

Introduction to the Eclipse IDE + PTP

M. D. Jones, Ph.D.

Center for Computational Research
University at Buffalo
State University of New York

High Performance Computing I, 2014

What is Eclipse?

- Vendor neutral open source development platform
- Platform for tool integration
- Plug-in based framework to create, integrate, and use software tools (called “Equinox,” part of the Open Services Gateway initiative, or OSGi):
 - Plug-ins form a “bundle”
 - bundle (un)installation, update
 - bootstrap, launching, and extension registry

Eclipse History

- Borland, IBM, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft and Webgain formed eclipse.org board in 2001
- Open-sourced under Eclipse Public License (EPL), based on IBM Common Public License
- 2004 consortium reorganized into not-for-profit corporation, Eclipse foundation
- Four classes of membership - strategic developers, strategic consumers, add-in providers, open source project leaders
- Currently many projects and subprojects

Eclipse Platform

- Core framework for all plug-in extensions
- Common facilities:
 - Workbench (user interface)
 - Portable UI libraries
 - Language independent debugging
 - CVS/subversion/git ... support
 - Dynamic update/install service

Plug-ins

There are a lot of eclipse plug-ins available:

- C/C++ development tools (CDT)
- Fortran development tools (Photran, now part of PTP)
- Parallel tools platform (PTP)
- Java development tools (JDT)
- Plug-in development environment (PDE)
- many more ...

Installing eclipse

www.eclipse.org

- Fortran support is through the **Photran** project (now part of Parallel Tools Platform):

www.eclipse.org/photran

- The Parallel Tools Platform (PTP) is an exciting development, but also can be somewhat tricky to integrate, especially with remote resources.
- Latest release of eclipse (Juno) has a package for parallel application developers that incorporates CDT and PTP

A Few More Installation Comments

- Can install multiple versions of eclipse without any difficulty (just watch out for overlapping workspaces)
- Not really intended as a “server” application - more of a locally installed development tool - that is why the remote tools are interesting/useful
- Remote development tools are still under (very) active development - great idea, though

Installing Eclipse & PTP

Steps for installing and configuring Eclipse plus the PTP, best to follow the latest on the PTP Wiki:

<http://wiki.eclipse.org/PTP>

The release notes for the current version generally have the most up-to-date instructions. Follow the instructions for installing Eclipse and PTP, running updates, enabling PTP update site, etc.

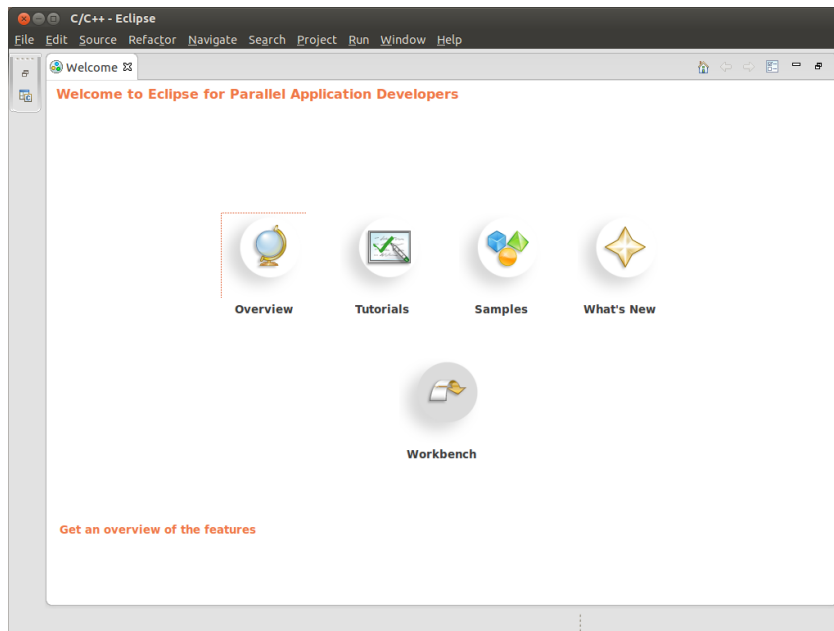
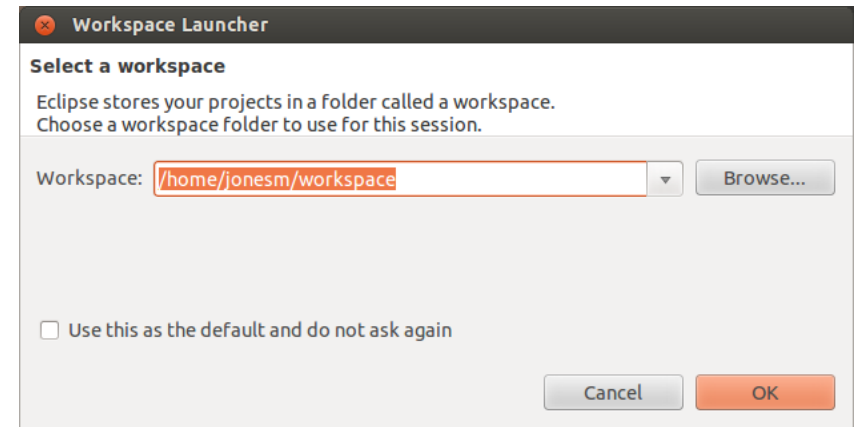
Some Issues

