Using Eclipse for C, MPI, and Suzaku

Modification date May 30, 2015 B. Wilkinson

General. Eclipse is an IDE with plugs for various programming environments including Java and C. Eclipse-PTP (Eclipse with the tools for Parallel Applications Developers) provides features to compile MPI programs. Eclipse is installed in the virtual machine with both Java and C perspectives including the tools for Parallel Applications Developers. Eclipse can also be used on a local computer to develop and execute programs on a remote cluster.

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

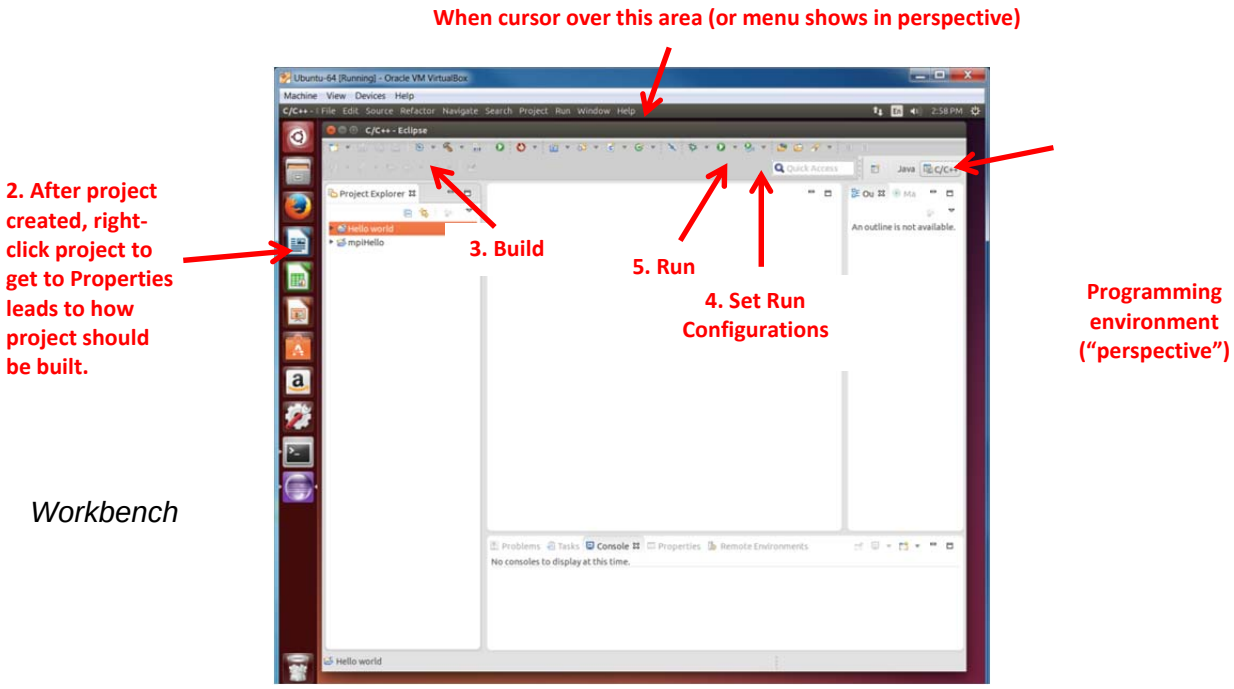
eclipse

Eclipse organizes program development activities into projects in a "workspace" directory. The workspace directory first needs to be created or selected when Eclipse is started. Different workspaces can be selected afterwards. (**File > Switch Workspace**).

Then, the basic steps to compile and execute a program are:

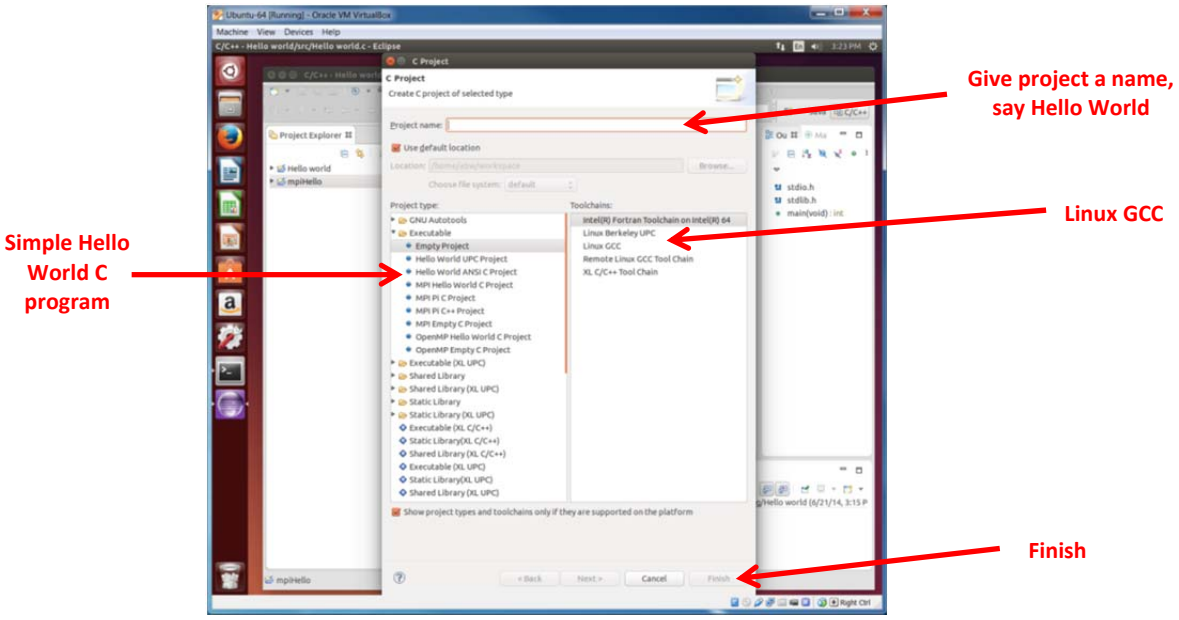
1. Create a project with the source file and any required libraries
2. Set how to build (compile) project in **Properties > ... Build**
3. Build project (compile to create executable)
4. Set how to execute compiled program in **Run Configurations**
5. **Run** (execute) using the specified run configurations

We will demonstrate these steps in various programming environments ("perspectives").

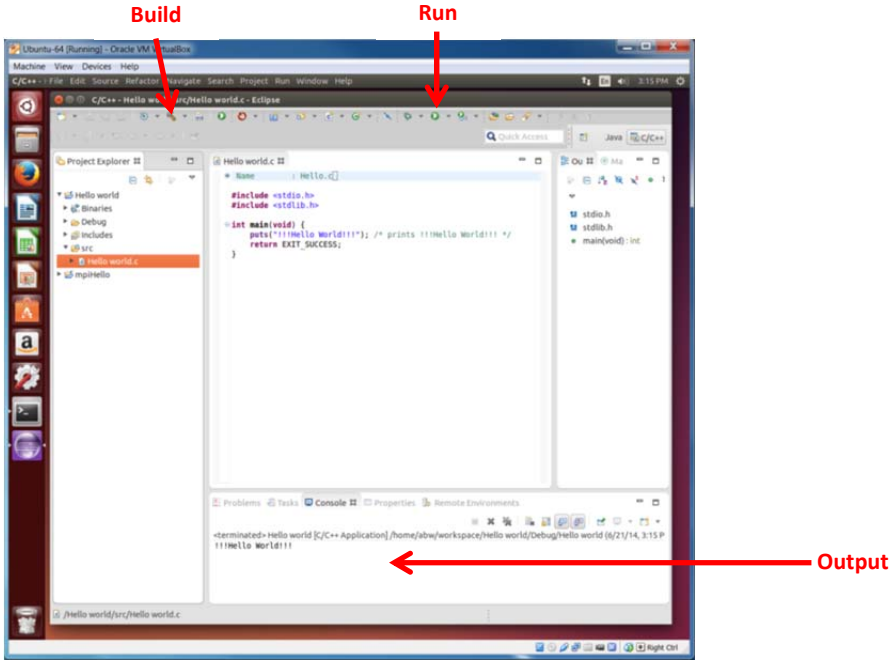


C Programming

Select a workbench location (say **~/ParallelProg/C/workspace).** To demonstrate using Eclipse for C programs, we will test the environment with 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 **gcc** compiler. Create the "Hello world" C project with a name:



For simple C programs, the build path is already set up, so we can simply select build. We can then select **Run As > Local C/C++ Application**. This will create a Run Configuration called Hello World Configuration and run the program.

