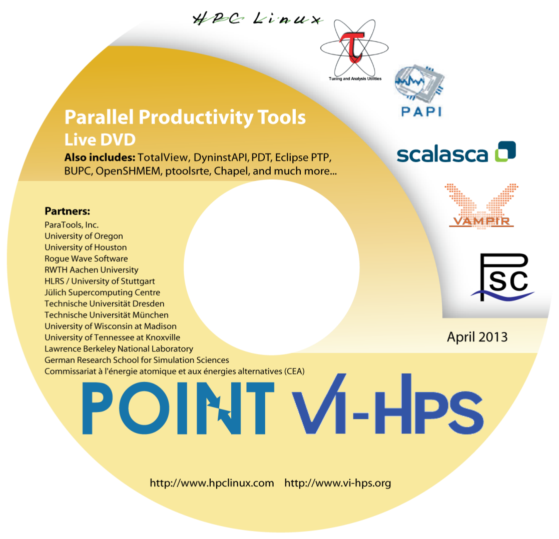
Setup of VirtualBox Development Environment[*]

University of Oregon, Department of Computer and Information Science
Parallel Programming Course

Rev: 1.0, January 2014



Parallel Productivity Tools
Live DVD
Also includes: TotalView, DyninstAPI, PDT, Eclipse PTP, BUPC, OpenSHMEM, ptoolsrte, Chapel, and much more...

Partners:
ParaTools, Inc.
University of Oregon
University of Houston
Rogue Wave Software
RWTH Aachen University
HLRS / University of Stuttgart
Jülich Supercomputing Centre
Technische Universität Dresden
Technische Universität München
University of Wisconsin at Madison
University of Tennessee at Knoxville
Lawrence Berkeley National Laboratory
German Research School for Simulation Sciences
Commissariat à l'énergie atomique et aux énergies alternatives (CEA)

April 2013

http://www.hpclinux.com http://www.vi-hps.org

[*]This document is meant to be used as a reference only. Information presented here was accurate and true at the time of its creation, but all software changes and evolves, so use your best judgment in the creation of your development environment.

Table of Contents

1 Introduction

This document has been designed to go through all of the steps necessary to both install *VirtualBox* and the development environment provided by the *HPCLinux OVA* file, as well as how to load and run the provided example code.

This document is broken into distinct sections to break up the work. First, we will go over when to find and download *VirtualBox* in Section2, and then how to download and initially load the *HPCLinux OVA* file in Section 3. The remainder of the document is dedicated to how to interact with the environment and run the example code in Section 4.

1.1 System Specifications

This section will detail some of the recommended minimum system specifications for running the development environment. The environment is based on Fedora 16, so these minimum specifications are the same.

1.1.1 Processor and memory requirements for x86 Architectures

Fedora 16 may be installed on most ”modern” x86 processors. (There are some “secondary architectures” supported by special interest groups for processors like Power PC, System/390 and ARM).

The minimum processor speed depends on the end use, the method of installation, and the specific hardware. Although some configurations might work on a Pentium 3, most users should consider a Pentium 4 or more modern processor, or the equivalent processor from other manufacturers. Fedora 16 is able to take full advantage of modern, multi-core architectures.

- Minimum RAM for text-mode: 768 MiB
- Minimum RAM for graphical: 768 MiB
- Recommended RAM for graphical: 1152 MiB

1.1.2 Processor and memory requirements for x86_64 architectures

- Minimum RAM for text-mode: 768 MiB
- Minimum RAM for graphical: 768 MiB
- Recommended RAM for graphical: 1152 MiB

1.1.3 Hard disk space requirements for all architectures

For this installation, it is recommended to have approximately 50GB of free space for installation. The *HPCLinux OVA* file is approximately 10GB, and will create a virtual disk once in virtual box for more user storage.

Additional space is also required for any user data, and at least 5% free space should be maintained for proper system operation.

2 Installing VirtualBox

The first step to setup the class development environment on your personal computer will be to download and install *VirtualBox*. *VirtualBox* is free to download and isntall, and is available for most platforms. The following two subsections (2.1 and 2.2) provide basic information about *VirtualBox* and may be safely skipped if you so desire.

2.1 What is VirtualBox

Oracle VM *VirtualBox* is a virtualization software package for x86 and AMD64/Intel64-based computers from Oracle Corporation as part of its family of virtualization products. It was created by innotek GmbH, purchased in 2008 by Sun Microsystems, and now developed by Oracle. It is installed on an existing host operating system as an application; this host application allows additional guest operating systems, each known as a Guest OS, to be loaded and run, each with its own virtual environment.