Depending on your operating system and environment, some parts of eclipse and PTP may or may not be available to you - some issues that I have encountered:

- Windows - I can not help you much there, but it is claimed to run under Windows
- Fortran support under Mac OS (and very likely Windows as well) - need to deploy Fortran compiler that you want to use
- MPI support - you need a working MPI to use the MPI features of PTP (depends on your platform)
- Virtual machine installs can be quite handy for dealing with some of these limitations - e.g., use VirtualBox and your favorite Linux distribution inside a VM without messing around with your base machine

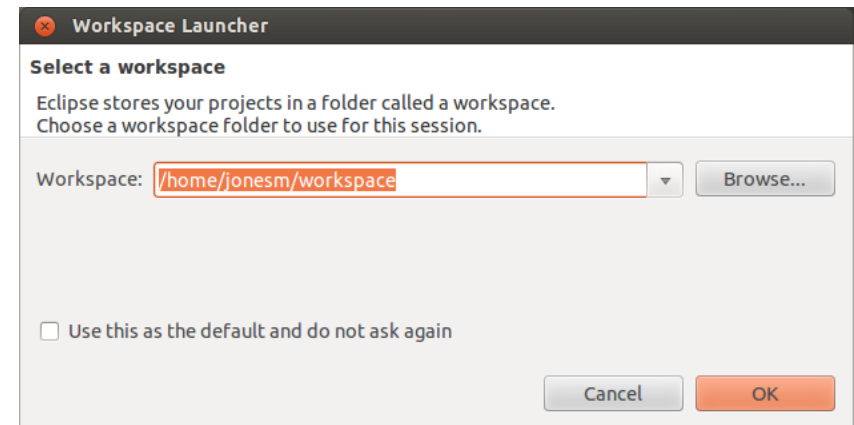
Workspace

When you start `eclipse` it asks you for a workspace to use - a directory in which it will store working files. I tend to accept the default (the directory is browsable from the command line)

