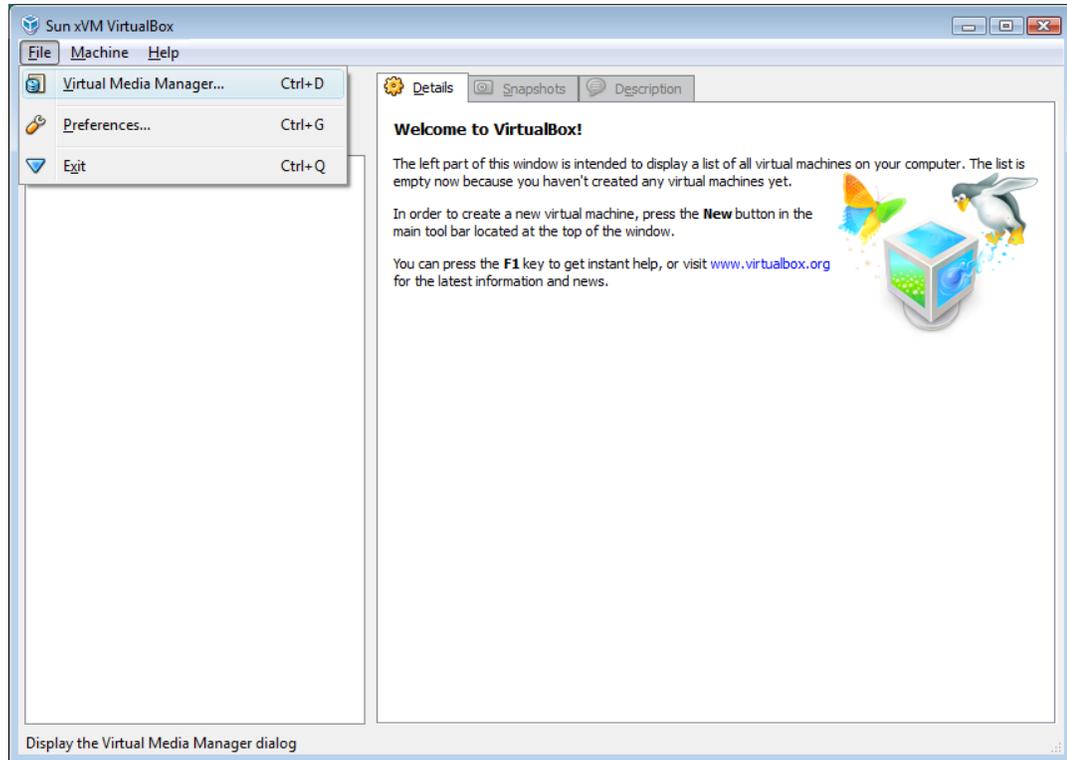


Figure 36

Select the **Add** icon (second on the left) as shown below.

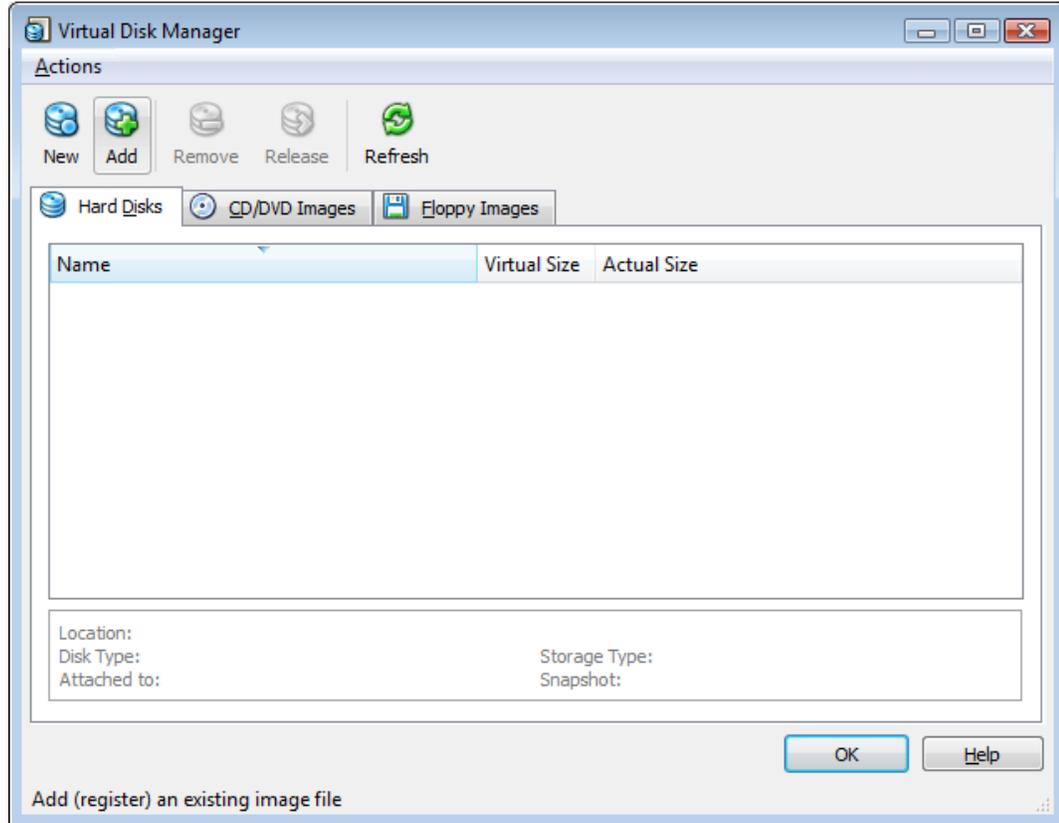


Figure 37

Double-click on VDI (as shown in Figure 38)

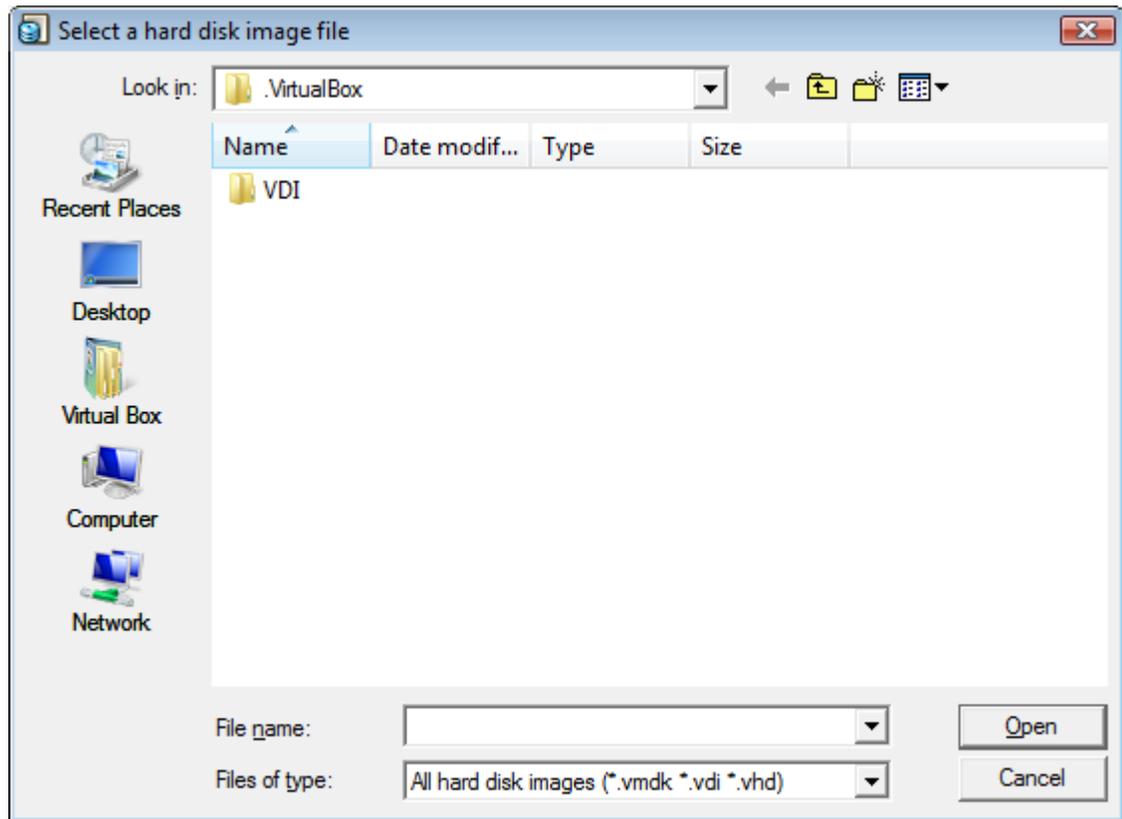


Figure 38

Select the **Ubuntu** virtual disk which has been placed in the VDI directory of the Virtual Box settings, as shown below (Figure 39) and press “*Open*”).

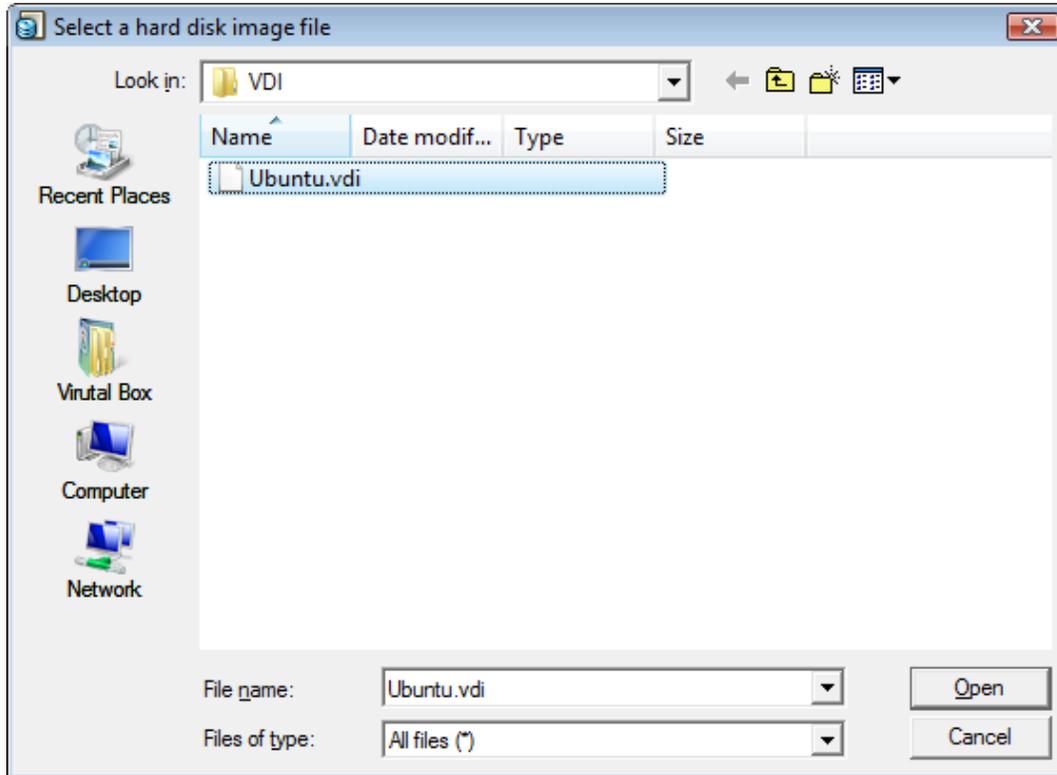


Figure 39

Now select the Ubuntu disk and press “OK” (Figure 40).

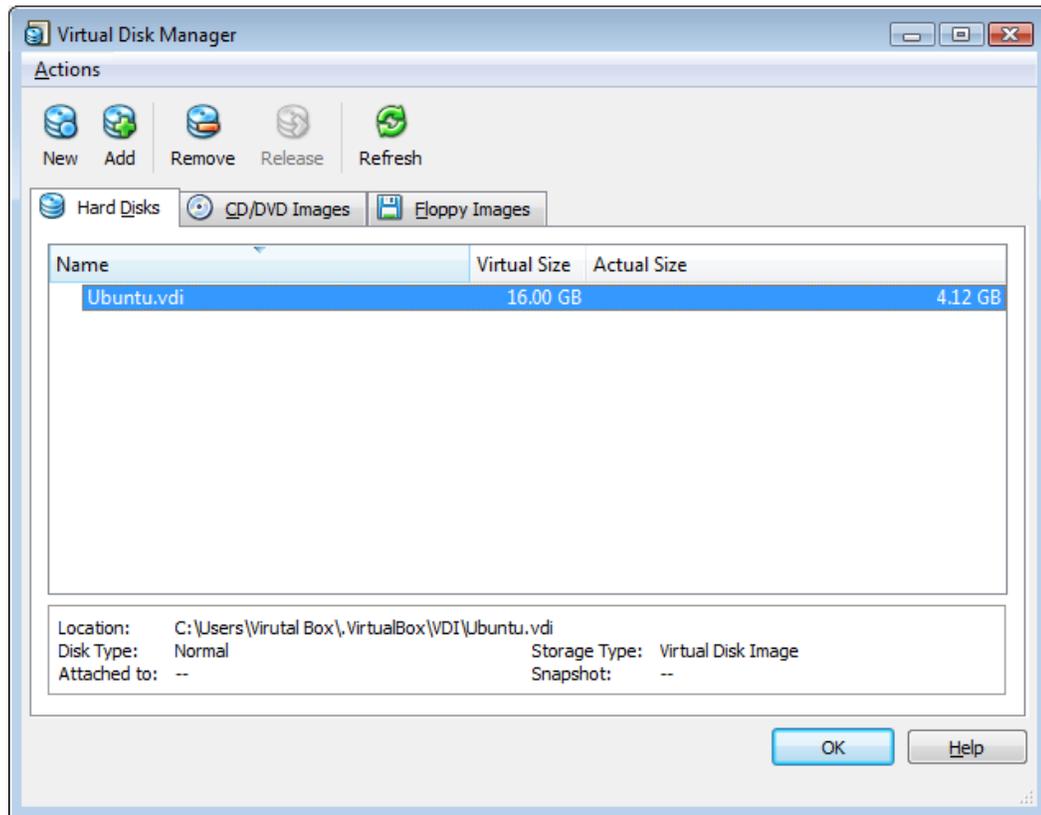


Figure 40

Create a new Virtual Machine by selecting the **New** button in the upper left hand corner of the VirtualBox console screen, as shown below.