Supported host operating systems include Linux, Mac OS X, Windows XP, Windows Vista, Windows 7, Windows 8, Solaris, and OpenSolaris. Supported guest operating systems include versions and derivations of Windows, Linux, BSD, OS/2, Solaris and others. Since release 3.2.0, *VirtualBox* also allows limited virtualization of Mac OS X guests on Apple hardware, though OSX86 can also be installed using *VirtualBox*.

Since version 4.1, Windows guests on supported hardware can take advantage of the recently implemented WDDM driver included in the guest additions; this allows Windows Aero to be enabled along with Direct3D support.

2.2 Licensing

With version 4 of *VirtualBox*, released in December 2010, the core package is free software released under GNU General Public License version 2 (GPLv2). This is the fully featured package, excluding some proprietary components not available under GPLv2. These components provide support for USB 2.0 devices, Remote Desktop Protocol (RDP) and Preboot Execution Environment (PXE) for Intel cards and are released as a separate "VirtualBox Oracle VM VirtualBox extension pack" under a proprietary Personal Use and Evaluation License (PUEL), which permits use of the software for personal use, educational use, or evaluation, free of charge.

Oracle defines personal use as any situation in which one person installs the software, and only that individual, and their friends and family, use the software. Oracle does not care if that use is for commercial or non-commercial purposes.[19] Oracle would consider it non-personal use, for example, if a network administrator installed many copies of the software on many different machines, on behalf of many different end-users. That type of situation would require purchasing a special volume license.

Although *VirtualBox* has experimental support for Mac OS X guests, the end user license agreement of Mac OS X does not permit the operating system to run on non-Apple hardware, enforced within the operating system by calls to the Apple System Management Controller (SMC) in all Apple machines, which verifies the authenticity of the hardware.

2.3 Obtaining VirtualBox

VirtualBox can be downloaded at virtualbox.org (`https://wwww.virtualbox.org/wiki/downloads`). Select, download, and install the binary that corresponds to your system.

3 Installing Development Environment

This section will detail how to load the *HPCLinux OVA* file into VirtualBox and how to begin development on the system.

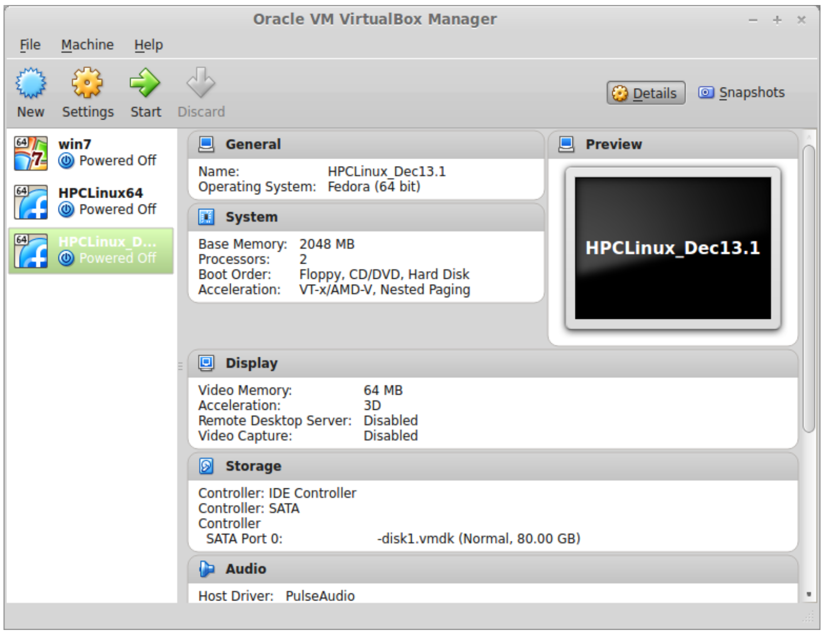
3.1 Download Development Environment 'OVA' File

The most recent *HPCLinux OVA* file can be downloaded form ParaTools at `http://www.paratools.com/HPCLinux#Download`. Select whichever mirror you most prefer, and download the file.

3.2 Using 'OVA' File with VirtualBox

Once the *HPCLinux OVA* file has been retrieved, it is time to import the file into *VirtualBox* using the following steps:

Step 1 Start *VirtualBox*. You will come to a screen similar to that of Figure 1.



win7
Powered Off
HPCLinux64
Powered Off
HPCLinux_D...
Powered Off

General

Name: HPCLinux_Dec13.1
Operating System: Fedora (64 bit)

System

Base Memory: 2048 MB
Processors: 2
Boot Order: Floppy, CD/DVD, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging

Preview

HPCLinux_Dec13.1

Display

Video Memory: 64 MB
Acceleration: 3D
Remote Desktop Server: Disabled
Video Capture: Disabled

