parallel tools platform

http://eclipse.org/ptp

Developing Scientific Applications Using Eclipse and the Parallel Tools Platform

Greg Watson, IBM
g.watson@computer.org

Jay Alameda, NCSA
jalameda@ncsa.uiuc.edu

Beth Tibbitts, IBM
tibbitts@us.ibm.com

Jeff Overbey, UIUC
overbey2@illinois.edu

November 16, 2009

Portions of this material are supported by or based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under its Agreement No. HR0011-07-9-0002, the United States Department of Energy under Contract No. DE-FG02-06ER25752, and the Blue Waters sustained petascale computing project, which is supported by the National Science Foundation under award number OCI 07-25070.

SC Portland, OR 2009

Tutorial Outline

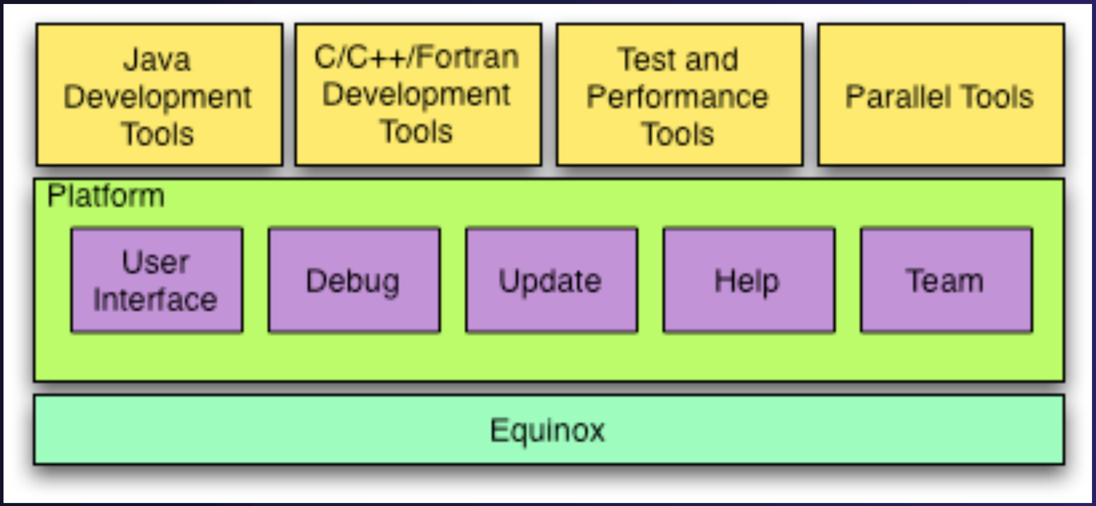
Time (Tentative!)	Module	Topics	Presenter
8:30-9:00	1. Overview of Eclipse and PTP	✦ Introduction to Eclipse/PTP	Greg
9:00-10:00	2. Installation	✦ Prerequisites, Installation	Greg
10:00-10:30	BREAK		
10:30-11:15	3. CDT: Working with C/C++	✦ Eclipse basics; Creating a new project ✦ Building and launching	Beth
11:15-12:00 1:30-2:15	4. Working with MPI (incl. Remote)	✦ CVS, Makefiles, autoconf, PLDT MPI tools ✦ Resource Managers ✦ Launching a parallel application	Jay
12:00 - 1:30	Lunch		
2:15-3:00	5. Debugging	✦ Debugging an MPI program	Greg
3:00-3:30	BREAK		
3:30-4:00	6. Fortran; Refactoring	✦ Photran overview; comparison w/ CDT ✦ Refactoring support	Jeff
4:00 – 4:45	7. Advanced Features: Performance Tuning & Analysis Tools	✦ PLDT (MPI, OpenMP, UPC tools) (10 min) ✦ TAU, ETFw (15), PPW (5) ✦ ISP (15)	Beth Wyatt/Max Alan
4:45- 5:00	8. Other Tools, Wrapup	✦ NCSA HPC Workbench, Other Tools, website, mailing lists, future features	Jay/Beth

Module 1: Introduction

- Objective
  - To introduce the Eclipse platform and PTP
- Contents
  - What is Eclipse?
  - What is PTP?

What is Eclipse?

- A vendor-neutral open-source workbench for multi-language development
- A extensible platform for tool integration
- Plug-in based framework to create, integrate and utilize software tools



Eclipse Platform

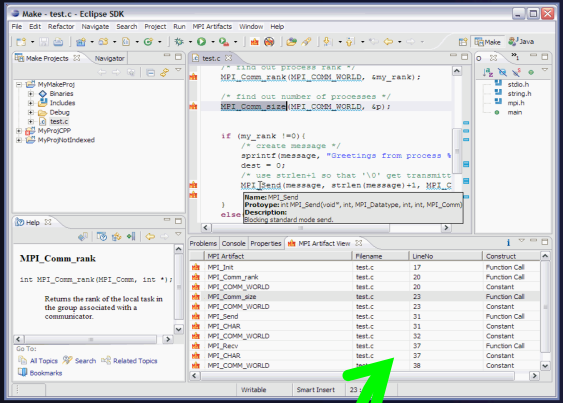
- Core frameworks and services with which all plug-in extensions are created
- Represents the common facilities required by most tool builders:
  - Workbench user interface
  - Project model for resource management
  - Portable user interface libraries (SWT and JFace)
  - Automatic resource delta management for incremental compilers and builders
  - Language-independent debug infrastructure
  - Distributed multi-user versioned resource management (CVS supported in base install)
  - Dynamic update/install service

Plug-ins

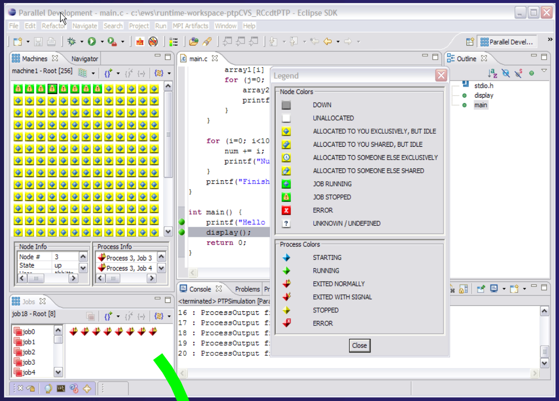
- Java Development Tools (JDT)
- Plug-in Development Environment (PDE)
- C/C++ Development Tools (CDT)
- Parallel Tools Platform (PTP)
- Fortran Development Tools (Photran)
- Test and Performance Tools Platform (TPTP)
- Business Intelligence and Reporting Tools (BIRT)
- Web Tools Platform (WTP)
- Data Tools Platform (DTP)
- Device Software Development Platform (DSDP)
- Many more…

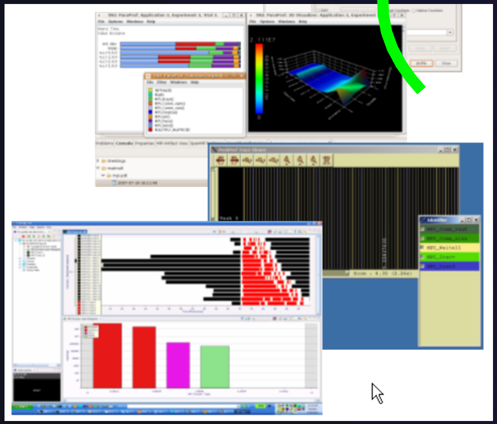
Eclipse Parallel Tools Platform (PTP)