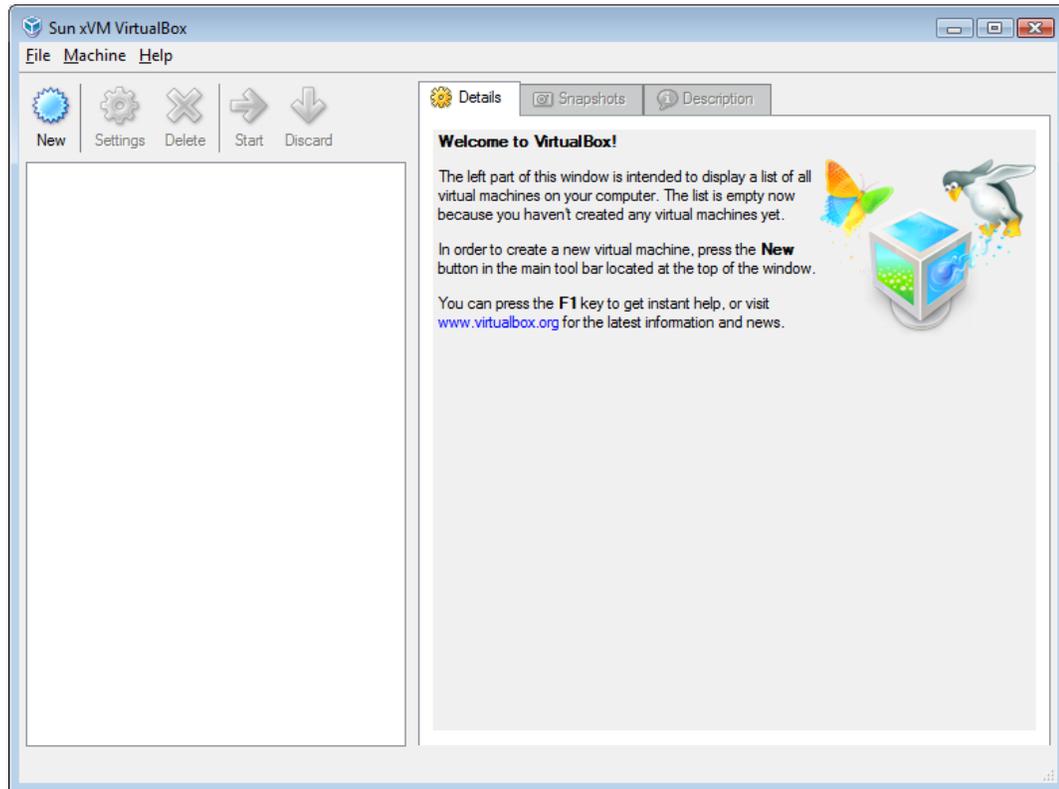


Figure 41

The *New Virtual Machine Wizard* will now begin. Follow the steps as shown below:

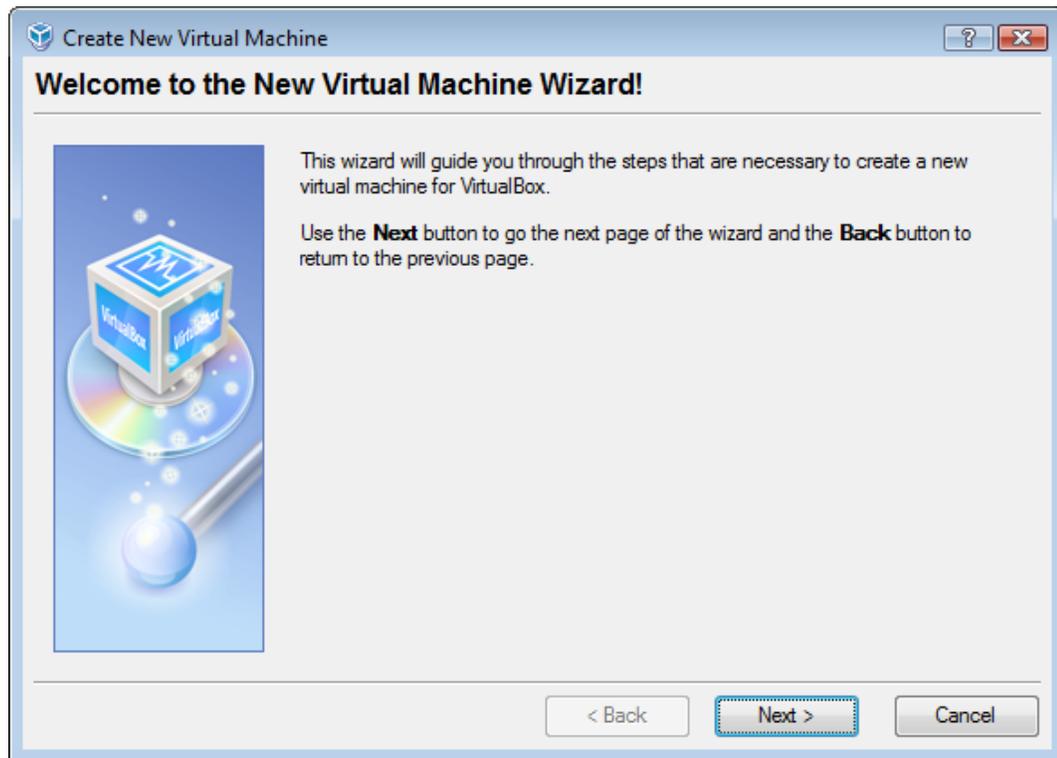


Figure 42

Select **Next** to proceed.

Give a name and OS type.

- Select *Ubuntu* for the name.
- Select *Linux* for the Operating System
- Select *Ubuntu* for the Version

Press **Next** as shown below.

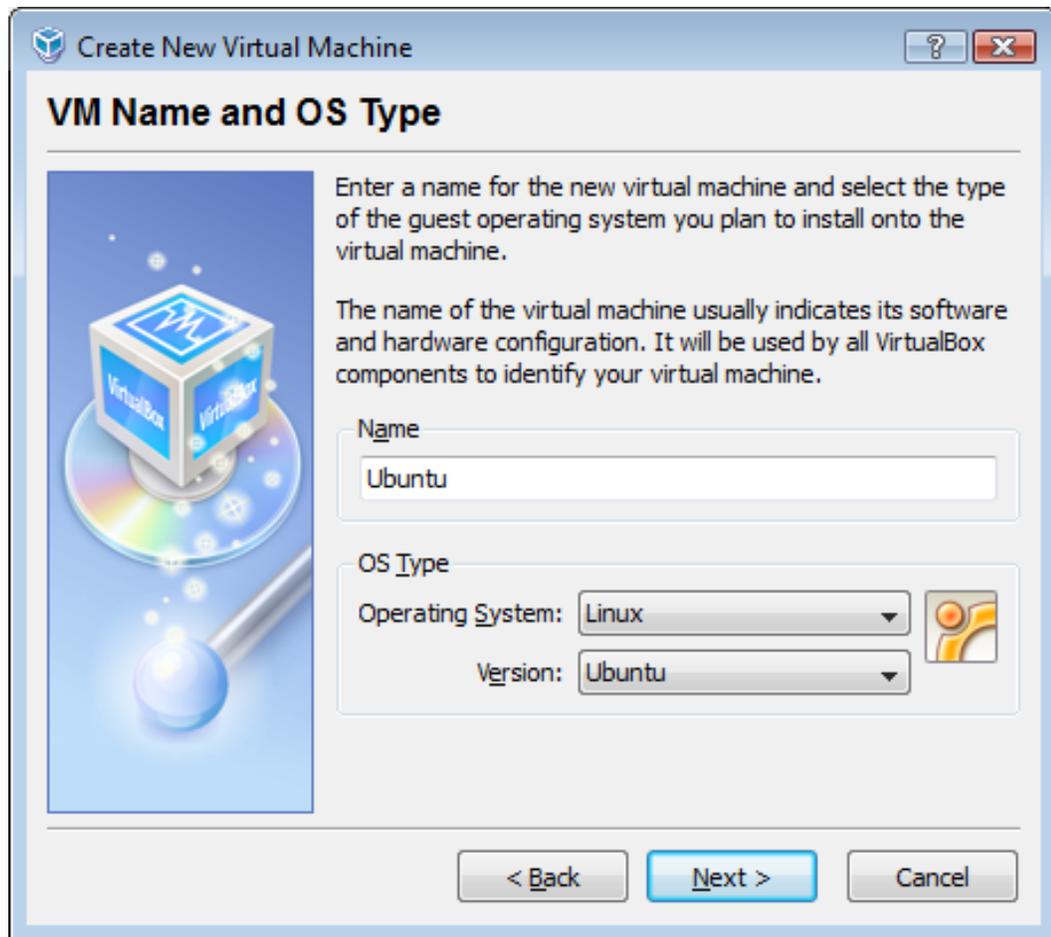


Figure 43

In the next step of the wizard, we are going to “borrow” some memory from our real machine for the Ubuntu Virtual machine. Whenever the virtual machine is running, this memory will not be available to the host computer.

Select **512 MB** for the amount of RAM for the virtual machine. If you have only 512 of real RAM in your system (which incidentally is quite low), select **256 MB** to be used for the virtual machine. Although **384 MB** is the wizard's recommendation, **512 MB** is optimum for running the Ubuntu client. See the screenshot below.

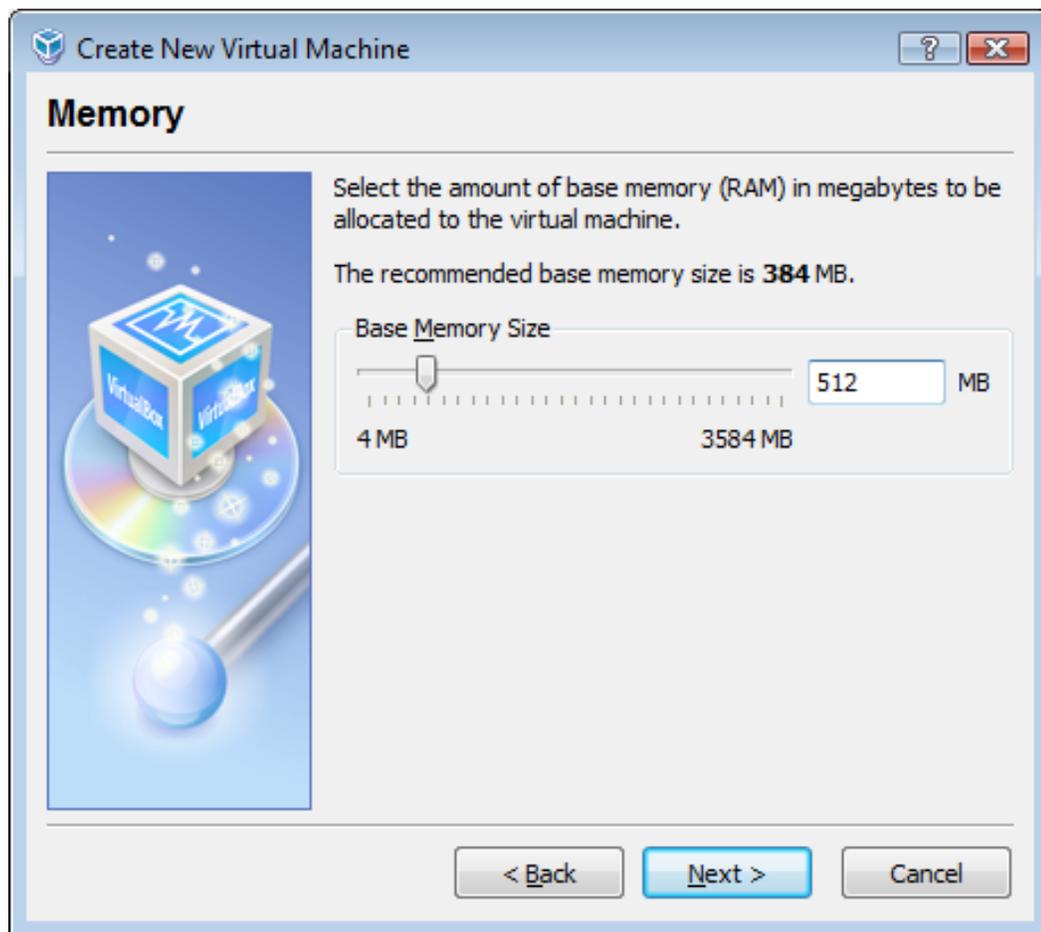


Figure 44

Select **Next** to proceed.

Select **Ubuntu.vdi**, and press Next to continue, as shown below.

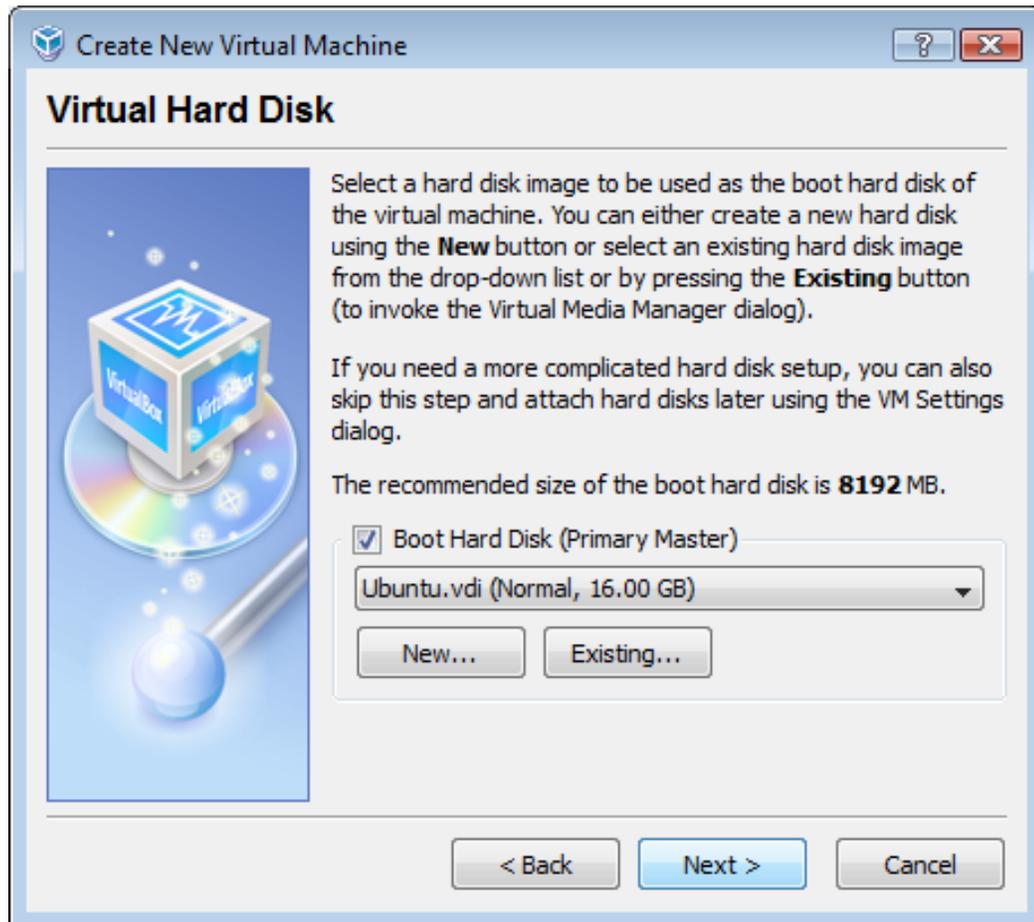


Figure 45

You will now be shown a *Summary* screen Click **Finish**.

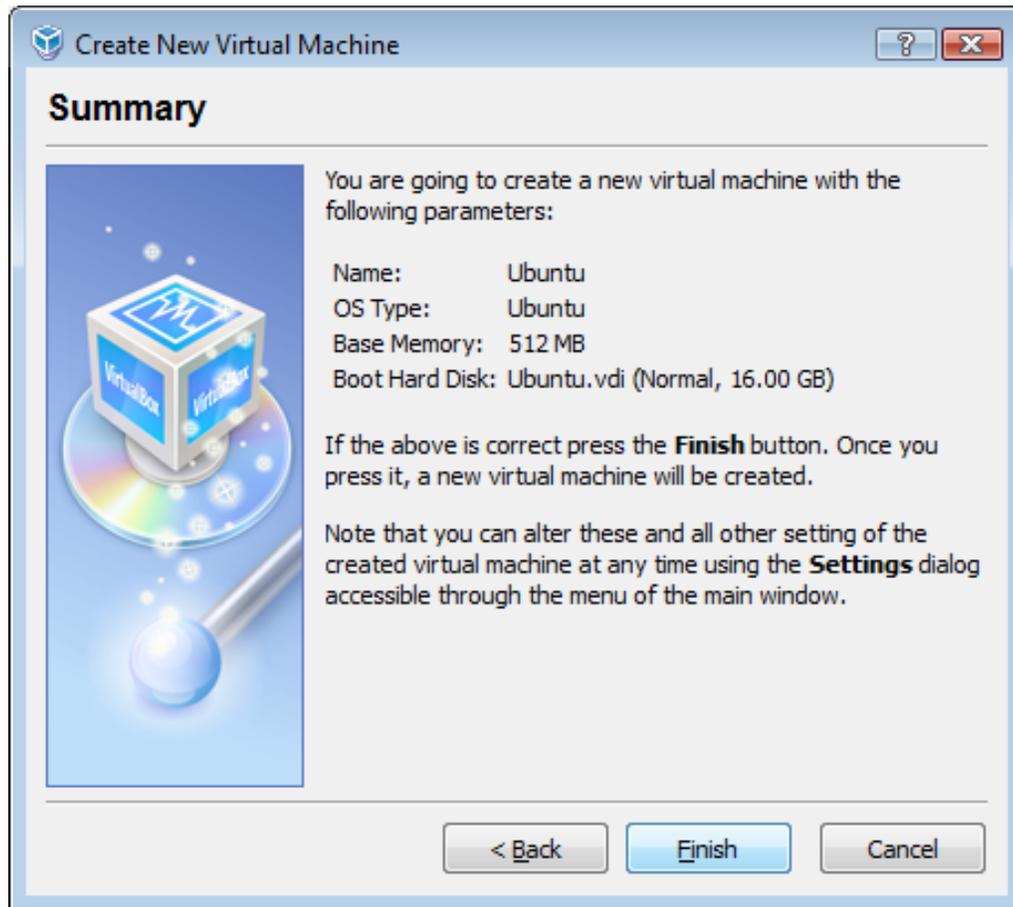


Figure 46

Congratulations! You have now completed the installation of VirtualBox and the Ubuntu client.

