Parallel Programming Pre-Assignment

Setting up the Software Environment

Author: B. Wilkinson Modification date: January 3, 2016

Software

The purpose of this "pre-assignment" is to set up the software environment on your computer that will be used in subsequent assignments. You will not be able to do the subsequent assignments as written without this environment. Hence it is essential that you complete it by the posted deadline. ***It is very important to report issues that stop you from progressing immediately so that they can be resolved quickly.***

In the course assignments, various parallel programming tools and programs will be tried out on your own computer (as well as on remote servers). Some of the software requires a C compiler and an MPI message-passing environment. As a convenience, a virtual machine with everything you need has been created as an .ova file that can be imported into VirtualBox (and probably other virtualization products). Virtualbox is a virtualization software that enables "guest" operating systems to be installed on the underlying "host" OS without disturbing the host. It will install on all host platforms (Windws, Mac, and Linux). Our guest operating system will be the Linux distribution Ubuntu. The provided virtual machine does separate the software for this course from existing software and protects your existing installation but you can use a lab computer if you do not want to use your own computer.

There are two important links regarding the course software on course home page or Moodle:

- Parallel Programming Software with links to the course software (excluding VirtualBox)
- Additional Information with more information on using the software.

There are many documents provided, some of which you will need for subsequent assignments.

The links require a username and password -- Username: pablo Password: rp19zb39

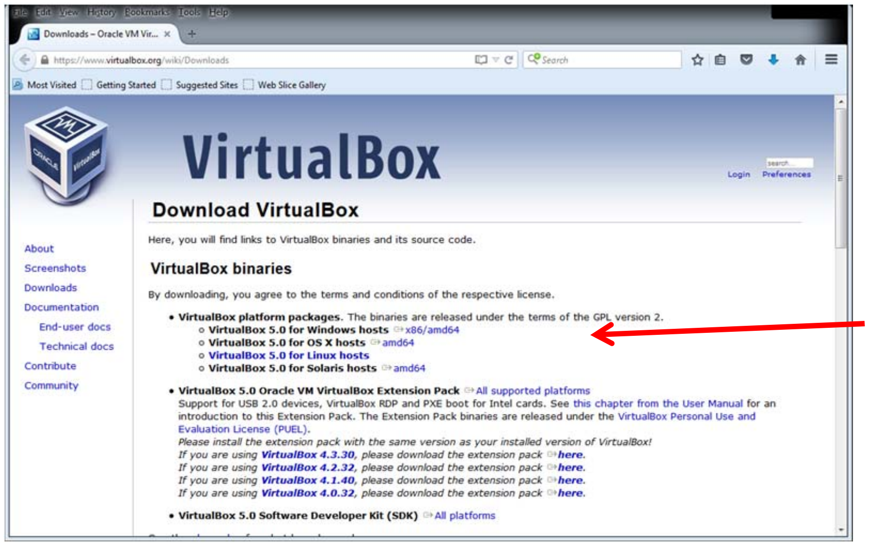
Installing everything from source. If you do not want to use the provided virtual machine and want to install the software onto an existing Linux platform, see the "Installing Software" section at "Additional Information" and then go to Part 2 Testing Environment.

Part 1 Installing VirtualBox

Go to:

https://www.virtualbox.org/wiki/Downloads

and download VirtualBox to suit your computer. Our testing used VirtualBox virtual machine version 5.0.10 but other versions should be fine. Run the installation package (double-clicking the downloaded executable file).


File Edit View History Bookmarks Tools Help
Downloads – Oracle VM Vir...
https://www.virtualbox.org/wiki/Downloads
Most Visited Getting Started Suggested Sites Web Slice Gallery
VirtualBox
Login Preferences
About
Screenshots
Downloads
Documentation
End-user docs
Technical docs
Contribute
Community

Download VirtualBox

Here, you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

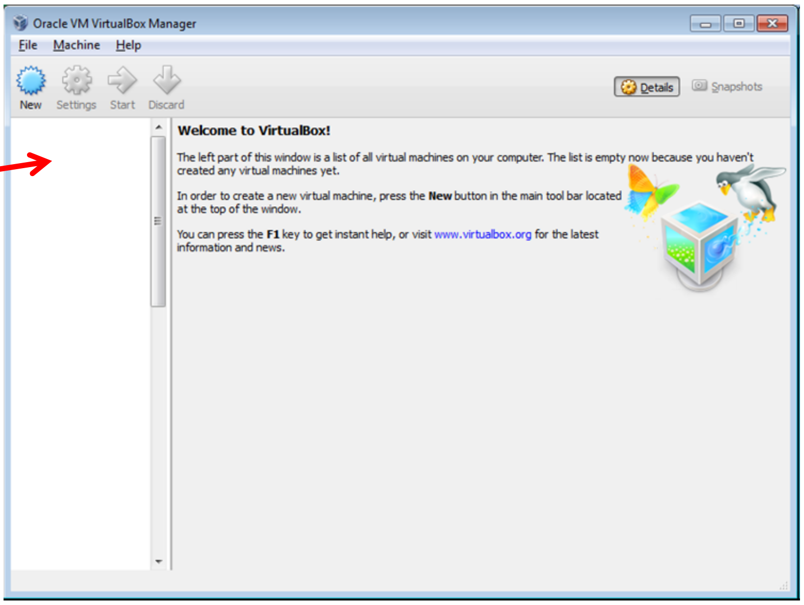
By downloading, you agree to the terms and conditions of the respective license.

- **VirtualBox platform packages.** The binaries are released under the terms of the GPL version 2.
  - **VirtualBox 5.0 for Windows hosts** x86/amd64
  - **VirtualBox 5.0 for OS X hosts** amd64
  - **VirtualBox 5.0 for Linux hosts**
  - **VirtualBox 5.0 for Solaris hosts** amd64
- **VirtualBox 5.0 Oracle VM VirtualBox Extension Pack** All supported platforms
  Support for USB 2.0 devices, VirtualBox RDP and PXE boot for Intel cards. See this chapter from the User Manual for an introduction to this Extension Pack. The Extension Pack binaries are released under the VirtualBox Personal Use and Evaluation License (PUEL).
  Please install the extension pack with the same version as your installed version of VirtualBox!
  *If you are using **VirtualBox 4.3.30**, please download the extension pack **here**.*
  *If you are using **VirtualBox 4.2.32**, please download the extension pack **here**.*
  *If you are using **VirtualBox 4.1.40**, please download the extension pack **here**.*
  *If you are using **VirtualBox 4.0.32**, please download the extension pack **here**.*
- **VirtualBox 5.0 Software Developer Kit (SDK)** All platforms

Download to suit your platform and execute

The package installs in **C:\Program files\Oracle\VirtualBox** by default on Windows. Starting VirtualBox for the first time, you should see:

Blank, no OS's installed yet



Welcome to VirtualBox!