Coding & Analysis

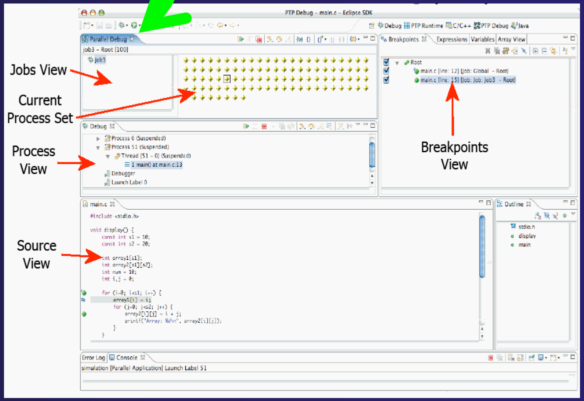


Launching & Monitoring





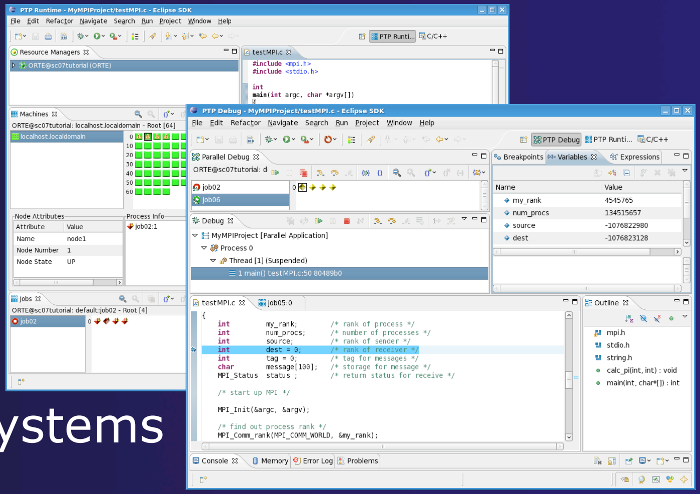
Performance Tuning



Debugging

Parallel Tools Platform (PTP)

- The Parallel Tools Platform aims to provide a highly integrated environment specifically designed for parallel application development
- Features include:
  - An integrated development environment (IDE) that supports a wide range of parallel architectures and runtime systems
  - A scalable parallel debugger
  - Parallel programming tools (MPI/OpenMP)
  - Support for the integration of parallel tools
  - An environment that simplifies the end-user interaction with parallel systems
- http://www.eclipse.org/ptp


PTP Runtime - MyMPIProject/testMPI.c - Eclipse SDK
PTP Debug - MyMPIProject/testMPI.c - Eclipse SDK
File Edit Refactor Navigate Search Run Project Window Help
PTP Debug
PTP Runti...
C/C++
Resource Managers
Machines
Jobs
Parallel Debug
Breakpoints
Variables
Expressions
Debug
Outline
Console
Memory
Error Log
Problems

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

int
main(int argc, char *argv[])
```

Name	Value
my_rank	4545765
num_procs	134515657
source	-1076822980
dest	-1076823128

```
MyMPIProject [Parallel Application]
  Process 0
    Thread [1] (Suspended)
      1 main() testMPI.c:50 80489b0
```

```
{
    int         my_rank;        /* rank of process */
    int         num_procs;      /* number of processes */
    int         source;         /* rank of sender */
    int         dest = 0;       /* rank of receiver */
    int         tag = 0;        /* tag for messages */
    char        message[100];   /* storage for message */
    MPI_Status  status ;        /* return status for receive */

    /* start up MPI */

    MPI_Init(&argc, &argv);

    /* find out process rank */
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
```

PTP Features Demo…

- Check out a project from CVS
- Team features
- Content assist, searching, include browser
- Building the project
- Launching an MPI program
- Debugging an MPI program

Module 2: Installation

- Objective
  - To learn how to install Eclipse and PTP
- Contents
  - System Prerequisites
  - Software Prerequisites
  - Eclipse Installation
  - PTP Installation

About PTP Installation

- PTP 3.0 isn’t “official” until Nov. 30.
  @SC09 Tutorial: we’re installing a pre-release of PTP 3.0
- Note: up-to-date info on installing PTP and its pre-reqs is available from the release notes:

  http://wiki.eclipse.org/PTP/release_notes/3.0

- The above information may supersede these slides
- *@SC09 Tutorial: specific instructions will follow, referencing files available on USB drive or CD during the tutorial*

System Prerequisites

- Local system (running Eclipse)
  - Linux (just about any version)
  - MacOSX (10.5 Leopard or 10.6 Snow Leopard)
  - Windows (XP on)
- Remote system (running/debugging application)
  - Must be supported by a resource manager
  - Open MPI 1.2+
  - MPICH 2
  - IBM PE & LoadLeveler (AIX or Linux)
  - SLURM (Linux)

Software Prerequisites

- Java (1.5 or later)
- Cygwin or MinGW (for local development on Windows)
- Unix make or equivalent
- Supported compilers (gcc, gfortran, Intel, etc.)
- Gdb for debugging (or a gdb-like interface)
- Gcc for building the debugger and SLURM proxies from source
- IBM C for building the PE/LoadLeveler proxies from source

Java Prerequisite

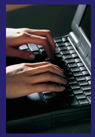
- Eclipse requires Sun or IBM versions of Java
  - Only need Java runtime environment (JRE)
  - Java 1.5 is the same as JRE 5.0
  - The GNU Java Compiler (GCJ), which comes standard on Linux, will not work!

Eclipse and PTP Installation

- Eclipse is installed in two steps
  - First, the base Eclipse package is downloaded and installed
  - Additional functionality is obtained by adding ‘features’
    - This can be done via an `update site’ that automatically downloads and installs the features
    - Update site archives can be downloaded to install features offline.
- PTP requires the following Eclipse features
  - C/C++ Development Tools (CDT)
  - Remote Systems Explorer (RSE) end-user runtime
    - Required only if you are remotely *developing*

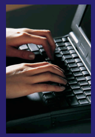
Eclipse Packages

- Eclipse is available in a number of different packages for different kinds of development
- Two packages are more relevant for HPC:
  - Eclipse Classic
    - The full software development kit (SDK), including Java and Plug-in development tools
  - Eclipse IDE for C/C++ developers
    - Base Eclipse distribution
    - Base C/C++ Development Tools (CDT) (does not include UPC, but it can be added)
    - Smaller and less cluttered than full SDK
- Either is fine for PTP use



Eclipse Installation

- The current version of Eclipse is 3.5.1 (Galileo)
  - PTP 3.0 will only work with this version
- Eclipse is downloaded as a single zip or gzipped tar file from http://eclipse.org/downloads
  @SC09 Tutorial: available on USB or CD.
- You *must* download the correct version to suit your local environment
  - Must have correct operating system version
  - Must have correct window system version
- Unzipping or untarring this file creates a directory containing the main executable



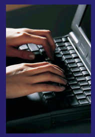
Eclipse Installation Files

Install from one of these files, depending on your platform, and SDK or C/C++ version:

- eclipse-SDK-3.5.1-win32.zip
- eclipse-SDK-3.5.1-macosx-cocoa[-x86_64].tar.gz
- eclipse-SDK-3.5.1-linux-gtk[-x86_64].tar.gz

- eclipse-cpp-galileo-SR1-win32.zip
- eclipse-cpp-galileo-SR1-linux-gtk[-x86_64].tar.gz
- eclipse-cpp-galileo-SR1-macosx-cocoa.tar.gz

Unzip or untar on your machine.
Creates ‘eclipse’ directory containing executable



Starting Eclipse

- **Linux**
  - From a terminal window, enter