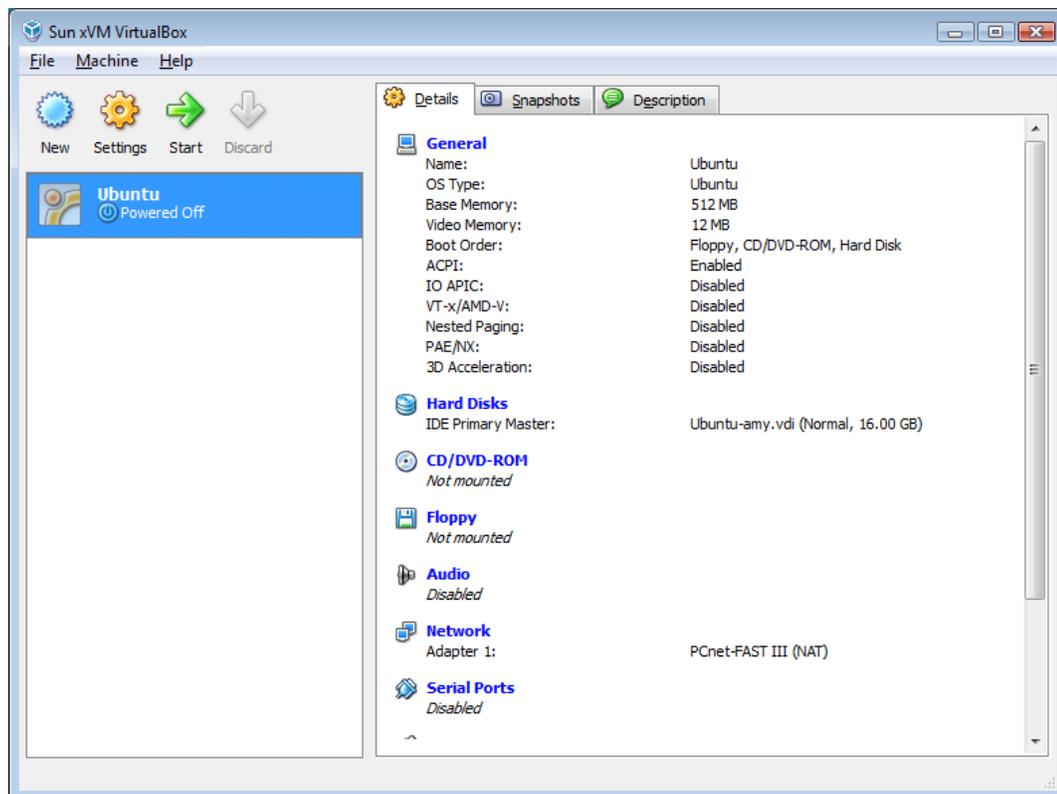


Figure 47

You have successfully built a system for Ubuntu!

- Select “*Start*” (the green arrow) to begin using the system.
- To login to the system, the username and password are both: *student*. (The root password is also *student*.)
- As you run a virtual machine, you may get various message windows from VirtualBox. If you wish you may check the box to disable those messages.

6.3. How to Uninstall the Software

To uninstall from Windows go to the *Control Panel* and then choose *Add/Remove Programs*. From the choices that appear select *remove VirtualBox*.

To uninstall from Macs, run the installer again and then double click on the *Uninstall* icon.

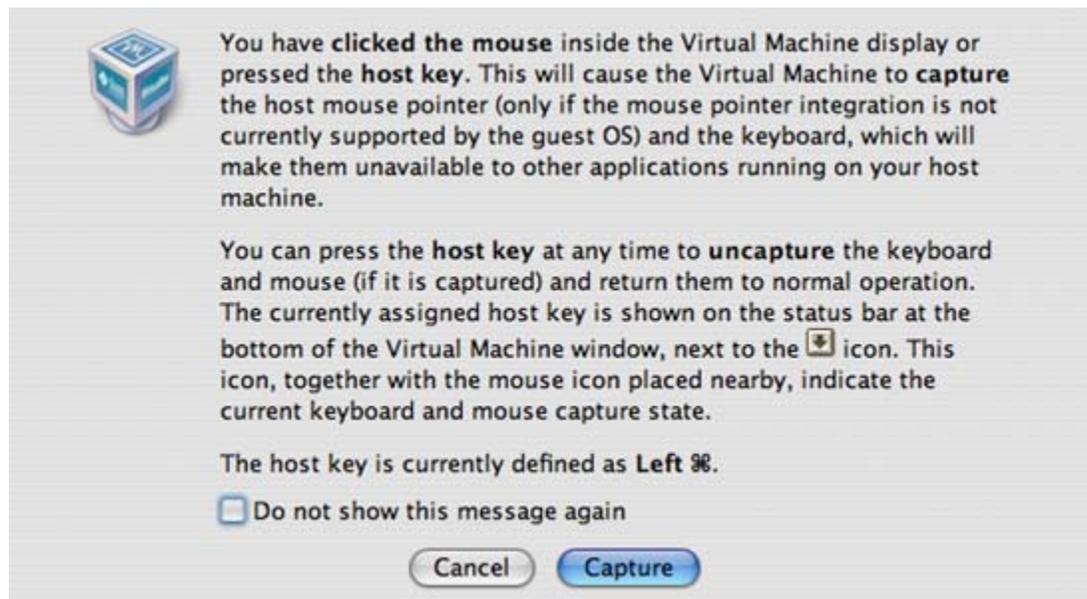
In both cases, find the location of the *.vdi* file and then delete it.

Section 7: First Use

Once you have completed the installation, you are ready to begin programming for the Lego Robot. In the following examples we will use screen shots from a Windows XP host, however, the first screen shots for those using a Mac OS X host will be comparable. Once you begin using the Ubuntu Linux guest, the screen shots will be identical on all systems.

Special note for Max OS X users of VirtualBox

As you are no doubt aware you on a single button Mac you ‘*right-click*’ by holding down the *Apple key* and simultaneously pressing the mouse button. However, while in VirtualBox, the *Apple key* is considered to be a special key. In order not to be annoyed by the following screen:



Check the “*Do not show this message again*” option.

1. Start the VirtualBox software (click on the VirtualBox icon on your *desktop* or on the menu). The resulting window is shown below (Figure 48).



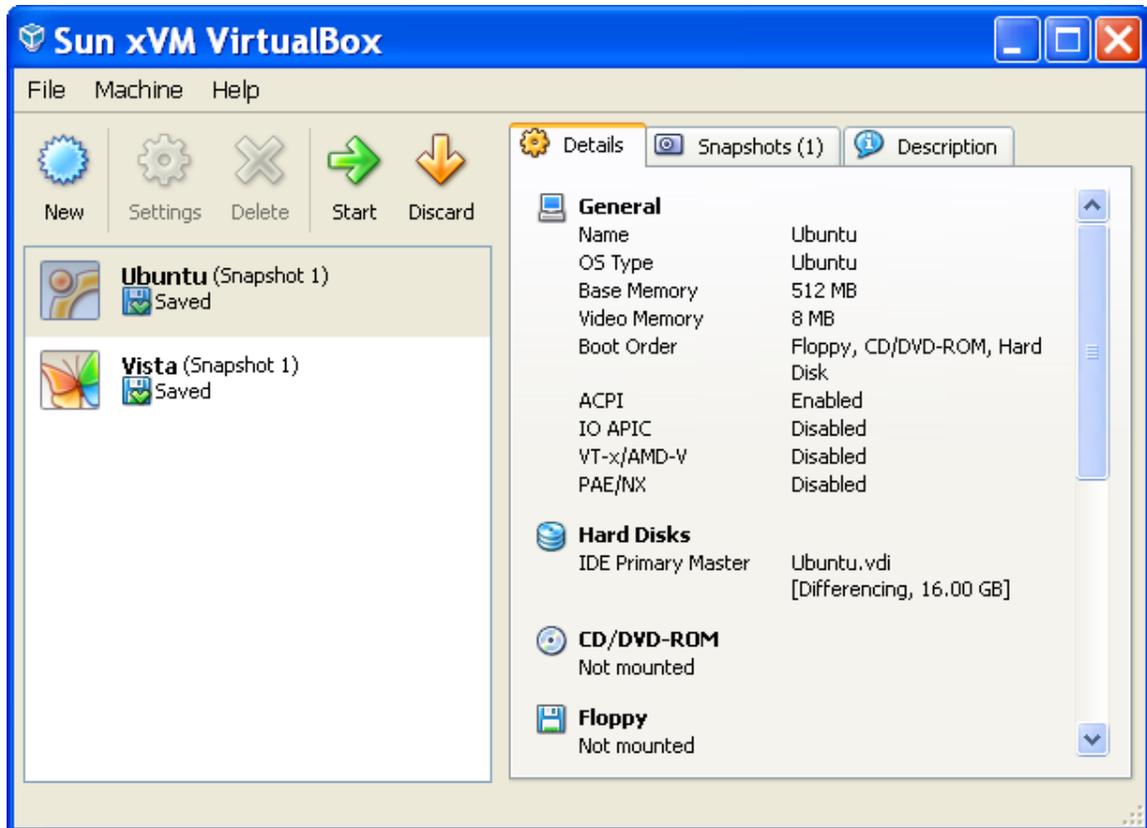


Figure 48 - Sun VirtualBox console

Your screen may appear somewhat differently, depending on the configuration of your hardware and the number of guest operating systems installed.

2. Start the Ubuntu Linux guest by clicking on the appropriately named icon in the console. After some time you will be asked to enter the username and your password. These have both been set to 'student' (all lowercase). After some additional time and appropriate sound effects you will be confronted with the Ubuntu Linux desktop, shown below (Figure 49).

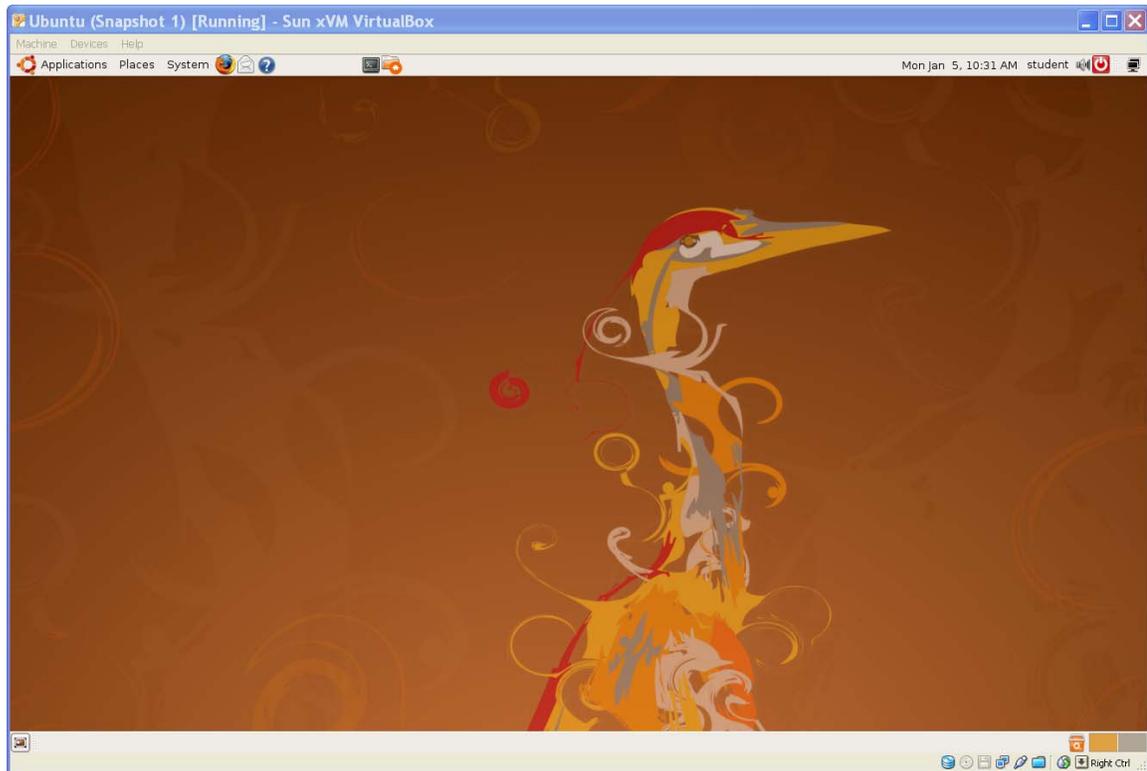


Figure 49 - Ubuntu Linux Desktop

- You may at this point open the window full screen since from this point on we will be working only within Ubuntu. (Of course you may adjust the window to whatever size is convenient in order for you to work in the host and the guest simultaneously. Isn't virtualization wonderful?!)
3. Start the Code::Blocks IDE. (Using the mouse click on *Applications/Programming/CodeBlocks IDE*)
 4. Start the BrickEMU (Using the mouse click on *Applications/Programming/Brick Emulator*)

5. Adjust the windows so that both the emulator and the CodeBlocks IDE are visible. Your screen should now look like the following (Figure 50).

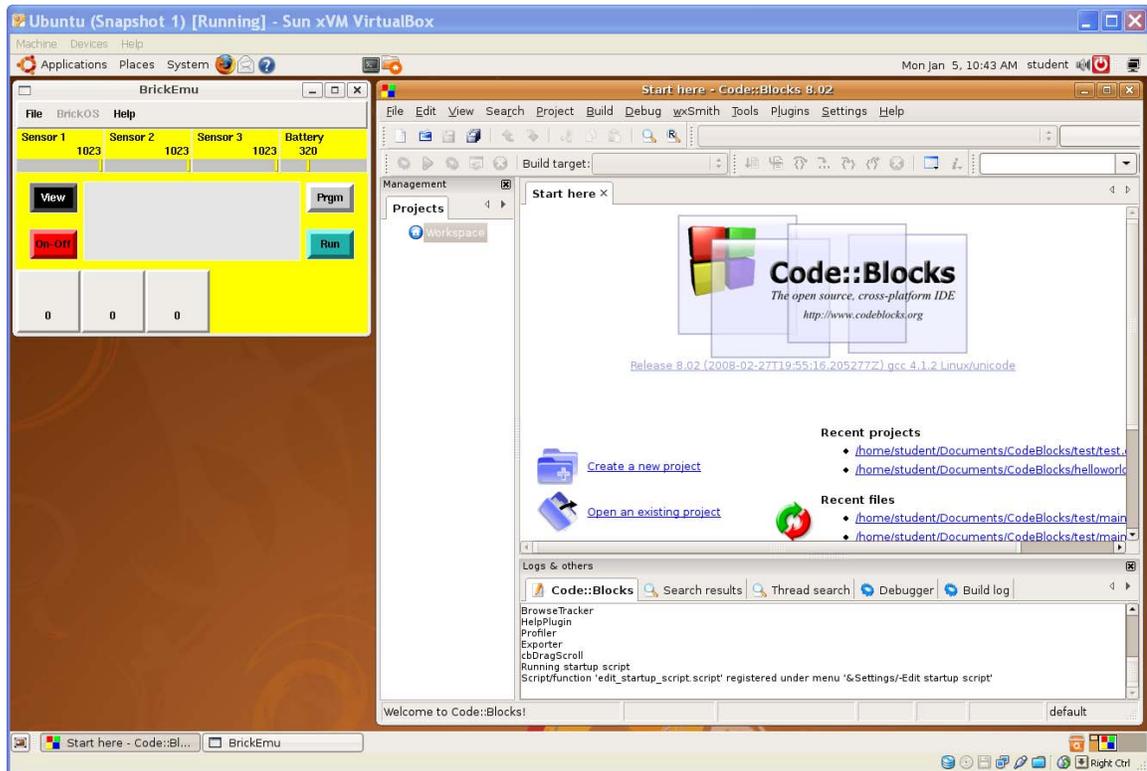


Figure 50 - Ubuntu Desktop with the CodeBlocks and BrickEMU applications

Press the *Run* button on the emulator. You should see the word *None* flash briefly on the LCD screen. This indicates that you have not loaded any programs into the emulator. We will do this in Section 10.