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

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

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

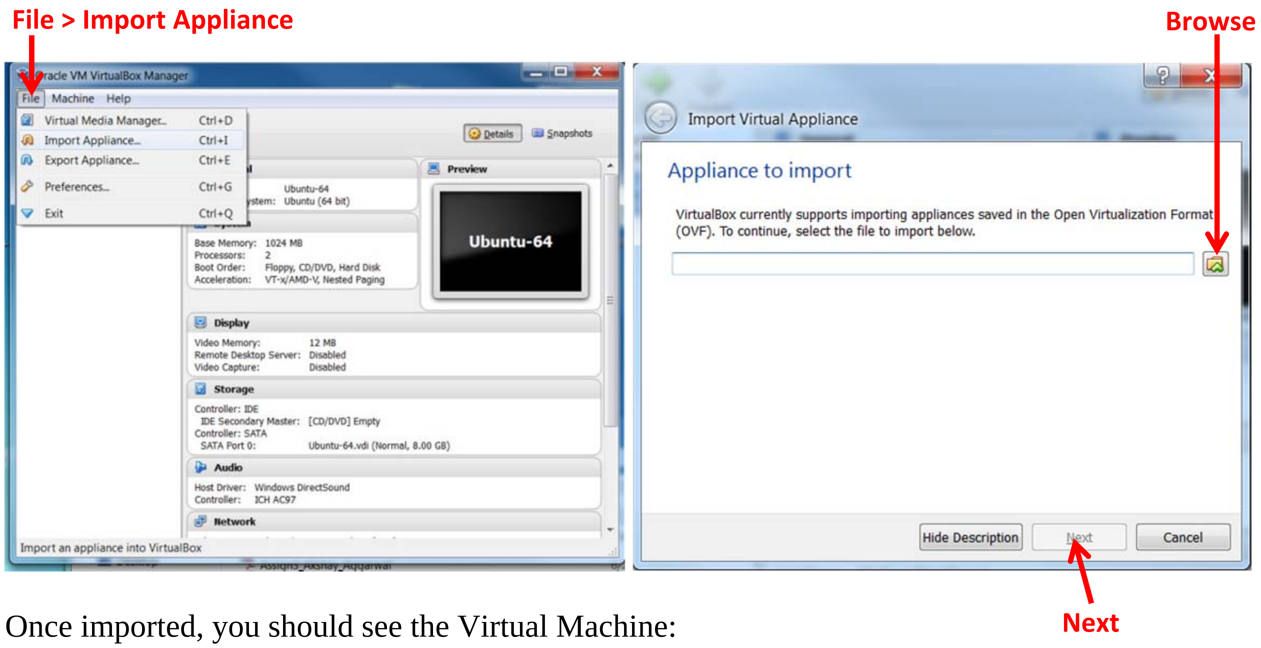
Now you have to import the virtual machine.

Importing Preconfigured Virtual Machine

The VM is provided at the “Parallel Programming Software” link on the course home page on Moodle:

http://coit-grid01.uncc.edu/ParallelProgSoftware/

as **ParallelProg-32.ova**, which is a 32-bit appliance that does not require hardware virtualization support on the processor. Download and save the .ova file. This is a big file (2.5 GB) and may take some time to download (around 30 minutes at 10Mbs). On VirtualBox, go to **File > Import Appliance...** and browse for the downloaded .ova file (probably in the Downloads directory). Click **Next** and **Import**.



Ubuntu-64
Operating System: Ubuntu (64 bit)

Base Memory: 1024 MB
Processors: 2
Boot Order: Floppy, CD/DVD, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging

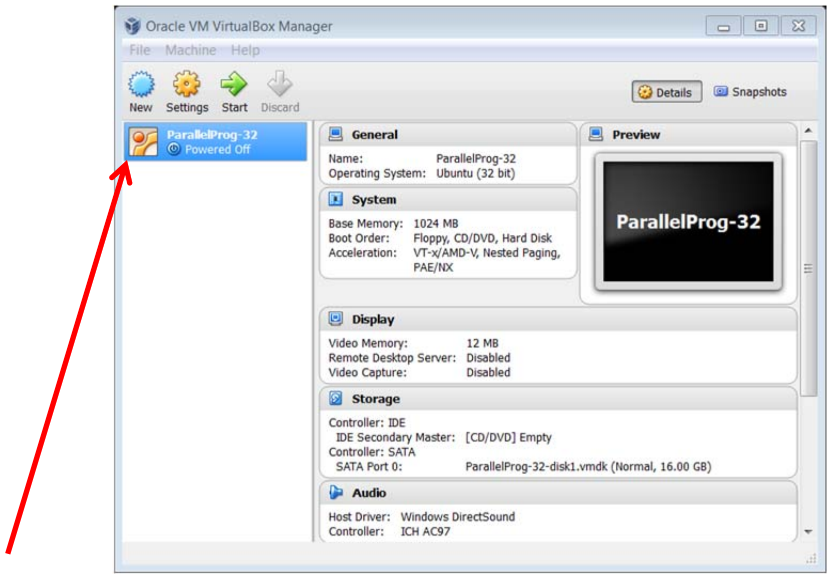
Video Memory: 12 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Controller: IDE
IDE Secondary Master: [CD/DVD] Empty
Controller: SATA
SATA Port 0: Ubuntu-64.vdi (Normal, 8.00 GB)

Host Driver: Windows DirectSound
Controller: ICH AC97

VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

Once imported, you should see the Virtual Machine:



Name: ParallelProg-32
Operating System: Ubuntu (32 bit)

Base Memory: 1024 MB
Boot Order: Floppy, CD/DVD, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX

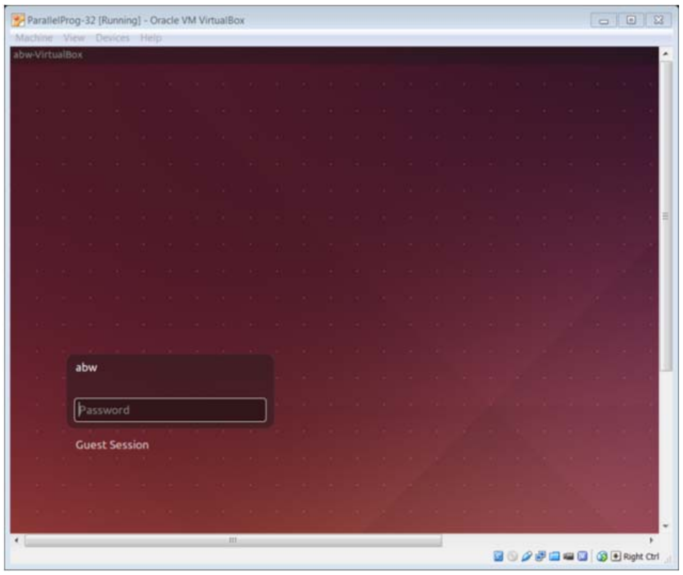
Video Memory: 12 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Controller: IDE
IDE Secondary Master: [CD/DVD] Empty
Controller: SATA
SATA Port 0: ParallelProg-32-disk1.vmdk (Normal, 16.00 GB)

Host Driver: Windows DirectSound
Controller: ICH AC97

Double click to start.