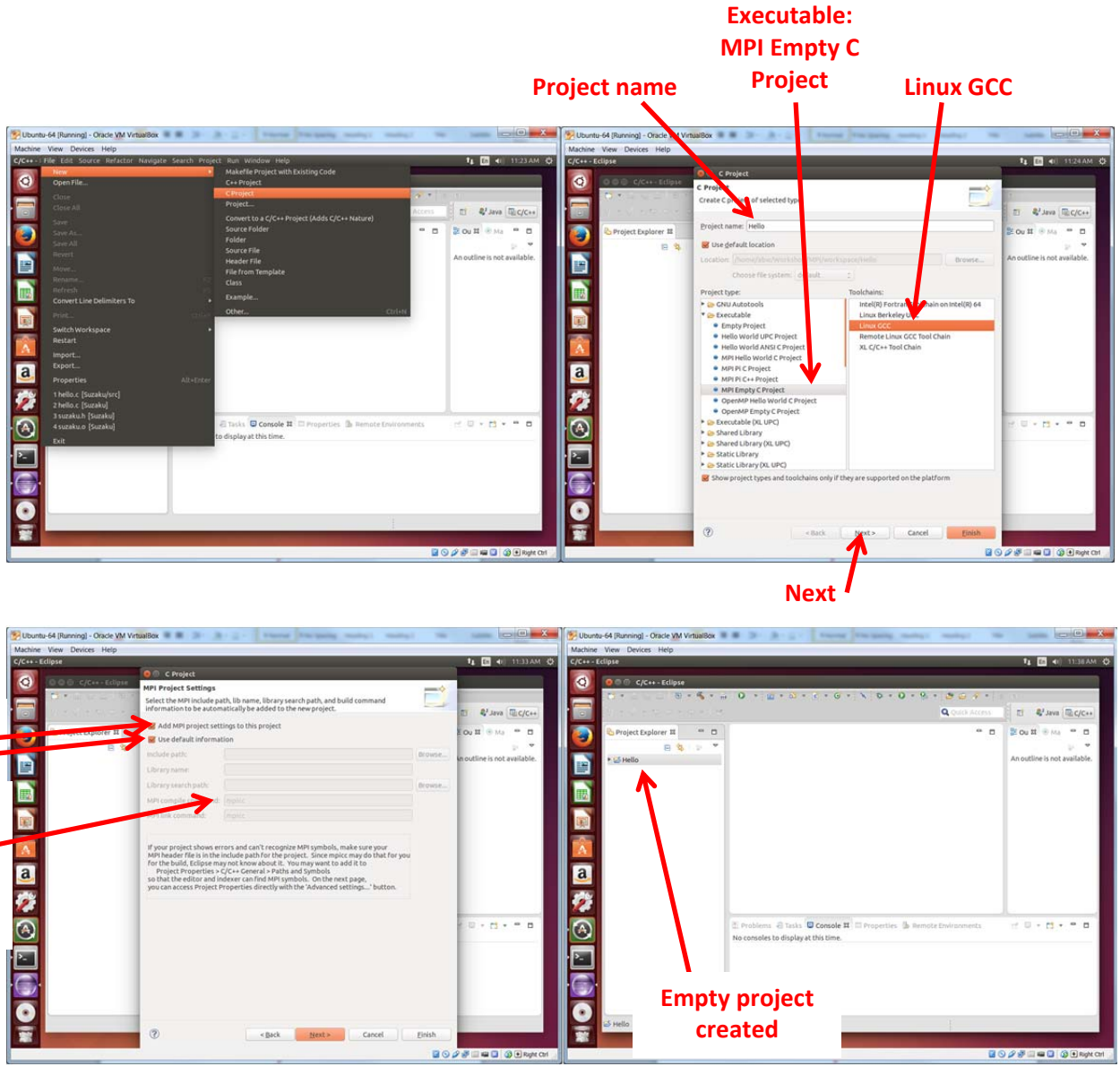


MPI

Eclipse-PTP (Eclipse with the tools for Parallel Applications Developers) can be used to compile and execute MPI programs. MPI programs are done as C/C++ projects with build/compilation for MPI pre-configured in Eclipse-PTP. Programs here will be C projects. Select a workbench location (say **~/ParallelProg/MPI/workspace)**.

(a) Creating project and adding source files

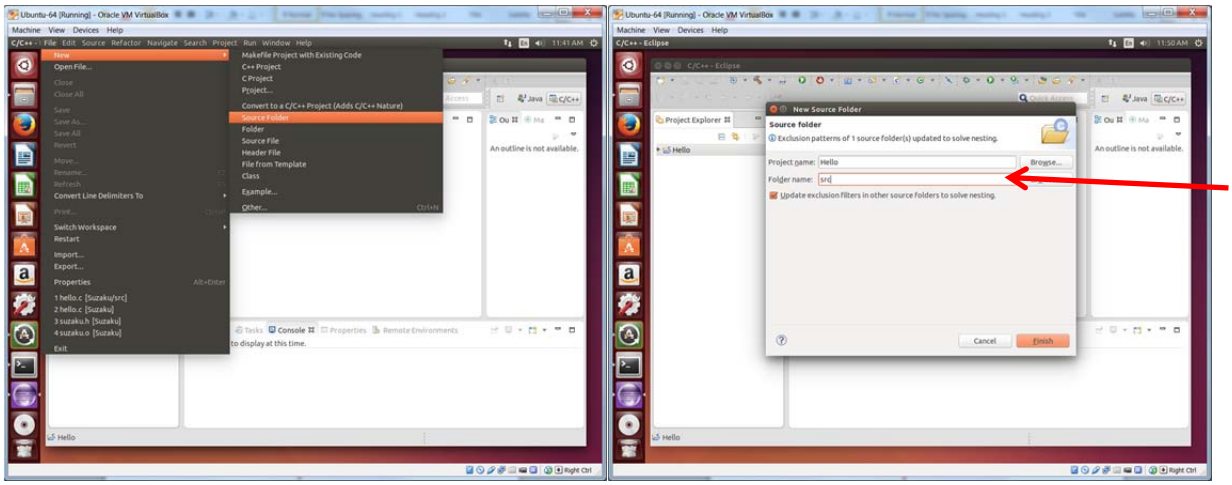
Create new C project (**File > New > C project**) called **Hello** of type "**MPI Empty C project**"[1] with default settings:

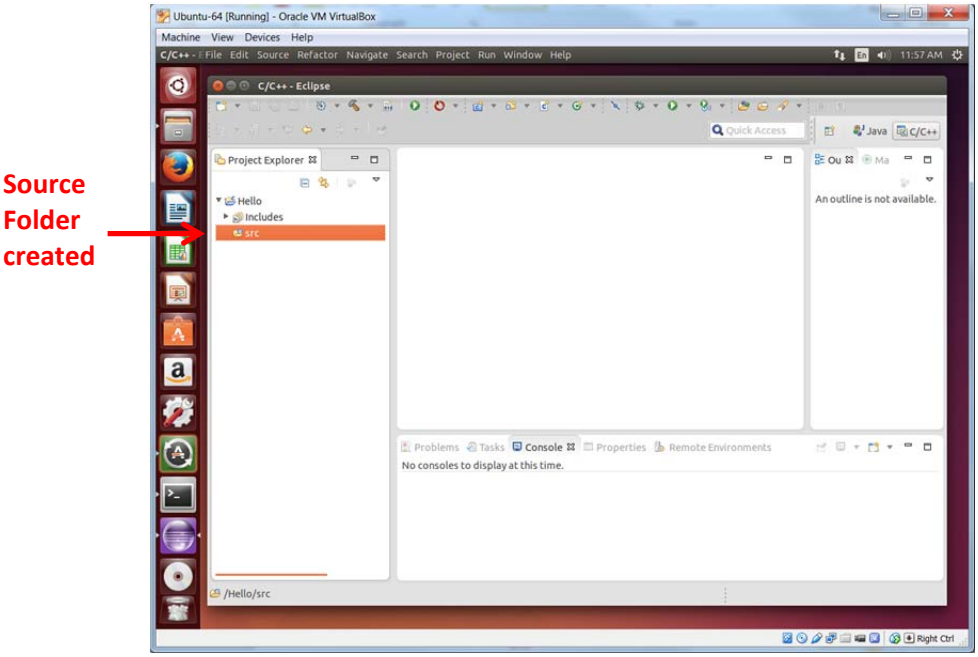


[1] Eclipse comes with sample programs pre-installed for C, OpenMP, and MPI ("Hello World" and MPI Pi), which are useful for testing the environment.

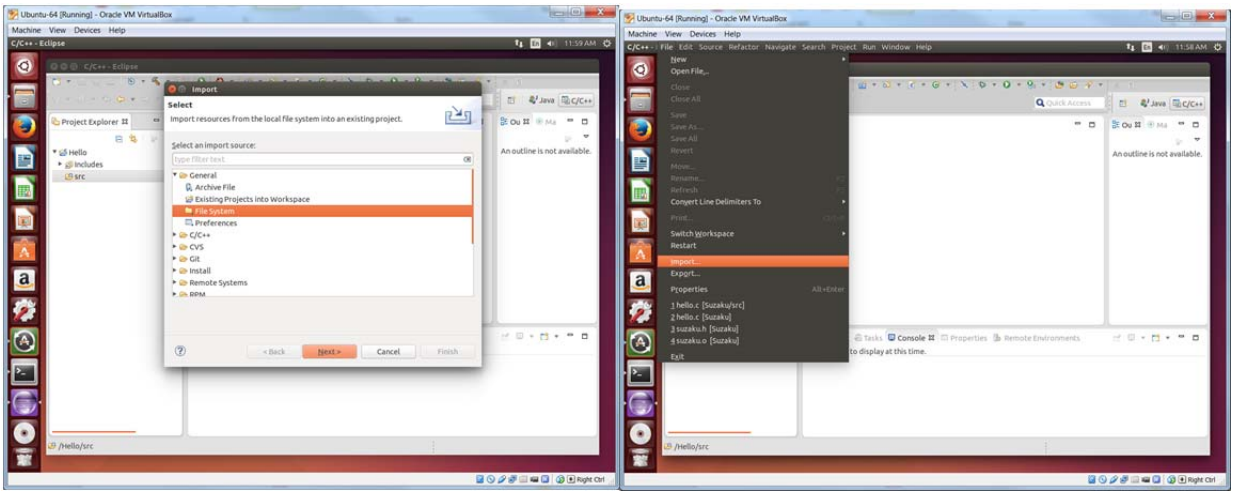
Adding program source file

Select Hello project and create a source folder called **src (File > New > Source Folder** or **right click project > New > Source Folder)**:

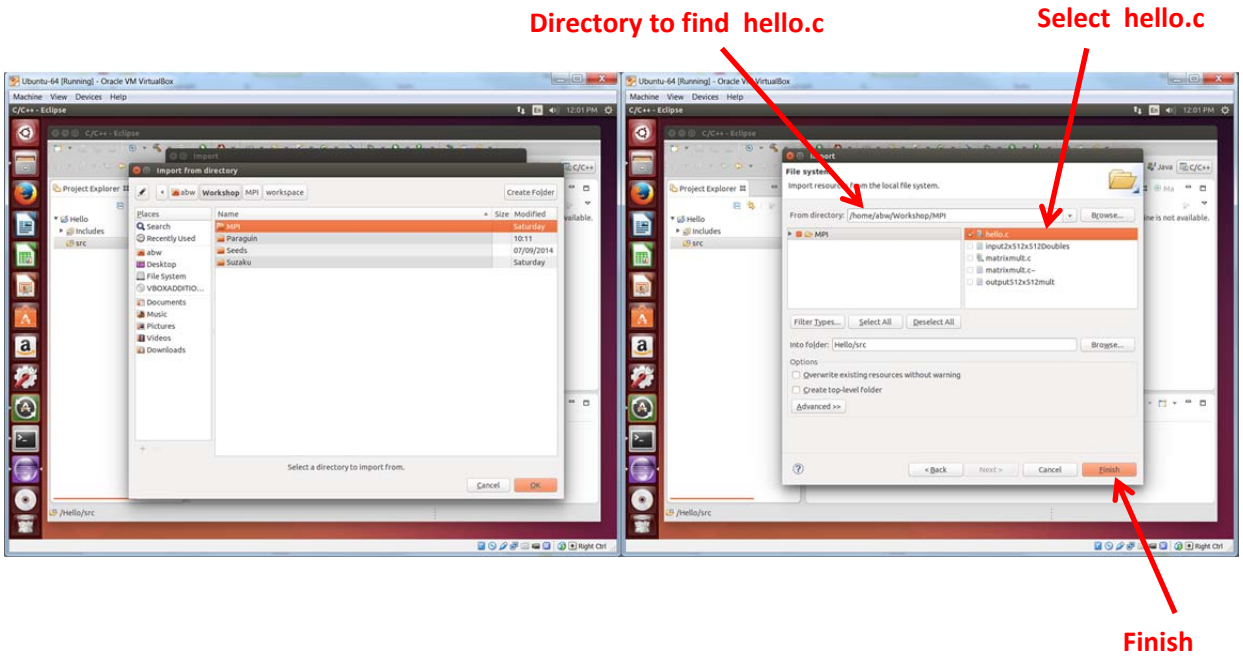




Expand the **Hello** project and select the newly generated **src** folder. Select **File > Import > General > File System:**