Section 8: The Ubuntu Linux Desktop

Learning to use Linux is an essential component of the education of the programmer. A detailed introduction to Linux is beyond the scope of this manual. Students wishing to learn more about Linux should consult one of the many excellent textbooks and website that are devoted to Linux in general and its Ubuntu variant in particular. In the Appendix we have presented several sources to get you started.

Having said the above, a detailed knowledge of Linux is not necessary for you to program the Lego Robot. In this section we take a brief look at the Ubuntu Linux desktop. In many ways it is similar to the desktop used in Windows XP/Vista or Apple Mac OS X. You select items using a *left-mouse click* and you use *right-mouse click* to open a context sensitive menu from which further selections may be made.

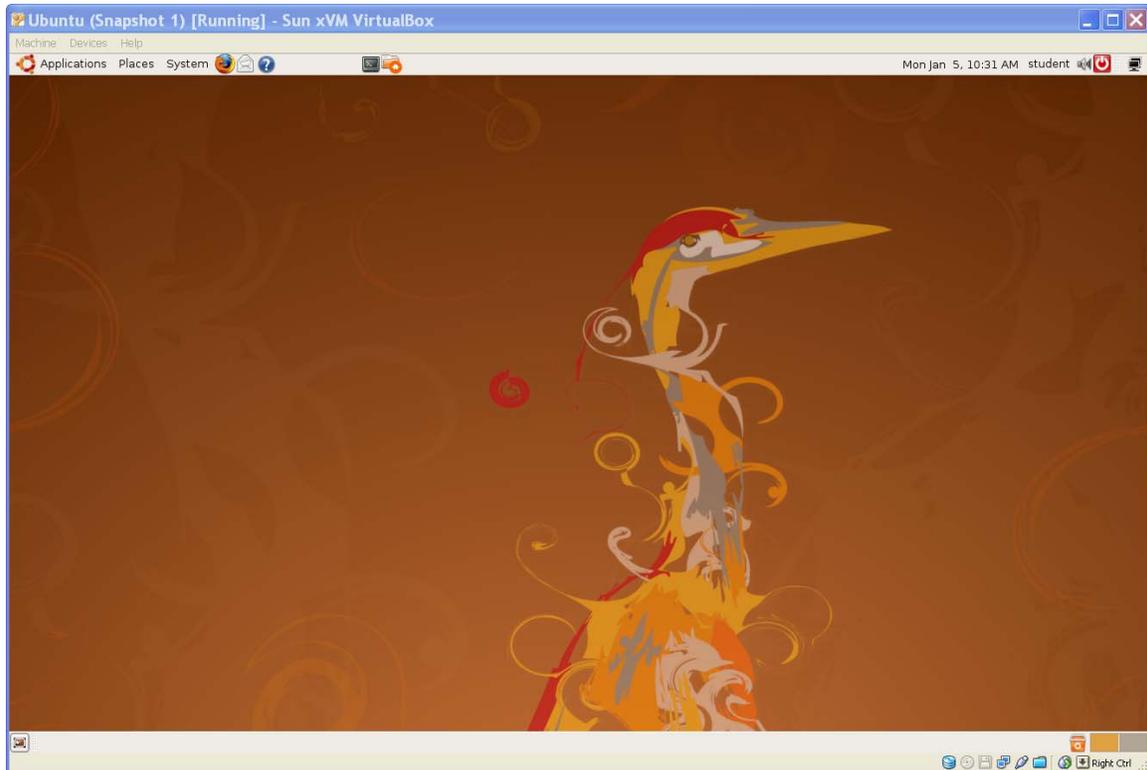


Figure 51 - The Ubuntu Desktop

The desktop is made up of several panels. At the top of the desktop you will see:



Figure 52 - Ubuntu Desktop Top Panel

From left to right we find:

- *Applications* menu – providing access to the programs installed on your computer
- *Places* menu – providing access to various locations on your computer including files, standard folders, the network as well as system drives.
- *Systems* menu – providing options for configuration of your preferences, system administration and help.
- The middle section contains several program icons allowing access to commonly used applications. You may add additional applications by right-clicking on the panel.
- The right side of the panel contains the current user, a volume control, a network status icon, the date and time, and the power button.
- When updates are available an additional icon appears. Click on that icon and a wizard will guide you through the update process.

Items on the panel may be rearranged and additional items may be added.

At the bottom of the desktop you will see:

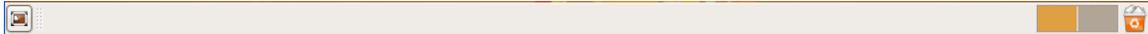


Figure 53 - Ubuntu Desktop Bottom Panel

- The icon on the far left is used to hide and show all windows on the desktop.
- Ubuntu allows the user to have virtual desktops (as if you would have multiple monitors). The squares display, in miniature, the contents of each desktop. You may move back and forth between virtual desktops by clicking the appropriate square.
- At the far right, appears the ubiquitous trash bin.
- As applications are opened, a button will appear for that application on the center portion of this panel.

The central portion of the screen contains your wallpaper (which may be changed) as well as any open application windows.

Additional information about the desktop can be found here:

http://www.techotopia.com/index.php/Ubuntu_Desktop_Essentials . For our purposes all interaction with the Ubuntu Linux system will take place using the graphical user interface described above. Advanced Linux users will want to take advantage of the linux *terminal*, shown below. The terminal may be started by selecting *Applications/Accessories/Terminal* from the menu on the top panel. Many powerful and complex commands can be issued

using the terminal. However, this lies well beyond the scope of this manual. The interested student is urged to consult the resources in Appendix I.

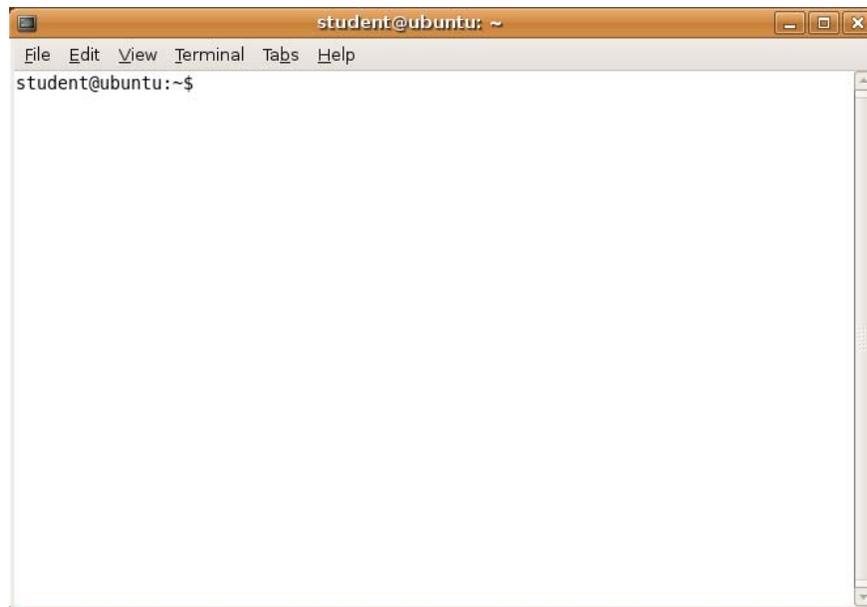


Figure 54 - The Ubuntu Terminal Application

Section 9: How to Shutdown the Ubuntu Client and the VirtualBox Console

1. The Ubuntu guest may be shutdown in one of two ways.

Method 1 - Press the Ubuntu *Shutdown* button, followed by either *Shutdown* (completely shuts down the Ubuntu virtual machine) or *Hibernate* (saves the current state of the machine before shutting down) on the screen which subsequently appears, *Figure 55*.

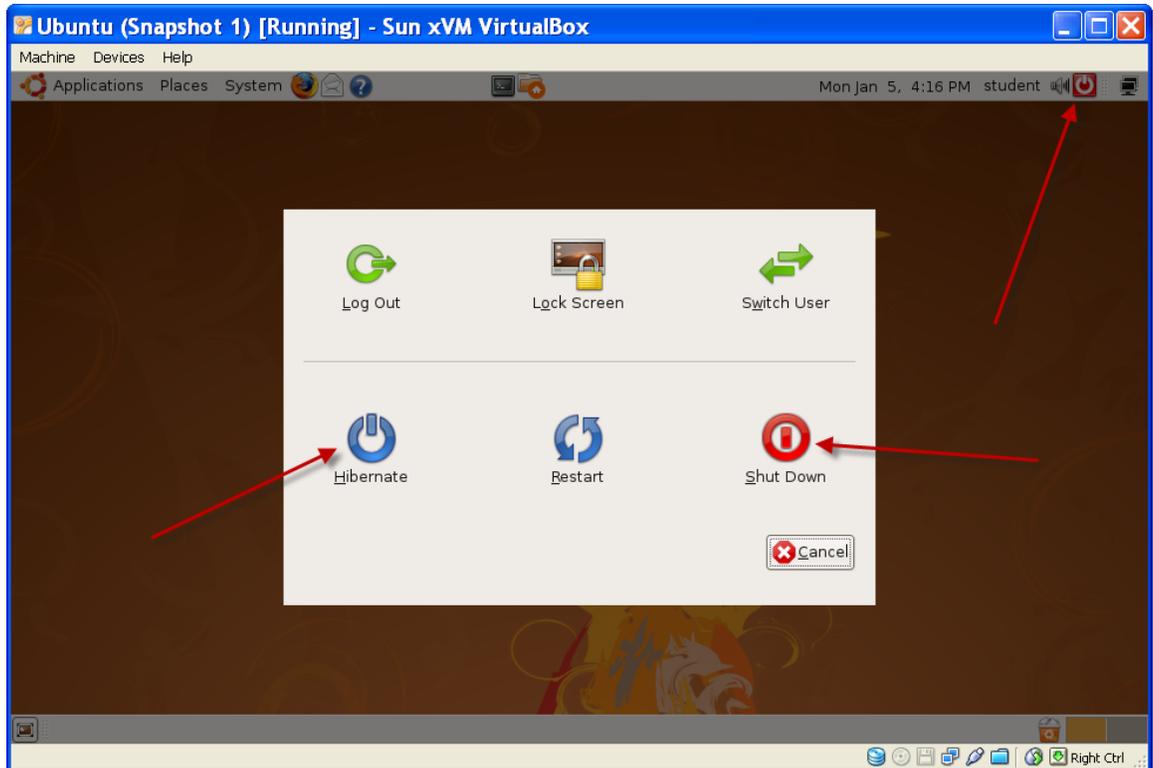


Figure 55 - Shutting down Ubuntu (Method 1)

Method 2 – Press the VirtualBox Windows Close button, followed by OK in the dialogue that follows, Figure 56. This will shut down the Ubuntu client, saving its state. The next time you start the Ubuntu client, you will find the virtual machine in the same state as it was when you last shut it down.

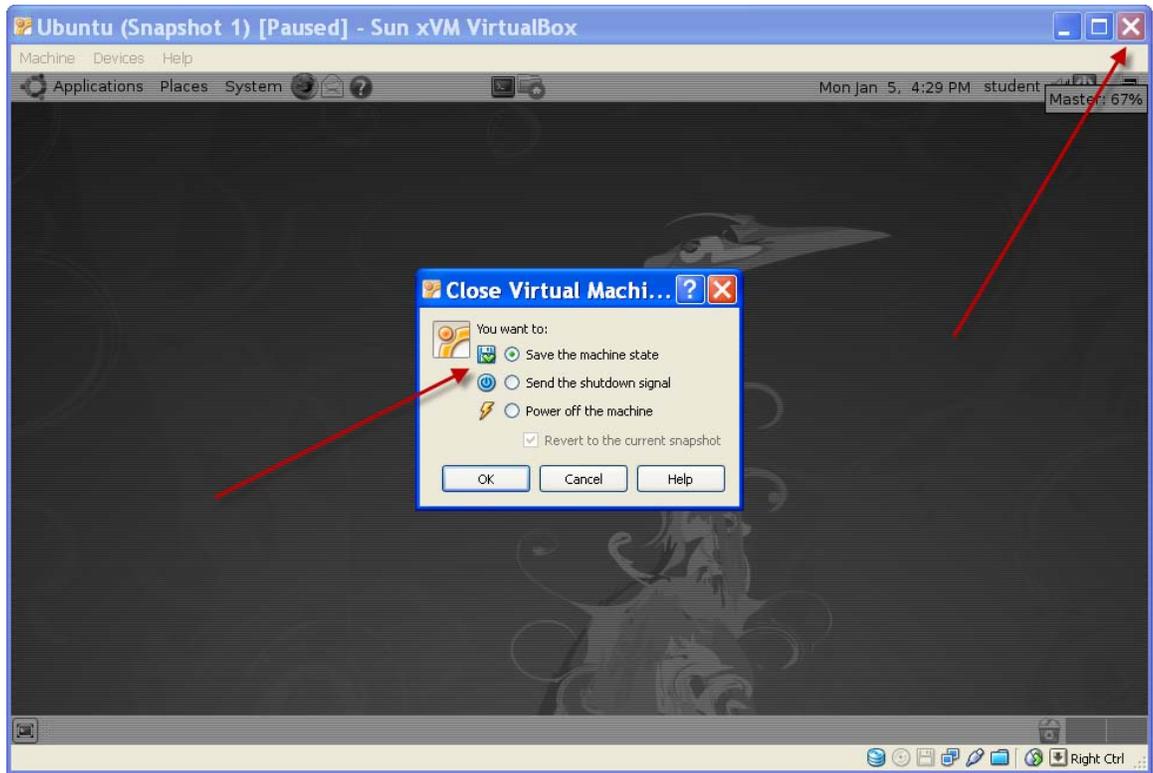


Figure 56 - Shutting down Ubuntu (Method 2)

2. Finally close the VirtualBox console.

Section 10: First BrickOS Program

In this section, we will go thru the steps involved in writing a complete C++ program using BrickOS for the Lego Robot. We will first test our program on the emulator and then (for those who have an actual Lego RCS Robot, we will transfer the program to Robot. Programming for the Robot is very similar to ordinary programming in C++. You should already be familiar with the Code::Blocks IDE. (You may wish to refresh your memory by reviewing the material in the Code::Blocks manual available here: <http://www.sci.brooklyn.cuny.edu/~goetz/codeblocks/codeblocks-instructions.pdf>)

As is traditional, we will write a simple program, called *HelloRobotWorld* to illustrate the basic features of brick programming.

1. Create a new project called *HelloRobotWorld*. Either choose the icon from the Code::Blocks IDE opening screen or select *File/New/Project* from the menu.



2. From the resulting screen (Figure 57) select *BrickOS*



Figure 57 - Code::Blocks Project Template menu

3. After the usual sequence of Code::Blocks menus, type in the following program as the *main.cpp*:

```

/*  Name: HelloRobotWorld
    Description: A simple program that illustrates the
                features of BrickOS. The program will:
                1. Display "Hello World" on the LCS
                2. Roll foward and then back
                3. Wait until the either Sensor 1 or
                   Sensor 2 is "touched"
*/

// Include files containing the BrickOS library functions
#include<conio.h>
#include<dmotor.h>
#include<dsensor.h>
#include<unistd.h>

using namespace std;

// function prototypes
void hello();
void world();
void roll();

int main() {

    hello();           // Display "Hello"
    sleep(5);         // Pause for 5 seconds
    world();           // Display "World"
    sleep(5);         // Pause for 5 seconds
    roll();            // Roll forward and backwards

    // Display "HUH" until such time as either sensor 1
    // or sensor 2 is activated. Display "OUCH" whenever
    // sensor 2 is activated. Display "BYE" and end the
    // program when sensor 1 is activated.
    while (1) {
        if (TOUCH_2)
            cputs("OUCH ");
        else if (TOUCH_1) {
            cputs("BYE ");
            break;
        } else cputs("Huh ");
    } // end while

    return 0;
}

```

```
// function hello displays "HELLO" on the LCD
void hello() {
    cputs("HELLO");
}

// function world displays "WORLD" on the LCD
void world() {
    cputs("WORLD");
}

// function roll instructs the robot to roll forward
// and then to reverse
void roll() {
    motor_a_dir(fwd);
    motor_a_speed(200);
    motor_c_dir(fwd);
    motor_c_speed(200);
    sleep(5);
    motor_a_dir(rev);
    motor_a_speed(200);
    motor_c_dir(rev);
    motor_c_speed(200);
    sleep(5);
    motor_a_dir(off);
    motor_c_dir(off);
}
```

4. Use the Code::Blocks IDE to cross compile the program for the Lego Brick. Select *Build/Compile Current File*. If your program contains no errors your screen should look like Figure 58 at the end of the *Build/Compile* process.