```
<eclipse_installation>/eclipse/eclipse &
```

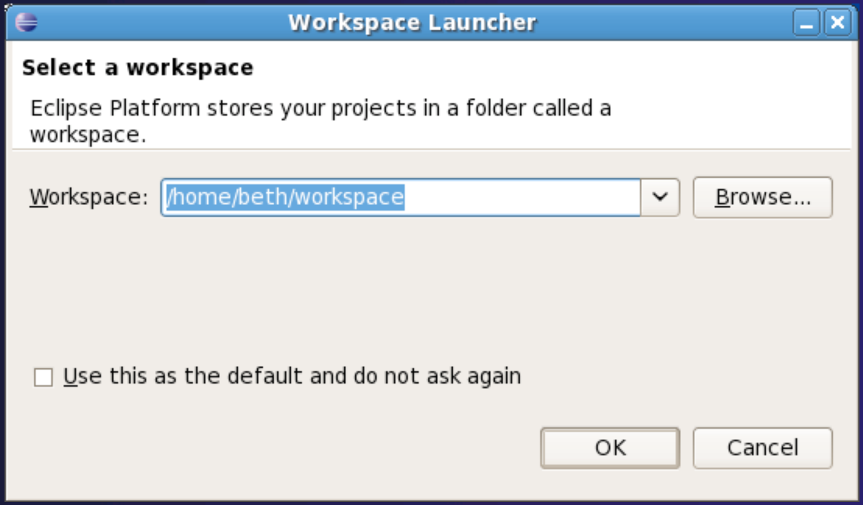
- **MacOS X**
  - From finder, open the **eclipse** folder where you installed
  - Double-click on the **Eclipse** application
  - Or from a terminal window
- **Windows**
  - Open the **eclipse** folder
  - Double-click on the **eclipse** executable

- Accept default workspace when asked
- Select workbench icon from welcome page

Specifying A Workspace

- Eclipse prompts for a workspace location at startup time
- The workspace contains all user-defined data
  - Projects and resources such as folders and files

The prompt can be turned off



Select a workspace

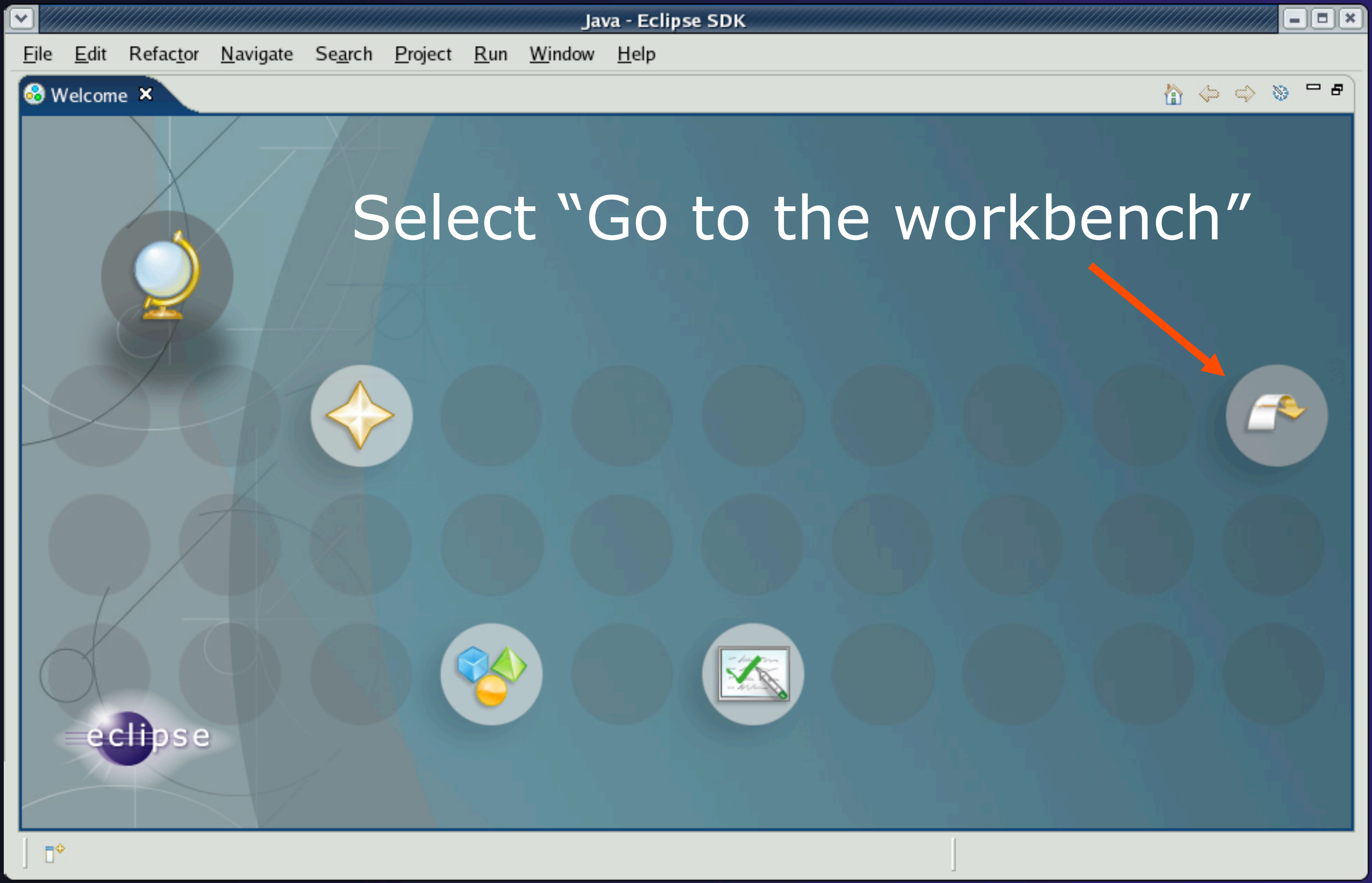
Eclipse Platform stores your projects in a folder called a workspace.

Workspace: /home/beth/workspace

☐ Use this as the default and do not ask again

Eclipse Welcome Page

- Displayed when Eclipse is run for the first time


Java - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
Welcome
Select "Go to the workbench"
eclipse

Adding Features

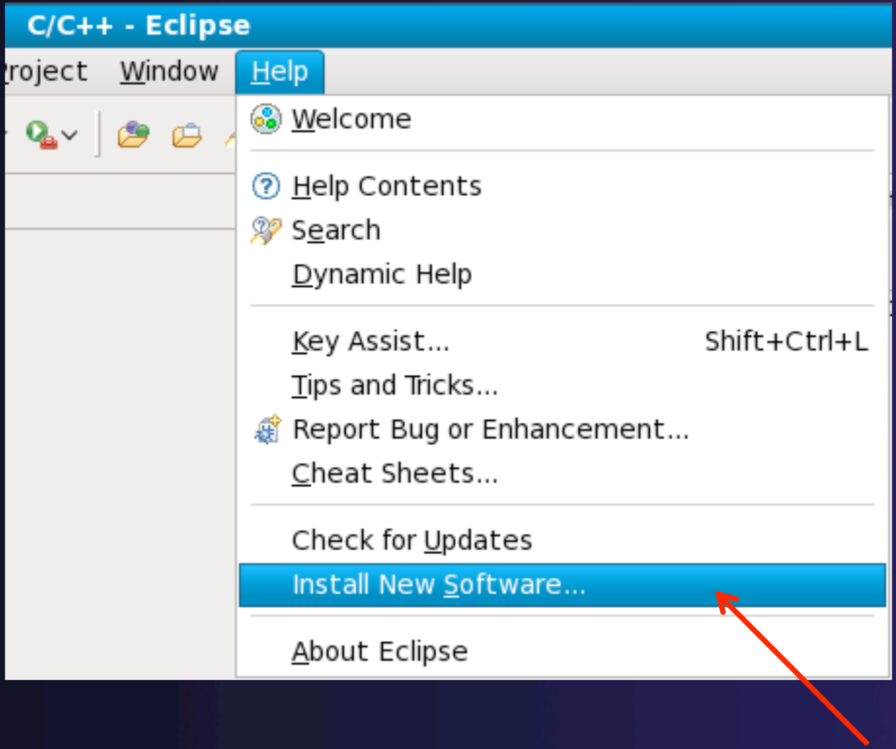
- New functionality is added to Eclipse using *features*
- Features are obtained and installed from an update site (like a web site)
- Features can also be installed from a local copy of the update site (which can be zipped archives)
  - allows for offline installation
  - *@SC09 Tutorial: we will install from archived update sites*

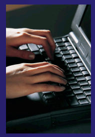
Installing Eclipse Features from an Update Site

- Three types of update sites
  - **Remote** - download and install from remote server
  - **Local** - install from local directory
  - **Archived** - a local site packaged as a zip or jar file
- Eclipse 3.5 comes preconfigured with a link to the **Galileo** Update Site
  - This is a remote site that contains a large number of official features
  - Galileo projects are guaranteed to work with Eclipse 3.5
- Many other sites offer Eclipse features
  - Use at own risk

Installing from an Update Site

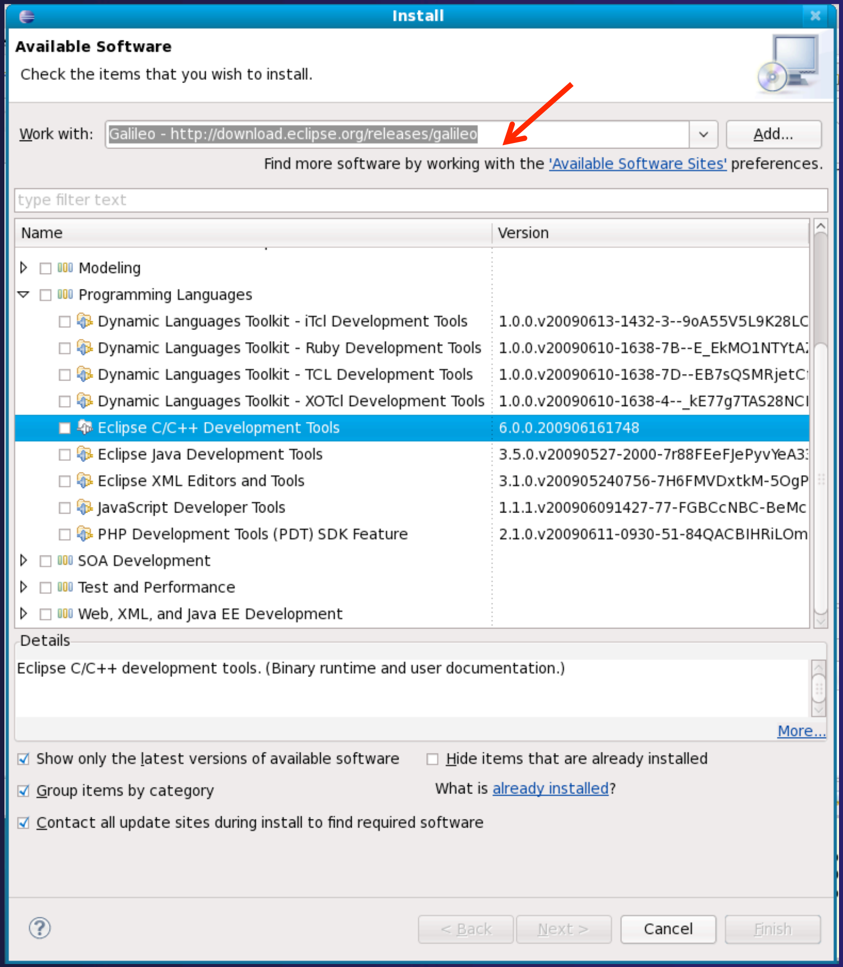
- From the **Help** menu, choose **Install New Software…**





Galileo Update Site

- The Galileo site comes already configured with Eclipse
- For example, some of the contents of the Galileo site: →
- You can get C/C++ Dev. Tools from the Galileo site, but…
  - Basic tools only, does not include UPC
  - More complete CDT install shown later



Install

Available Software

Check the items that you wish to install.

Work with: Galileo - http://download.eclipse.org/releases/galileo Add...

Find more software by working with the 'Available Software Sites' preferences.

type filter text

Name	Version
▷ ☐ Modeling	
▽ ☐ Programming Languages	
☐ Dynamic Languages Toolkit - iTcl Development Tools	1.0.0.v20090613-1432-3--9oA55V5L9K28LC
☐ Dynamic Languages Toolkit - Ruby Development Tools	1.0.0.v20090610-1638-7B--E_EkMO1NTYtA;
☐ Dynamic Languages Toolkit - TCL Development Tools	1.0.0.v20090610-1638-7D--EB7sQSMRjetC