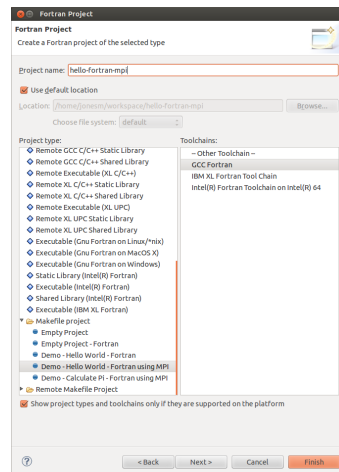


Start New Project

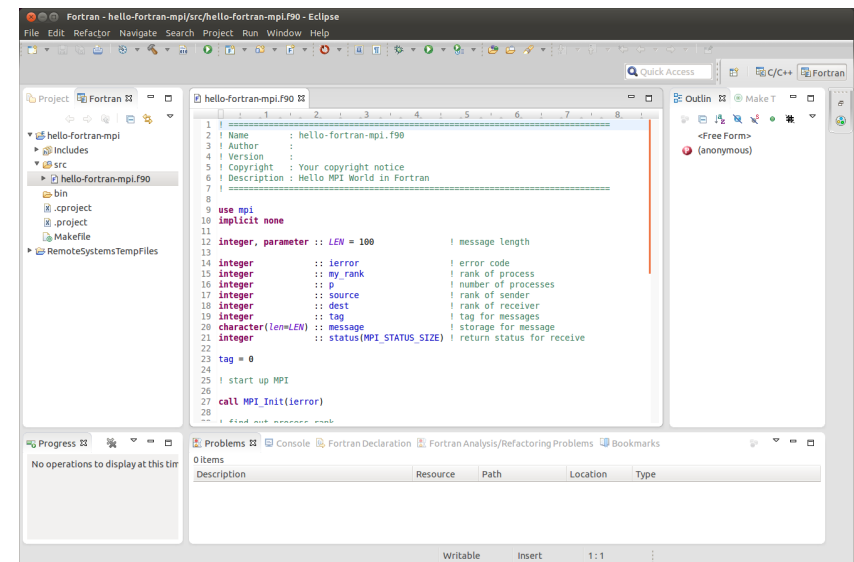
File->New->Project will start a new project in your workspace.



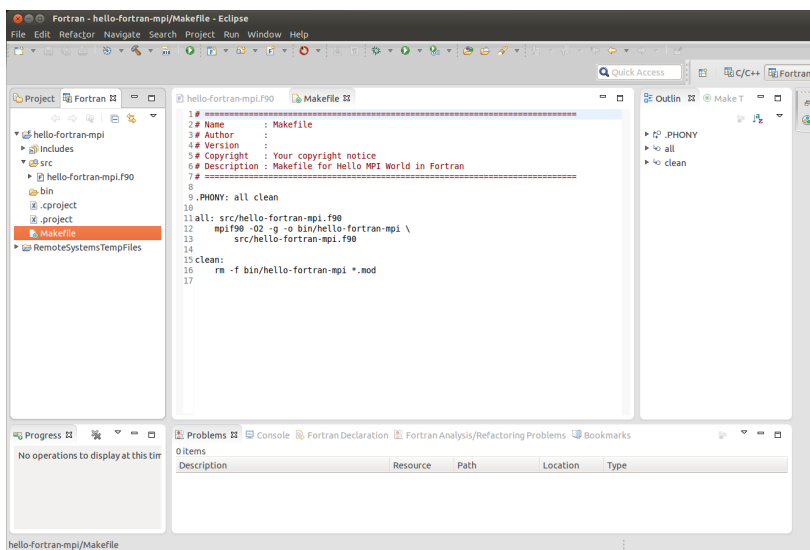
Selecting a project type determines the toolchain (compilers, linker, etc.) and tabs that eclipse will use and display. Note that you can first open the perspective (Window->Open Perspective) and then start a new project (File->New->Fortran Project->Makefile Project):



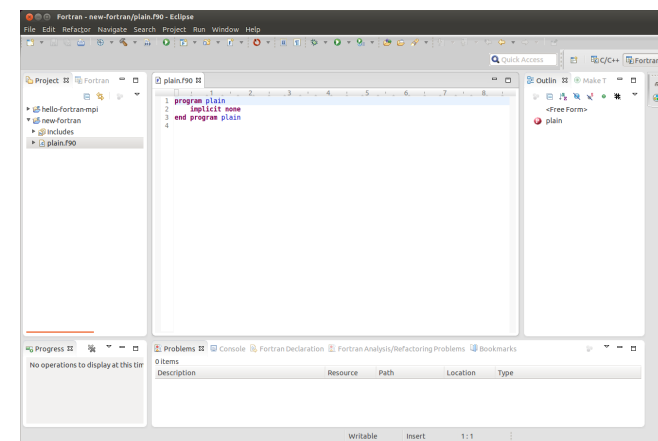
In this case I will load up a prepared sample MPI code that should look pretty familiar:



and you can edit the Makefile if you want to tweak the various settings (note the built-in editor with syntax highlighting and other nice features):