Storage

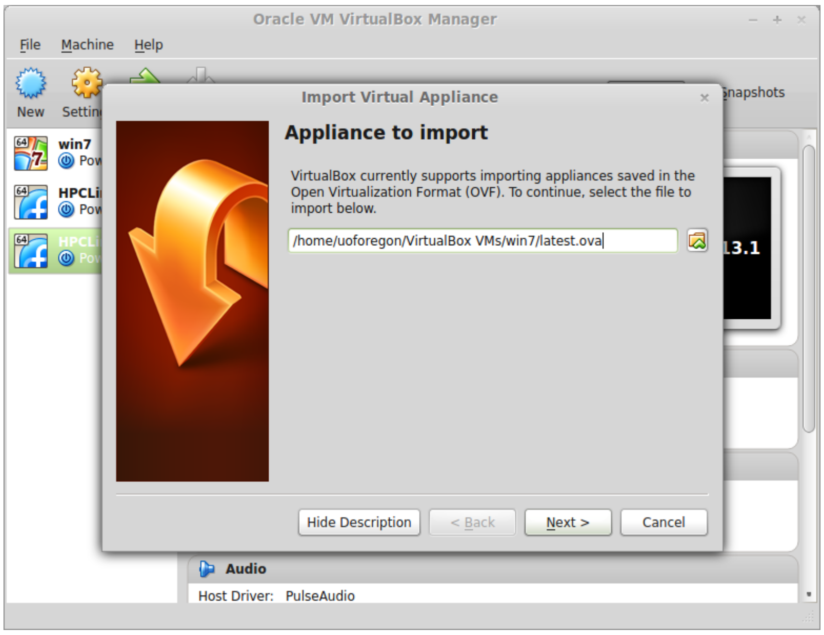
Controller: IDE Controller
Controller: SATA
Controller
SATA Port 0: -disk1.vmdk (Normal, 80.00 GB)

Audio

Host Driver: PulseAudio

Figure 1: Initial VirtualBox start screen

Step 2 Load the 'OVA' file into *VirtualBox*. Click **File ->Import Appliance** and locate your downloaded copy of the 'OVA' file with the file explorer (See Figure 2).



Appliance to import

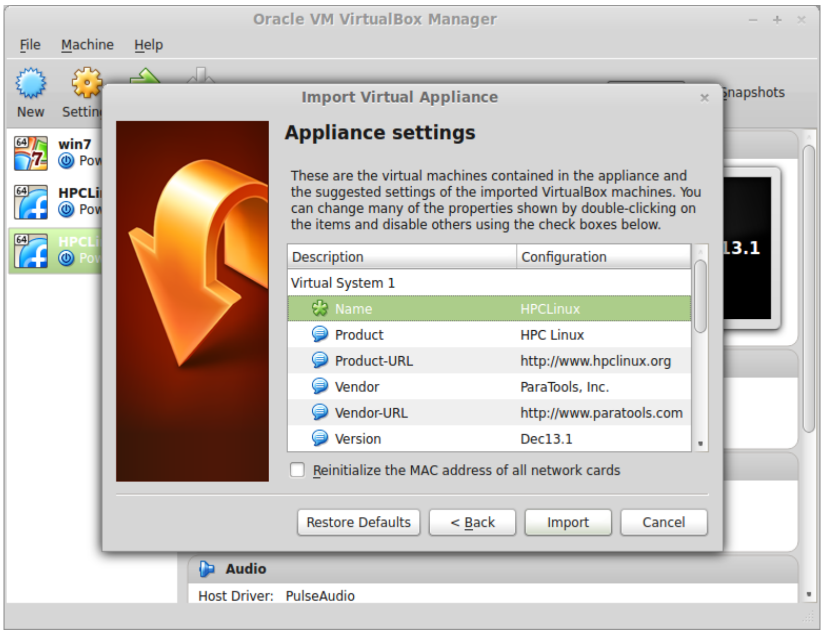
VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

/home/uoforegon/VirtualBox VMs/win7/latest.ova

Host Driver: PulseAudio

Figure 2: Importing the 'OVA' file

Step 3 Click **Next**. Here, you can modify any of the base attributes of the system such as its *Name* number of *cpus* etc. For most systems the defaults will suffice, and can be changed later if needed (See Figure 3).



Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

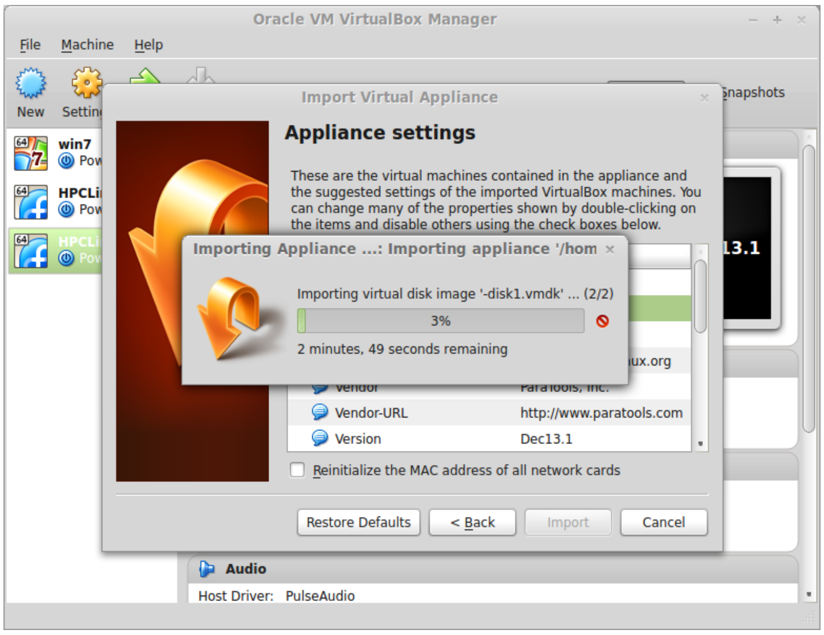
Description	Configuration
Virtual System 1	
Name	HPCLinux
Product	HPC Linux
Product-URL	http://www.hpclinux.org
Vendor	ParaTools, Inc.
Vendor-URL	http://www.paratools.com
Version	Dec13.1

☐ Reinitialize the MAC address of all network cards

Host Driver: PulseAudio

Figure 3: After clicking **Next** during 'OVA' import

Step 4 Click **Import** to finish the import process (See Figure 4). The appliance will then take up to several minutes to import and ready the virtual environment for use.



These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

Importing virtual disk image '-disk1.vmdk' ... (2/2)

3%

2 minutes, 49 seconds remaining

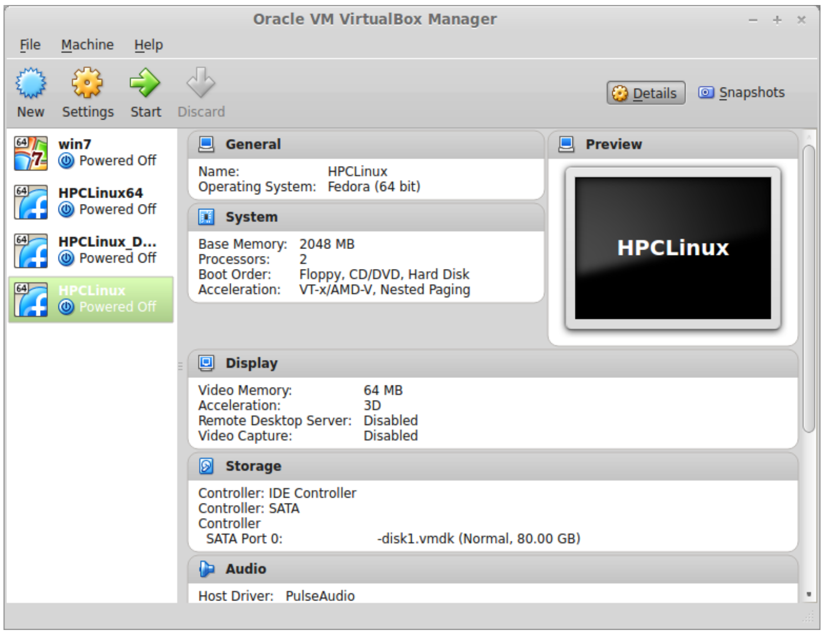
Vendor	ParaTools, Inc.
Vendor-URL	http://www.paratools.com
Version	Dec13.1

☐ Reinitialize the MAC address of all network cards

Host Driver: PulseAudio

Figure 4: After clicking **Import** during 'OVA' import

Step 5 Once the import finishes, you will have a functional virtual environment ready for use (See Figure 5).