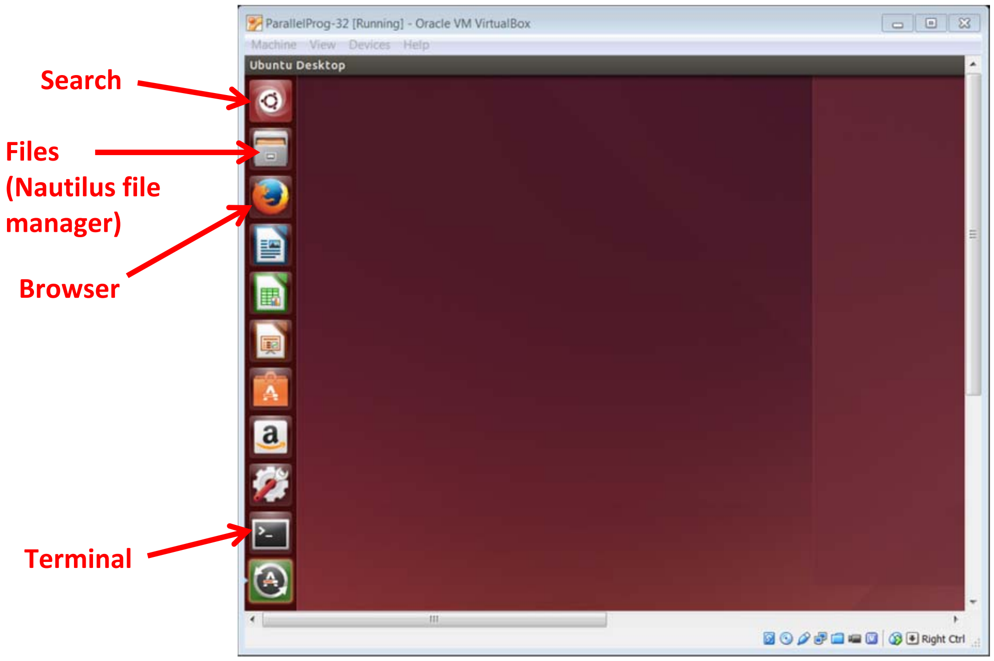
When Ubuntu starts:[1]


ParallelProg-32 [Running] - Oracle VM VirtualBox
Machine View Devices Help
abw-VirtualBox
abw
Password
Guest Session
Right Ctrl

Enter the password.

Username: **abw**
Password: **abc123**

You should see the desktop:


ParallelProg-32 [Running] - Oracle VM VirtualBox
Machine View Devices Help
Ubuntu Desktop
Search
Files
(Nautilus file
manager)
Browser
Terminal
Right Ctrl

More information on using Ubuntu within Virtualbox can be found at the link “Using Ubuntu including within VirtualBox” from “**Additional Information**” on the course home page on Moodle.

[1] For issues, see “**Course FAQ**” at the “**Additional Information**” link.

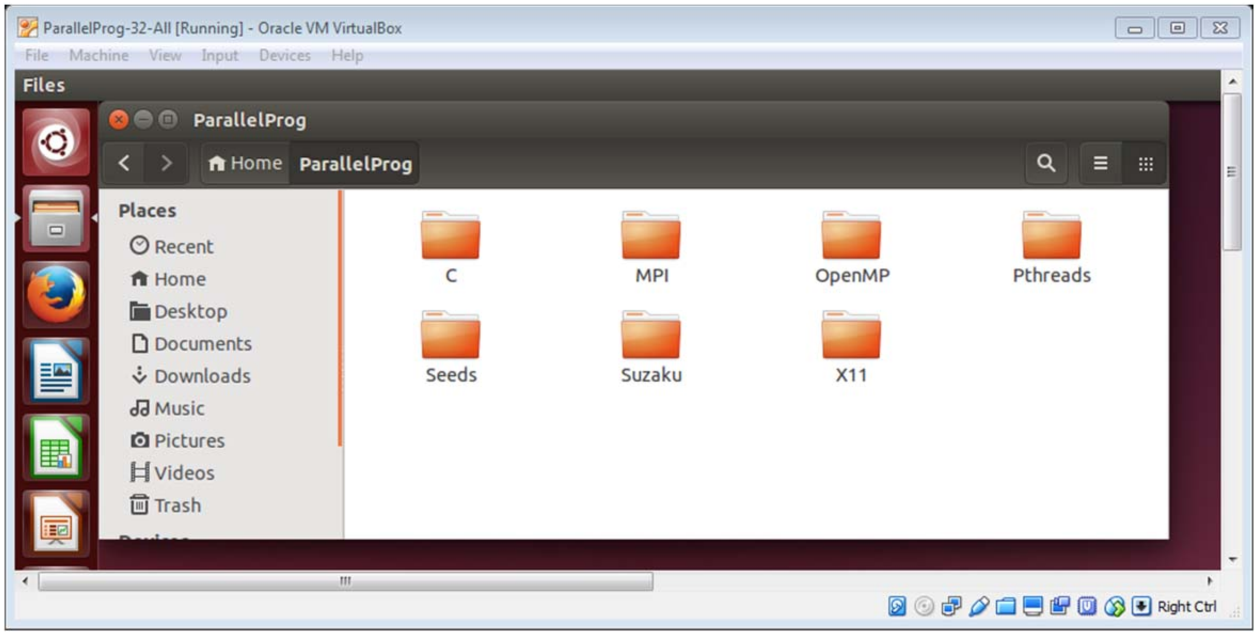
Software

Within the provided virtual machine is the following software stack:[2]

- 32-bit Ubuntu Linux OS version 14.04, with VirtualBox guest additions
- OpenMPI version 1.8.1
- Java version 1.7
- Eclipse version 4.3.2 (Kepler) with plug-ins for both Java and C programming, and the parallel tools platform (PTP) for MPI and OpenMP

Within the directory **~/ParallelProg**, there is one directory for each programming environment we may need in assignments:

- OpenMP
- MPI
- Seeds
- Suzaku



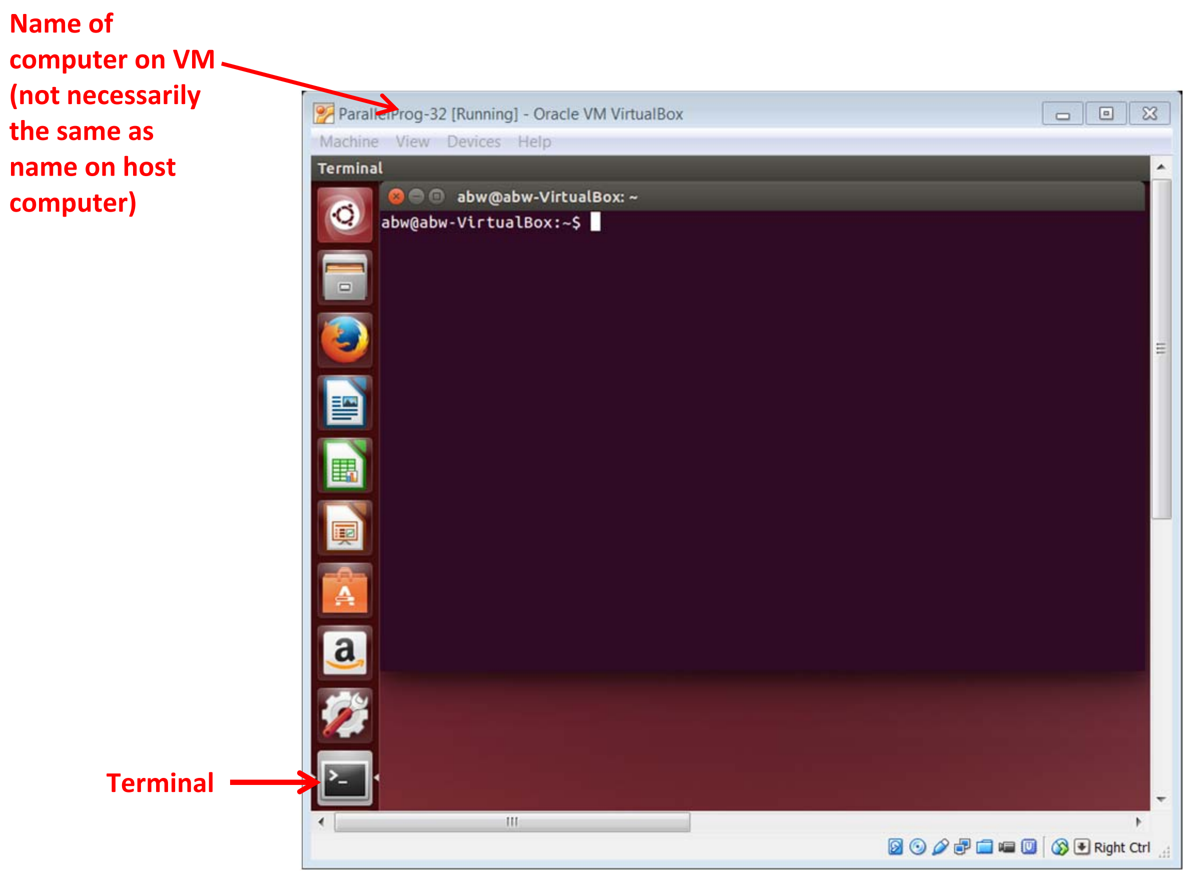
Within each of these directories are sample programs and test files for the corresponding environment (e.g. **hello.c**, **matrixmult.c**, etc.) and a **workspace** directory used for compiling and running the programs as Eclipse projects. The other directories provide sample code for C, X11, and Pthreads. You will need the provided X11 macro in later assignments but we will not do any Pthreads programming in assignments.

[2] The versions of the various software packages may be updated over time, and generally you can use more recent versions as they become available.

Part 2 Testing the VM environment

Your environment will be tested with the provided sample programs. The environment and its use will be described in more detail in subsequent assignments.

Terminal window. Start a terminal by double clicking on the Terminal icon if present on the leftside ribbon of icons ("Launcher") otherwise with **Control** + **Alt** + **t**, or click the search icon (top icon on left side ribbon) to search on "Term" to find the terminal and start it. For convenience, lock the terminal to the launcher by clicking on the terminal on the left side ribbon and selecting lock to launcher.


Name of computer on VM (not necessarily the same as name on host computer)
ParallelProg-32 [Running] - Oracle VM VirtualBox
Machine View Devices Help
Terminal
abw@abw-VirtualBox: ~
Terminal
Right Ctrl

```
abw@abw-VirtualBox:~$
```

Many of the programming assignments ask you to compile and execute programs using a command line. Command line is actually more convenient in many cases than using an IDE such as Eclipse although we although do use Eclipse sometimes, and certainly for Java.

Basic Linux commands are given under "**Additional Information**" on the home page. The most likely commands you will need are **cd** (change directory) and **ls** (list contents of a directory). Use **cd ..** to moves up one directory. *Note you do not need to type the full name of a file or directory - start with the first few letters of the name and use the Tab key to get the system to fill in the rest once the letters refer to a unique name. For example* **cd Pa** *will fill out to* **cd ParallelProg/**.

Task 1 Command line execution of a C program

Let's try a really simple C program. The following C program called **hello.c** should be in **~ParallelProg/C**:

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
     printf("Hello World\n");
     return 0;
}
```

cd to **~ParallelProg /C**. If the program is not there, create it using an editor (typically **gedit** in Ubuntu).

Compile the program with the command:[3]

```
cc hello.c -o hello
```

On a Linux system this usually invokes the gcc compiler.

Execute the program with the command:

```
./hello
```

Note **./** (the current directory) is usually necessary because the current directory is purposely not in the path. Save a screenshot of the output for your submission document.

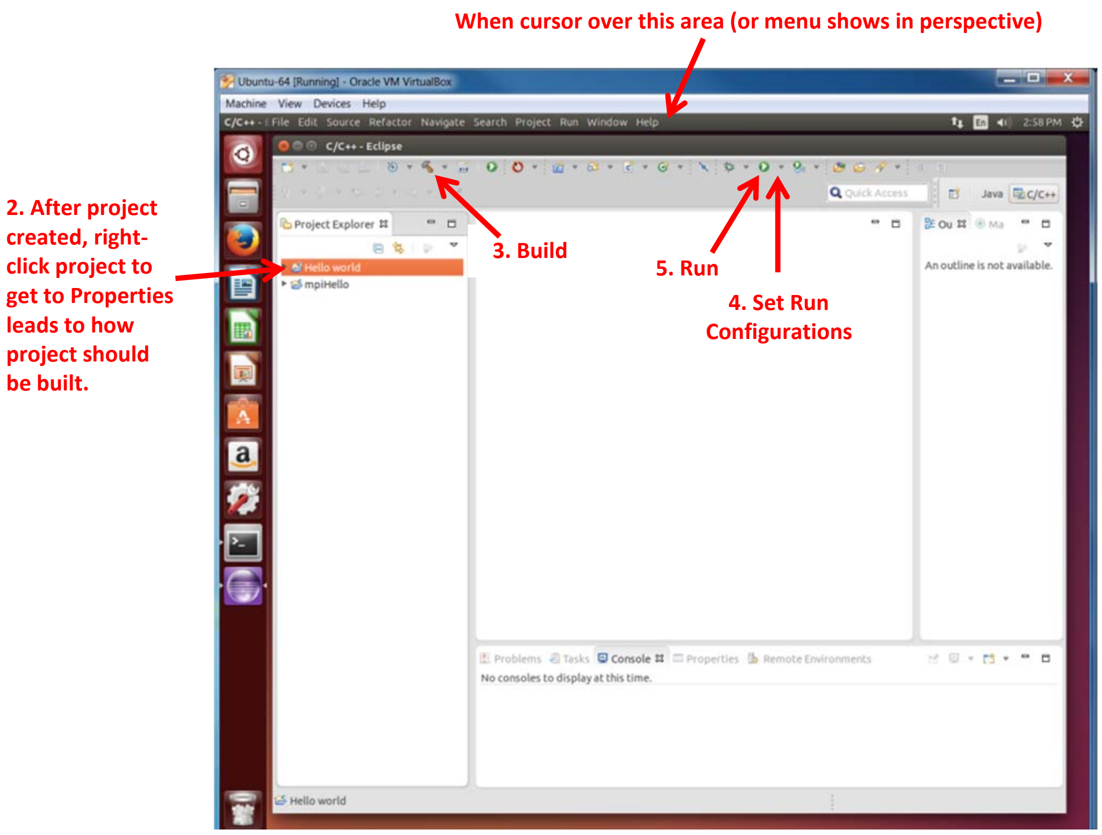
Task 2 Execution of a C program using Eclipse

Certain assignments also ask you to use the Eclipse IDE, so let try that.

General. Eclipse has environments called "Perspectives" for different programming languages. It is installed in the virtual machine with both Java and C perspectives, including for parallel programming tools. Here we will use the C perspective. The basic steps to compile and execute a program are:

1. Create a project with the source file and any required libraries
2. Set how to build (compile) project in **Properties > ... Build**
3. Build project (compile to create executable)
4. Set how to execute the compiled program in **Run Configurations**
5. **Run** (execute) using the specified run configurations

[3] Whether the output option **-o hello** is before or after the source file is personal preference. It usually works either way. Strictly options are defined in gcc before the source file. In the course documentation you may see it either way.


When cursor over this area (or menu shows in perspective)
2. After project created, right-click project to get to Properties leads to how project should be built.
3. Build
5. Run
4. Set Run Configurations
Ubuntu-64 [Running] - Oracle VM VirtualBox
Machine View Devices Help
C/C++ - File Edit Source Refactor Navigate Search Project Run Window Help
C/C++ - Eclipse
Quick Access
Java
C/C++
Project Explorer
Hello world
mpiHello
Ou
Ma
Problems
Tasks
Console
Properties
Remote Environments
Hello world

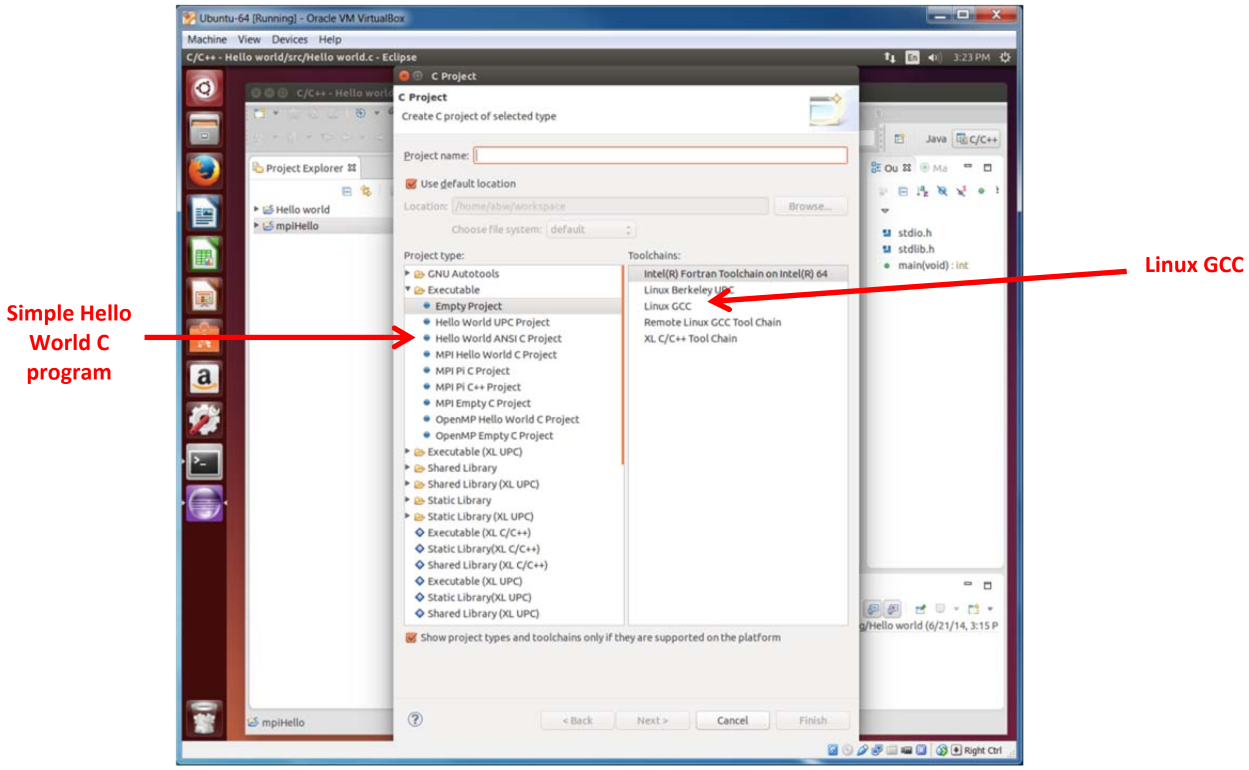
An outline is not available.

No consoles to display at this time.

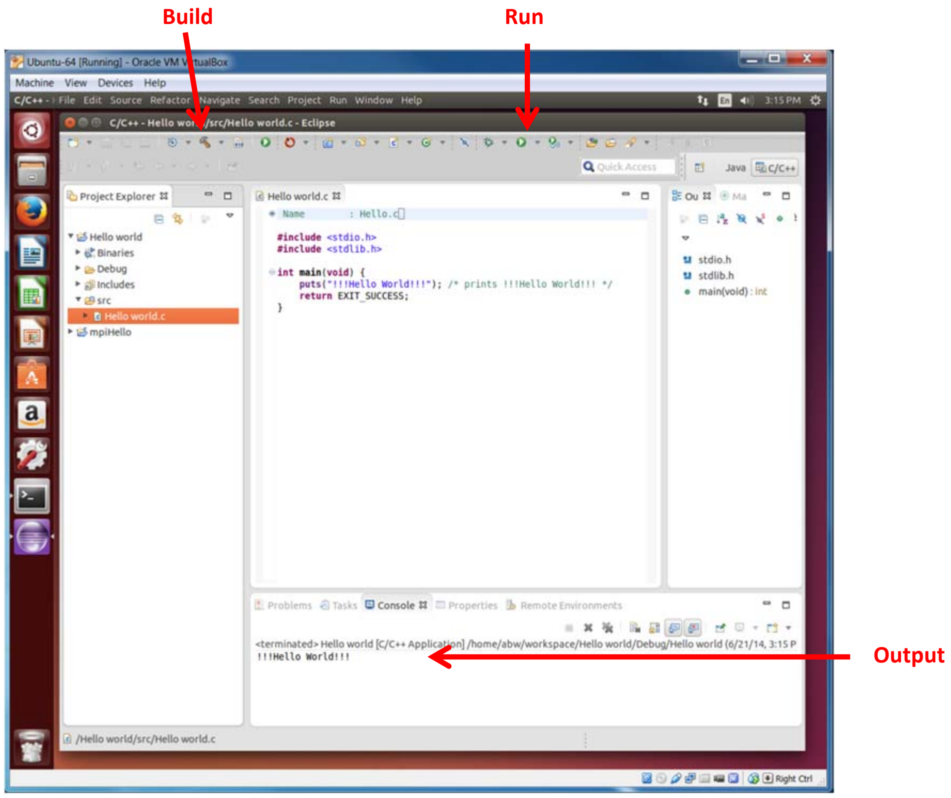
Using Eclipse. Start Eclipse on the command line by typing:

```
eclipse
```

Create a new workspace called **~/ParallelProg/C/workspace** and go to the workbench. We will test the environment using the sample C program (Hello World) given in Eclipse. Create a new C project (**File** > **New** > **C Project**), and select the "**Hello World ANSI C** Project" type and the **gcc** compiler:



Create the “Hello world” C project with a name, built it, and run:



```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
	puts("!!!Hello World!!!"); /* prints !!!Hello World!!! */
	return EXIT_SUCCESS;
}
```

```
<terminated> Hello world [C/C++ Application] /home/abw/workspace/Hello world/Debug/Hello world (6/21/14, 3:15 P
!!!Hello World!!!
```

Parallel programs (OpenMP, MPI, etc) will be tested in subsequent main assignments.

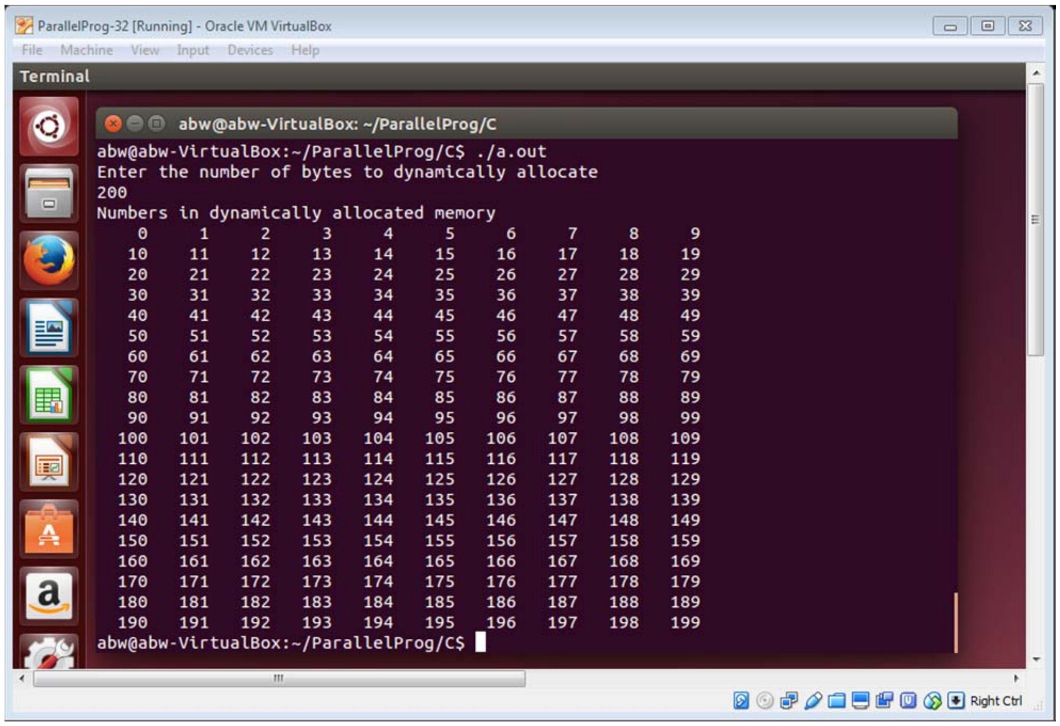
Task 3 Input/output, pointers, and dynamic memory (10%)

In subsequent assignments, you will need to incorporate keyboard input and display output in your programs. You will need to understand pointers and dynamic memory in C. So let us refresh ourselves on these C features. High performance computing as in this course is mostly done with C.

In C, display output is achieved with **printf()**. Keyboard input is achieved with **scanf()**. The **&** symbol, when used with a variable name, is the address operator and returns the address of the location. **scanf** needs the address of the location holding the keyboard input as the second parameter. (The first parameter is the formatting string.)

An array declared as **int A[100];** declares a 1-dimensional array **A** with 100 elements statically, that is, with fixed size of 100 integers at compile time. The array name used alone is a pointer to the first location in the array. C now allows **int A[n];** where **n** is a variable set to a value during runtime but the array is then held on the stack instead of the heap and is more limited in size. To declare an array dynamically in memory whose size can be altered during runtime, use **malloc,** which returns a pointer pointing to the first location of the array. Pointers are declared by prefixing the variable name with a *****, e.g. **int *p;** declares a pointer called **p** pointing to an integer. One can still use the construction **p[i]** to reach the *i*th location of a 1-dimensional array (or ***(p + i)**). Refer to C programming texts or online for more information.

Write a C program that reads an integer, *n*, at the keyboard and dynamically allocates memory for *n* integers using **malloc**. Initialize the *i*th location to *i*. Create the output such as shown below:


ParallelProg-32 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
abw@abw-VirtualBox: ~/ParallelProg/C
Right Ctrl