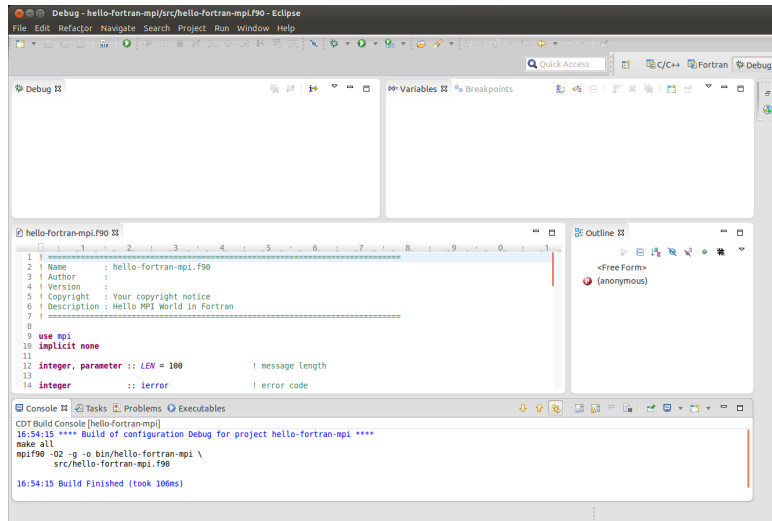
... and then you can edit a new file in your project by right-clicking on the project folder and selecting **New->File From Template**. You can then type in some new code:



You can follow a similar path to create a Makefile (new projects can automatically create a Makefile if you select that type of project initially).

Changing to the Debug Perspective

Selecting the **Project->Build Project** should build a binary, and then you can run it (**Run As ...**) and change to the *Debug Perspective*:



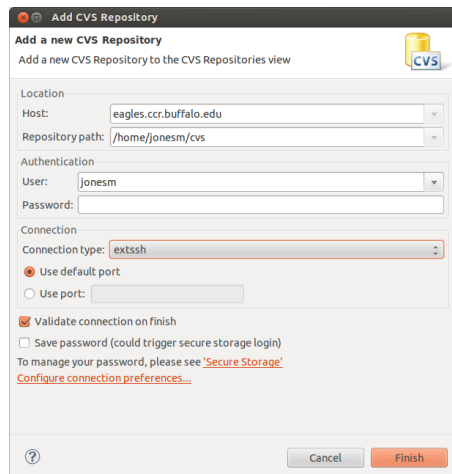
CVS?

If you are not using CVS (or a close relative), you should be:

- Concurrent Version System
- Shared access for multiple developers to develop single code “repository” with version control, branching, etc.
- Popular in open source software development (even “anonymous” read-only access to online CVS repositories)
- Git, subversion close competitors (eclipse has git support, is planning on adding subversion support)

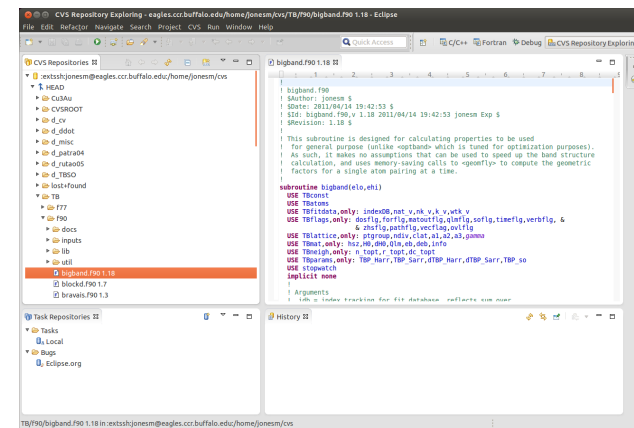
Adding CVS Repo

Window->Open Perspective -> Other -> CVS Repository Exploring changes to CVS perspective