Name: HPCLinux
Operating System: Fedora (64 bit)

Base Memory: 2048 MB
Processors: 2
Boot Order: Floppy, CD/DVD, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging

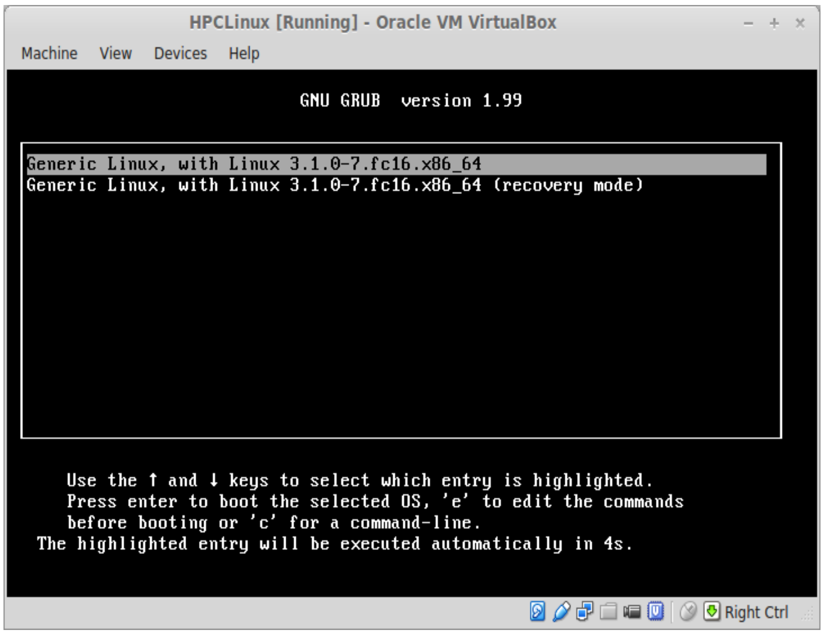
Video Memory: 64 MB
Acceleration: 3D
Remote Desktop Server: Disabled
Video Capture: Disabled

Controller: IDE Controller
Controller: SATA Controller
SATA Port 0: -disk1.vmdk (Normal, 80.00 GB)

Host Driver: PulseAudio

Figure 5: After import finishes environment is ready for use

Step 6 Highlight the virtual environment that you want to use and slick **Start** at the top of the screen. This will launch the virtual environment. You will be taken to a Grub screen and can select OS is loaded. The default selection is what we want, so either hit **enter**, or wait for it to autostart (See Figure 6).


HPCLinux [Running] - Oracle VM VirtualBox
Machine View Devices Help
Right Ctrl

```
GNU GRUB  version 1.99

Generic Linux, with Linux 3.1.0-7.fc16.x86_64
Generic Linux, with Linux 3.1.0-7.fc16.x86_64 (recovery mode)

    Use the ↑ and ↓ keys to select which entry is highlighted.
    Press enter to boot the selected OS, 'e' to edit the commands
    before booting or 'c' for a command-line.
 The highlighted entry will be executed automatically in 4s.
```

Figure 6: After clicking start and the loading of the Grub screen

Step 7 Once the virtual system starts, you will be at the login screen (See Figure 7). Select ***livetau*** and enter the password ****************. Your system is now ready for use. For instructions on running example code see Section 4.

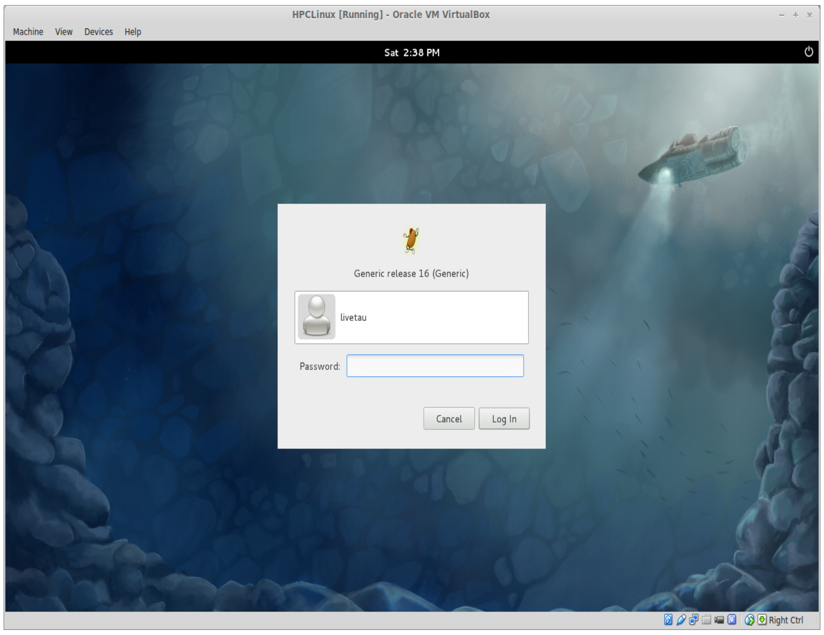

HPCLinux [Running] - Oracle VM VirtualBox
Machine View Devices Help
Sat 2:38 PM
Generic release 16 (Generic)
livetau
Password:
Cancel
Log In
Right Ctrl

Figure 7: System ready for login

4 Environment Interaction and Running Example Code

In this section we will go over how to make and run the example code for this course. This section is important as it goes over specific modules that must be loaded each time you start the system in order to have access to the appropriate libraries and compilers.