From **"Import from directory"**, browse for the directory that holds your **hello.c** files (**~/Workshop/MPI/)**. Click **OK**. Select the **hello.c** file to copy into the **MPI/src** project folder. [2]

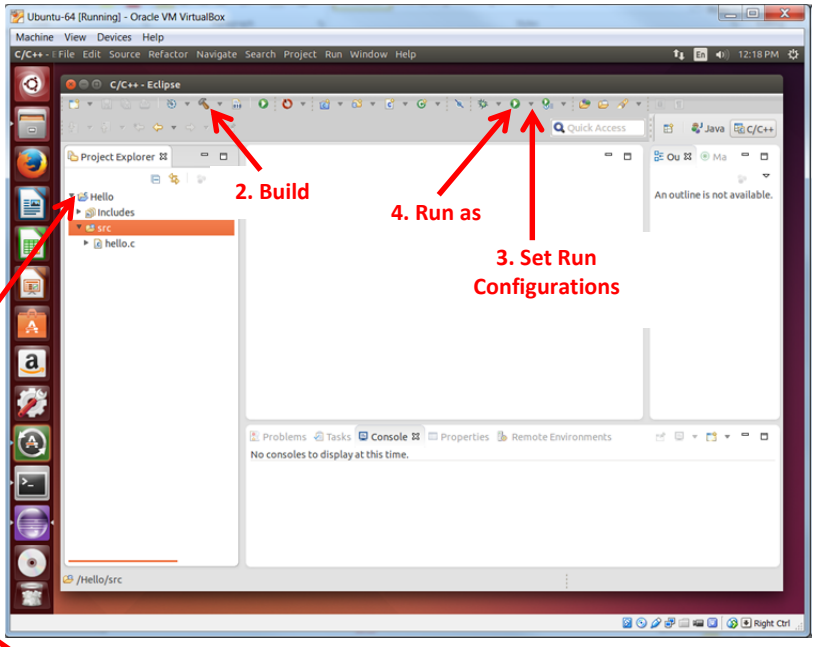


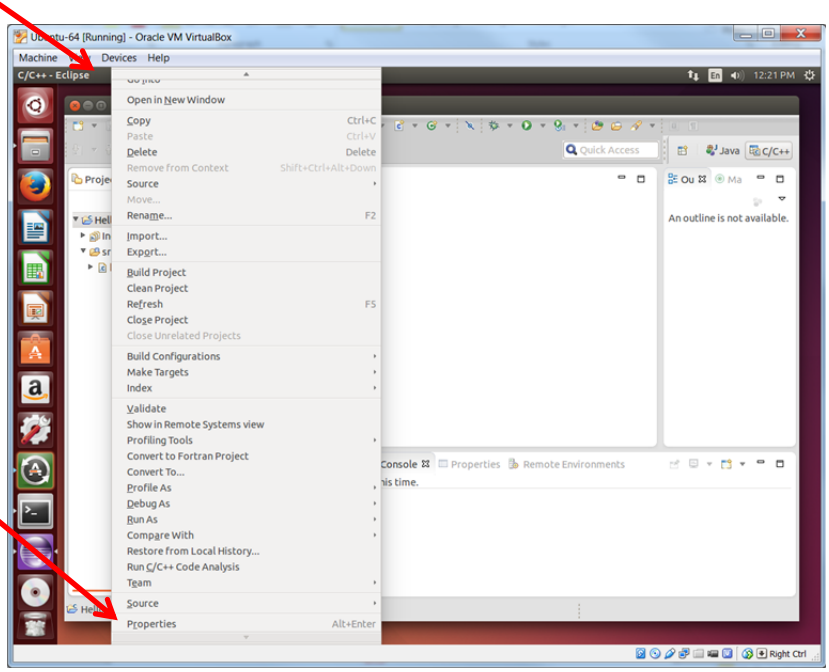
[2] In a later example, we will demonstrate using a link to the source file at their original location rather than copying it, which is usually better.

(b) Build and Compile

Basic steps:

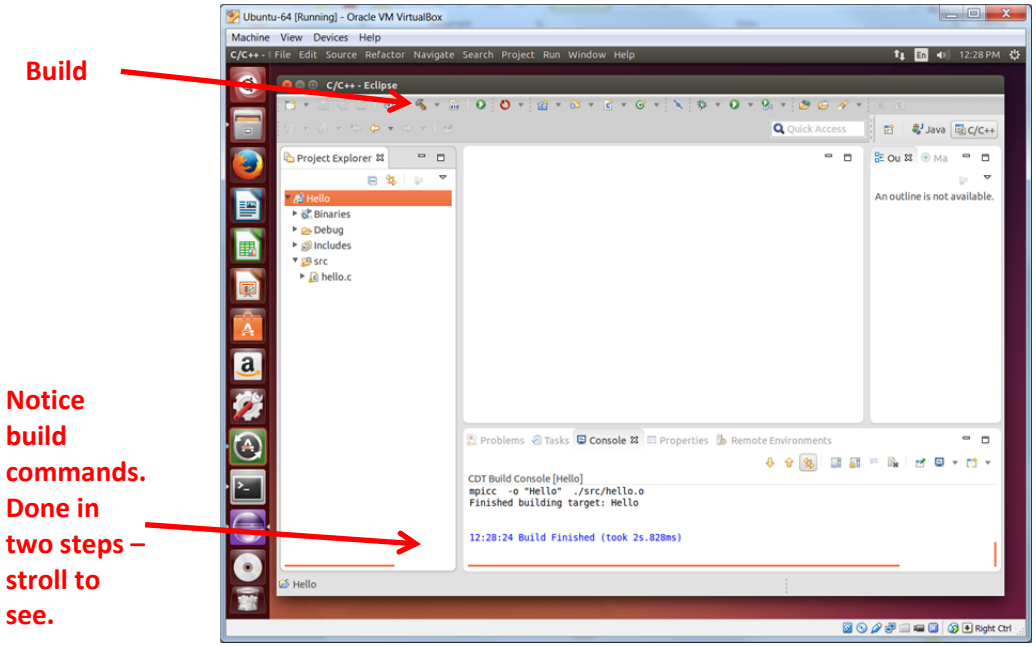
1. Set how to build (compile) project in **Properties > ... Build**
2. Build project (compile to create executable)
3. Set how to execute compiled program in **Run Configurations**
4. **Run** (execute) using the specified run configurations





Build project

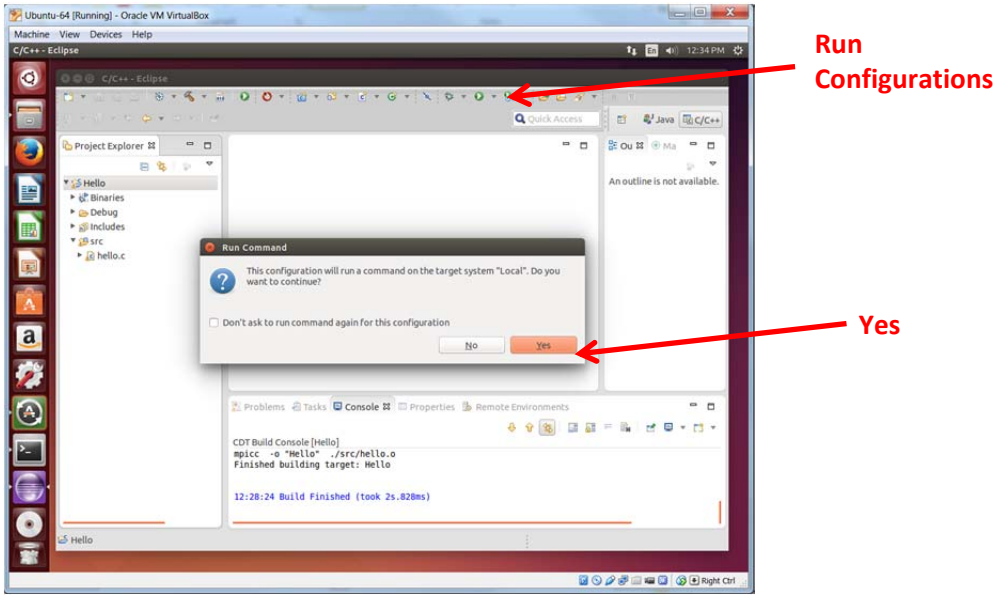
Click **Build** icon (Hammer) to build the project (default “Debug” option):



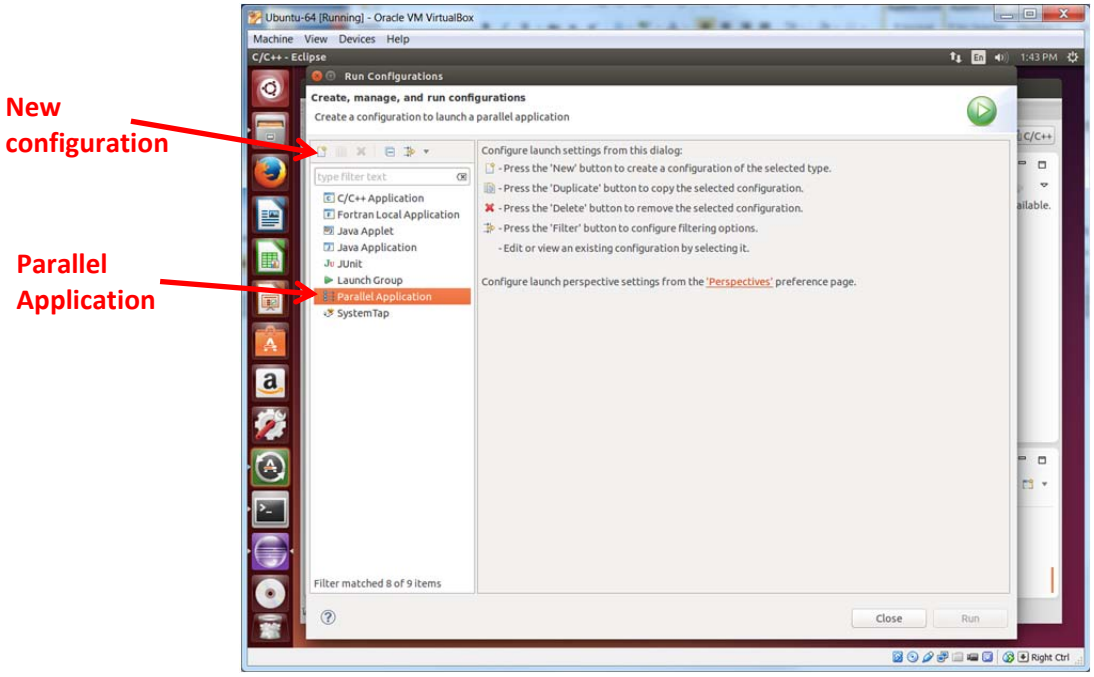
Although build will happen automatically when a project is executed next, sometimes it is handy to know if there are any build errors first.