☐ Dynamic Languages Toolkit - XOTcl Development Tools	1.0.0.v20090610-1638-4--_kE77g7TAS28NCI
☐ Eclipse C/C++ Development Tools	6.0.0.200906161748
☐ Eclipse Java Development Tools	3.5.0.v20090527-2000-7r88FEeFJePyvYeA3
☐ Eclipse XML Editors and Tools	3.1.0.v200905240756-7H6FMVDxtkM-5OgP
☐ JavaScript Developer Tools	1.1.1.v200906091427-77-FGBCcNBC-BeMc
☐ PHP Development Tools (PDT) SDK Feature	2.1.0.v20090611-0930-51-84QACBIHRiLOm
▷ ☐ SOA Development	
▷ ☐ Test and Performance	
▷ ☐ Web, XML, and Java EE Development	

Details

Eclipse C/C++ development tools. (Binary runtime and user documentation.)

More...

☑ Show only the latest versions of available software ☐ Hide items that are already installed

☑ Group items by category What is already installed?

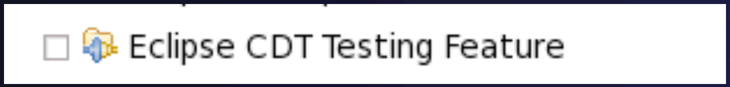
☑ Contact all update sites during install to find required software

Installation: CDT

- If you installed Eclipse classic (full SDK)
  - you will now need to install CDT.
- If you installed the C/C++ IDE instead
  - You will need to update CDT to get 6.0.2 (required for PTP 3.0)
  - You may want to install other features too, for example: the UPC feature.
    The C/C++ IDE includes only the most basic CDT features.

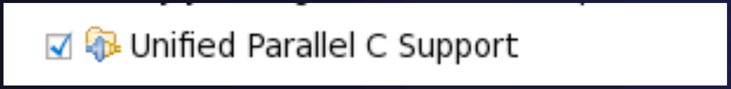
Installation: CDT (2)

- PTP 3.0 needs CDT 6.0.2
  - Update site contains only 6.0.1 as of this writing
  - Update site: http://download.eclipse.org/tools/cdt/releases/galileo
- *CDT 6.0.2 archived update site file is available from: download.eclipse.org/tools/cdt/builds*
  @SC09 Tutorial:
  Add… Archive… cdt-master-6.0.2.zip*
- Install any features you want
  - Suggestion: install everything *but:* Omit the testing feature:

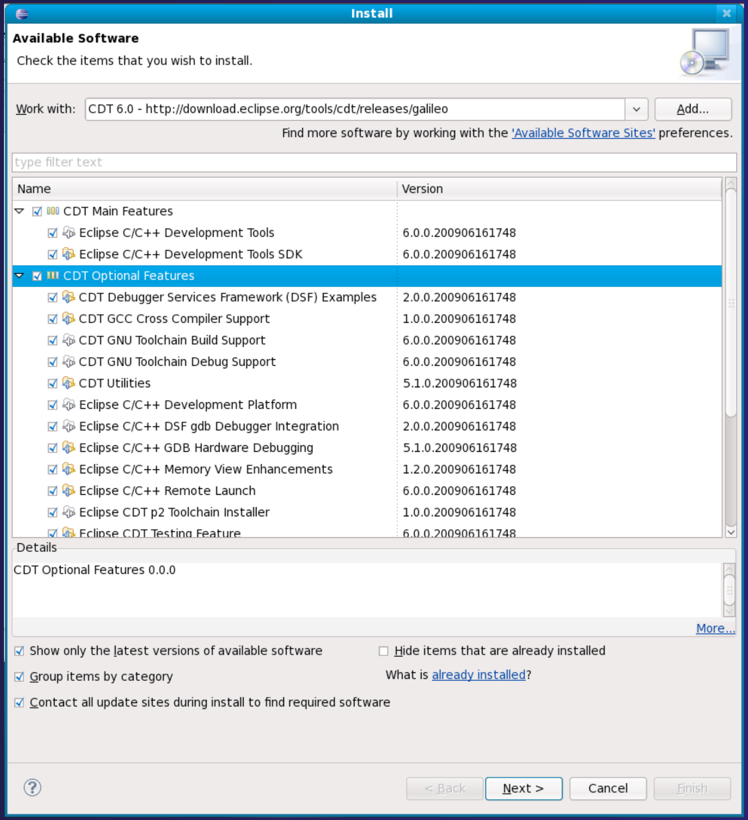


☐ Eclipse CDT Testing Feature

  - This includes the UPC feature (optional)



☑ Unified Parallel C Support



Available Software

Check the items that you wish to install.

Work with: CDT 6.0 - http://download.eclipse.org/tools/cdt/releases/galileo

Find more software by working with the 'Available Software Sites' preferences.

Name	Version
☑ CDT Main Features	
☑ Eclipse C/C++ Development Tools	6.0.0.200906161748
☑ Eclipse C/C++ Development Tools SDK	6.0.0.200906161748
☑ CDT Optional Features	
☑ CDT Debugger Services Framework (DSF) Examples	2.0.0.200906161748
☑ CDT GCC Cross Compiler Support	1.0.0.200906161748
☑ CDT GNU Toolchain Build Support	6.0.0.200906161748
☑ CDT GNU Toolchain Debug Support	6.0.0.200906161748
☑ CDT Utilities	5.1.0.200906161748
☑ Eclipse C/C++ Development Platform	6.0.0.200906161748
☑ Eclipse C/C++ DSF gdb Debugger Integration	2.0.0.200906161748
☑ Eclipse C/C++ GDB Hardware Debugging	5.1.0.200906161748
☑ Eclipse C/C++ Memory View Enhancements	1.2.0.200906161748
☑ Eclipse C/C++ Remote Launch	6.0.0.200906161748
☑ Eclipse CDT p2 Toolchain Installer	1.0.0.200906161748
☑ Eclipse CDT Testing Feature	6.0.0.200906161748

Details

CDT Optional Features 0.0.0