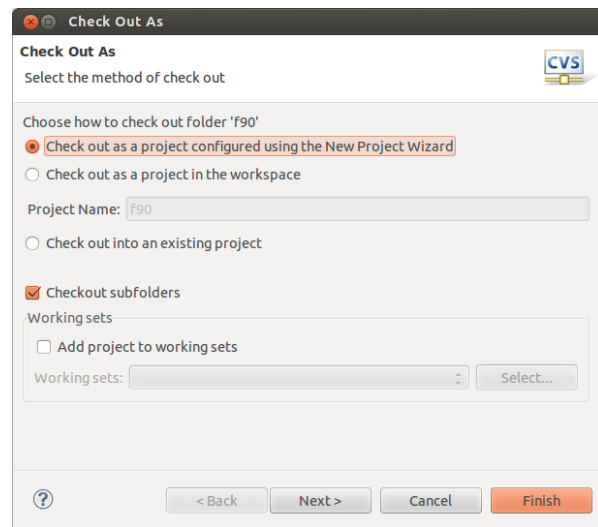


Hit the **Add CVS Repository** button ...

Provided that you authenticate properly (I am using ssh), you can then browse the CVS repo:



(note that the above is just browsing the files in the repository). Right-click on the project that you want, and select check out to get a copy ...



and you now have a rich set of CVS functionality within eclipse (diffs, commits, patching, etc.)

More CVS Information

CVS is a topic in its own right - some links to more CVS documentation:

- <http://offog.org/cvsintro.html>
- <http://cvsbook.red-bean.com/>
- http://en.wikipedia.org/wiki/Concurrent_Versions_System

Git, subversion are also integrated with eclipse.

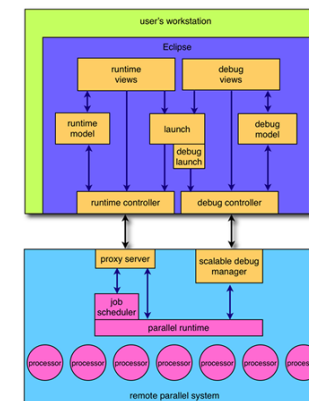
Parallel Tools Platform

Lofty goals:

- Support range of parallel architectures and runtime systems
- Scalable parallel debugger
- Support integration of parallel tools
- Simplify end-user interaction with parallel systems

PTP Architecture

Schematic overview of PTP:



Tricky bit (unsurprisingly) is launching parallel codes (few standards) on remote execution systems (many different job schedulers)...

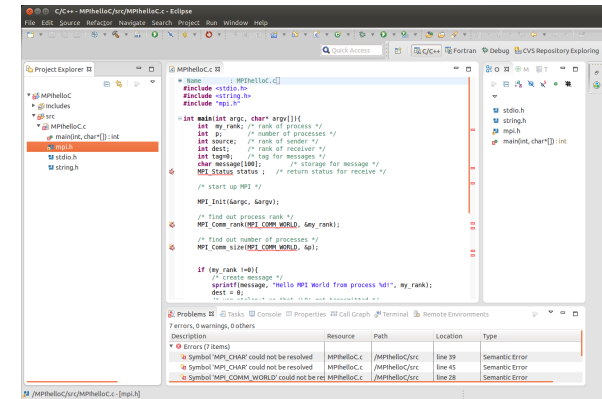
PTP Demonstration

Version 7.0 (officially released mid-2013) is the latest version, and what I will attempt to use for demonstration purposes.

- Best bet - follow carefully the steps outlined in the release notes for the latest PTP release (or pre-release if there is a significant feature that you want that has not yet made it into the released version). There are also frequent tutorials that are worth walking through if you are completely unfamiliar with the interface.

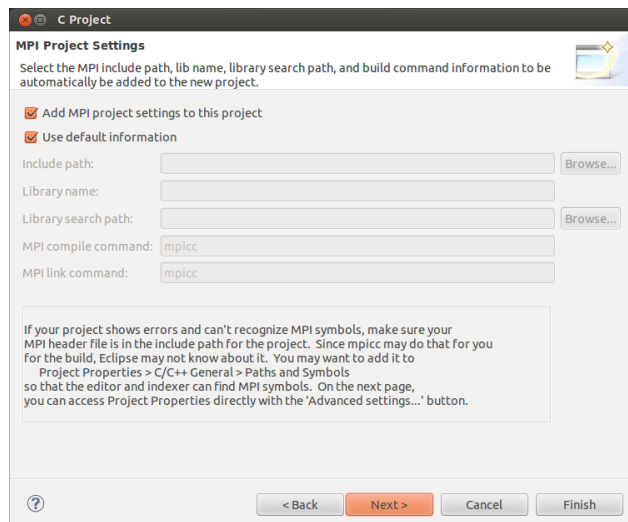
Simple MPI Example

Note that mousing over MPI functions shows calling syntax (also known as “Content Assist” - also can use CTRL-space), but you may need to fix the path settings first ...

