Instalation of Eclipse Galileo and compatible debuggers with OpenMPI and OpenMP

Parallel and Distributed y Programming

Granada's University

LSI Departement

By:
Daniel Guerrero Martínez
Sergio Rodríguez Lumley

Index of content

Introduction

We are about to install an IDE (*integrated development environment*) for C and C++ to safely debug our projects. This is all the software we are installing:

- OpenMPI (Interface's implementation of MPI)
- Eclipse Galileo (Version 3.5.x)
- Eclipse Galileo's C Development Tools plugin (CDT) 6.0.2
- Eclipse Galileo's Parallel Tools Platform plugin (PTP) 3.0
- Parallel debugger based in gdm (GNU standard debugger), SDM.

Prerequisites:

- Have installed a Java Virtual Machine (For example: sun-java6-jre and sun-java6-jdk)
- C/C++ Compiler, we strongly recommend GNU GCC v4.4.1 compiler or superior.

Although it will be indicated in every case, we'll assume that the installation will run on a Ubuntu (Linux/Unix) OS (Operating System), so a package manager and the tool “sudo” (as superuser) will be available.

1. OpenMPI installation

Open MPI is an open source MPI-2 implementation that is developed and maintained by a consortium of academic, research, and industry partners. Open MPI is therefore able to combine the expertise, technologies, and resources from all across the High Performance Computing community in order to build the best MPI library available.

You can download the last version from it's official page:

http://www.open-mpi.org/

We recommend you to see our installation manual if you haven't already installed this tool. Here we will explain how to install it only for Ubuntu and similar OS. If you don't use it or you are not sure, please take a look at our OpenMPI Installation Guide.

Necessary packages are:

openmpi-bin: Parallel code executor program (mpirun).
Installs: *openmpi-common libopenmpi1.3*

openssh-client, openssh-server: Communicating programs (control and presentation routines) between processes.
libopenmpi-dbg: Debug information generator for MPI.
libopenmpi-dev: Necessary to develop parallel programs based on MPI (mpicc command...).

Quick command:

```
sudo apt-get install openmpi-bin openmpi-common openssh-client openssh-server libopenmpi1.3 libopenmpi-dbg libopenmpi-dev
```

Please note that if you are using Ubuntu, it will automatically install a C/C++ compiler, and it will check for compatibilities and installed components.

2. Eclipse Galileo installation

Eclipse Galileo is an open sourced, Java based IDE. It's development is open and is carried out by an Eclipse community. We are interested in it for C/C++ development with OpenMPI and OpenMP and for it's capabilities of debugging parallel programs through plugins.

The program can be directly downloaded through it's official page:

http://www.eclipse.org

There are many preconfigured Eclipse packages to download. As we are only interested on it's functionality with C and C++, we'll download C and C++ Developers IDE. Please note that any preconfigured IDE can be downloaded because the functionality we are looking for is all installed through plugins.

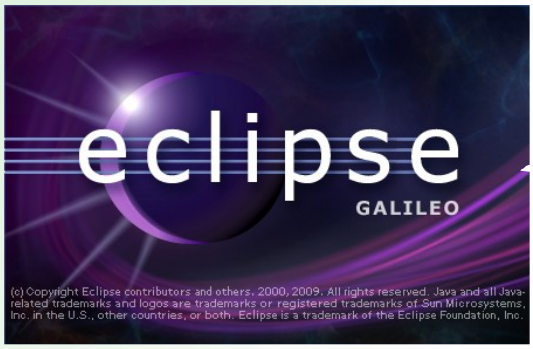
For Ubuntu users, it's install is as easy as running the next command:

```
sudo apt-get install eclipse
```

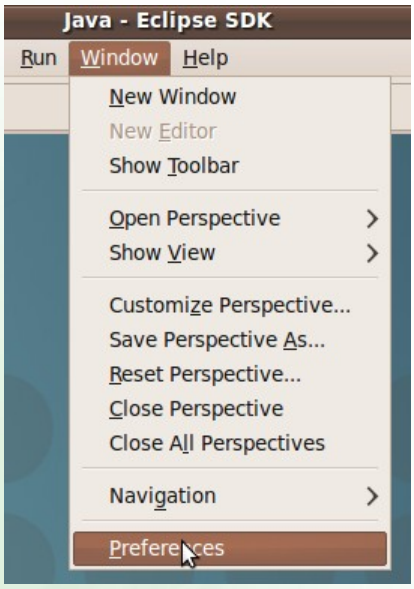
Adding repositories to Eclipse

This task is something we'll do very often, so we'll explain it the first to get us in the job the sooner possible. The repositories we are adding are all from the official page.

We'll start opening the program, it should show an image similar to the one showed at the right image. It will let us select out preferred 'working space', we can select any folder we'd like to work in, we'll select the 'default' one.