```
abw@abw-VirtualBox:~/ParallelProg/C$ ./a.out
Enter the number of bytes to dynamically allocate
200
Numbers in dynamically allocated memory
     0     1     2     3     4     5     6     7     8     9
    10    11    12    13    14    15    16    17    18    19
    20    21    22    23    24    25    26    27    28    29
    30    31    32    33    34    35    36    37    38    39
    40    41    42    43    44    45    46    47    48    49
    50    51    52    53    54    55    56    57    58    59
    60    61    62    63    64    65    66    67    68    69
    70    71    72    73    74    75    76    77    78    79
    80    81    82    83    84    85    86    87    88    89
    90    91    92    93    94    95    96    97    98    99
   100   101   102   103   104   105   106   107   108   109
   110   111   112   113   114   115   116   117   118   119
   120   121   122   123   124   125   126   127   128   129
   130   131   132   133   134   135   136   137   138   139
   140   141   142   143   144   145   146   147   148   149
   150   151   152   153   154   155   156   157   158   159
   160   161   162   163   164   165   166   167   168   169
   170   171   172   173   174   175   176   177   178   179
   180   181   182   183   184   185   186   187   188   189
   190   191   192   193   194   195   196   197   198   199
abw@abw-VirtualBox:~/ParallelProg/C$
```

Task 4 Shared directory, if you are using a virtual machine (5%, otherwise extra for native installation)

It is really important that your keep backups of the work. If you are using a virtual machine, follow the instructions "Using Ubuntu including within VirtualBox" from "**Additional Information**" on the course home page on Moodle to set up a shared folder (directory) on your host machine to hold backups of your files and make sure you can transfer files between the host and VM.

Part 4 Connecting to remote server

After programs have been tested on your own computer, in subsequent assignments you may also be asked to connect to the parallel programmer cluster called **cci-gridgw.uncc.edu** and test you programs there using multiple processors. So let us make sure you can do that. First *carefully review* the notes on logging onto the cluster at **"Additional Information"** > **"Using UNC-C cci-gridgw.uncc.edu."** The information on OpenMP and MPI will be used in subsequent assignments.

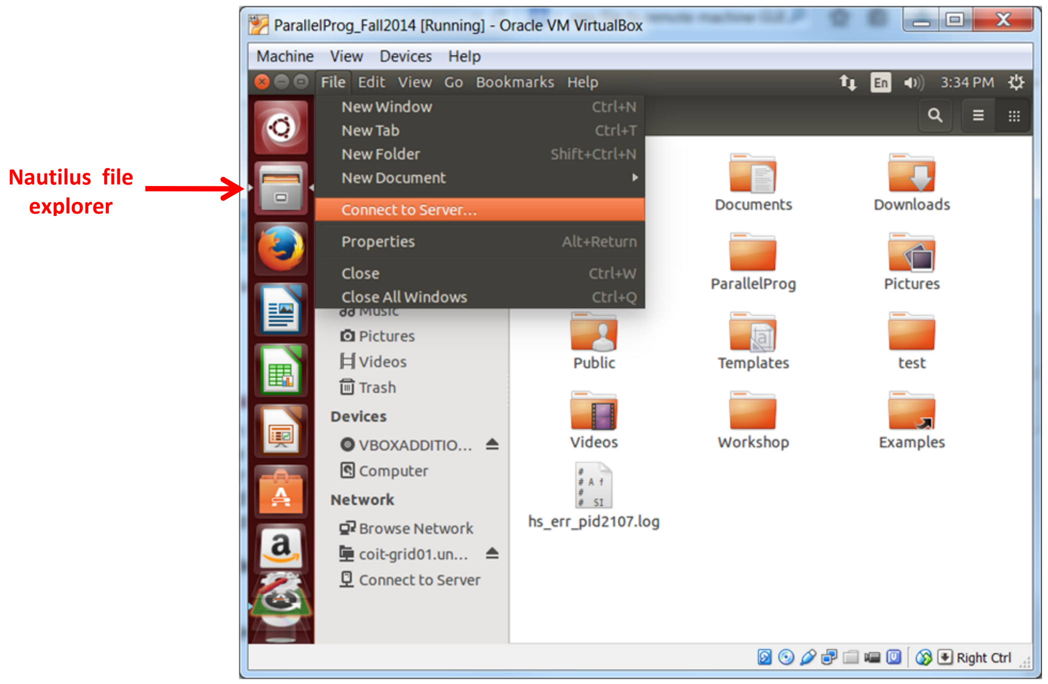
Command line. While it is possible to use clients such as Putty and WinSCP on Windows systems,[4] the most convenient approach for file transfers to and from the virtual machine and the remote server is to make the connection directly from the virtual machine. A virtual machine terminal can be used to access remote severs using the command:

ssh –l *username* cci-gridgw.uncc.edu

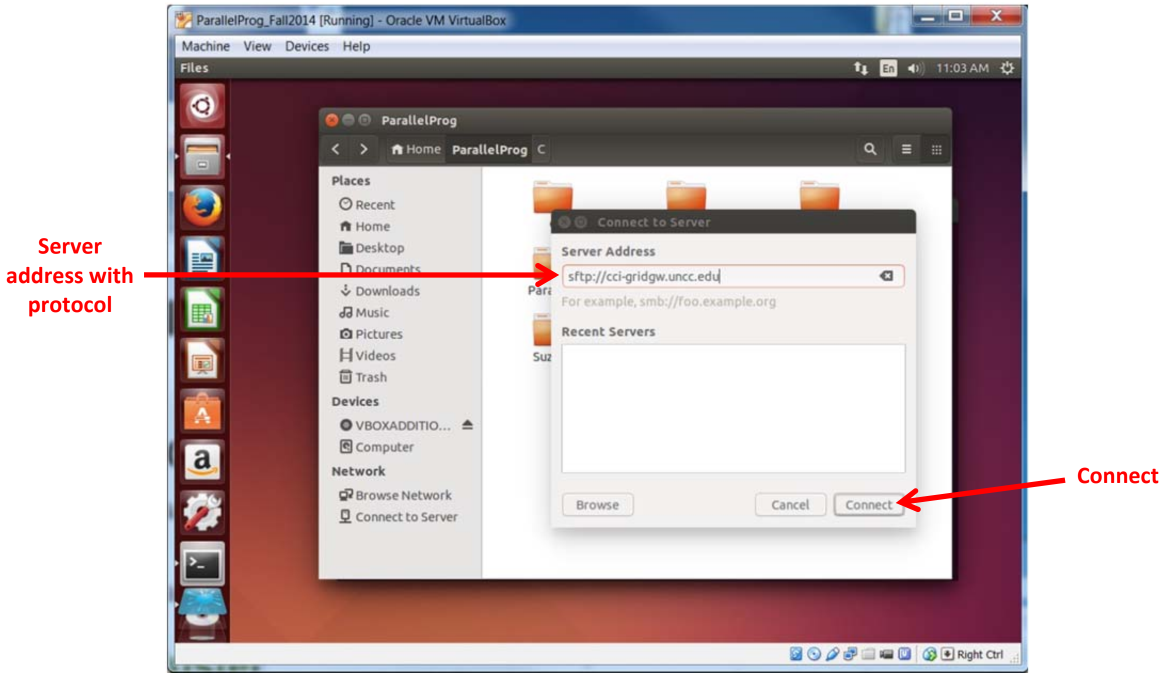
where ***username*** is your user name on the server and **cci-gridgw.uncc.edu** is the server name. Notice the **–l** option is necessary as your username on the remote server will be different to that on you virtual machine (or you could use ***username*@cci-gridgw.uncc.edu**). You will be prompted for your password on the remote server. Then Linux commands can be issued. For file transfers to and from the virtual machine one could use Linux commands such as **scp** but a more convenient approach is to use the Ubuntu Nautilus file explorer. Go **to File** > **Connect to Server**:

Part 4 can only be done once you receive your credentials (user name and password) for the parallel programming cluster. Accounts are set up by the first of classes but may not include those who recently added the course. Contact the TA responsible for accounts if you do not have an account as soon as possible.