4.1 Setup After Startup

After *VirtualBox* development environment is installed and started, there are a few options for compiling and running the example code. This example code is from the book **Structured Parallel Programming**, and is useful to ensure that your system is ready for development. These examples will ensure that the appropriate compilers and libraries are available, and will run several example tests.

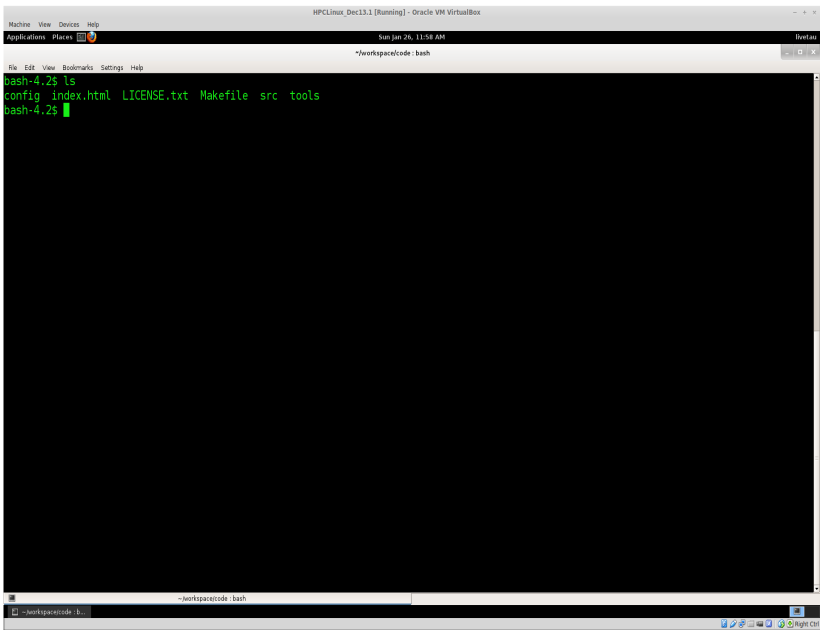
4.1.1 Get the Example Code

The example code for the **Structured Parallel Programming** book is available at `parallelbook.com/student`. Download the most recent version of the code using the link under the heading ***CODE EXAMPLES FROM BOOK***. Once the code is downloaded, extract the top level directory to your most favorite place to work.

4.1.2 Compiling the Examples

The following steps will go over compiling and running the examples and tests with the various available options.

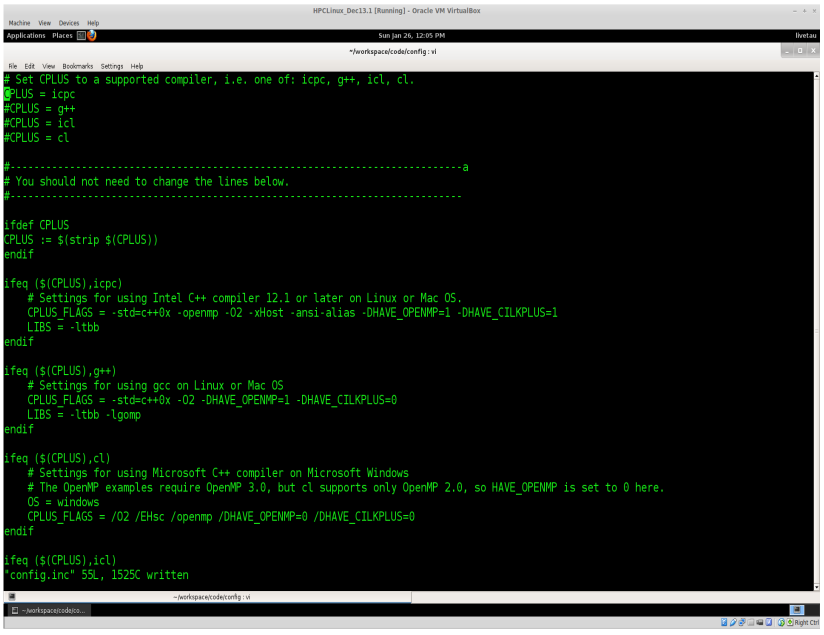
Step 1 Go to the top level of the examples directory, you should see three directories and three files (See Figure 8).



```
bash-4.2$ ls
config  index.html  LICENSE.txt  Makefile  src  tools
bash-4.2$ 
```