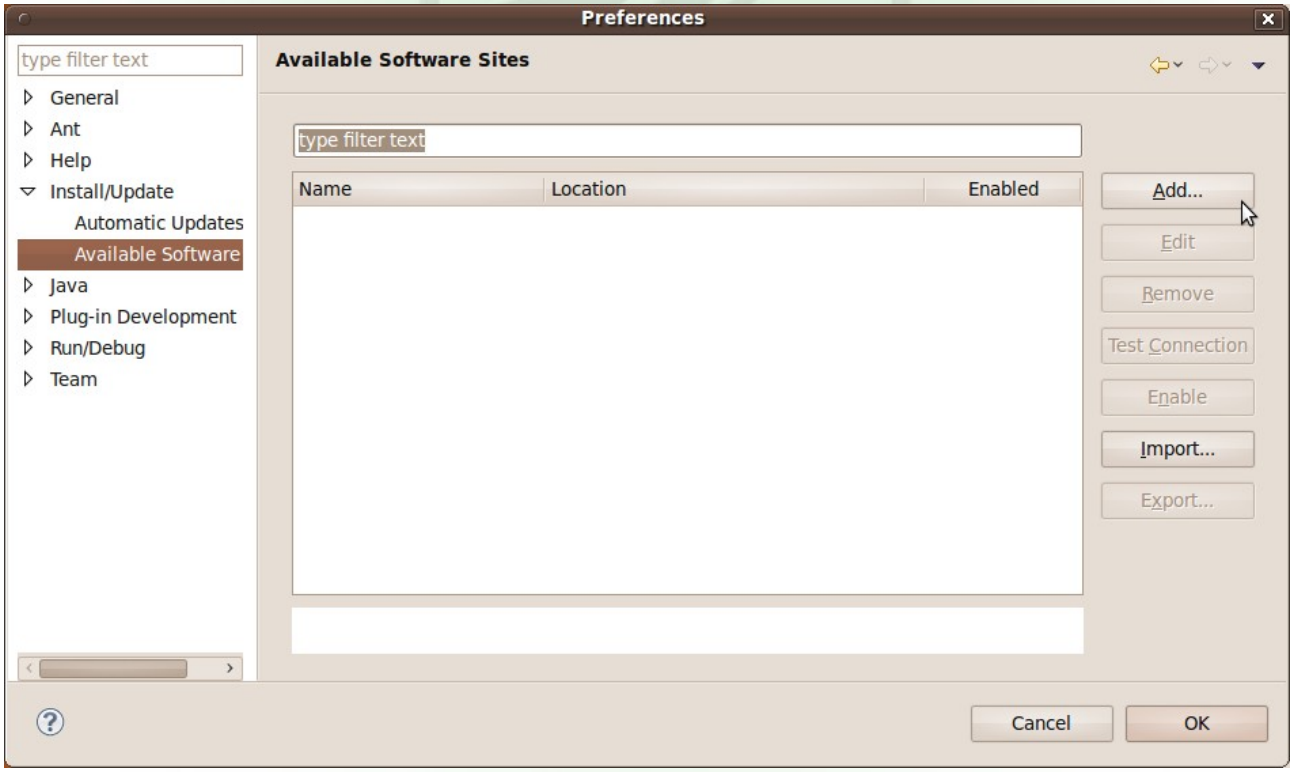


Once opened, we'll open the Preferences window, selecting the following Window → Preferences.



1. Preferences location.

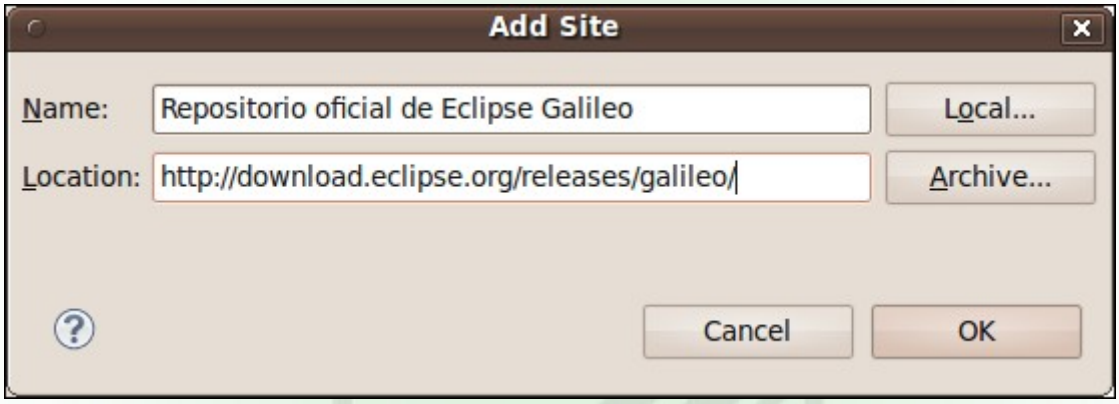
The next window will show up. We have to look at the left bar for the available repositories, to do that, we'll do the next Install/Update → Available Software.



We'll press the Add button, specify the name we prefer (optional) and the direction (obligatory) which can be from the Internet or a local file. In our case, we'll add the next:

Name:	Eclipse Galileo official repository
Location:	http://download.eclipse.org/releases/galileo/

The next image shows how should it be:



3. Adding a repository to Eclipse

Once done, we press on the 'Test Connection' button, this will alert us if there is any problem with the connection, or inform us of the success. In this case, it should all be Ok.

3. (CDT) Get “C Development Tools” plugin for Eclipse

This plugin is necessary to develop in C and C++. Even if we downloaded the Eclipse for C and C++ IDE it's strongly recommended to install this plugin. This will ensure us that we are using the last version of it. If you installed Eclipse for C and C++ IDE, you can go to the next step, if something is wrong you can always jump back here and install it manually.

There are two ways of obtaining this plugin, adding the online repository of CDT, or downloading the file and adding it locally (which is what we recommend).