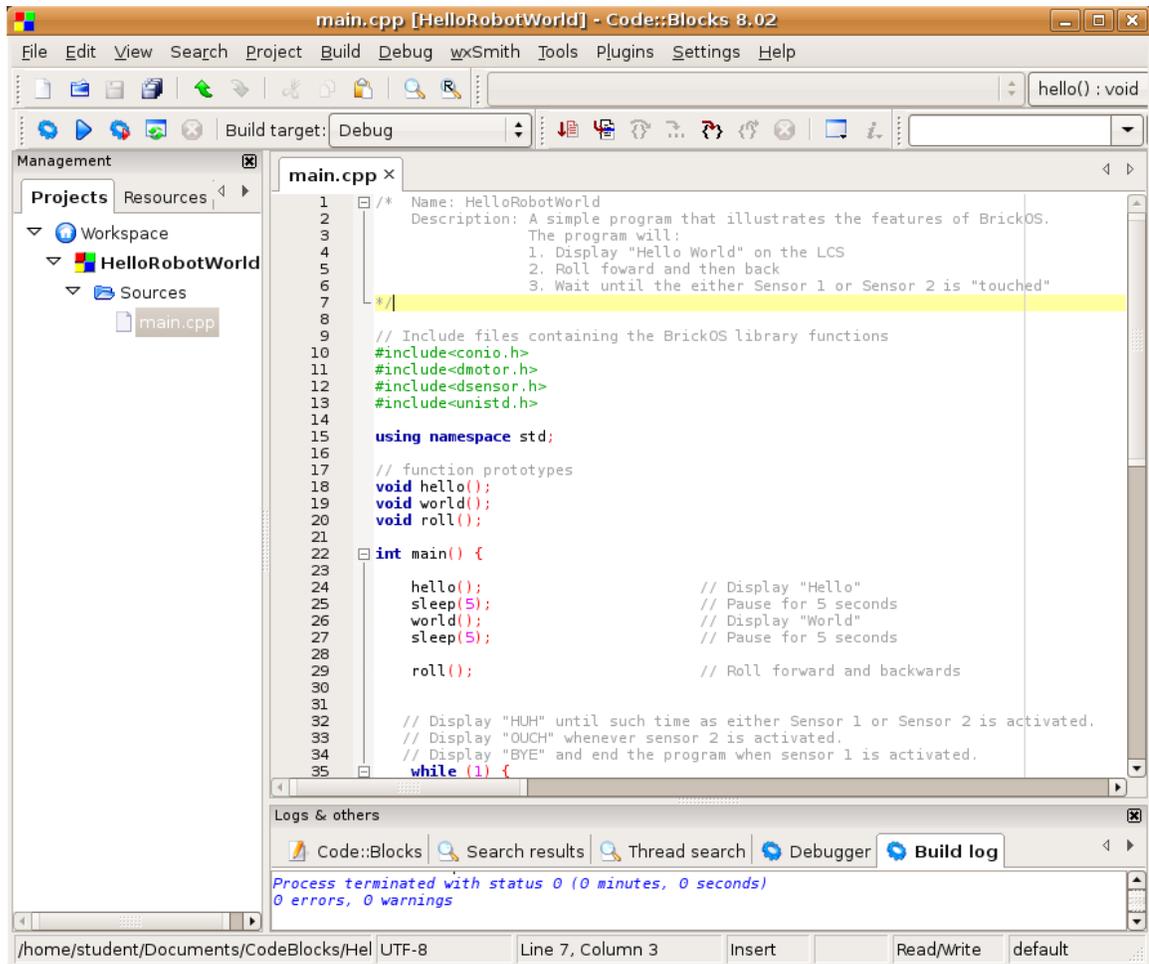


Figure 58 - Code::Blocks indicating a successful compilation

Be sure not to *Build* (or *Build and Run*) the program (as you might do during ordinary C++ program development. The program we have written is designed to run on the Brick, not on the computer. Building your code in the usual fashion would result in object code and executable code designed for the computer that Code::Blocks is running on and not the computer residing in the Brick. This is certainly not what we want.

Assuming your source code compiled correctly for the Brick emulator, you will find a file called *main.lx* in your *HelloRobotWorld* project directory. This is the file that needs to be transferred to the BrickEMU and will run on the emulated computer. This transfer is shown in the next step.

5. Make sure the BrickEMU is on (press the On-Off button if necessary and that the BrickOS firmware has been loaded as described in Steps 6 and 7 on page 83.
6. Transfer your program to the BrickEMU. Select *Tools/RCX Transfer Emulator*. If all goes well you will see the following:

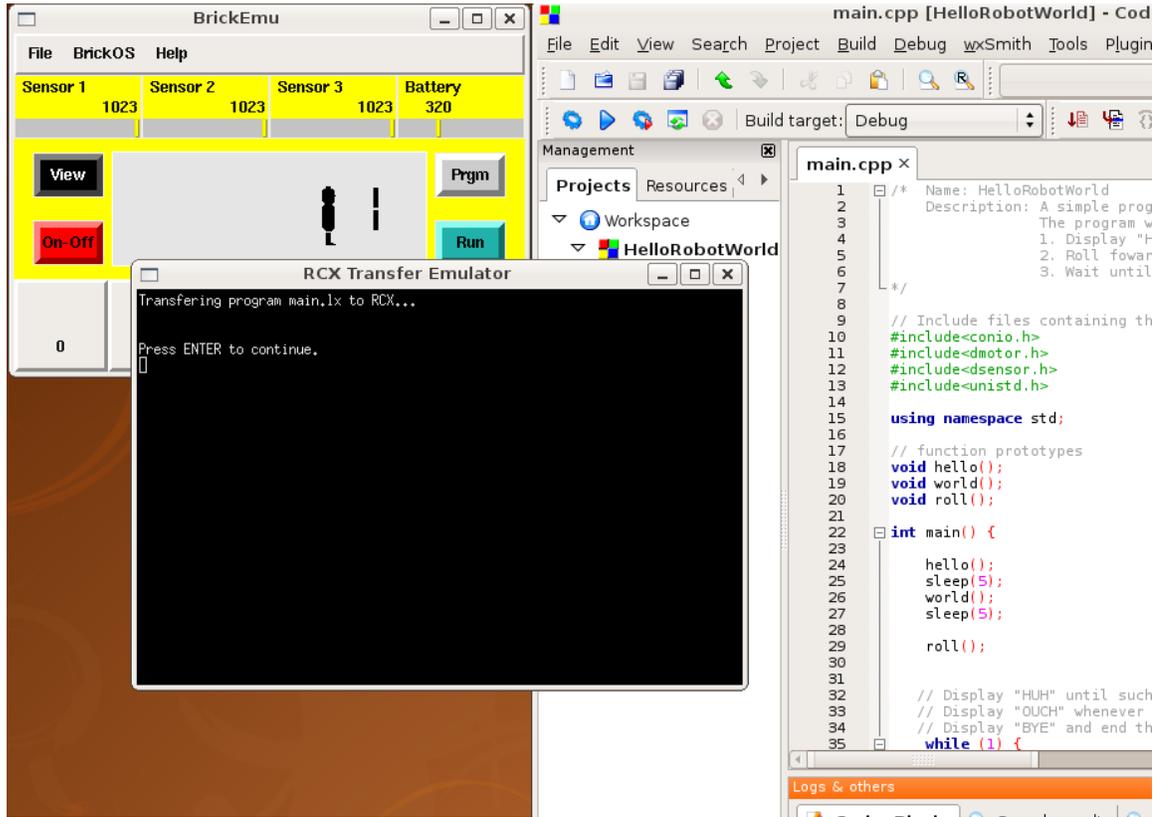


Figure 59 - Program successfully transferred to the BrickEMU

6. Press *Enter*

7. Run your Robot by pressing the *Run* button on the Brick EMU. Your program should display “Hello” – pause for 5 seconds – and then display “World”. After another 5 seconds the motor activity buttons will activate indicating first activity in the forward direction (see the numbers and arrows on Motor Port 1 and Motor Port 3) followed by activity in the reverse direction. After another 5 second pause the LCD will display “Huh” (Figure 60).

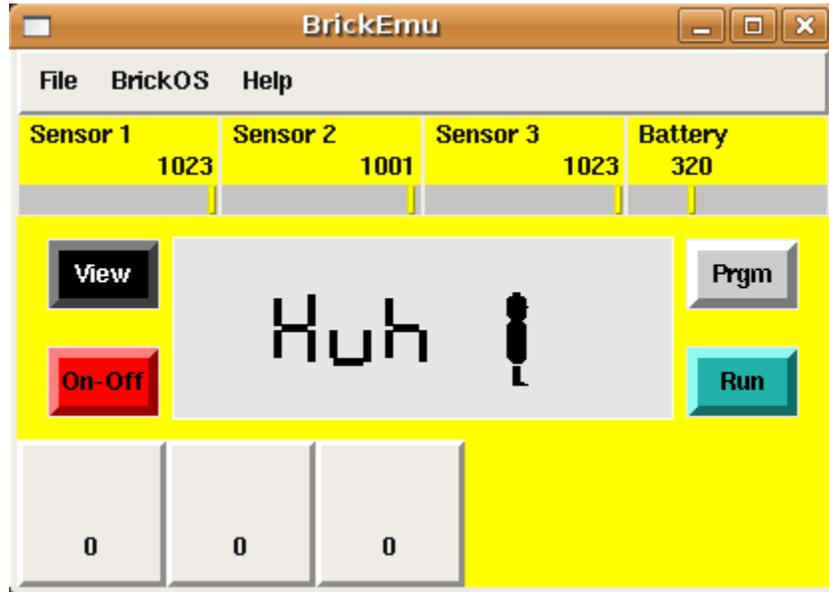


Figure 60 - BrickEMU running program

8. Play with Sensor 2 by moving the small bar back and forth. Each time Sensor 2 detects a 50% change in signal it will switch back and forth from displaying “Huh” to displaying “Ouch”

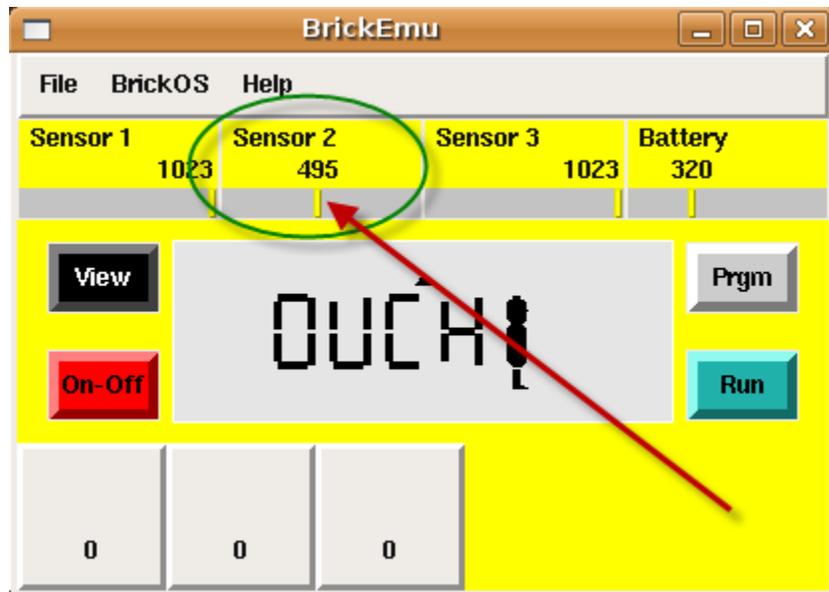


Figure 61 - BrickEMU running program. Note activation of Sensor 2.

9. Finally end the execution of the Robot's program by activating Sensor 1.

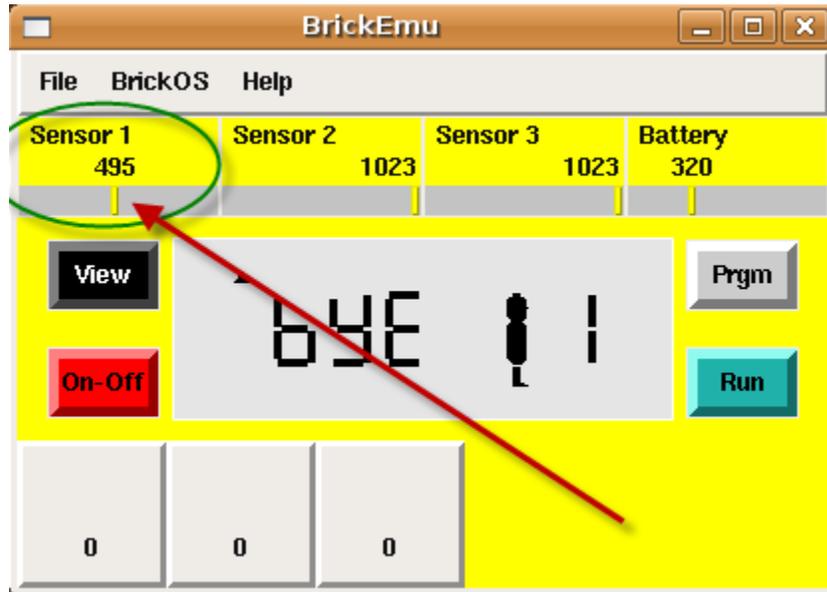


Figure 62 – BrickEMU program end. Note activation of Sensor 1.

10. Congratulations! You have just run your first BrickOS program on a robot emulator.
11. The next step is to learn how to down load your program into the real robot and watch it do its thing.

Section 11: Using the Lego RCX Brick

Before using the Lego RCX Brick for the first time it will have to be installed onto both the host and client machines. Follow the follow steps:

11.1 Installing the USB Tower

Do not plug in the tower until after the drivers have been installed.

- i. Unzip *Tower164.zip* (found on the System DVD in the *Windows Driver* directory) onto a temporary directory on your hard drive
- ii. Click on the “*Setup*” file
- iii. After the tower drivers are updated, plug in the tower.
- iv. Windows will report that it has found a new device and the “Found New Hardware Wizard” will appear



Figure 63

- v. Choose “*No, not this time*” and press “*Next*”

- vi. Allow the software install the software automatically, and press “Next”

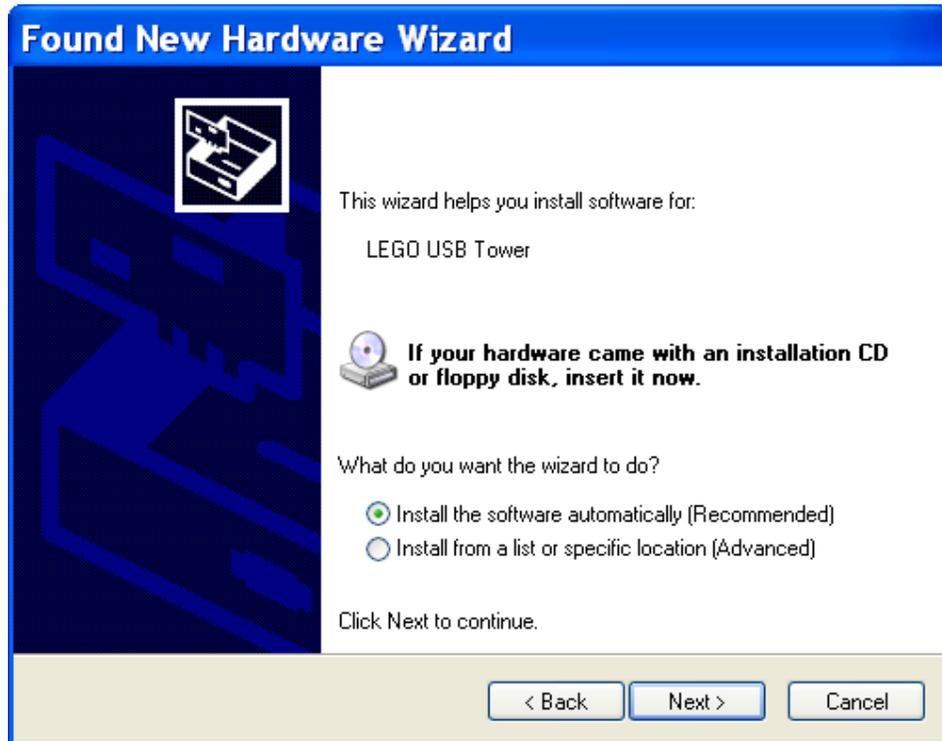


Figure 64

vii. After a short period of time you will see:



Figure 65

viii. Press *Finish* to close the wizard. Windows will notify you that it has successfully installed the Lego Tower. At this point, you may delete your temporary installation file.