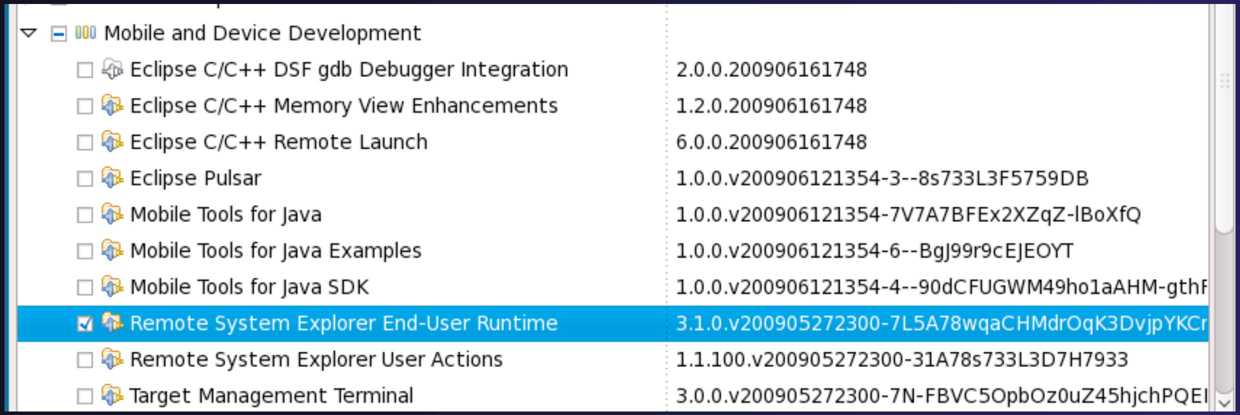
☑ Show only the latest versions of available software
☐ Hide items that are already installed
☑ Group items by category
What is already installed?
☑ Contact all update sites during install to find required software

Installation: CDT (3)

- Finish installing CDT:
  - Next, Confirm features, Next, Accept license terms, Finish
- You do *not* need to restart Eclipse, since we will now install RSE and PTP.

Installation: RSE

- Again: Help > Install New Software…
- The RSE End-User Runtime can be installed from the Galileo update site



☐ Mobile and Device Development	
☐ Eclipse C/C++ DSF gdb Debugger Integration	2.0.0.200906161748
☐ Eclipse C/C++ Memory View Enhancements	1.2.0.200906161748
☐ Eclipse C/C++ Remote Launch	6.0.0.200906161748
☐ Eclipse Pulsar	1.0.0.v200906121354-3--8s733L3F5759DB
☐ Mobile Tools for Java	1.0.0.v200906121354-7V7A7BFEx2XZqZ-lBoXfQ
☐ Mobile Tools for Java Examples	1.0.0.v200906121354-6--BgJ99r9cEJEOYT
☐ Mobile Tools for Java SDK	1.0.0.v200906121354-4--90dCFUGWM49ho1aAHM-gthF
☑ Remote System Explorer End-User Runtime	3.1.0.v200905272300-7L5A78wqaCHMdrOqK3DvjpYKCr
☐ Remote System Explorer User Actions	1.1.100.v200905272300-31A78s733L3D7H7933
☐ Target Management Terminal	3.0.0.v200905272300-7N-FBVC5OpbOz0uZ45hjchPQEI

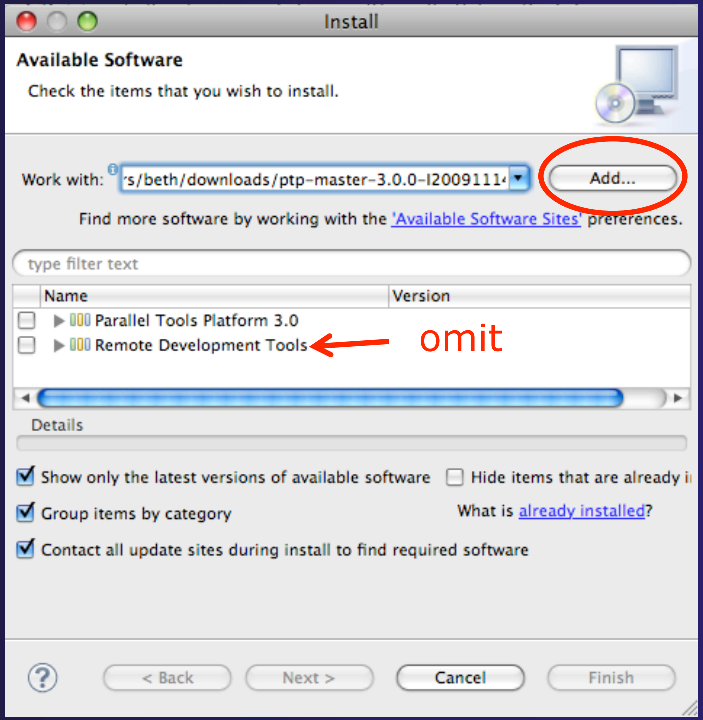
@SC09 Tutorial: Use archived update site
Add… Archive… choose rse-3.1-updateSite.zip file

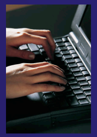
- Choose TM and RSE 3.1.1 / RSE End-User Runtime
  - If it tries to install 3.1.0 also, uncheck it and hit Next >
  - Accept license terms etc., no need to restart yet

Installing PTP

- Help > Install New Software hit **Add…**
- PTP update site:
  http://download.eclipse.org/tools/ptp/releases/galileo/
  @SC09 Tutorial: Select Archive…
  and enter ptp-master-.zip*
- Click **OK** and the list of features on the update site will be populated
- Select all the components you require.
  Suggestion: select only "Parallel Tools Platform 3.0"
  @SC09 Tutorial:
  Omit Remote Development Tools
- Click **Next>, Next>, Finish**
- Select **Yes** to restart Eclipse now

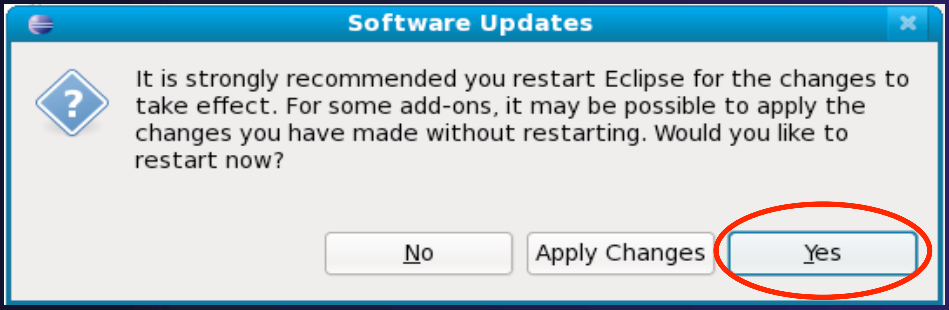
See PTP release notes for most recent info on installing 3.0
http://wiki.eclipse.org/PTP/release_notes/3.0


Install
Available Software
Check the items that you wish to install.
Work with: 's/beth/downloads/ptp-master-3.0.0-I2009111
Add...
Find more software by working with the 'Available Software Sites' preferences.
type filter text
Name
Version
Parallel Tools Platform 3.0
Remote Development Tools
omit
Details
Show only the latest versions of available software
Hide items that are already i
Group items by category
What is already installed?
Contact all update sites during install to find required software
< Back
Next >
Cancel
Finish



Installing PTP (2)

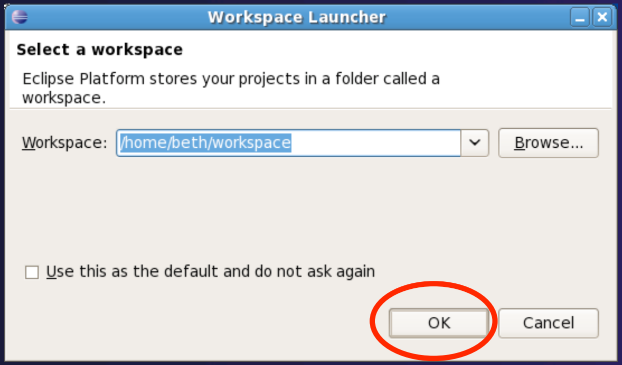
- After selecting **Finish…**
- Restart Eclipse when prompted



Software Updates