1. Downloading the file from it's official page (RECOMMENDED) :

- The options are select the newest package from the next link:
  http://download.eclipse.org/tools/cdt/builds/
- Or select the recommended package (The first one that appears on top), the link for the installation we did is the next:
  http://download.eclipse.org/tools/cdt/builds/6.0.2/index.html

One oppened, we'll add it to our IDE repositories as we did before, as if it was an online repository, but looking for the file we've just downloaded (which shouldn't move once you select it).

2. Or adding the online repository:

As we've explained before, we add the next link to our repository:

http://download.eclipse.org/tools/cdt/releases/galileo

Once we have it added, we jump to the next step, Installing the parallel development plugin known as PTP.

4. Installation of Parallel Tools Platform (PTP)

This tool is a set of programs to run and test parallel processes. The installation we are doing is to rung all processes locally, but it can manage remote processes, which is much more complex.

1 Install the Remote System Explorer (RSE) 3.1.1 Eclipse Galileo official repository.

1.1 Open "Help→ Install New Software".

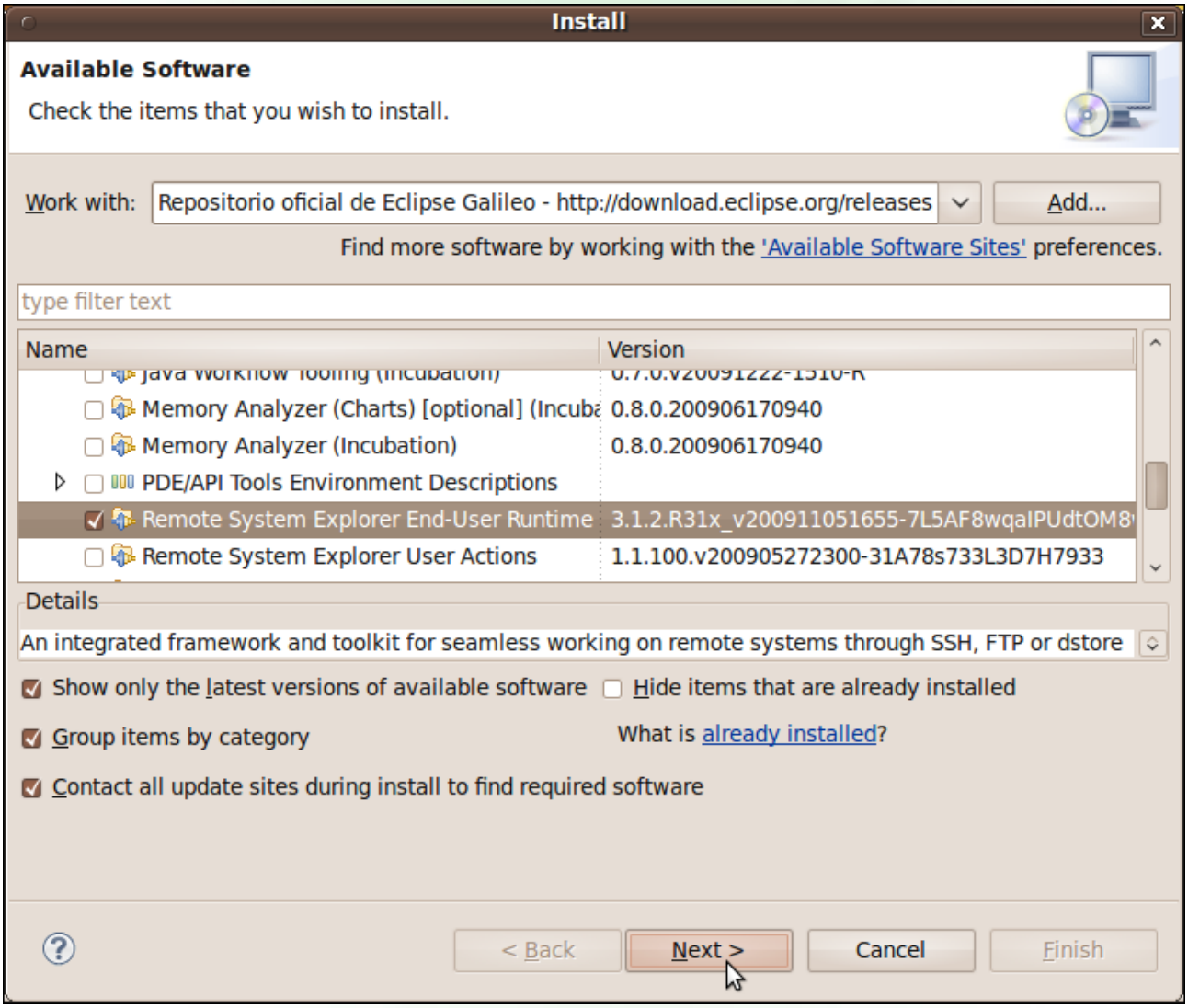
1.2 On the combo box of "Work With" select the 'Eclipse Galileo official repository' we first added.

1.3 Open "*General Purpose Tools*".

1.4 Select "*Remote System Explorer End-User Runtime*" .

1.5 End up the installation following the installation's steps.

1.5.1 It should NOT be necessary to restart Eclipse.



4. Selecting Remote System Explorer End-User Runtime

2 **Now we'll install C/C++ (CDT) 6.0.2, the C/C++ development tools (in this case, from the file downloaded from step 3** *"(CDT) Get "C Development Tools" plugin for Eclipse"***).** You can jump to the next step if you installed Eclipse for C/C++ and do not want to install this plugin manually, although we recommend to do it.