Figure 8: Top level directory of the example code

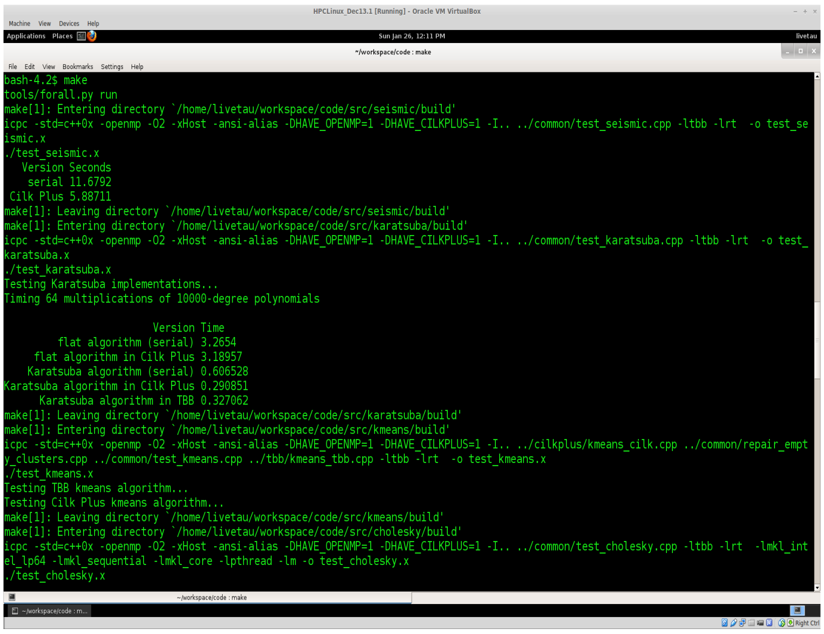
Step 2 The first thing that needs to be done, is the ‘config.inc’ file must be modified. Go to the **config** directory and open **config.inc**. To fully compile the example code and to take advantage of all the features, uncomment the second line in the file and set the compiler to ‘icpc’ (See Figure 9).. This will set the compiler to be the Intel compiler with support for Cilk Plus. If you do not want to compile the Cilk Plus specific code, uncomment the third line and set the compiler to ‘g++’.


HPCLinux_Dec13.1 [Running] - Oracle VM VirtualBox
Machine View Devices Help
Applications Places
Sun Jan 26, 12:05 PM
livetau
~/workspace/code/config : vi
File Edit View Bookmarks Settings Help
~/workspace/code/config : vi
~/workspace/code/co...
Right Ctrl

```
# Set CPLUS to a supported compiler, i.e. one of: icpc, g++, icl, cl.
CPLUS = icpc
#CPLUS = g++
#CPLUS = icl
#CPLUS = cl

#----------------------------------------------------------------------------a
# You should not need to change the lines below.
#----------------------------------------------------------------------------

ifdef CPLUS
CPLUS := $(strip $(CPLUS))
endif

ifeq ($(CPLUS),icpc)
    # Settings for using Intel C++ compiler 12.1 or later on Linux or Mac OS.
    CPLUS_FLAGS = -std=c++0x -openmp -O2 -xHost -ansi-alias -DHAVE_OPENMP=1 -DHAVE_CILKPLUS=1
    LIBS = -ltbb
endif

ifeq ($(CPLUS),g++)
    # Settings for using gcc on Linux or Mac OS
    CPLUS_FLAGS = -std=c++0x -O2 -DHAVE_OPENMP=1 -DHAVE_CILKPLUS=0
    LIBS = -ltbb -lgomp
endif

ifeq ($(CPLUS),cl)
    # Settings for using Microsoft C++ compiler on Microsoft Windows
    # The OpenMP examples require OpenMP 3.0, but cl supports only OpenMP 2.0, so HAVE_OPENMP is set to 0 here.
    OS = windows
    CPLUS_FLAGS = /O2 /EHsc /openmp /DHAVE_OPENMP=0 /DHAVE_CILKPLUS=0
endif

ifeq ($(CPLUS),icl)
"config.inc" 55L, 1525C written
```