Execution

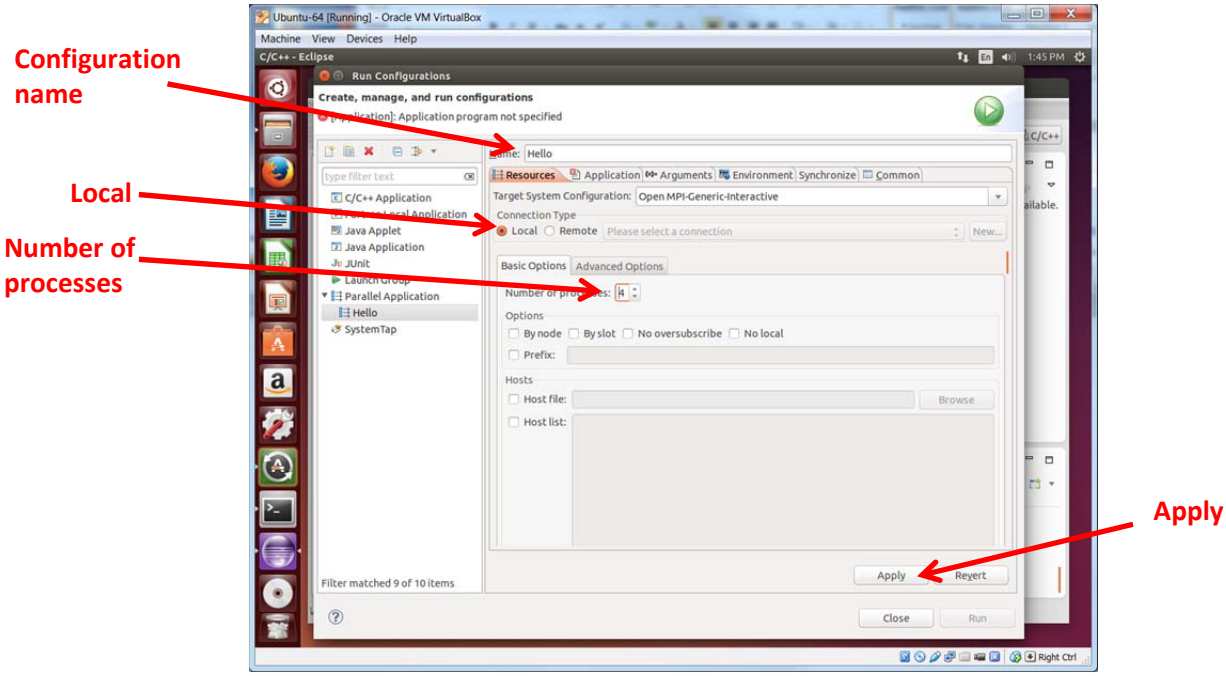
To execute the program, first the **Run Configurations** need to be set up that specify local execution, the software environment, etc. Select **Run Configurations**



Select “Parallel Application” and click the new configuration button:

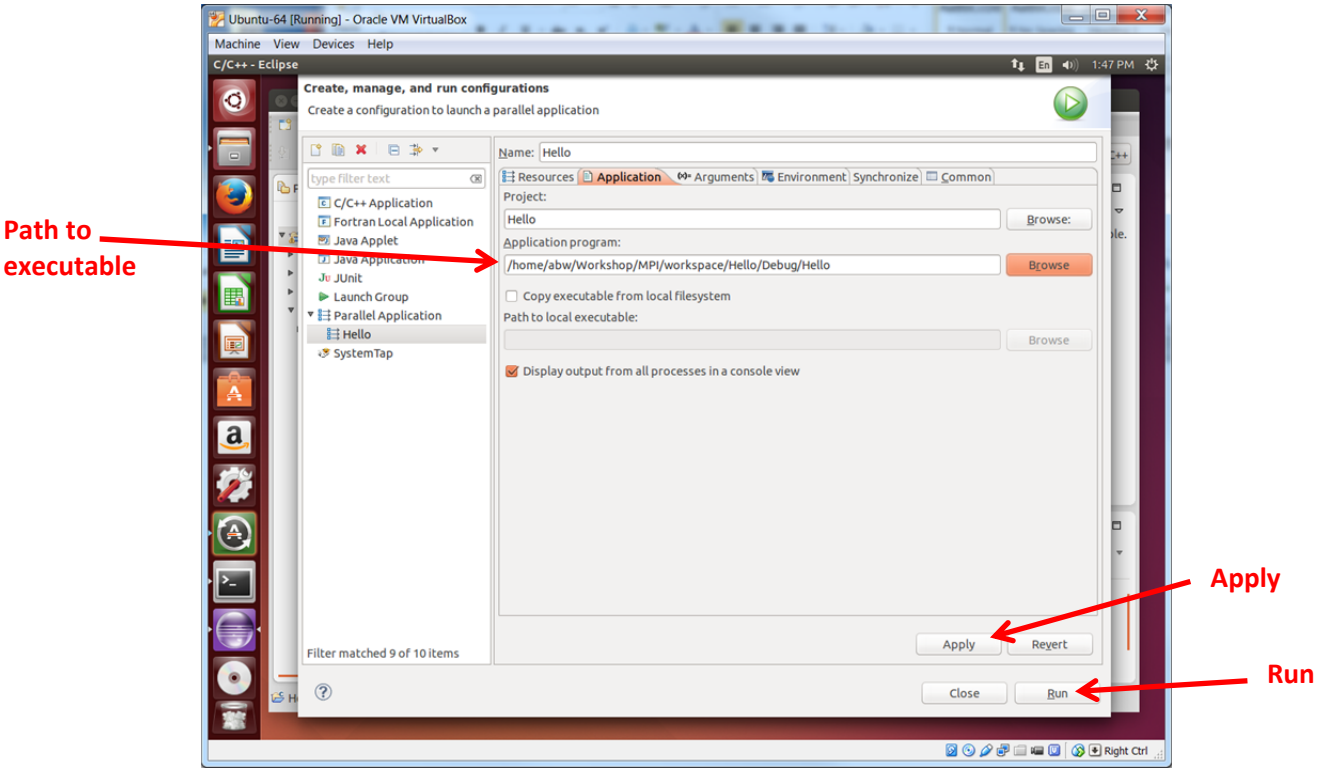


Create a new run configuration called **Hello**. In the *Resource* tab, select the Target Type as “**Open-MPI-Generic-Interactive**”, the connection type as “**Local”**.[3] Set the number of processes to say 4. Apply.

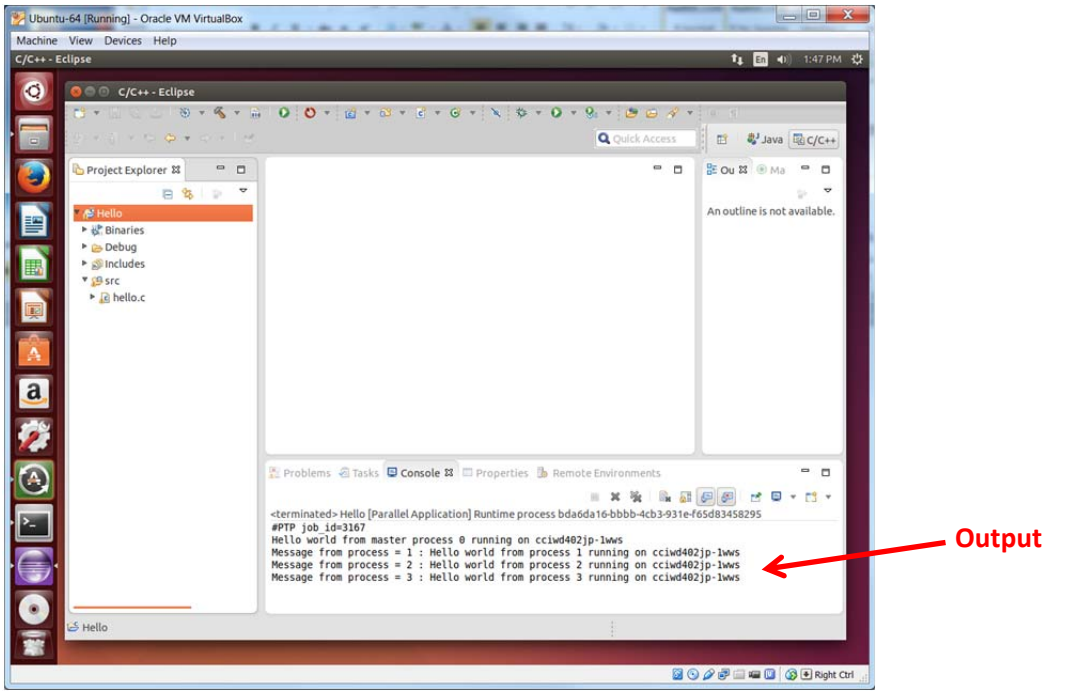


[3] Selecting “Local” will generate a message confirming you want this and create local resources to do this. Select “Don’t ask to run command again for this configuration” in the Run Command message when it appears to stop the message. The message is most relevant when doing both local and remote executions.

In the *Application* tab, set the Project name to “Hello” and browse for the full path to the executable.

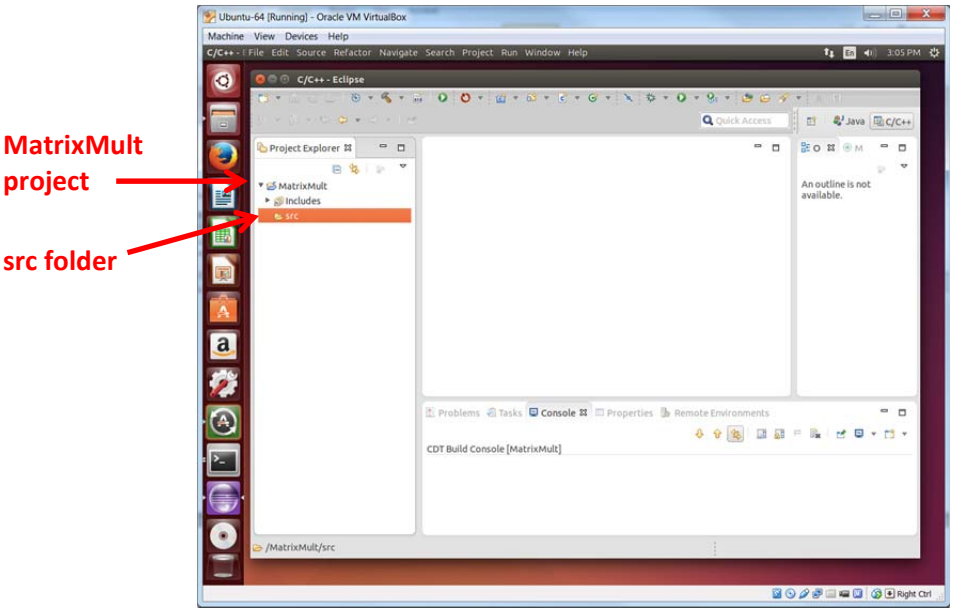


Click **RUN**. (If **RUN** is greyed out, there are build or compile errors that prevent execution.)