2.1 Open "Help → Install New Software".

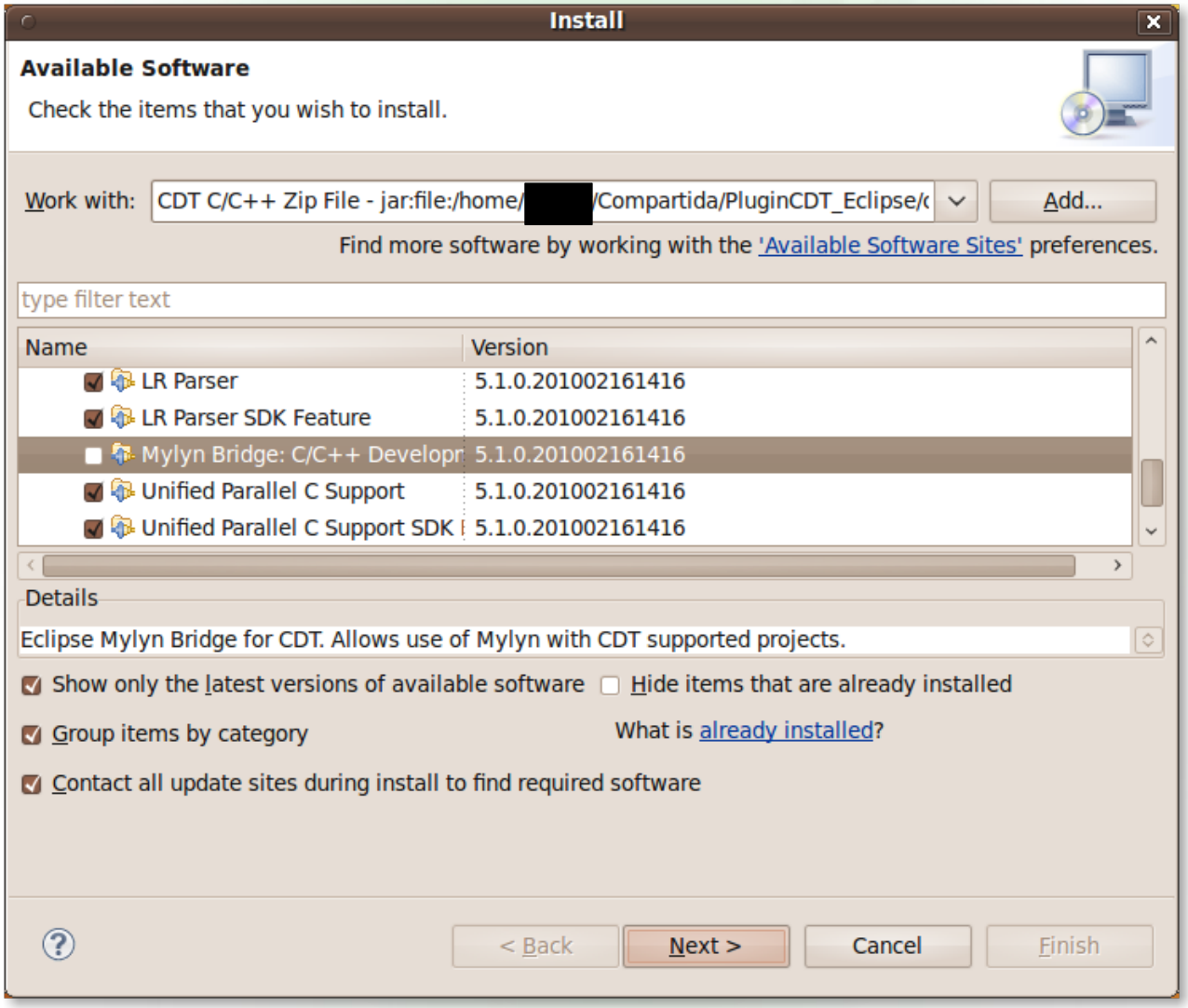
2.2 Click on "Add" button.

2.3 On the path selection dialog, we select the file we downloaded, which will be named something like "cdt-master-xxx.zip".

2.4 We'll select the features we want to install, or just select the main feature to install them all. In our case, we'll select every feature except for "Mylin Bridge", which is one that we cannot install here and we won't need it.