Figure 9: Uncommenting the compiler to allow for CILK Plus, TBB, and OpenMP code

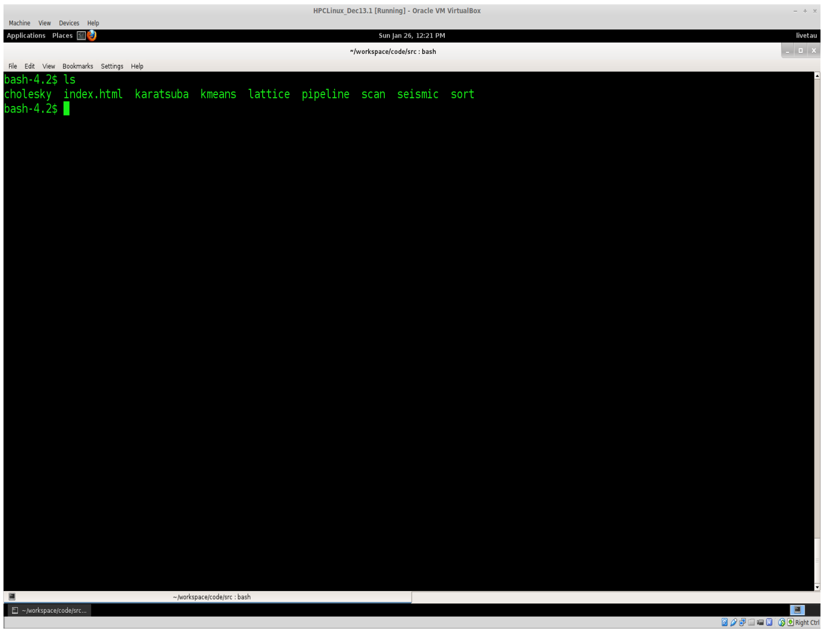
Step 3 Now, go back to the top level directory and type **make**. This will compile the example code with the compiler option that was set in *Step 2* and run some of the test programs (See Figure 10).



```
bash-4.2$ make
tools/forall.py run
make[1]: Entering directory `/home/livetau/workspace/code/src/seismic/build'
icpc -std=c++0x -openmp -O2 -xHost -ansi-alias -DHAVE_OPENMP=1 -DHAVE_CILKPLUS=1 -I.. ../common/test_seismic.cpp -ltbb -lrt  -o test_se
ismic.x
./test_seismic.x
   Version Seconds
    serial 11.6792
 Cilk Plus 5.88711
make[1]: Leaving directory `/home/livetau/workspace/code/src/seismic/build'
make[1]: Entering directory `/home/livetau/workspace/code/src/karatsuba/build'
icpc -std=c++0x -openmp -O2 -xHost -ansi-alias -DHAVE_OPENMP=1 -DHAVE_CILKPLUS=1 -I.. ../common/test_karatsuba.cpp -ltbb -lrt  -o test_
karatsuba.x
./test_karatsuba.x
Testing Karatsuba implementations...
Timing 64 multiplications of 10000-degree polynomials

                         Version Time
         flat algorithm (serial) 3.2654
     flat algorithm in Cilk Plus 3.18957
    Karatsuba algorithm (serial) 0.606528
Karatsuba algorithm in Cilk Plus 0.290851
      Karatsuba algorithm in TBB 0.327062
make[1]: Leaving directory `/home/livetau/workspace/code/src/karatsuba/build'
make[1]: Entering directory `/home/livetau/workspace/code/src/kmeans/build'
icpc -std=c++0x -openmp -O2 -xHost -ansi-alias -DHAVE_OPENMP=1 -DHAVE_CILKPLUS=1 -I.. ../cilkplus/kmeans_cilk.cpp ../common/repair_empt
y_clusters.cpp ../common/test_kmeans.cpp ../tbb/kmeans_tbb.cpp -ltbb -lrt  -o test_kmeans.x
./test_kmeans.x
Testing TBB kmeans algorithm...
Testing Cilk Plus kmeans algorithm...
make[1]: Leaving directory `/home/livetau/workspace/code/src/kmeans/build'
make[1]: Entering directory `/home/livetau/workspace/code/src/cholesky/build'
icpc -std=c++0x -openmp -O2 -xHost -ansi-alias -DHAVE_OPENMP=1 -DHAVE_CILKPLUS=1 -I.. ../common/test_cholesky.cpp -ltbb -lrt  -lmkl_int
el_lp64 -lmkl_sequential -lmkl_core -lpthread -lm -o test_cholesky.x
./test_cholesky.x
```

Figure 10: Uncommenting the compiler to allow for CILK Plus, TBB, and OpenMP code

Step 4 If there were no errors during the build and the tests completed, you should have an environment that is ready for development. If you want to look at or run any of the test programs individually they are available under the **src** directory (See Figure 11).


HPCLinux_Dec13.1 [Running] - Oracle VM VirtualBox
Machine View Devices Help
Applications Places
Sun Jan 26, 12:21 PM
livetau
~/workspace/code/src : bash
File Edit View Bookmarks Settings Help
~/workspace/code/src : bash
~/workspace/code/src...
Right Ctrl

```
bash-4.2$ ls
cholesky  index.html  karatsuba  kmeans  lattice  pipeline  scan  seismic  sort
bash-4.2$
```

Figure 11: Test code directories available under **src**