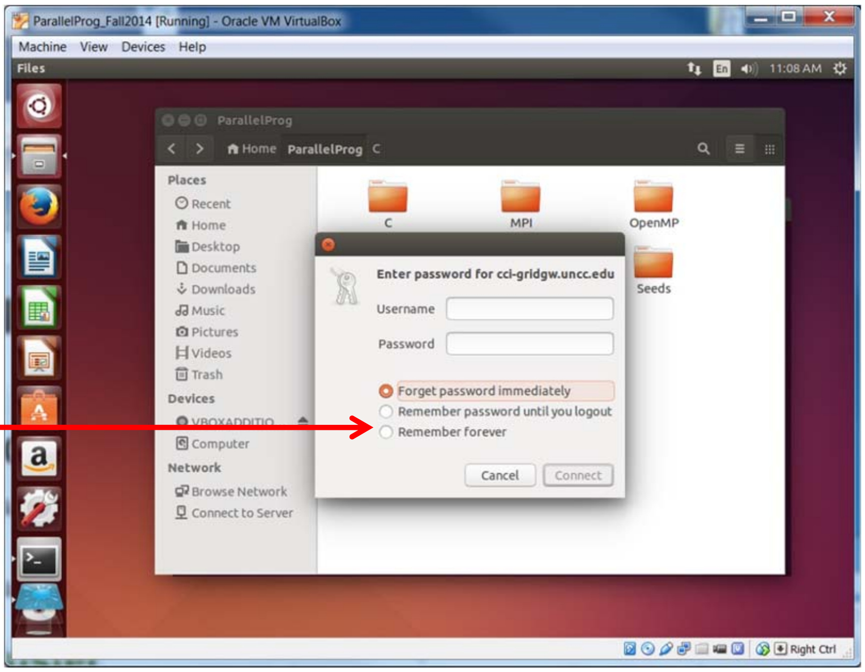
[4] You may wish to install Putty and WinSCP anyway so that you can also access the servers without the VM, see "Additional Information"



Enter the servers address with the **sftp** protocol: **sftp://cci-gridgw.uncc.edu.**



DO NOT set “Remember forever” as this would not be good if you lose your computer or others have access to it.



Enter password for cci-gridgw.uncc.edu

Username

Password

[✓ Forget password immediately] Remember password until you logout Remember forever

Once you have entered your credentials (username and password), you should see your home directory on the remote server and be able to move files between the remote server and the local machine.

If you have any problems at this stage, check the course FAQ for known issues. We have seen isolated issues using the Nautilus file manager on some platforms and that case you may need to use the command line for file transfers. It is sometimes easier to use PuTTy and WinSCP to access remote servers and transfer files to/from a Windows system. Additional information link “Accessing remote servers from a Windows platform (PuTTY, WinSCP, nano).”

Command Line. Create, compile, and execute the **hello.c** C program on the remote server (using the same commands you used on your local virtual machine).

Using Eclipse. It is possible to use Eclipse to make a remote connection and compile and execute programs on the remote server. It is even possible to synchronize local and remote copies of files. (WinSCP also has this feature.) However we will not be using Eclipse for remote access.

Assignment Submission

In subsequent assignments, there will be list of things that must be included in the submission, usually given after the task or part. This list is not comprehensive and you should always include you own code as a listing (not as screenshots). In general, we do not ask you to demonstrate the programs in front of us. IT IS CRITIALLY IMPORTANT THAT YOU DO NOT WORK TOGETHER. ASSIGNMENTS ARE INDIVIDUAL ASSIGMENTS UNLESS SPEFICALLY SPECIFIED OTHERWISE.

What to submit for this pre-assignment

Your submission document must include, but is not limited to, the following:

Your name and whether you are an undergraduate or graduate student.

Part 1 A screenshot showing you have installed the virtual machine, or have installed a native installation.

Part 2

(a) Task 1: A screenshot of the sample C program executing on your computer using the command line.

(b) Task 2: A screenshot of the sample C program executing on your computer using Eclipse

(c) Task 3: A listing of your C program for Task 3 (*not a screenshot*) and a sample screenshot of the output.

(c) Task 4: If you are using a virtual machine, a screenshot showing that you have created shared folder on the host machine and have transferred one file from the guest OS to the host.

Part 3:

(a) A screenshot demonstrating that you can connect to a remote server.

(b) A screenshot of the sample C program executing on the remote server using the command line

Clearly identify each part in your submission document.

Submit **ONE PDF** file containing your submission on the UNC-C Moodle-2 by the due date as described on the course home page. *It is extremely important that you do not submit in any other format as it is possible you will not get any credit. Also each part and sub-part is assigned points so missing a part/sub-part automatically reduces your grade.*