2.5 End up the installation following the installation's own help.

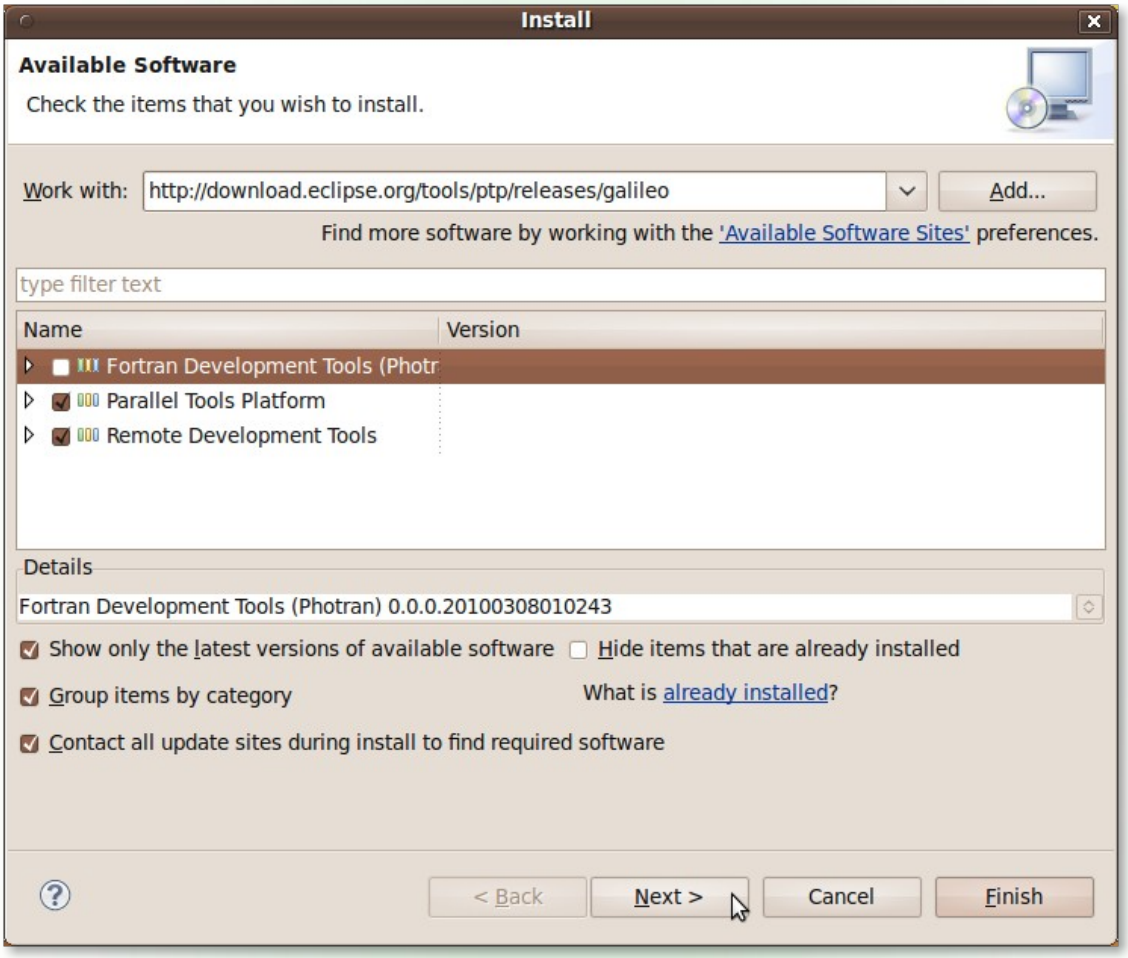
2.5.1 It should NOT be necessary to restart the IDE.



5. Installing CDT.

3 Installing PTP 3.0

3.1 We'll open once again "Help → Install New Software".
3.2 Click on "Add" to add a new update direction.
3.3 We'll add PTP's update page (with the name we want):
http://download.eclipse.org/tools/ptp/releases/galileo
3.4 Select "Parallel Tools Platform" and "Remote Development Tools" categories.
3.5 Follow the installation steps.
3.5.1 We SHOULD restart the IDE after this.



6. Updating our tools

4 Install SDM (Scalable Debug Manager) debugger.

This is a very difficult step, be sure to read it carefully.

4.1 We have to find Eclipse's plugins folder, by default it can be found on the user's personal folder ('home/<user>/.eclipse/'). Once found the folder, get inside the 'Plugins' folder.

4.2 We now have to find a folder with the name like the following "**org.eclipse.ptp.<os>.<arch>_3.0.0.YYYYMMDDHHmm**". The meaning of all this is the next:

<os> Operating System, in our case *linux*.
<arch> Architecture, *x86* for 32 bits, *x86_x64* for 64 bits and *ppc* for PowerPC.
3.0.0 The version, it doesn't matter if this number is superior.
AAAAMMDDHHmm The moment when the package was compiled, where Y (Year) M (Month) D (Day) H (Hour) m (minute).

4.3 If we are able to find it, we only have to get into it and run the command "sh BUILD".

```
sh BUILD
```

This will install all the binaries. Please check that they have been created in the folder "bin", a 'sdm' executable should have appeared. We'll have to remember the path to this executable to tell Eclipse later were to find it.