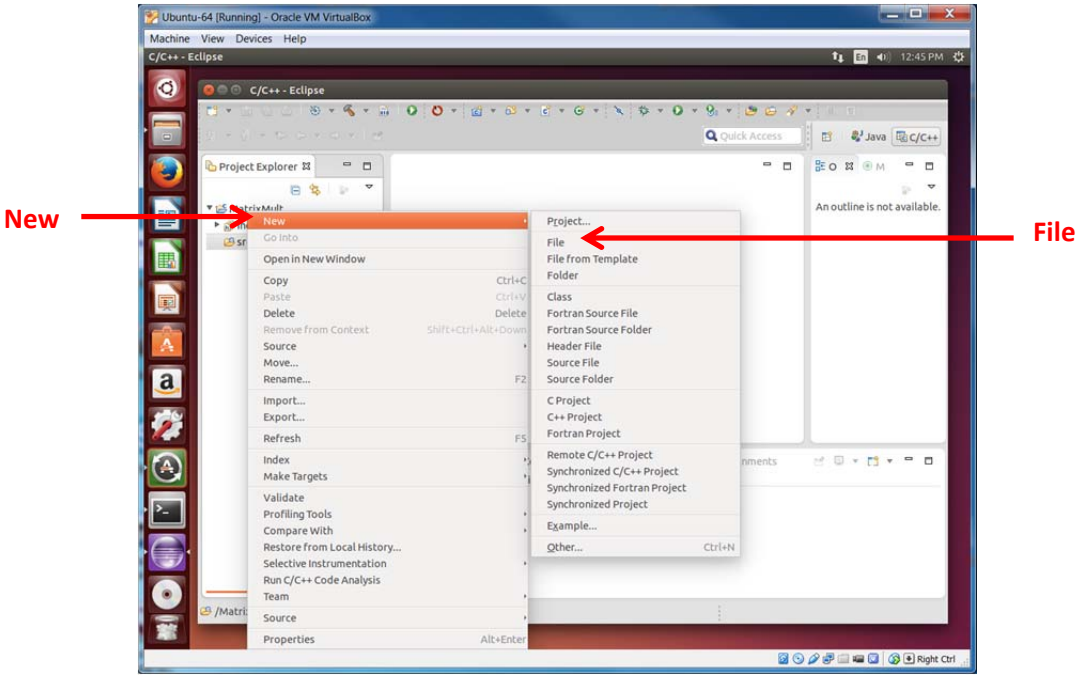


Matrix Multiplication

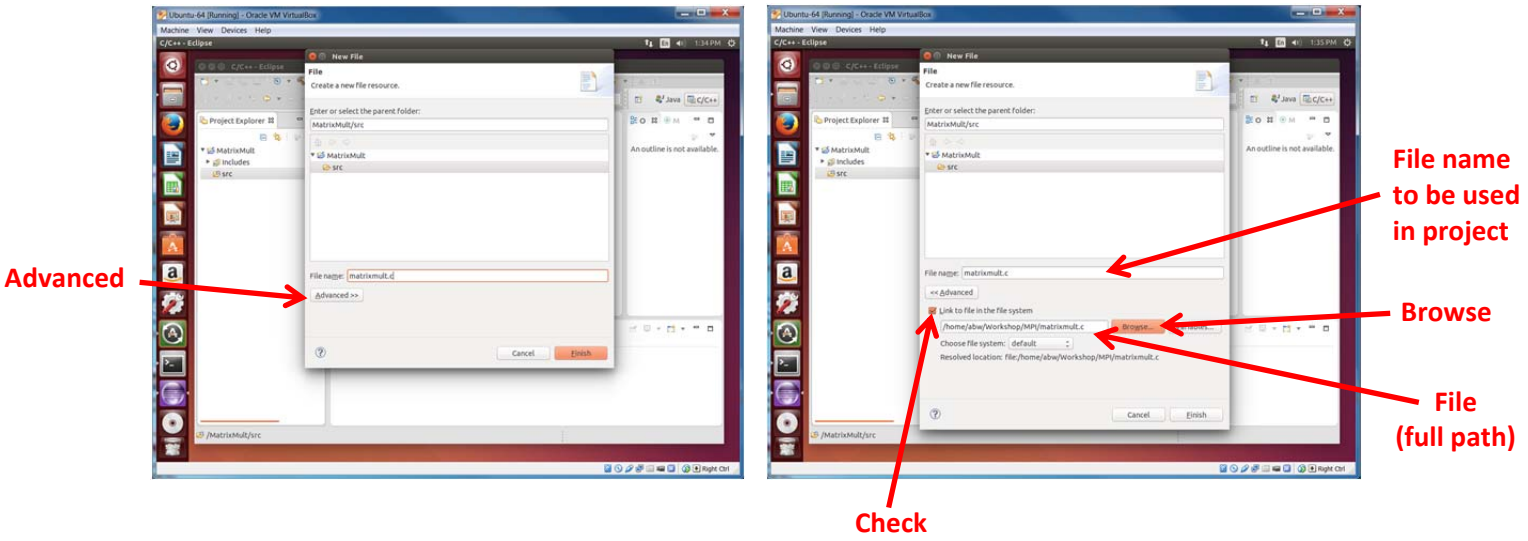
We will go through the steps, this time not copying the source file but referring to it in the original location, which generally is a better approach. We will also set up the program to read an input file and redirect the output to a file. The name of the input file is set as an argument in the Run Configuration. Create an MPI project called **MatrixMult** and a source directory called **src**:



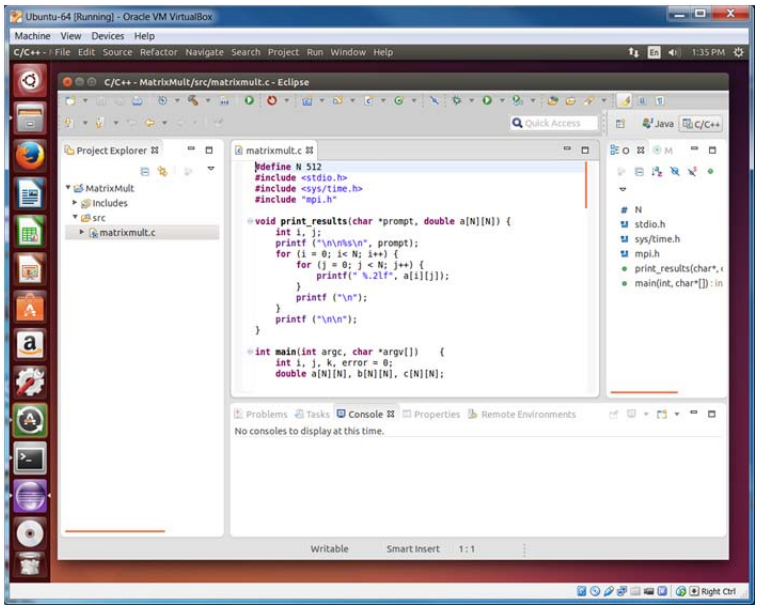
Now we will link source file **matrixmult.c** rather than copy it. Select the source folder **src**, right click, and select **NEW > FILE:**



Click on **Advanced>>** and enter the file name (**matrixmult.c**), check **"Link to file in the file system**" box and enter the full path to the source file (browse for file) and **Finish**:

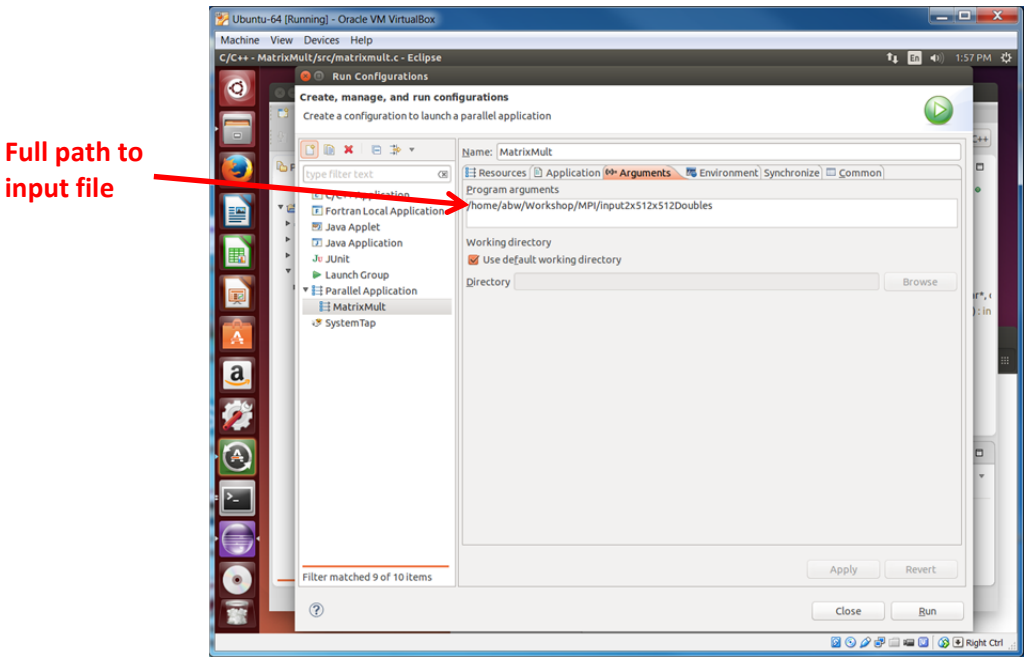


You should now see the source file:

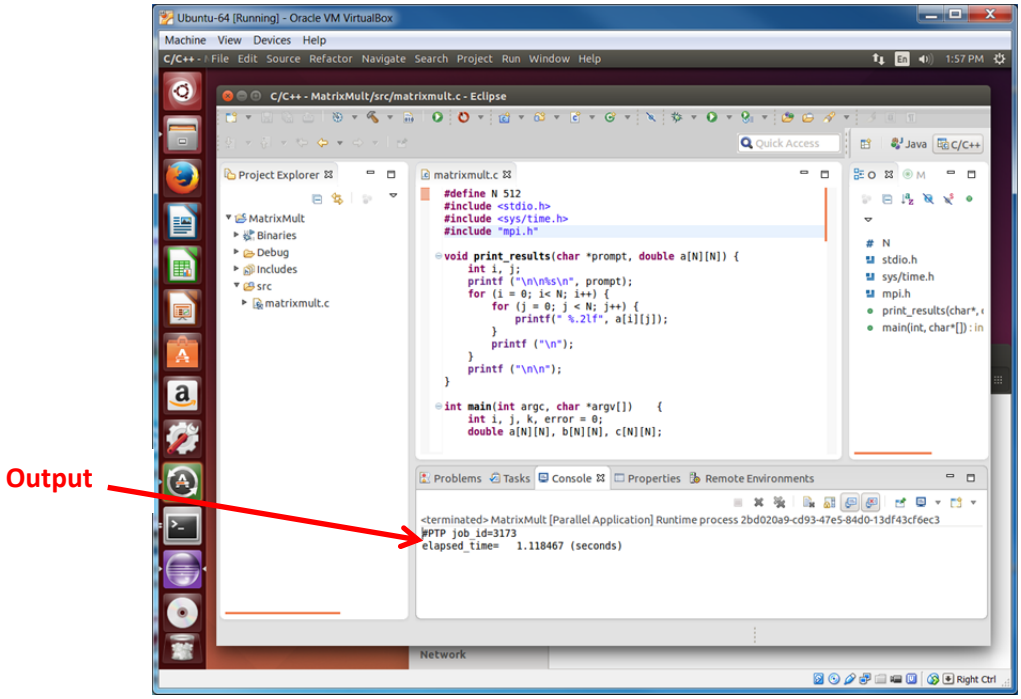


Run Configuration. Now create a run configurations (**MatrixMult**). As before, in the *Resource* tab, select the Target Type as "**Open-MPI-Generic-Interactive**", the connection type as "**Local" .** Set the number of processes.