- ix. Start the *VirtualBox* console. Be sure that the *Ubuntu Virtual Machine* is powered off, and scroll the *Details* window until the *USB* section is visible, as in the screenshot below:

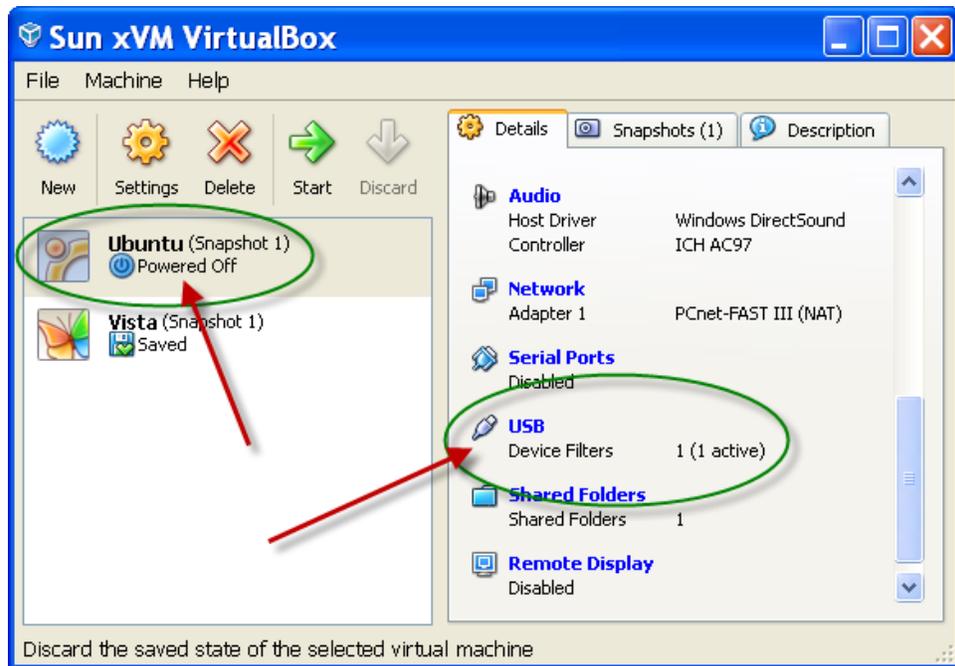


Figure 66

- x. Double-click on the *USB section* (Windows - Figure 67) or the *Ports section* (Mac OS X - Figure 68) and select the 2nd icon on the right to add USB Device Filters to the Virtual machine

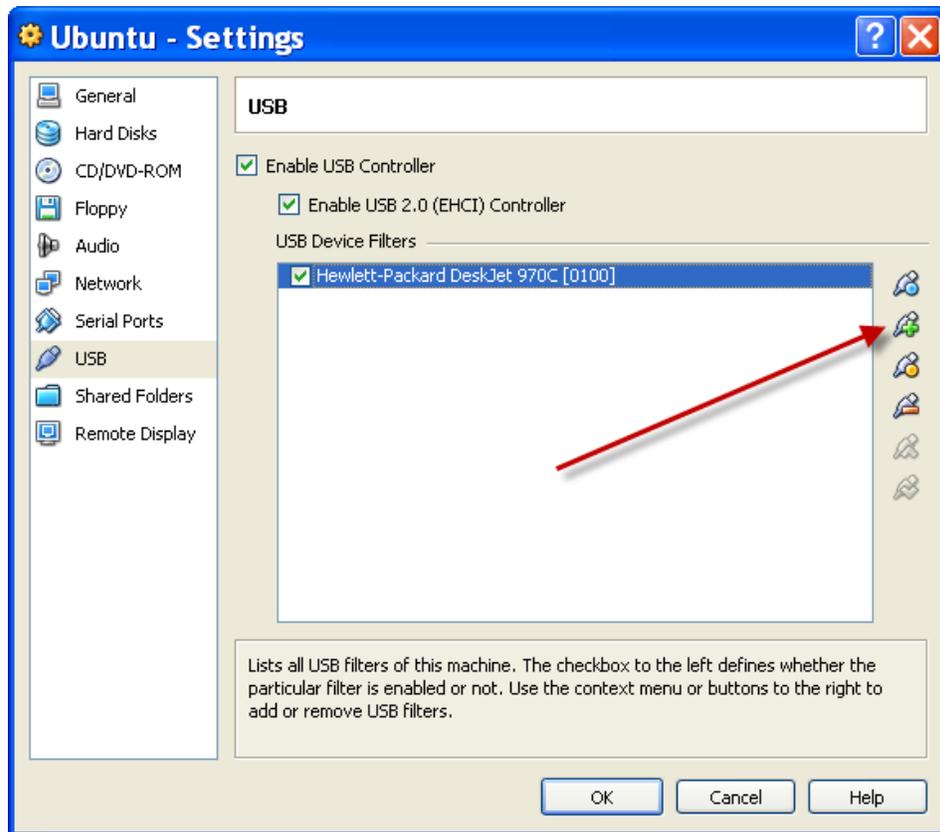


Figure 67 – VirtualBox (Windows) USB Setup

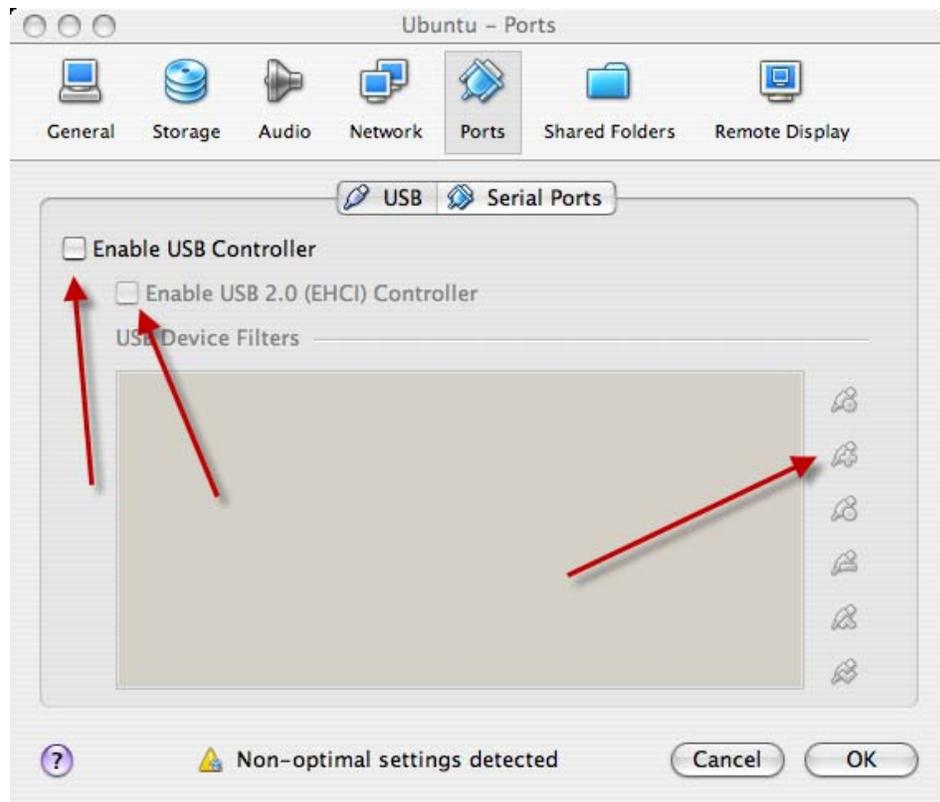


Figure 68 – VirtualBox (Mac OS X) USB Setup

Be sure to enable the USB Controller by checking both boxes.

- xii. Choose the *LEGO Group Lego USB Tower* from the list that appears. You may also wish to add your USB printer (if any) so that it becomes available to the Ubuntu Virtual Machine.

Note: Items selected will become unavailable to the host as long as the client is active. If you print anything from the host, it will be held in the printer queue until you shut down the virtual machine.

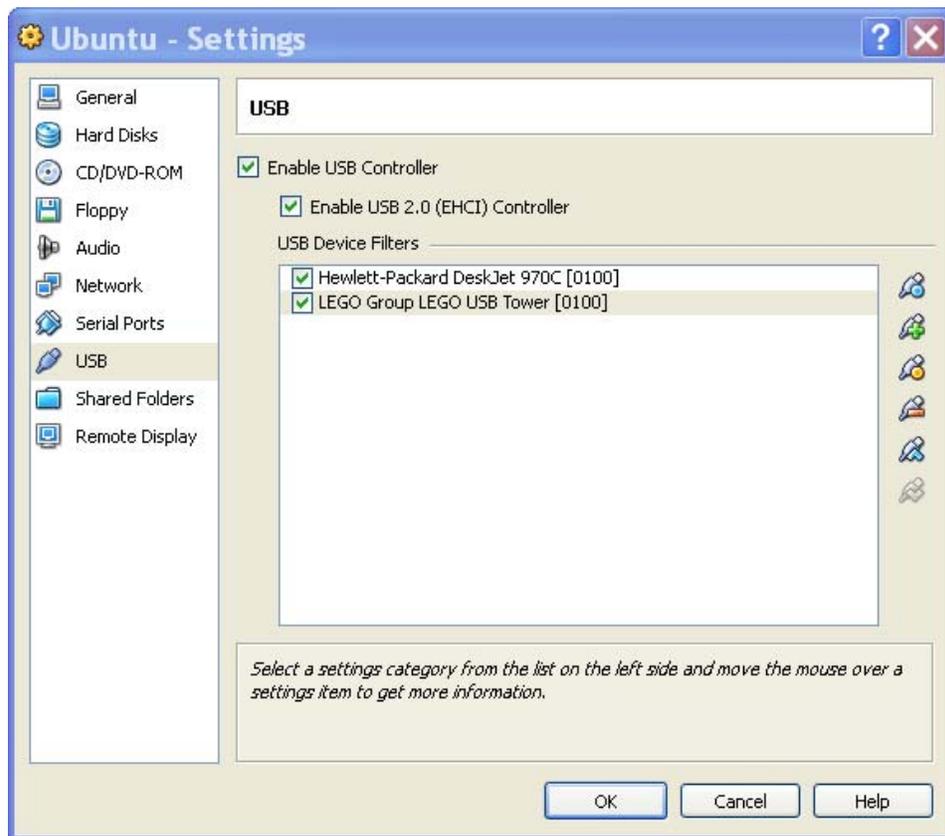


Figure 69

- xii. Choose OK to return to the VirtualBox console.

- xiii. Start the Ubuntu Virtual Machine, and *right-click* on the USB icon as shown below. In the resultant list of devices place a check mark by clicking to the left of the entry for the *LEGO Group Lego USB Tower*.

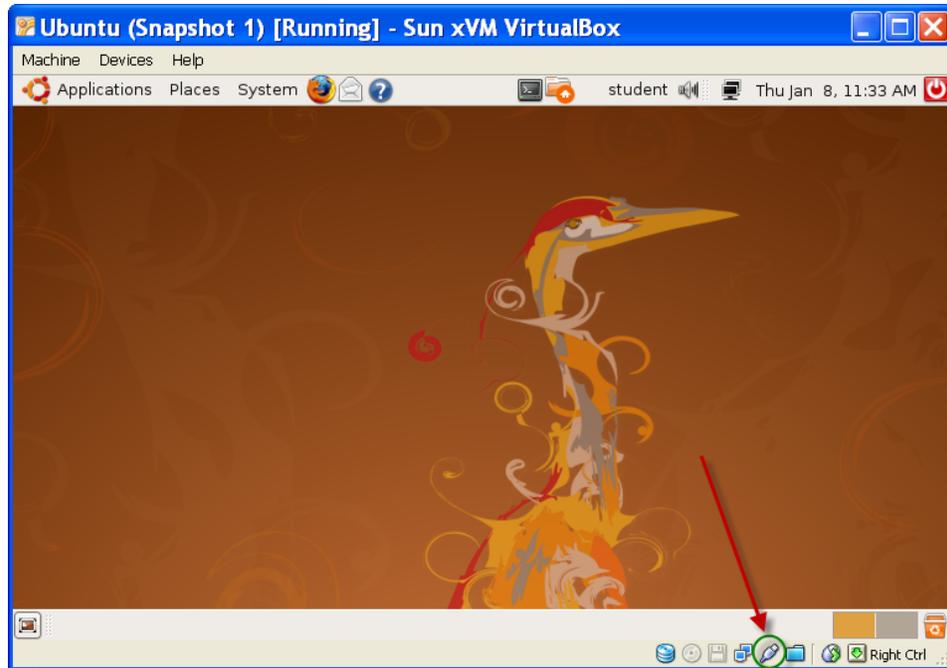


Figure 70

- xiv. If you get an error message as shown below, *shut down* the virtual machine and *reboot* the host computer. Repeat the previous step (xiii). This time the Tower will be captured by the virtual machine.
- xv. You now may proceed to load the firmware into the Lego RCX Brick. (See next section).

11.2 How to install the firmware

The first time you use the brick, you must download the BrickOS firmware into the actual brick. Attach the tower, and place the brick's infrared window approximately 3-4 inches in front of the tower. Turn the brick on. Start the Code::Blocks application and choose *Tools/RCX Load Firmware* from the Code::Blocks menu.

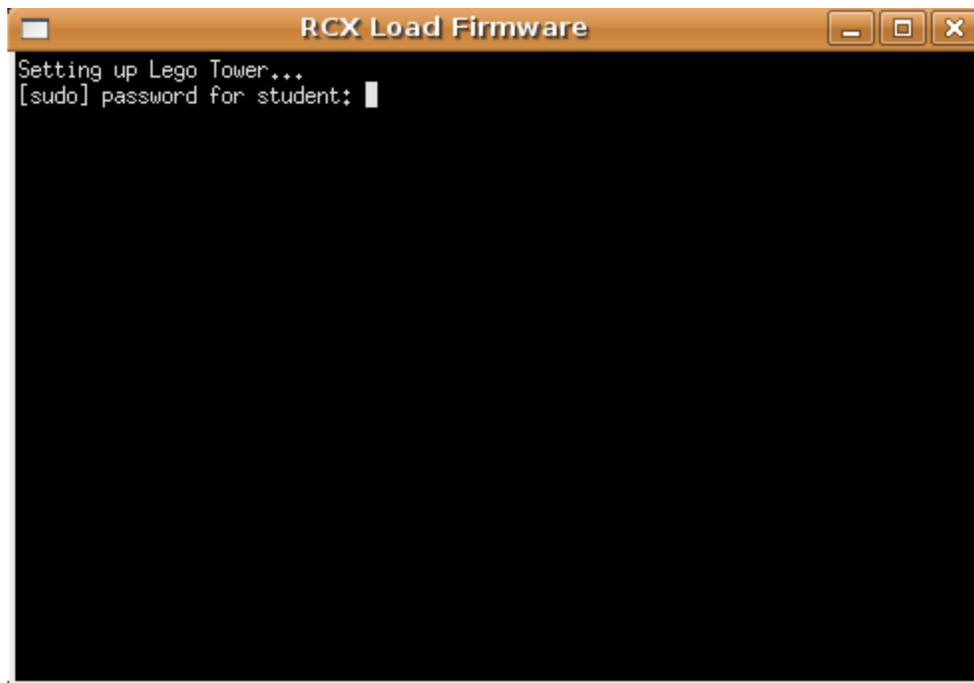


Figure 71 - Loading the BrickOS Firmware on to the RCX Brick

Enter the password (*'student'*) and wait for the process to complete.

11.3 How to transfer the compiled program to the LEGO Brick

Attach the tower, and place the brick's infrared window approximately 3-4 inches in front of the tower. Turn the brick on. Start the Code::Blocks application and choose *Tools/RCX Transfer Brick* from the Code::Blocks menu. A message window opens up, showing the progress of the transfer. When the transfer completes, press *'Enter'*.