5. Some common errors

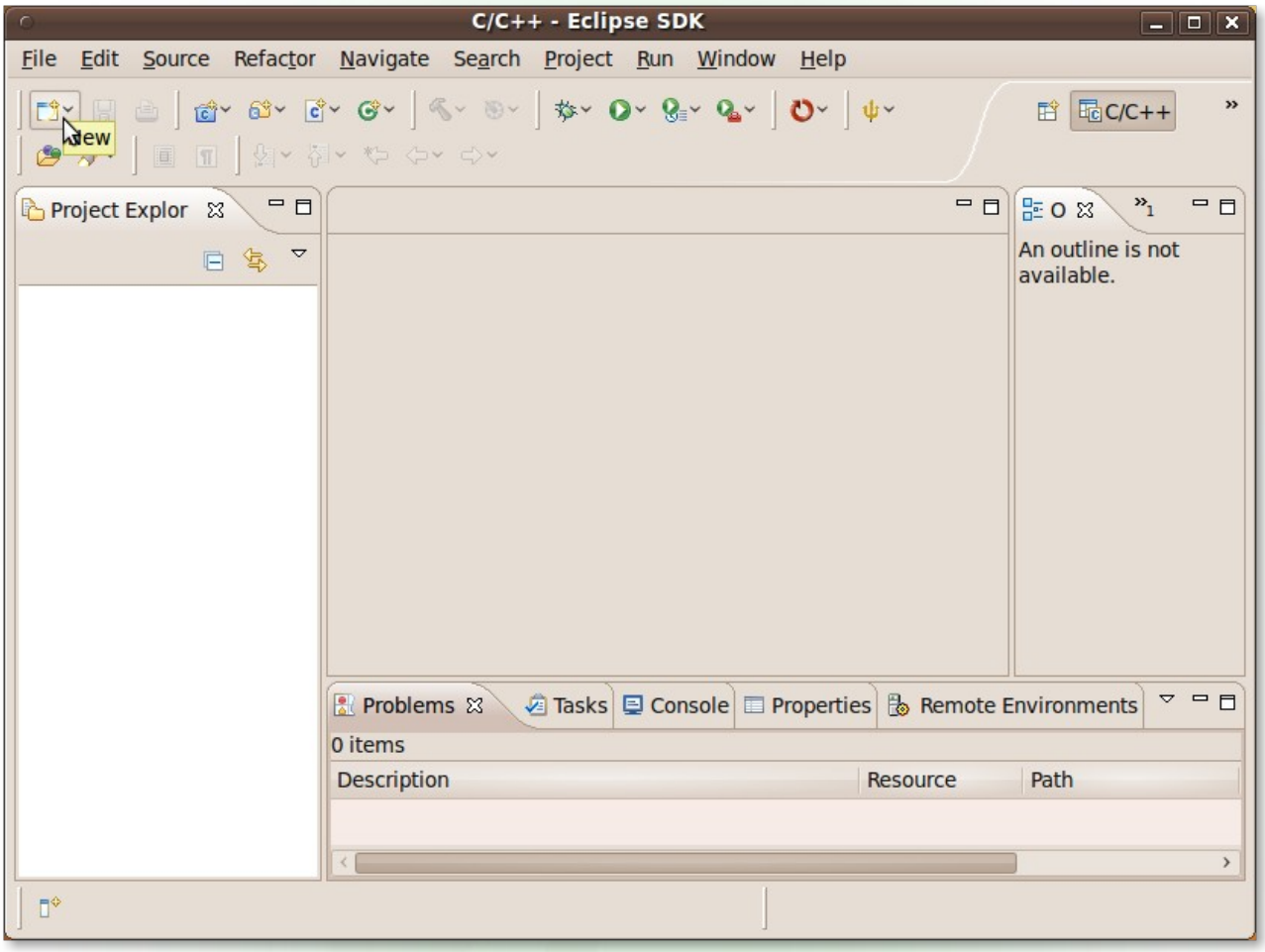
- ***"When I get to Install PTP step Eclipse's package manager shows many conflicts that cannot get solved"***

This can happen when the correct support for C/C++ develop is not found, if you jumped the CDT installation step, you'll notice now that you must install it manually. Go back to the "*(CDT) Get "C Development Tools" plugin for Eclipse*" step, download it and then install the features needed.

6. Configuring the environment

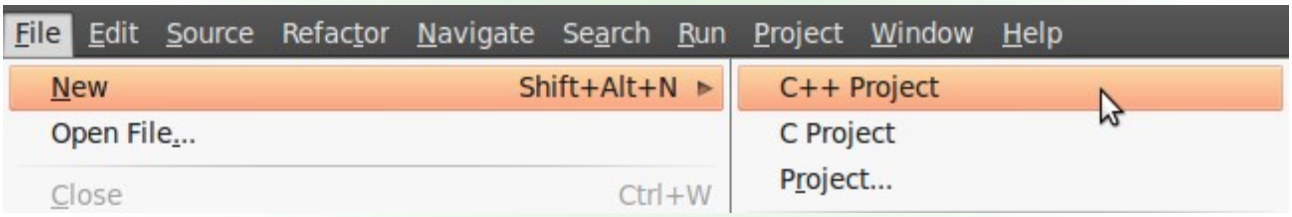
The easiest way of configuring the environment and test it is to create and run a MPI application.

1. Open the Eclipse IDE (we'll maybe have to run the -clean parameter to force it to recognize the new features and plugins).
2. Go to C/C++ Perspective (Window → Open Perspective → Other...).