In the *Arguments* tab, add the full path to the input file **input2x512x512Doubles** as an argument:

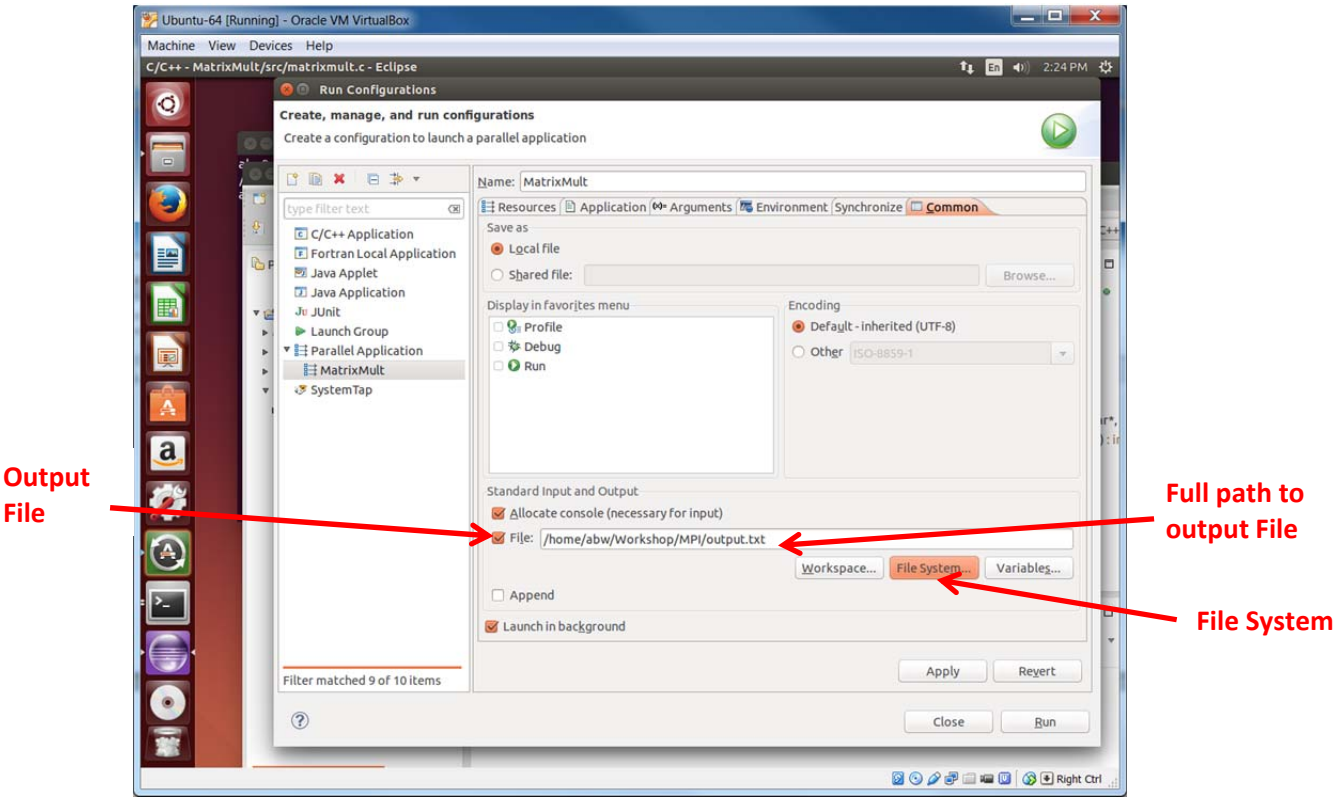


Make be sure to comment out the **print_result()** function call so that the output of the resulting matrix is not printed to the console. Then run the project:



Re-directing Output

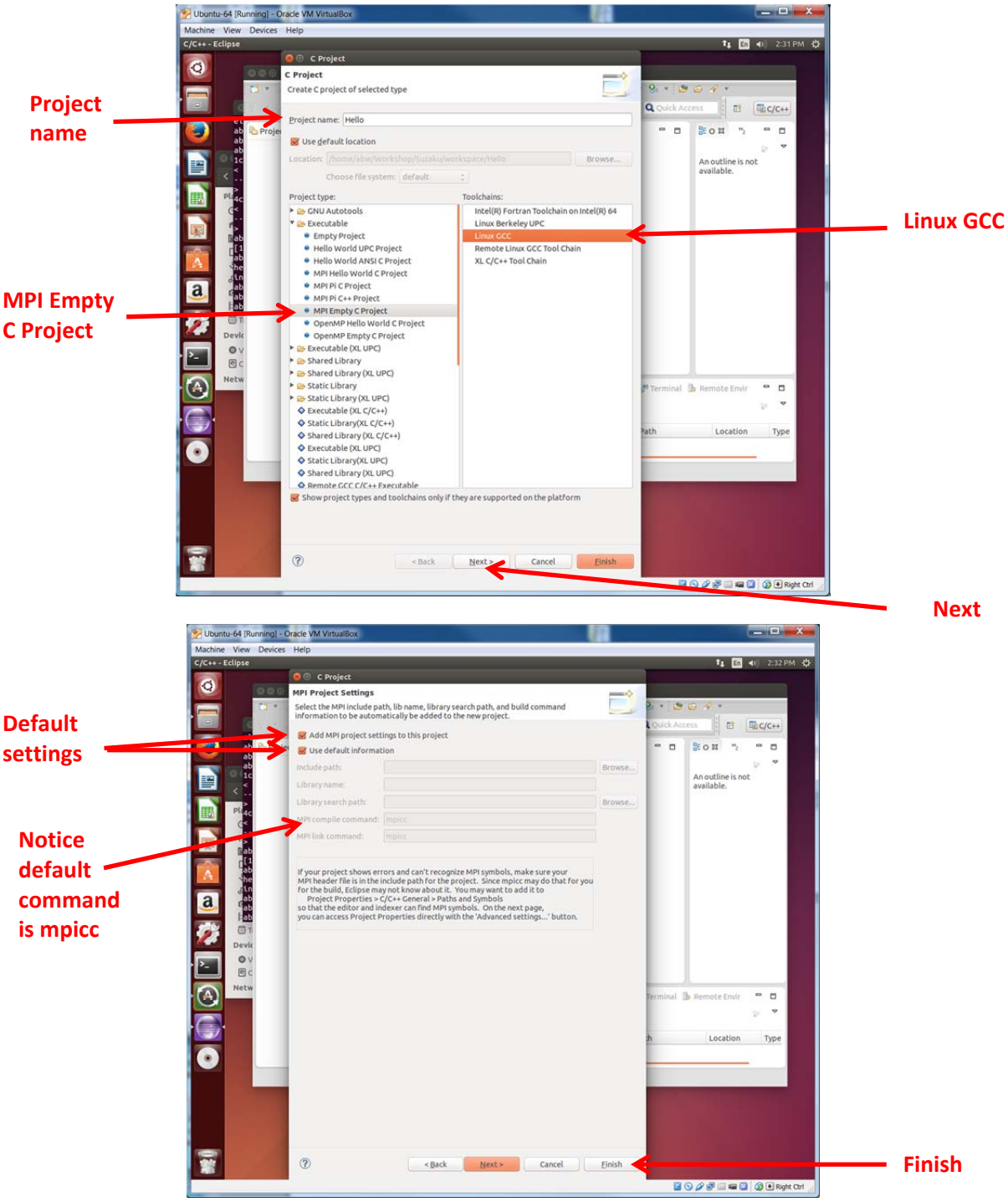
To check that the code produces the correct answer, the print statement is uncommented and the output re-directed to a file, which can then be compared with the provided output file, **output512x512Mult**. Re-directing output in Eclipse is done, *not as an argument*, but in the *Common* tab. Check **File** and **File System**. Browse for the file and select:



Uncomment the **print_result()** function call to output the result matrix and run the project. Using the **diff** command will have to be done on the comment line as before.