It is strongly recommended you restart Eclipse for the changes to take effect. For some add-ons, it may be possible to apply the changes you have made without restarting. Would you like to restart now?

- Select OK when Eclipse restarts, to use the same workspace



Workspace Launcher

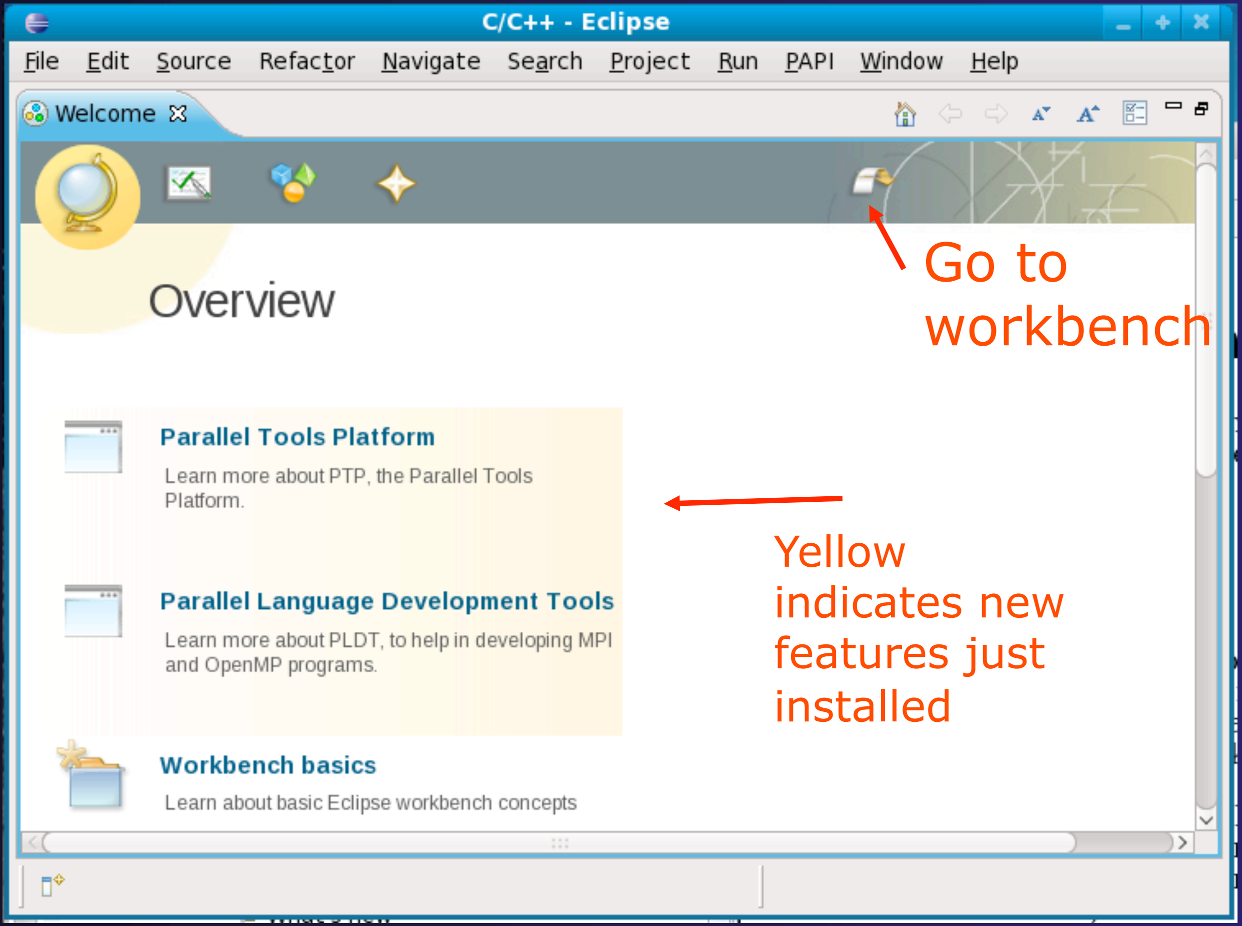
Select a workspace

Eclipse Platform stores your projects in a folder called a workspace.

Workspace: /home/beth/workspace

Restarting Eclipse

- Welcome page informs you of new features installed
- Select workbench icon to go to workbench


C/C++ - Eclipse
File Edit Source Refactor Navigate Search Project Run PAPI Window Help
Welcome
Overview
Go to workbench
Yellow indicates new features just installed

Parallel Tools Platform
Learn more about PTP, the Parallel Tools Platform.

Parallel Language Development Tools
Learn more about PLDT, to help in developing MPI and OpenMP programs.

Workbench basics
Learn about basic Eclipse workbench concepts

Installing Additional PTP Components

- PTP has a number of additional components depending on the installation
  - Scalable Debug Manager (SDM) – required for all platforms to support debugging
  - PE and LoadLeveler proxy – IBM systems only
  - SLURM proxy – systems using the SLURM resource manager
- Installation of these components is beyond the scope of the tutorial
- See the release notes for details of installing these components

Platform Differences

- Single button mouse (e.g. MacBook)
    - Use Control-click for right mouse / context menu
- Context-sensitive help key differences
    - Windows: use **F1** key
    - Linux: use **Shift-F1** keys
    - MacOS X
        - Full keyboard, use **Help** key
        - MacBooks or aluminum keyboard, create a key binding for **Dynamic Help** to any key you want
- Accessing preferences
    - Windows & Linux: **Window▶Preferences…**
    - MacOS X: **Eclipse▶Preferences…**

Module 3: Working with C/C++

- Objective
  - Learn how to use Eclipse to develop C programs
  - Learn how to launch and run a C program
- Contents
  - Brief introduction to the C/C++ Development Tools (CDT)
  - Create a simple application
  - Learn to launch a C application

Installation recap

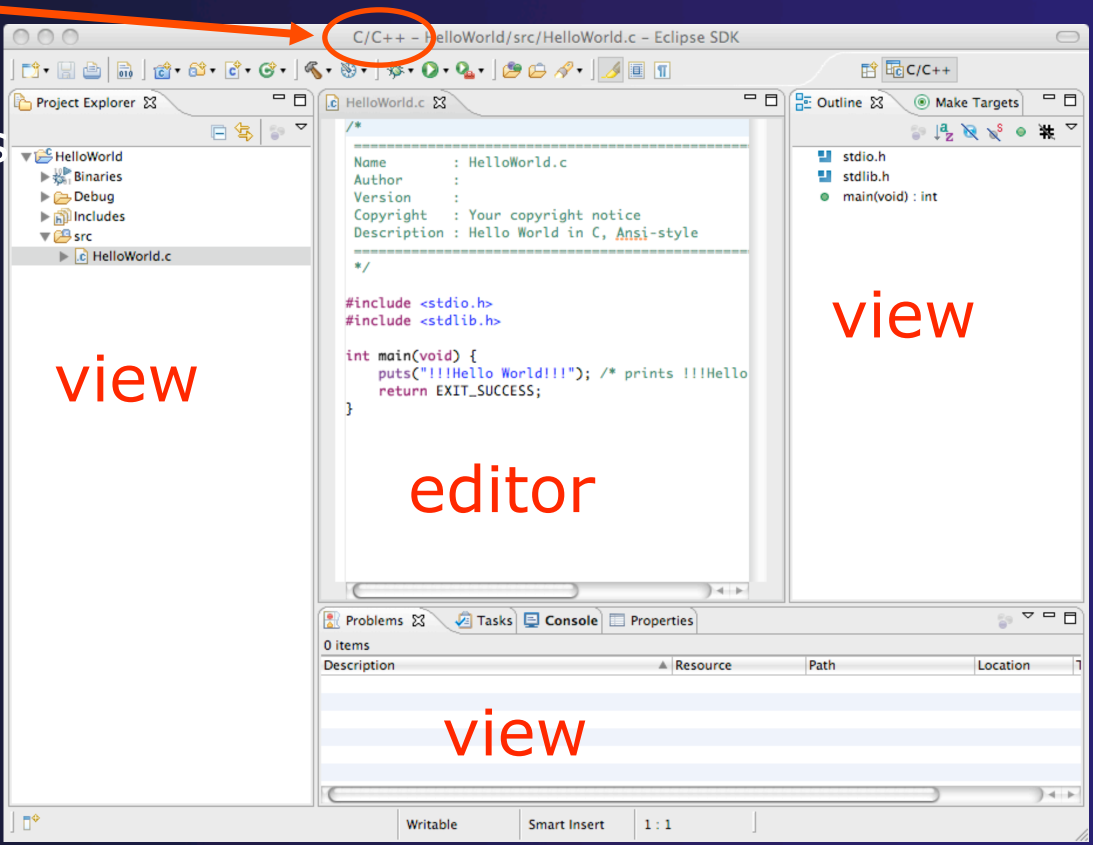
- Download and unzip/untar eclipse
- Use Help >Install new software -- to get
  - CDT for C/C++ tools
  - PTP and related tools for Parallel application work *
- Build PTP binary on target machine (local or remote) *
    - Only required if running parallel apps locally
- Launch eclipse!
  Run the 'eclipse' executable, from icon or from command line

* Not required for this module

Workbench

- A Workbench contains perspectives
- A Perspective contains views and editors

- The Workbench represents the desktop development environment
  - Contains a set of tools for resource mgmt
  - Provides a common way of navigating through the resources
- Multiple workbenches can be opened at the same time