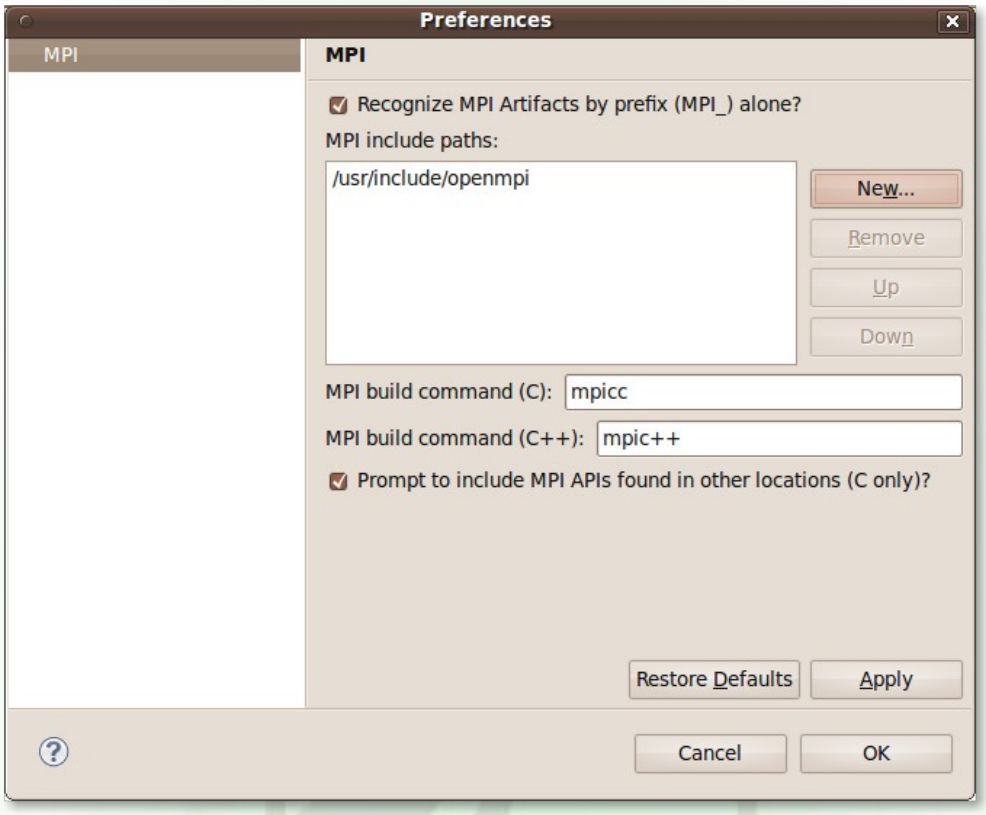


7. C/C++ Perspective looks.

3. Create a new MPI C project selecting File → New → C++ Project.

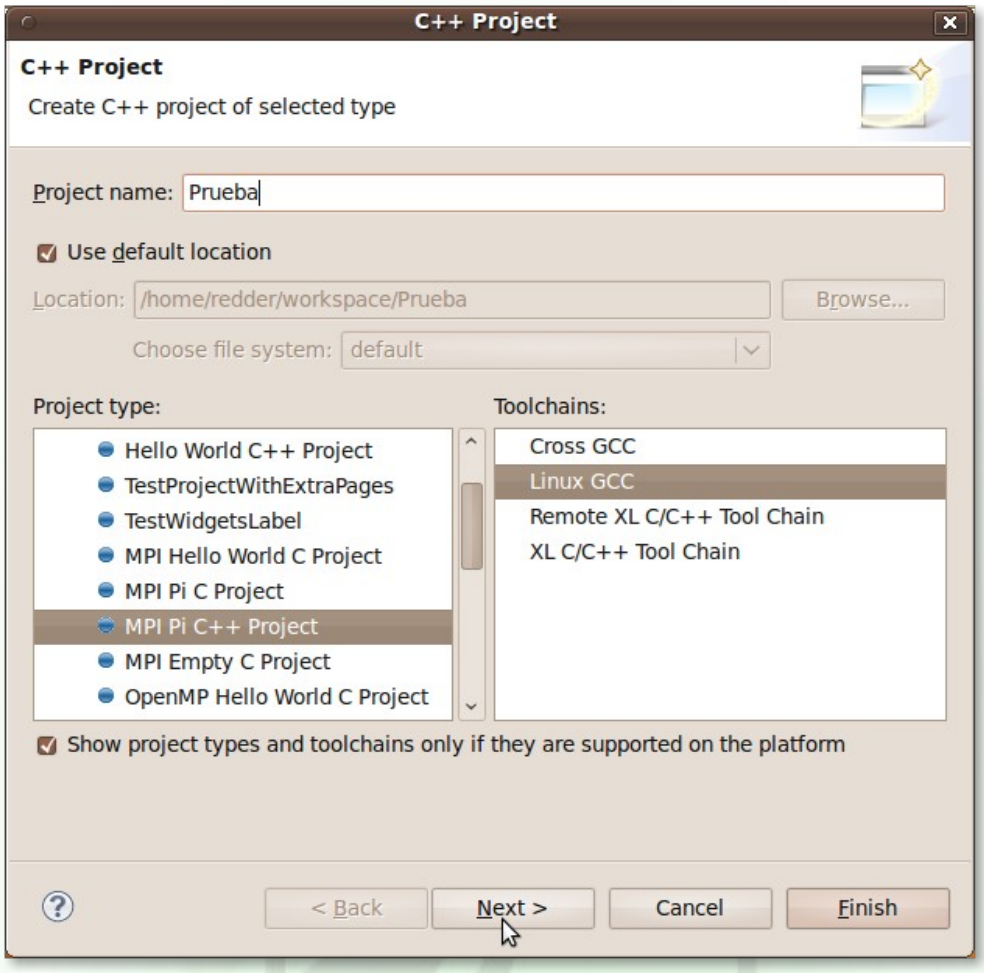


4. We'll select “MPI pi C++ Project”, a dialog will show asking us to configure MPI preferences. We accept and configure them leaving everything by default and including OpenMPI headers path, which will be the implementation we'll use. By default it should be “/usr/include/openmpi”, if it wasn't installed manually.