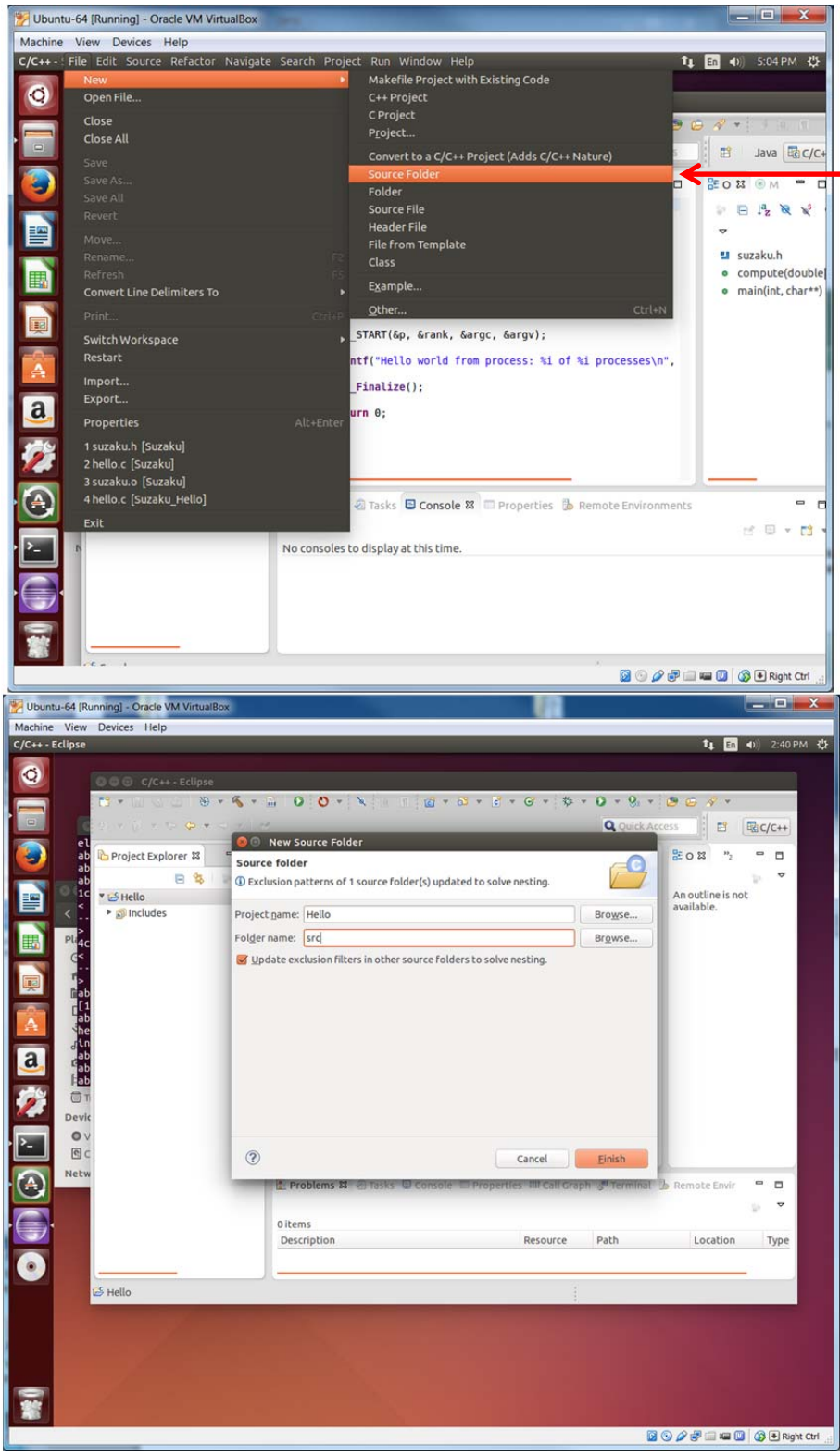
Suzaku

Using Eclipse-PTP for Suzaku is similar to using Eclipse for MPI except the **suzaku.o** and **suzaku.h** files need to be incorporated into the source folder. Select a workspace **/home/ParallelProg/Suzaku/workspace**. Create new C project (**File > New > C project**) called **Hello** of type "**MPI Empty C project**" with default settings:

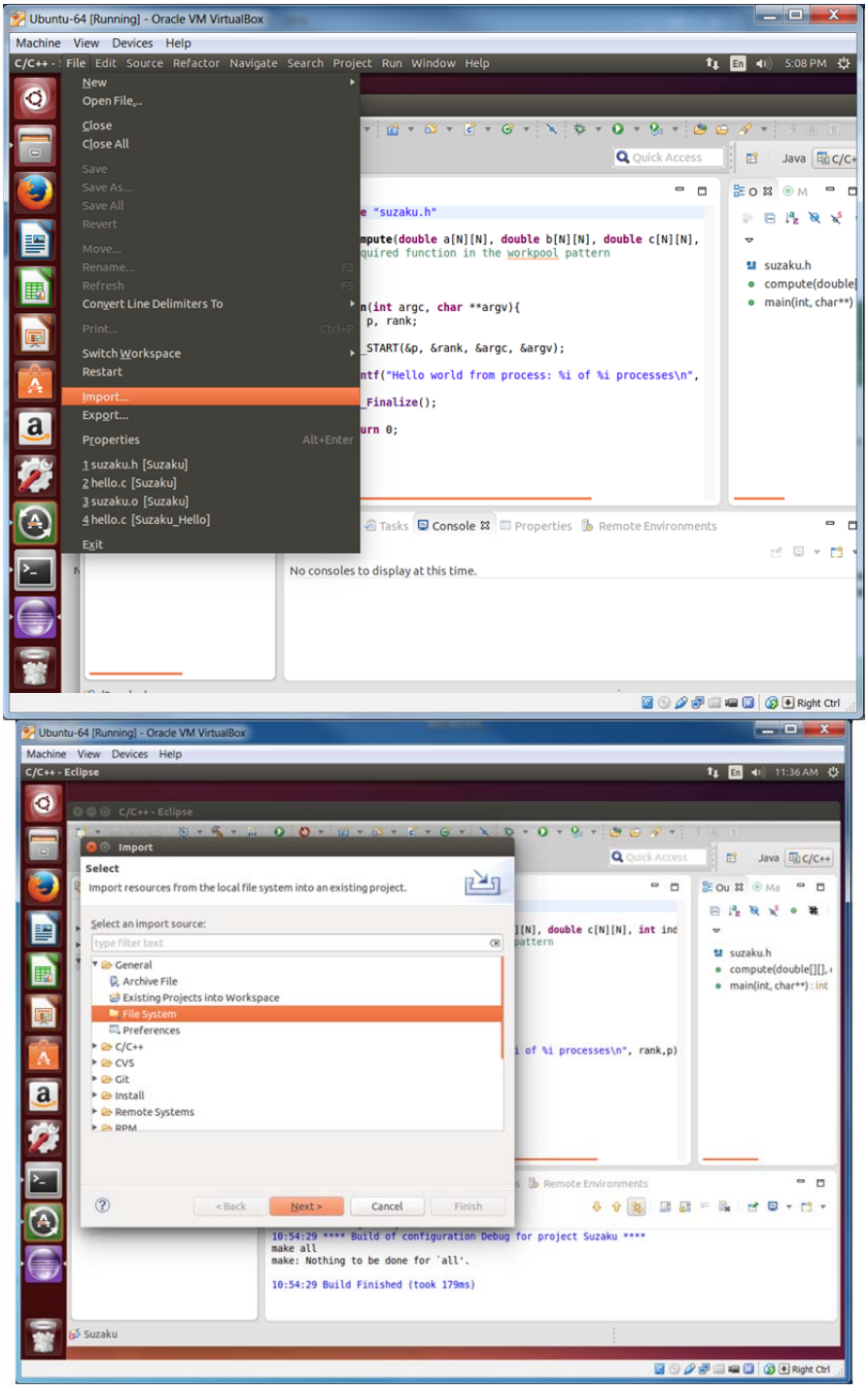


Adding program source and Suzaku files

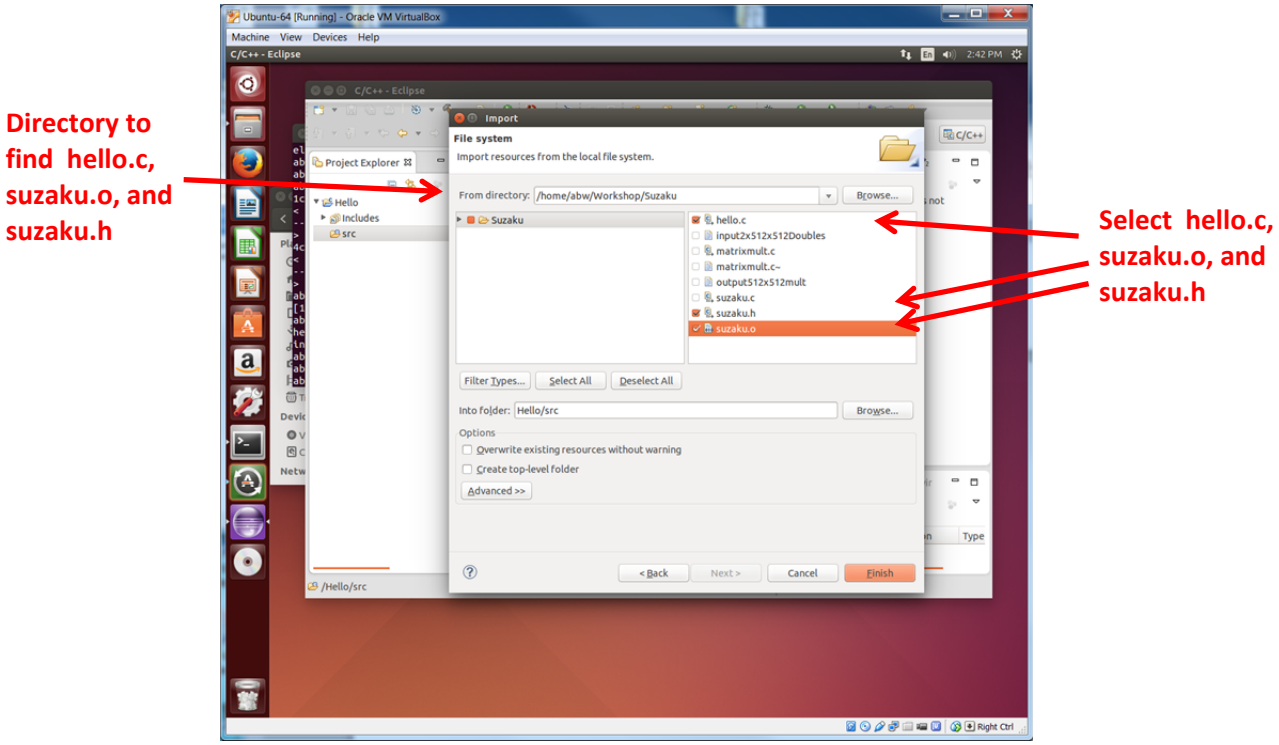
Create a source folder in the **Hello** project called **src (File > New > Source Folder)**:



Expand the **Hello** project and select the newly generated **src** folder. Select **File > Import > General > File System:**



From **“Import from directory”**, browse for the directory that holds your **hello.c** and **suzaku.o** and **suzaku.h** files (**~/Workshop/Suzaku/)**. Click **OK**. Select the **hello.c**, **suzaku.o**, and **suzaku.h** files to copy into the **Suzaku/src** project folder. [4]