```
/*
 ========================================================
 Name        : HelloWorld.c
 Author      :
 Version     :
 Copyright   : Your copyright notice
 Description : Hello World in C, Ansi-style
 ========================================================
 */

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

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

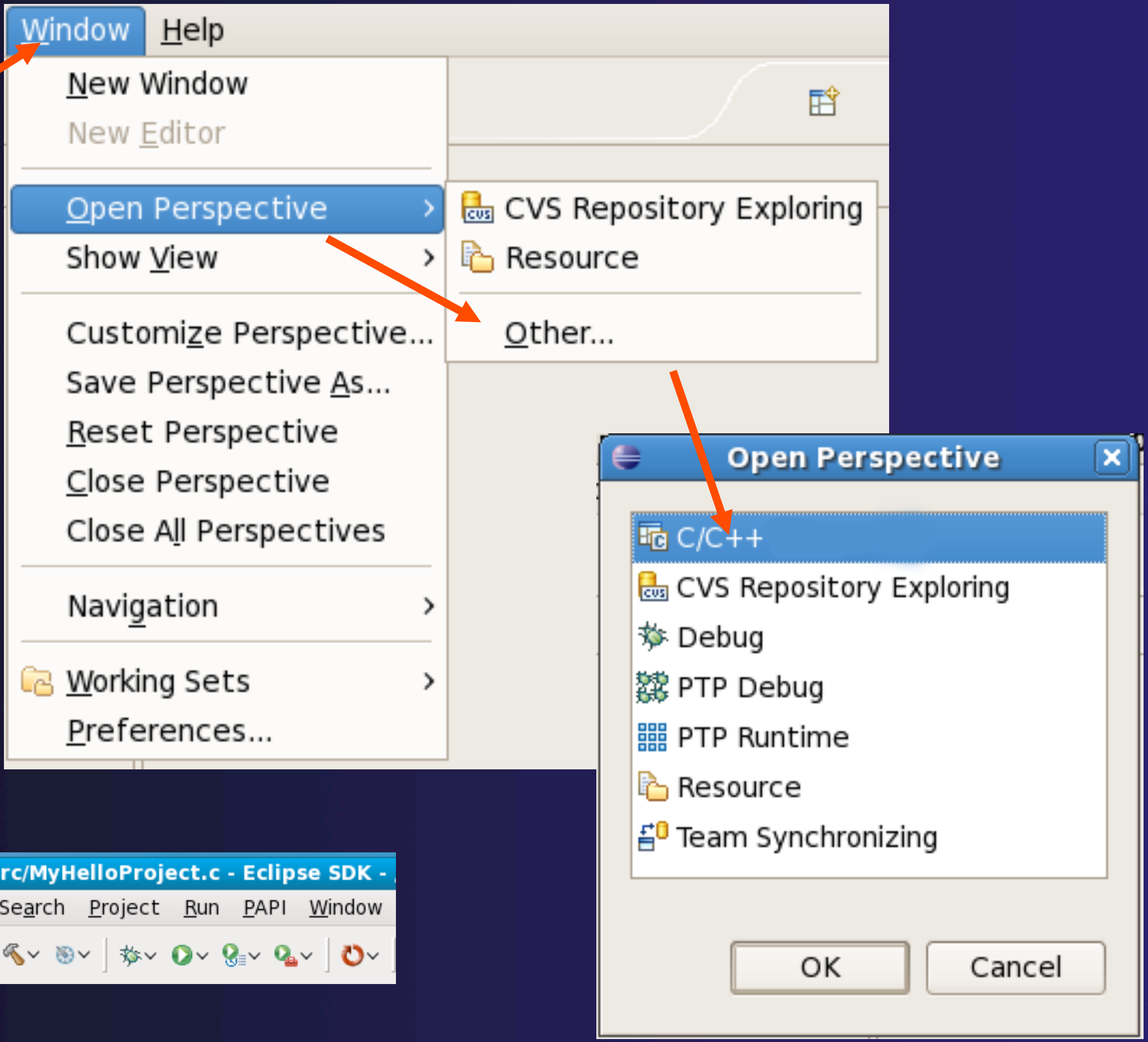
perspective

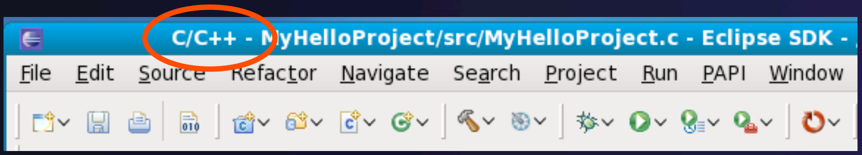
Perspectives

- Perspectives define the layout of views in the Workbench
- They are task oriented, i.e. they contain specific views for doing certain tasks:
  - There is a Resource Perspective for manipulating resources
  - C/C++ Perspective for manipulating compiled code
  - Debug Perspective for debugging applications
- You can easily switch between perspectives