11.4 Running your program

Turn the RCX Brick on, and press *"Run"*.

Section 12: Advanced Topics

12.1 Setting up the printer

Setting up the printer is very similar to setting up the Lego Tower (see Section 11.1 Step ix – page 64).

- i. First make sure that your USB printer is installed correctly for your host operating system.
- ii. Make sure your printer is powered ON.
- iii. Add the USB filter for your printer as described in Section 11.1 steps ix through xii.
- iv. Start your Ubuntu Virtual machine
- v. Choose *System/Administration/Printing* from the Ubuntu menu.
- vi. Choose the *New Printer* icon and follow the wizard to install your printer.

Note: Items selected in the VirtualBox USB settings section (Figure 69) will become unavailable to the host as long as the client is active. If you print anything from the host, it will be held in the printer queue until you shut down the virtual machine and only then will it print.

12.2 How to Setup Shared Folders

It is often useful to be able to transfer information between the host computer and the virtual Ubuntu machine. For simple transfer of text, *cut and paste*, will function just as it does on the host. Simply copy the text you wish to copy in the source window and paste it at the desired location in the destination window.

However, it is also useful to be able to exchange files between the host and the client. This is done by what is known as “*Shared Folders*”, that is a folder in the host and a folder in the client that mirror each other’s contents.

The procedure is similar for both a Windows and Mac hosts. In the following steps we will use a Windows XP screenshots to illustrate the process.

- a. Create a folder on the host. For this example we will create a folder called *My Documents\VirtualBoxShared\Ubuntu*.

b. From the VirtualBox's menu select *Machine/Settings* and from the list of icons on the left select *Shared Folders* (Figure 72). Press the button with the + symbol to add a new shared folder in the list. You will have to specify a folder name for each folder you add. Make sure you memorize that name because you will need it very soon.

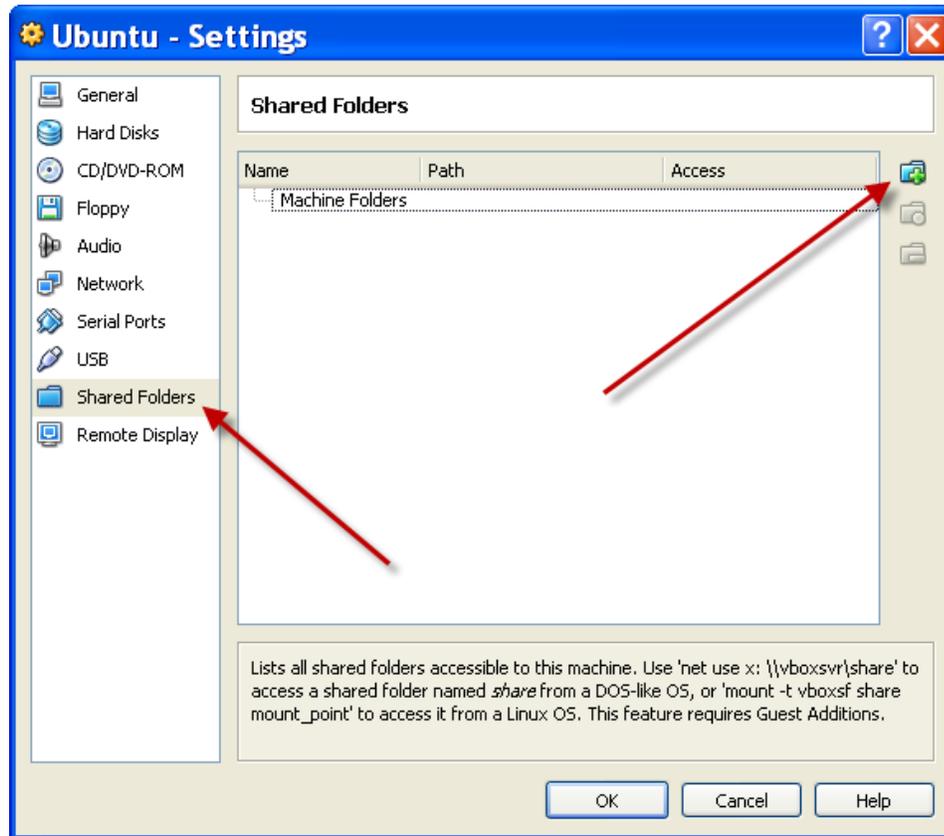


Figure 72 - Adding Shared Folders

- c. Using the pull down menu, select the location of the folder you created in step a.

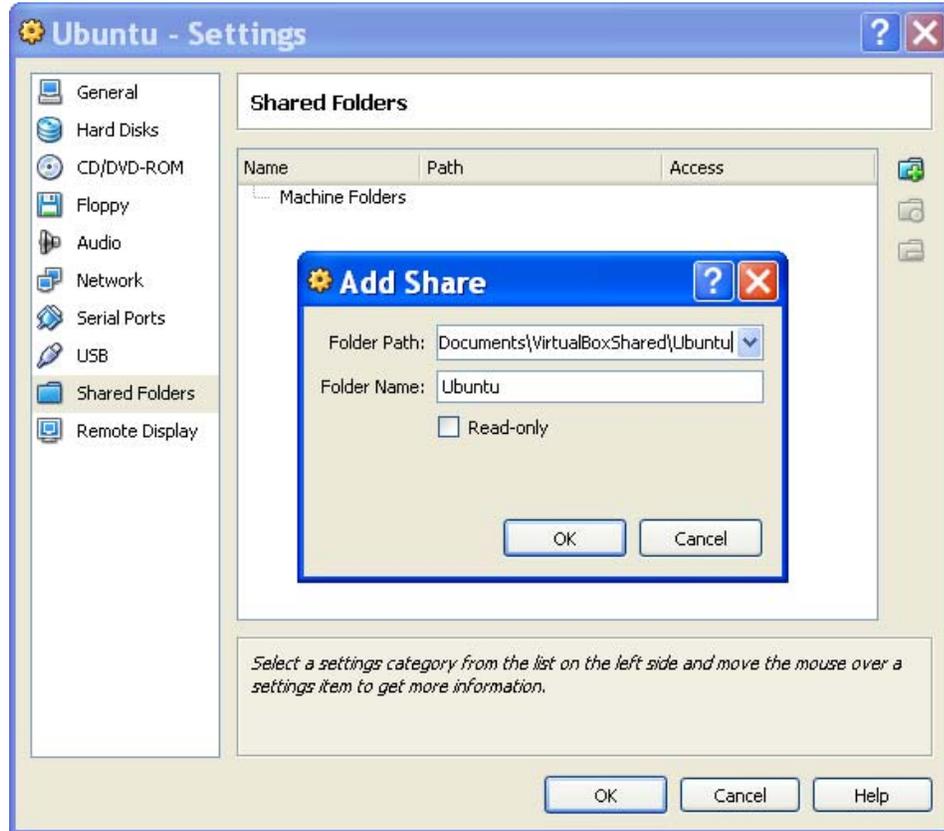


Figure 73 - Specifying the Shared Folder

d. After selecting OK, the window should appear as in Figure 74

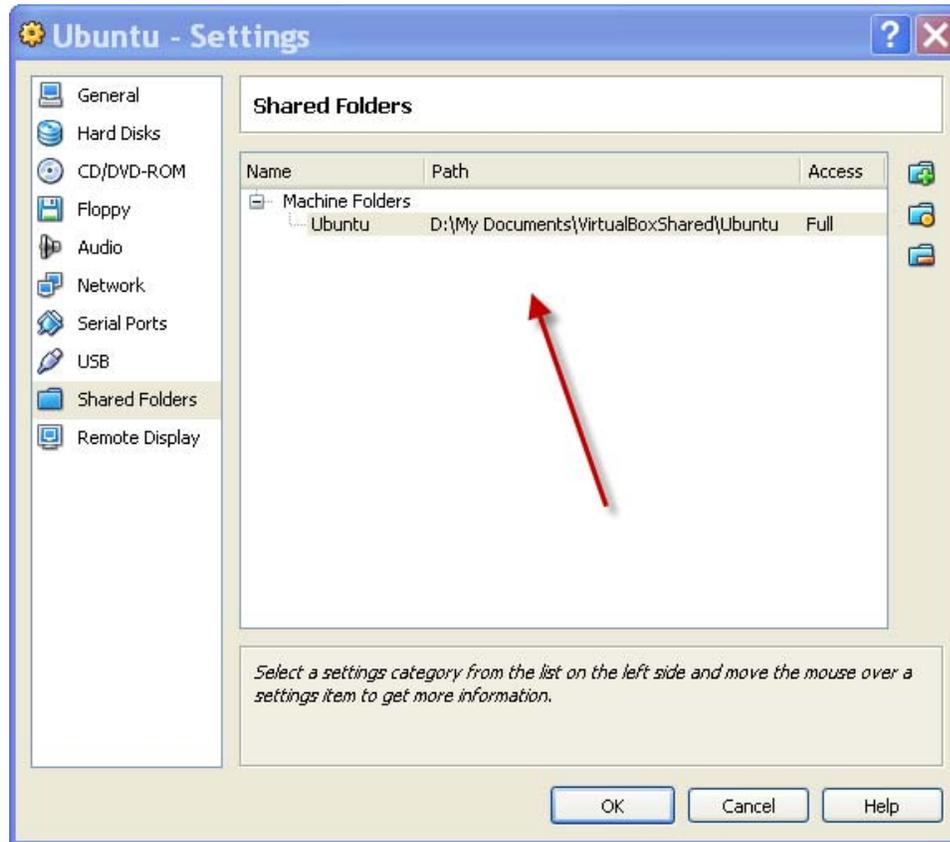


Figure 74 - Shared Folder Selected

e. Choose OK and then start the Ubuntu virtual machine.

- f. Choose Places/Home Folder from the Ubuntu menu and create the folder in your Home directory which you wish to share. (Note: *the File Browser* in Ubuntu is similar to *Windows Explorer*.) In our example I have called the folder *WindowsShare* as shown in Figure 75

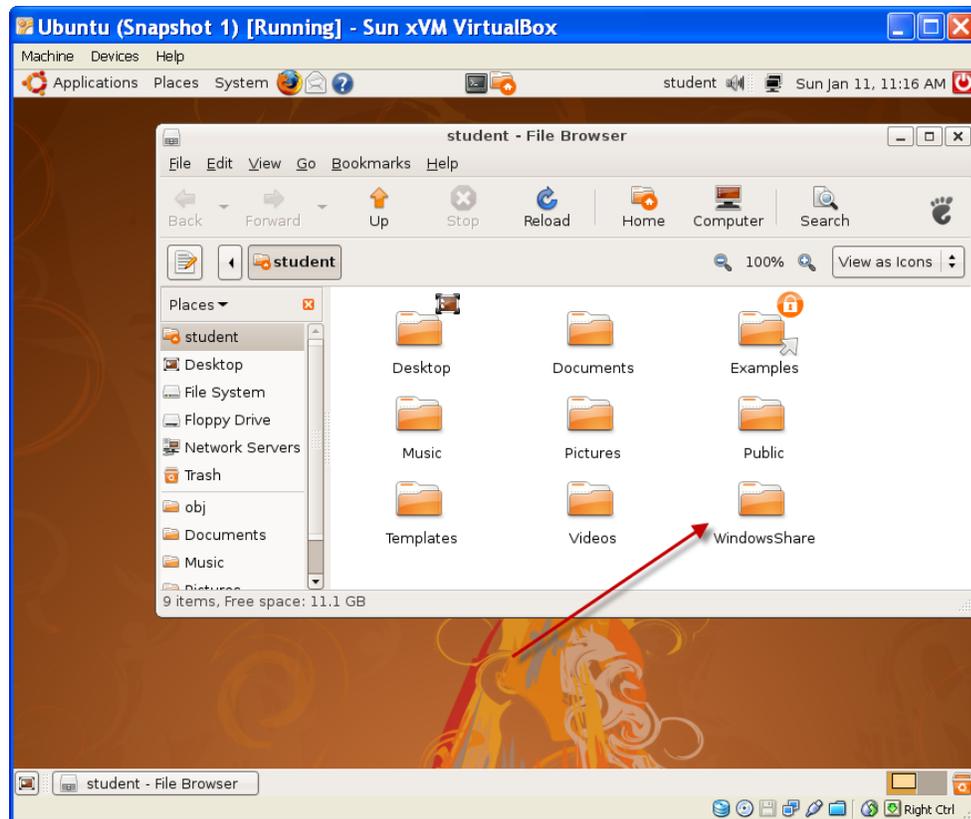


Figure 75 - Ubuntu File Browser

- g. Choose *Applications/Accessories/Terminal* from the Ubuntu menu and (assuming you names the folders as I did above) type the following line at the prompt: (Note: Linux is case sensitive, so you must use consistent capitalization.)

```
sudo mount -t vboxsf Ubuntu ~/WindowsShare
```

- h. Press *Enter* and enter your password at the prompt (*student*). Your screen should look like Figure 76

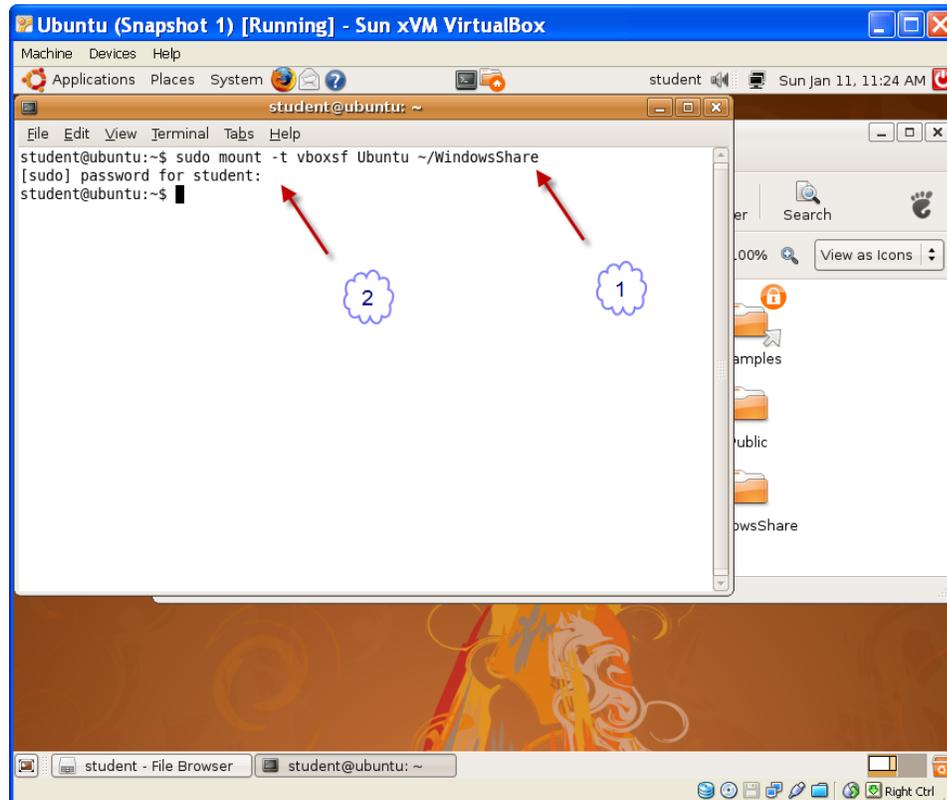


Figure 76 - Mounting the Shared Folder in Ubuntu

- i. You may now close the Ubuntu *Terminal* window. The *WindowsShare* folder in Ubuntu and the *C:\My Documents\VirtualBoxShared\Ubuntu* folder in Windows will now mirror each other.

Note: You will have to remount the shared folder each time you reboot the Ubuntu virtual machine. Users familiar with Linux may use the */etc/init.d/rc.local* script to execute these commands on startup to have the shared folders automatically mounted every time you start your Ubuntu VirtualBox.

12.3 Using the DDD Debugger

The section will be added later and is intentionally blank.