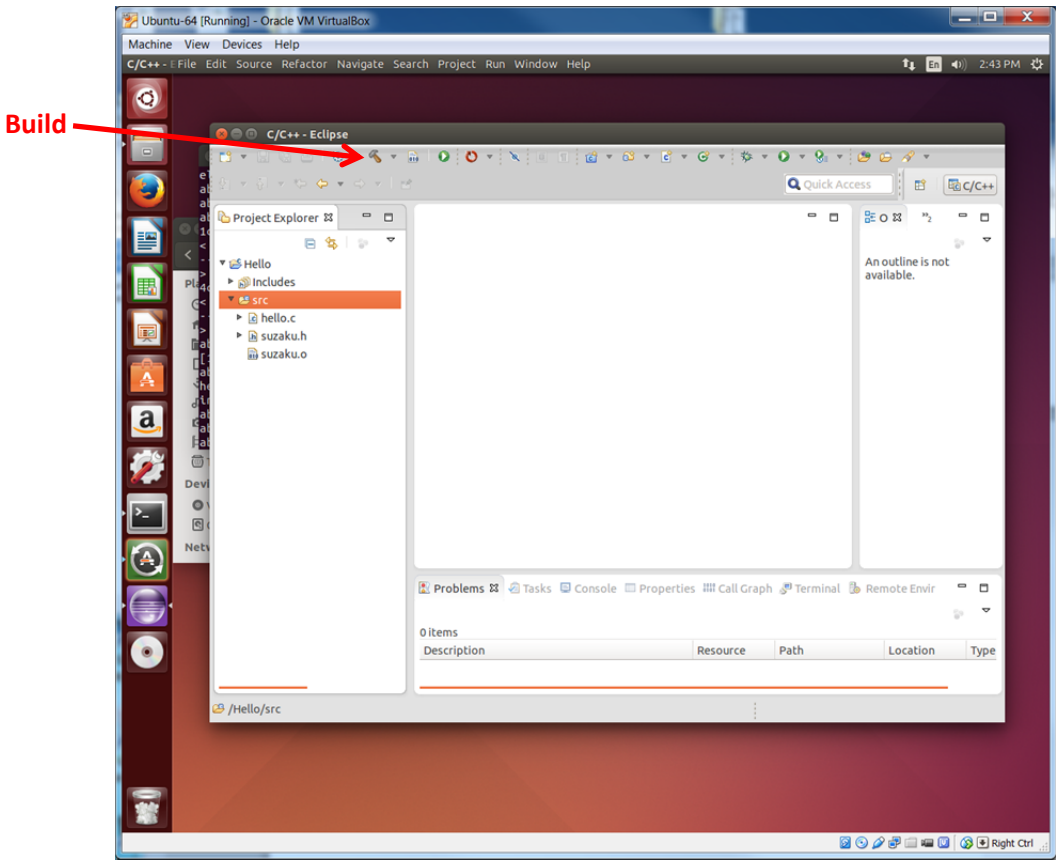


At this point you may get errors indicated in the source files, for example include file not found. These errors will disappear when the project is built.

Build project

Click **Build** icon (Hammer) to build the project (default “Debug” option):

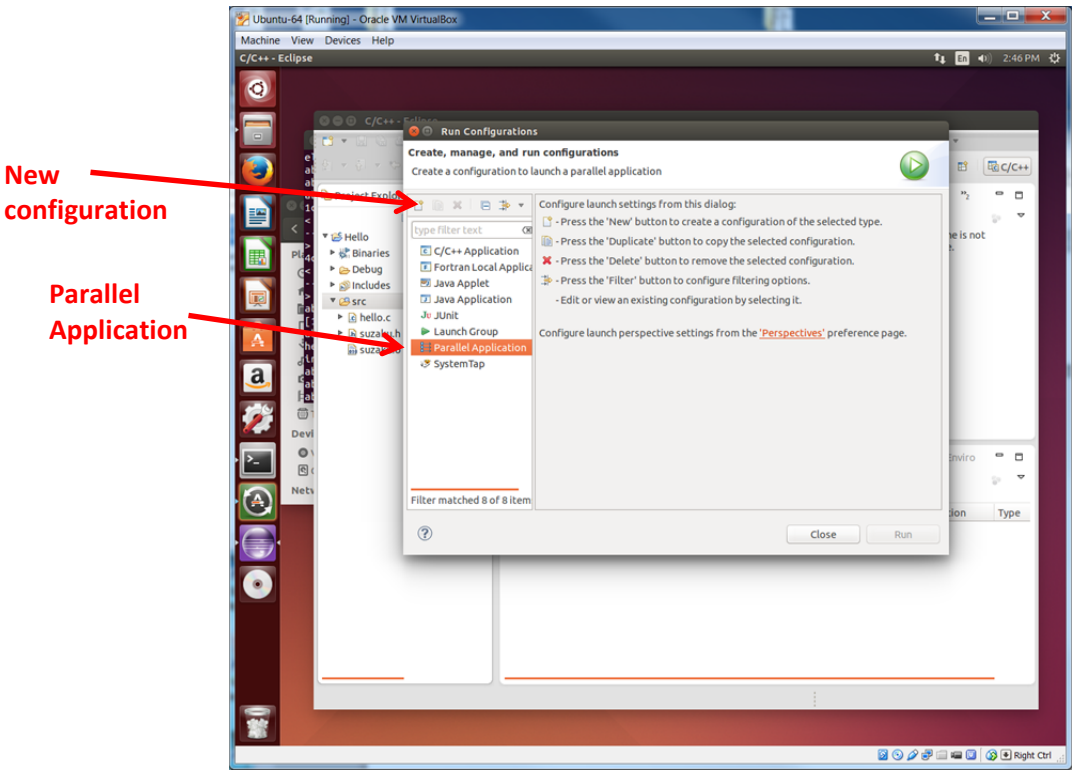
[4] It is probable that the Suzaku files can be placed in other locations within the project folder although Eclipse needs to be able to find them. It is also possible for the Suzaku files to be referred to at their original location rather than copying them, which would be better when developing multiple Suzaku programs.



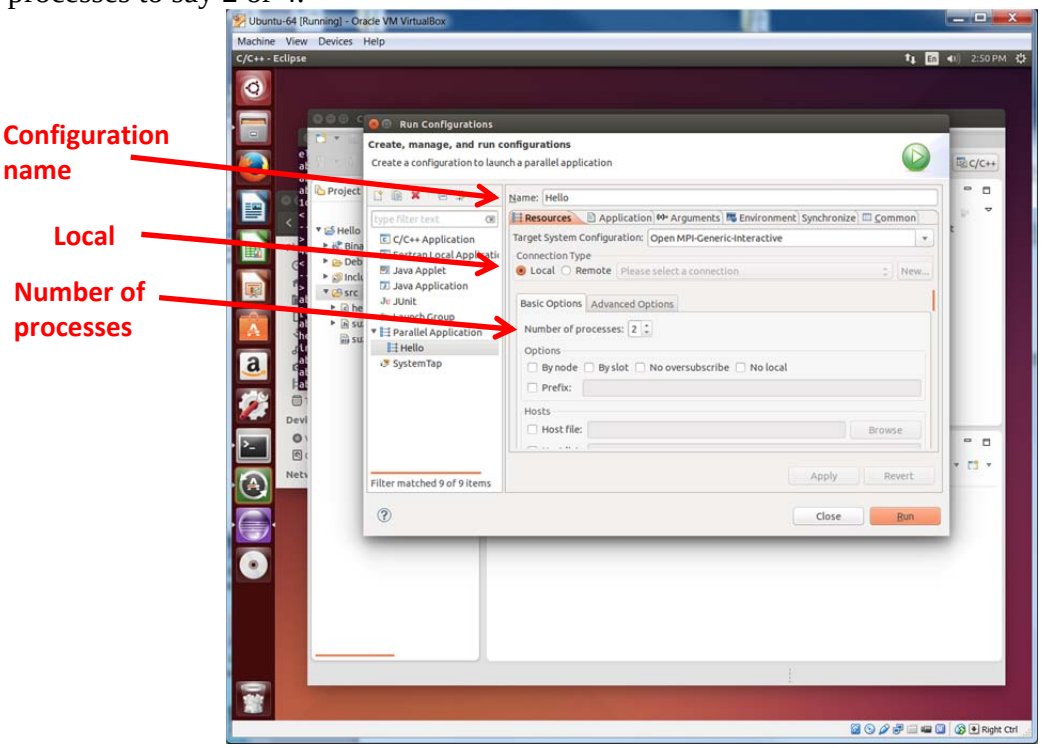
Execution

To execute the program, first the **Run Configurations** need to be set up that specify local or remote execution, software environment, etc.

Select **Run Configurations**. Select “Parallel Application” and click the new configuration button:

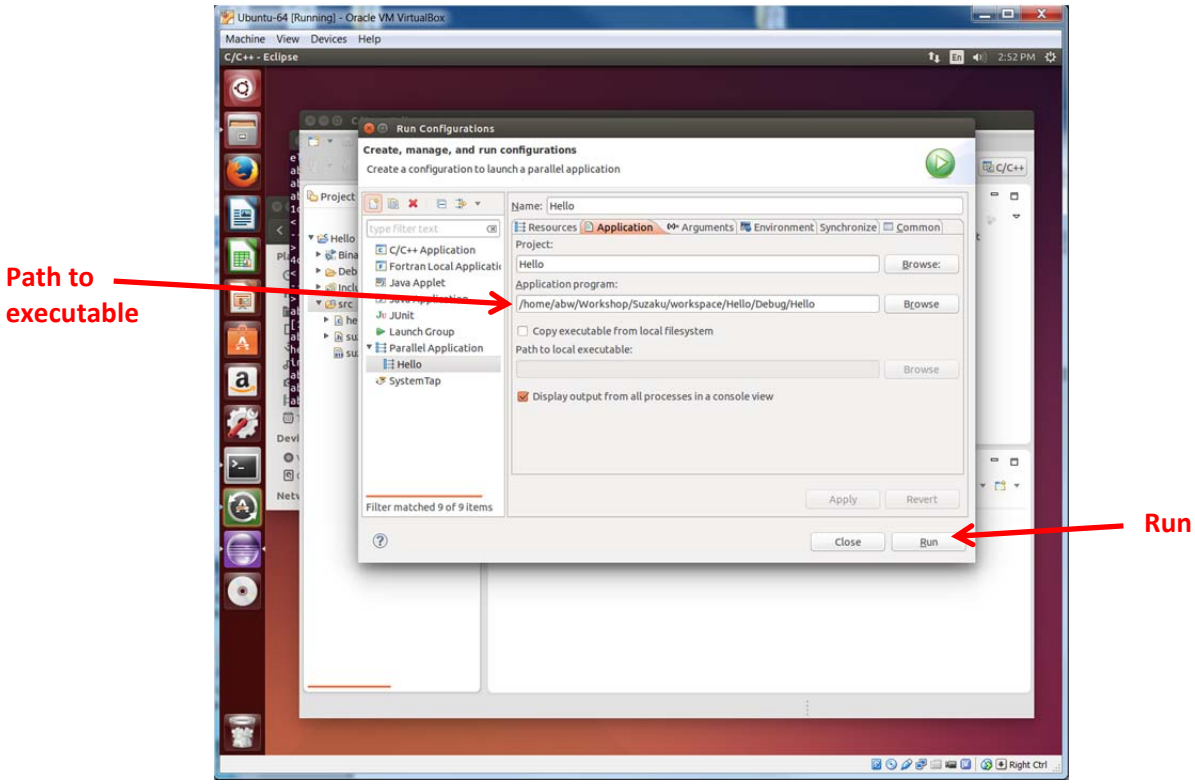


Create a new run configuration called **Hello**. In the *Resource* tab, select the Target Type as "**Open-MPI-Generic-Interactive**", the connection type as "**Local"**.[5] Set the number of processes to say 2 or 4.



[5] Selecting "Local" will generate a message confirming you want this. Eclipse can be used with remote systems.

In the *Application* tab, set the Project name and for “Application program” browse for the full path to the executable (**/home/abw/workspace/Suzaku/Debug/Suzaku**).



Click **RUN**. (If **RUN** is greyed out, there are build or compile errors that prevent execution.)