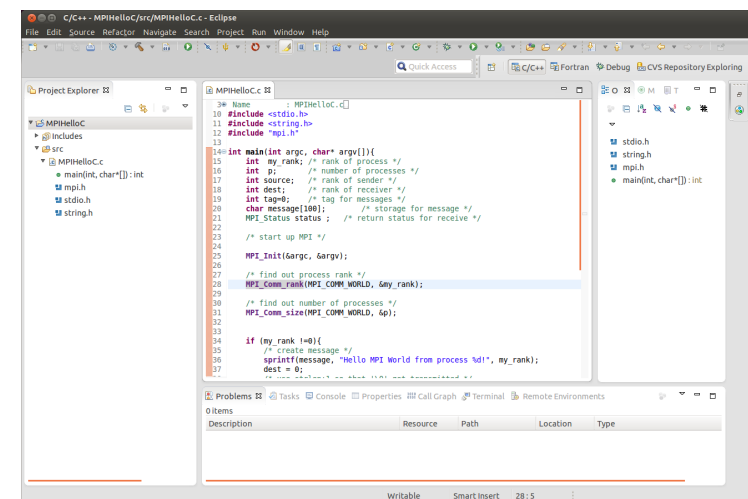


MPI Artifacts

Going through the project wizard gives you the instructions for fixing missing include/library paths:



Once you have cleaned up the settings:



MPI Features

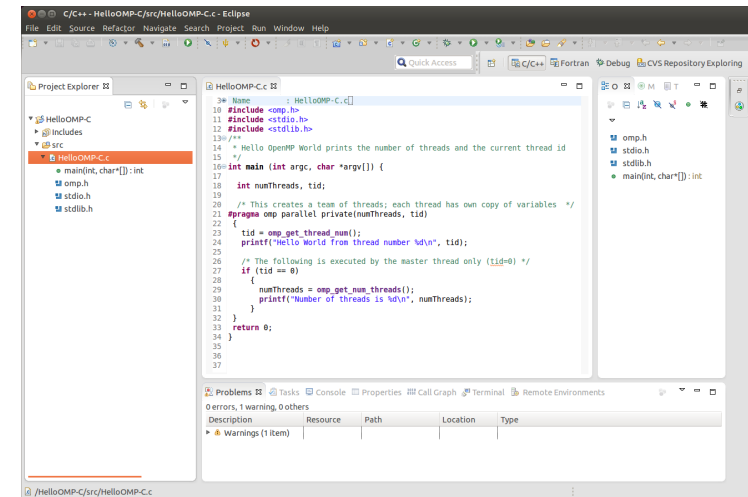
- "Mouse over" feature gives calling syntax
- "Control-space" keys give completion assistance in choosing MPI routines
- Press help key (typically F1) when clicking on a routine and help window should appear
- MPI Templates - e.g. type "mpisr <ctrl-space>" to automatically insert a template of MPI_Send and MPI_Recv ("mpi <ctrl-space> <ctrl-space>" should show all templates)

PTP Runtime and Debugging

- PTP runtime needs an install of MPI in your default environment
- Debugging - typically have to build the debugger (sdm) by hand - see PTP Wiki for steps (usually in the release notes for each version)
- Debug sessions are pretty flexible
- Remote development and debugging also supported through (new) synchronized projects, but setup can be more than a bit tricky (demo?)

OpenMP Example

Eclipse PTP has gotten a lot better at automatically detecting the environment, but of course an OpenMP compliant compiler is a necessary prerequisite.



More Information on PTP

- PTP Wiki,
<http://wiki.eclipse.org/ptp>
- PTP Home,
<http://www.eclipse.org/ptp/>
- PTP XSEDE14 Tutorial,
<http://wiki.eclipse.org/PTP/tutorials/XSEDE2014>