8. MPI preferences.

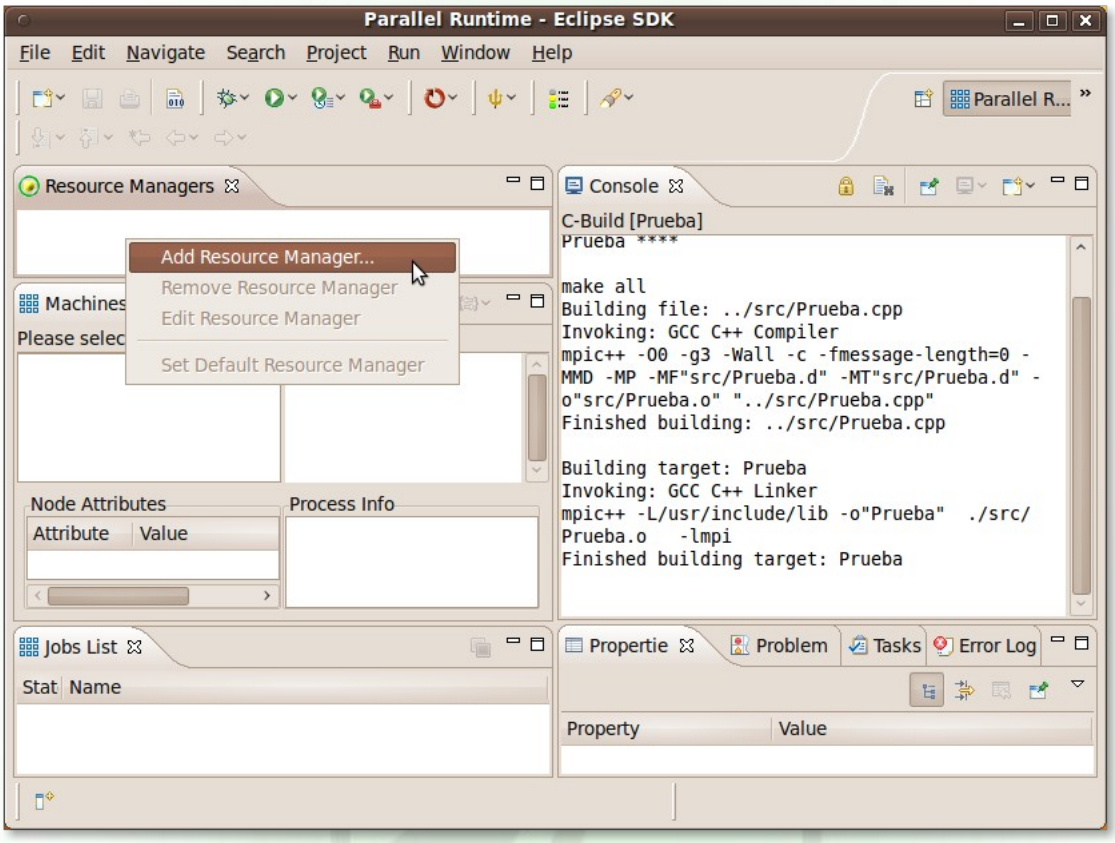
5. Once configured this, we create the project.



9. Create MPI Project in C++.

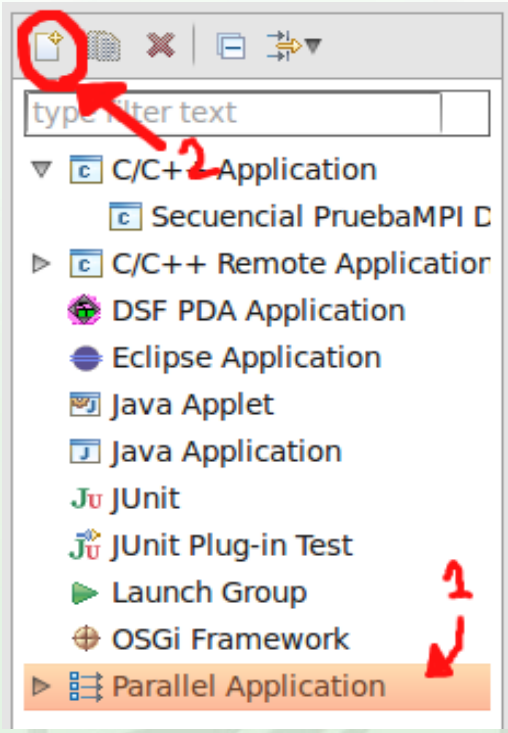
6. Open PTP Runtime Perspective from Window → Open Perspective → Other → "Parallel Runtime".

7. On this view, we now have to create a Resource Manager. Right-click on “Resource Managers” and select the “Add Resource Manager...” option. Select OpenMPI leaving every parameter by default and start it. A machine and one or more nodes should be shown on the “Machine View” panel.



10. Adding a Resource Manager.

8. Now we have to create a “Launch Configuration” on Run → Run Configurations. We select “Parallel Application” configuration and we press on the “new” button (a white paper with a star), up to the left.



11. Creating a Launch Configuration.

   1. We have to fill the information for every particular case, in ours, we'll leave everything by default, except for the next three points:
      1. On “Resources” tab, we have to select the number of processes (generally 4).
      2. On “Application” tab, we select our project for “Parallel Project” and the executable for the application.
      3. On the debugging tab, we have to look for the **sdm** debugger, the one which we built before, on the "Path to debugger executable”.
9. Now we can run the application by pressing the button “Play”.

More details on how to use PTP on Help → Help Contents, and click on “Parallel Tools Platform User Guide”.

You can also find help online for PTP.