Appendix I – References

- a. Programming the BrickOS
 - i. http://www-mdp.eng.cam.ac.uk/web/CD/engapps/brickos_reference.pdf
 - ii. <http://brickos.sourceforge.net/docs/APIs/html-c++/>
 - iii. Sample programs
- b. More on Ubuntu Linux
 - i. http://www.techotopia.com/index.php/Ubuntu_Linux_Essentials
 - ii. http://www.techotopia.com/index.php/Ubuntu_Desktop_Essentials
 - iii. <https://help.ubuntu.com/ubuntu/desktopguide/C/index.html>
 - iv. <http://ubuntuforums.org/index.php>
 - v. <http://www.ubuntu.com/>
 - vi. <http://ubuntutip.googlepages.com/>
- c. More on Virtual Box
 - i. <http://www.virtualbox.org/>
- d. More on CodeBlocks
 - i. <http://www.sci.brooklyn.cuny.edu/~goetz/codeblocks/>
 - ii. <http://www.sci.brooklyn.cuny.edu/~goetz/codeblocks/codeblocks-instructions.pdf>
 - iii. <http://www.codeblocks.org/>

Appendix II – Installing the BrickOS Firmware into the Emulator

Under ordinary circumstances, when starting the BrickEMU, the firmware will be installed automatically – Should you need to reload the firmware, follow the steps shown below:

1. Load the BrickOS firmware into the emulator. On the BrickOS Window select *File/Firmware*. You will now see a window allowing you to select the firmware which is to be loaded (Figure 77).

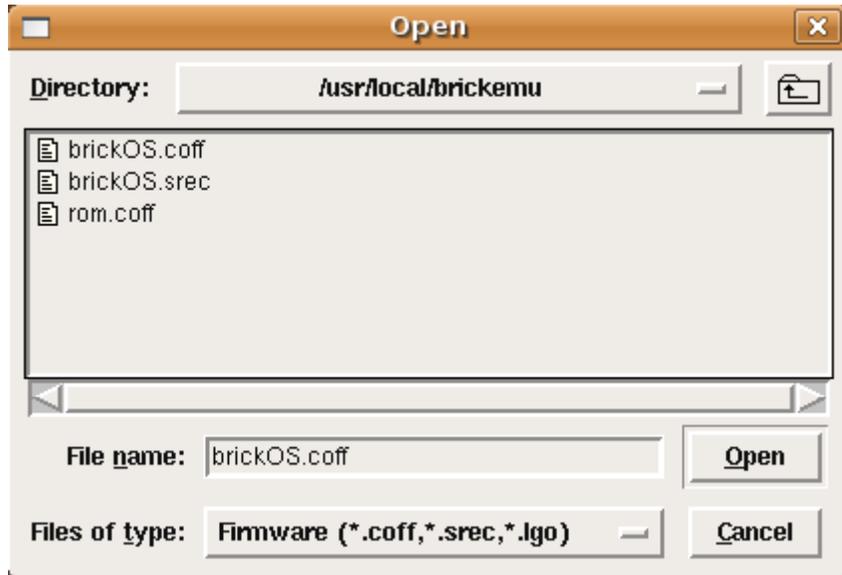


Figure 77 - Installation of BrickOS Firmware

2. Select brickOS.coff followed by Open. After a brief period of time the firmware will be loaded into the BrickEMU and its window will now look as in Figure 78.

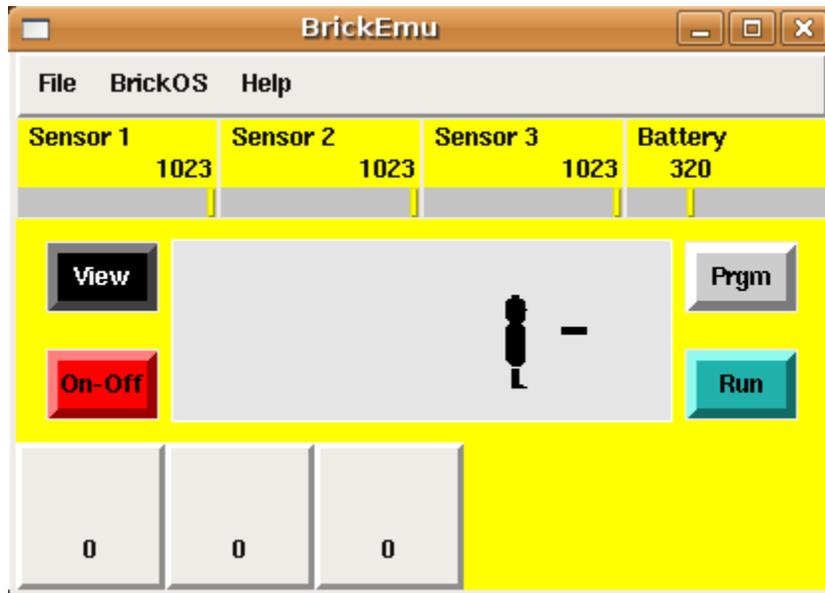


Figure 78 - BrickEMU with firmware installed

You have successfully installed the firmware and are ready to program for the Lego Robot!

Appendix III Installing VirtualBox Guest Additions

The “*VirtualBox Guest Additions*” are a set of drivers and utilities that are shipped as a subset of the VirtualBox for the purpose of being installed *inside* a *guest* to improve its performance and cooperation with the host.⁷ It provides such enhancements as a special video driver that allows for better performance and features such as dynamically adjusting the guest resolution when the virtual machine window is resized.⁸ Other features include support for mouse integration, communication, and shared folders.

When we originally installed the system, the guest additions for Ubuntu were automatically installed. If you update the Ubuntu client, it may be necessary to reinstall these guest additions. The following is the procedure: (Note we use screenshots of a Mac OS X host to illustrate the procedure. However, since the installation of the Guest Additions takes place completely within the client, the procedure will be the same on Windows based hosts.)

1. Go to the *Devices* Menu from Virtual Box and select *Install Guest Additions* (Figure 79).

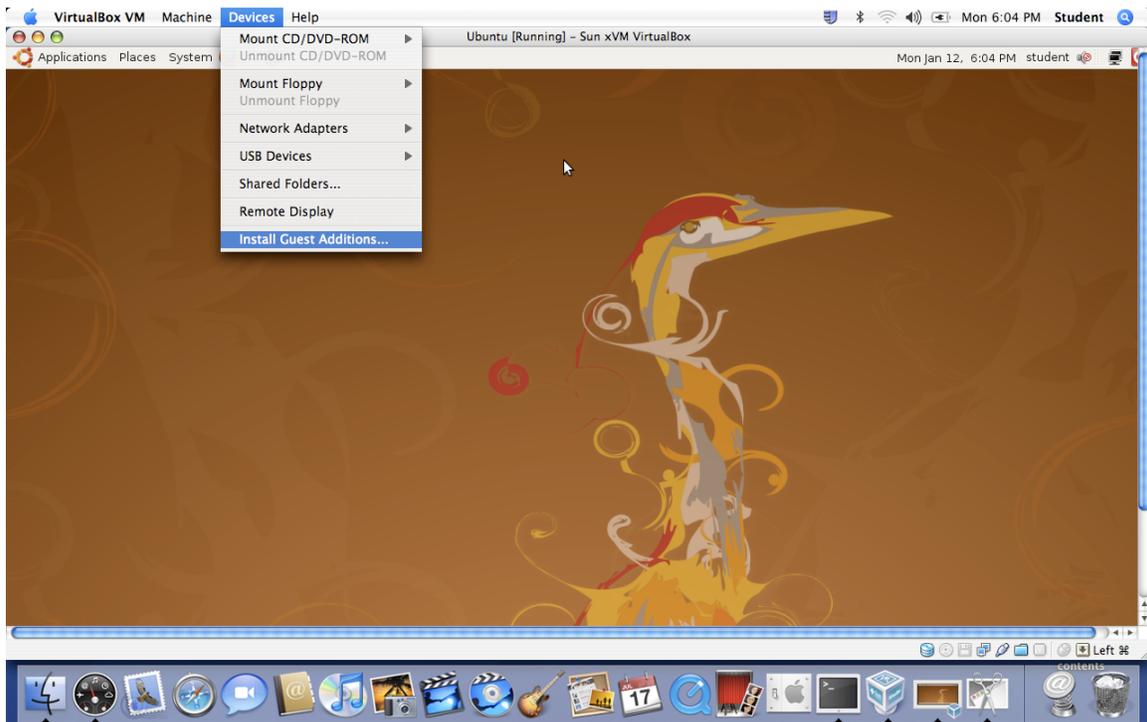


Figure 79 - Installation of Guest Additions

⁷ http://www.virtualbox.org/wiki/VirtualBox_PUEL

⁸ <http://en.wikipedia.org/wiki/VirtualBox>

2. Open up a terminal window by selecting *Applications/Terminal* (Figure 80).

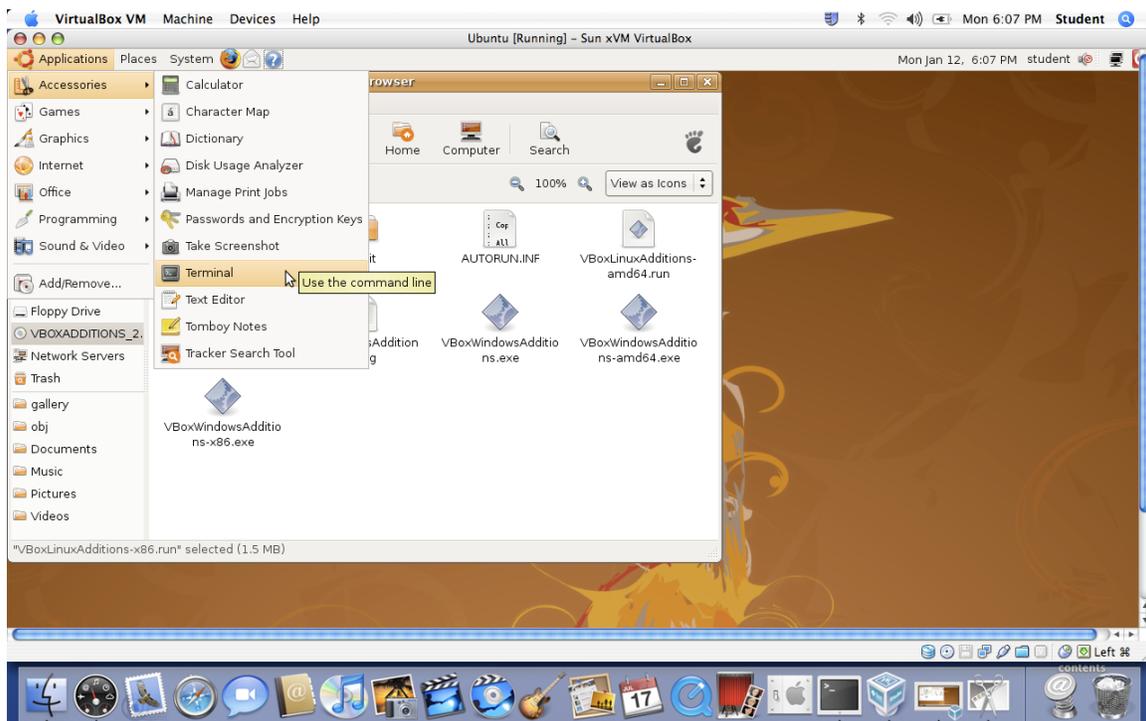


Figure 80 - Opening a Terminal Window

3. Enter the following command at the terminal prompt (Figure 81):

```
sudo /media/cdrom0/VBoxLinuxAdditions-x86.run
```

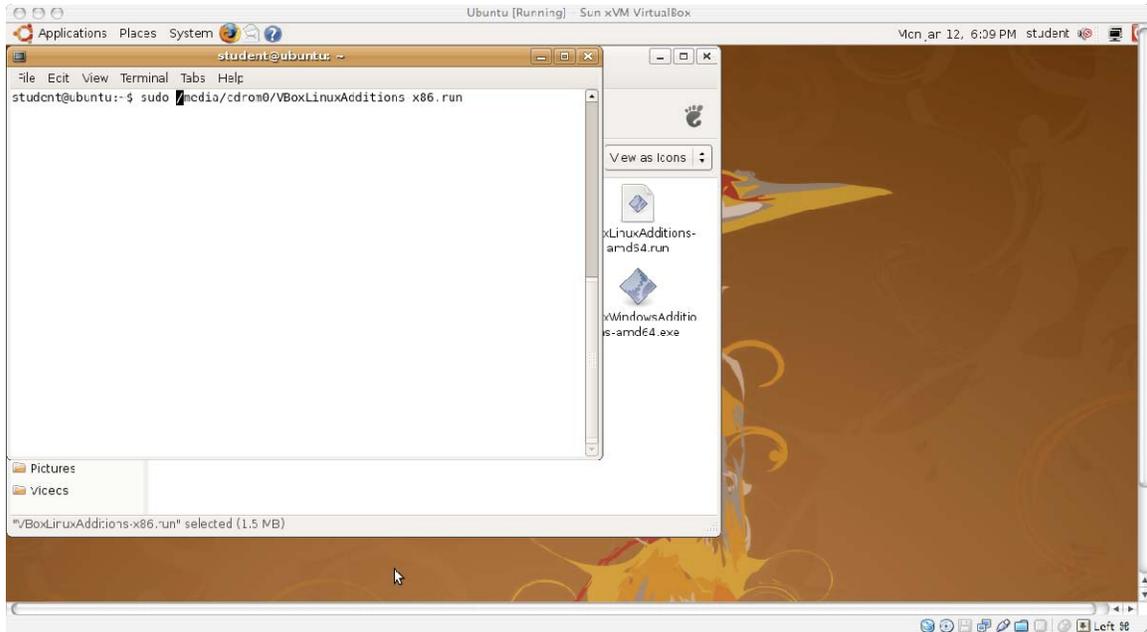


Figure 81 - Installing the VBoxLinuxAdditions

4. At the prompt, enter your password (*student*)

5. Right click the CD and select *Unmount* (Figure 82).

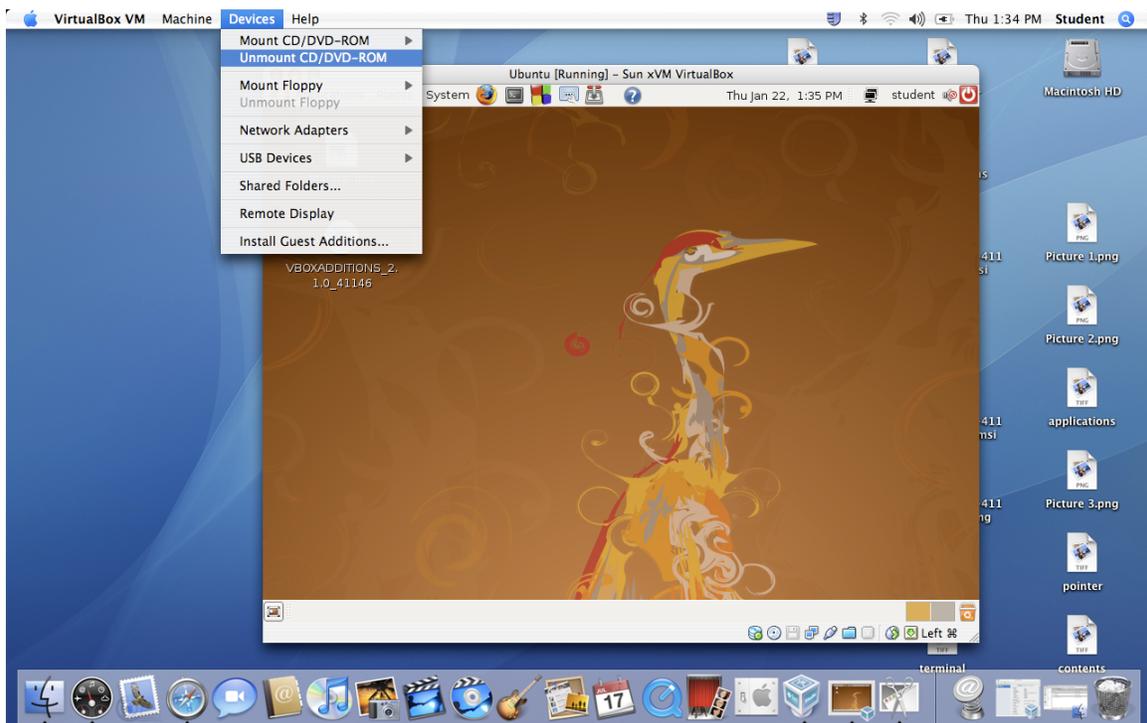


Figure 82 - Ejecting the Virtual Guest Additions CD

6. *Shutdown* the Ubuntu guest system and then restart Ubuntu for the guest additions to load.