Switch to C/C++ Perspective

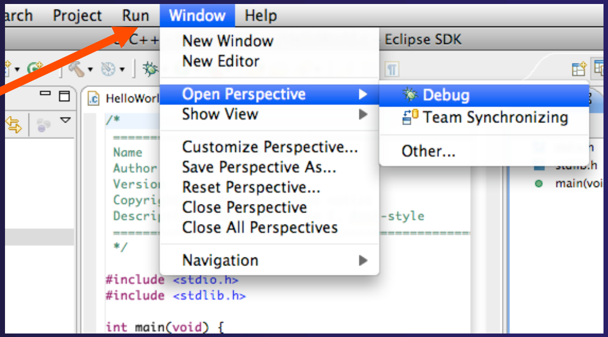
- Only needed if you're not already in the perspective
- What Perspective am in in? See Title Bar

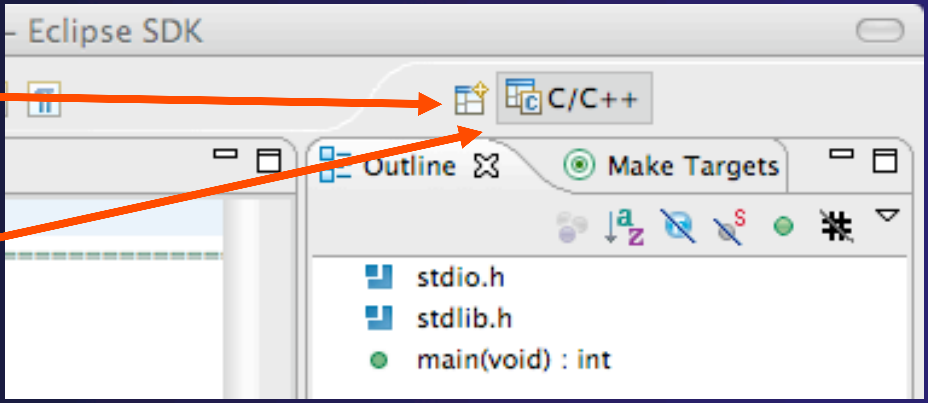




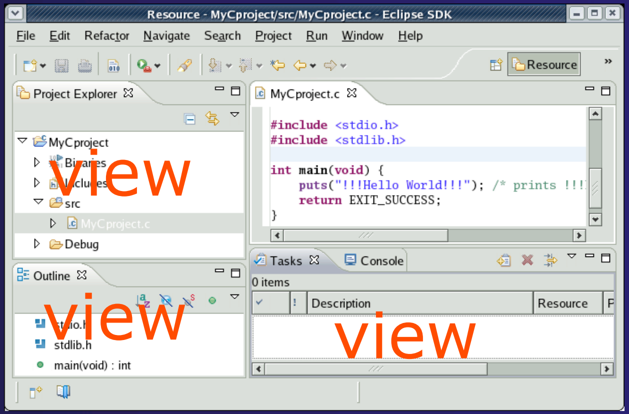
Switching Perspectives

- You can switch Perspectives by:
  - Choosing the **Window▸Open Perspective** menu option
  - Clicking on the **Open Perspective** button
  - Clicking on a perspective shortcut button



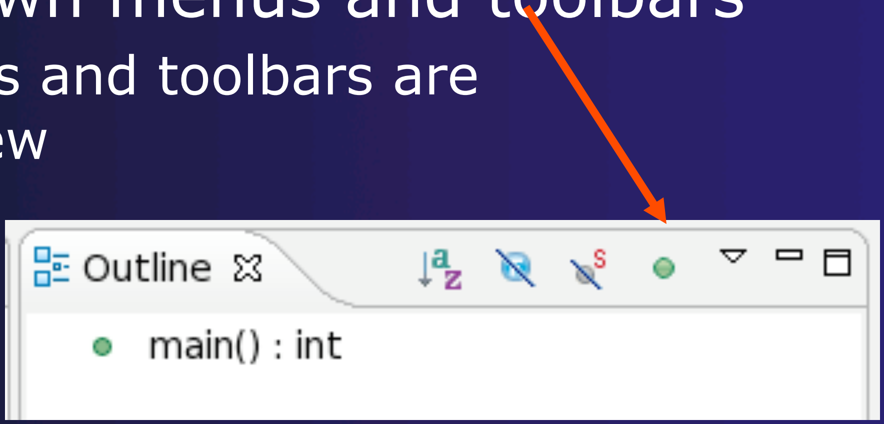


Views



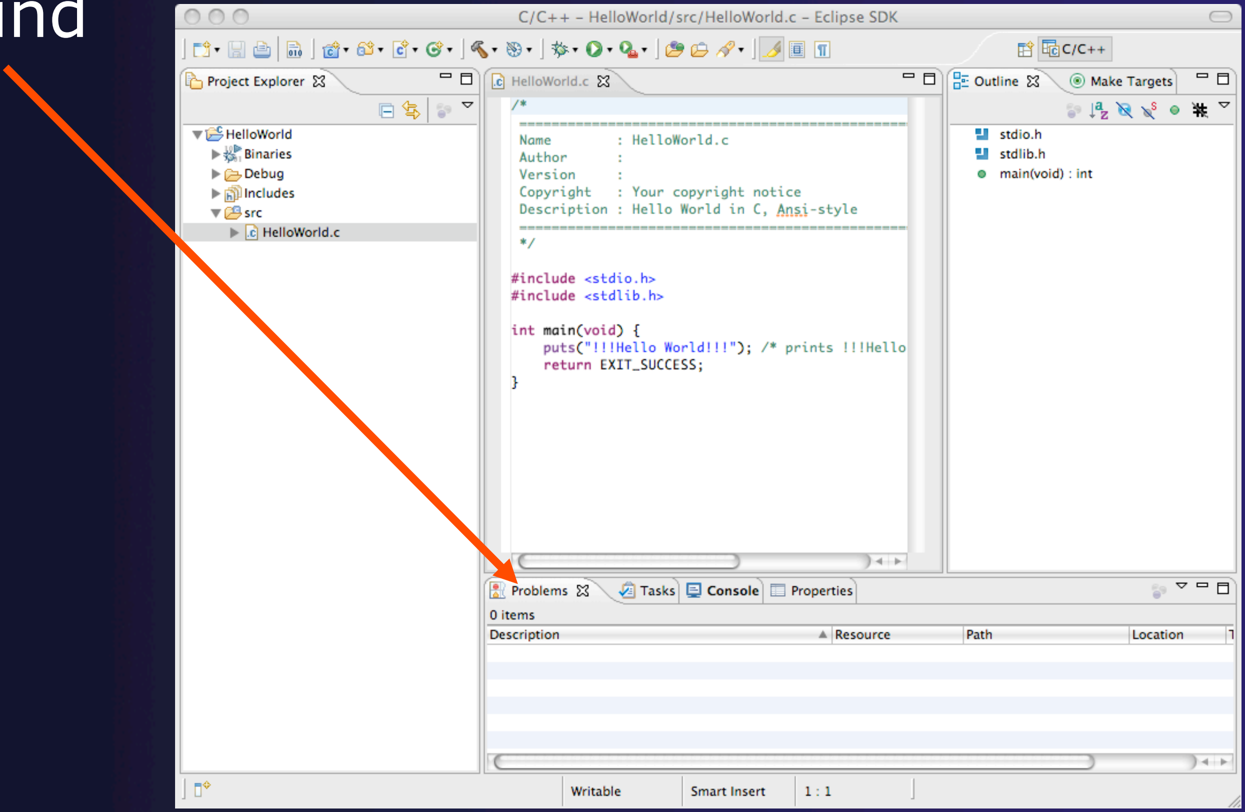
```
#include <stdio.h>
#include <stdlib.h>

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

- The workbench window is divided up into Views
- The main purpose of a view is:
  - To provide alternative ways of presenting information
  - For navigation
  - For editing and modifying information
- Views can have their own menus and toolbars
  - Items available in menus and toolbars are available only in that view
  - Menu actions only apply to the view
- Views can be resized



Stacked Views

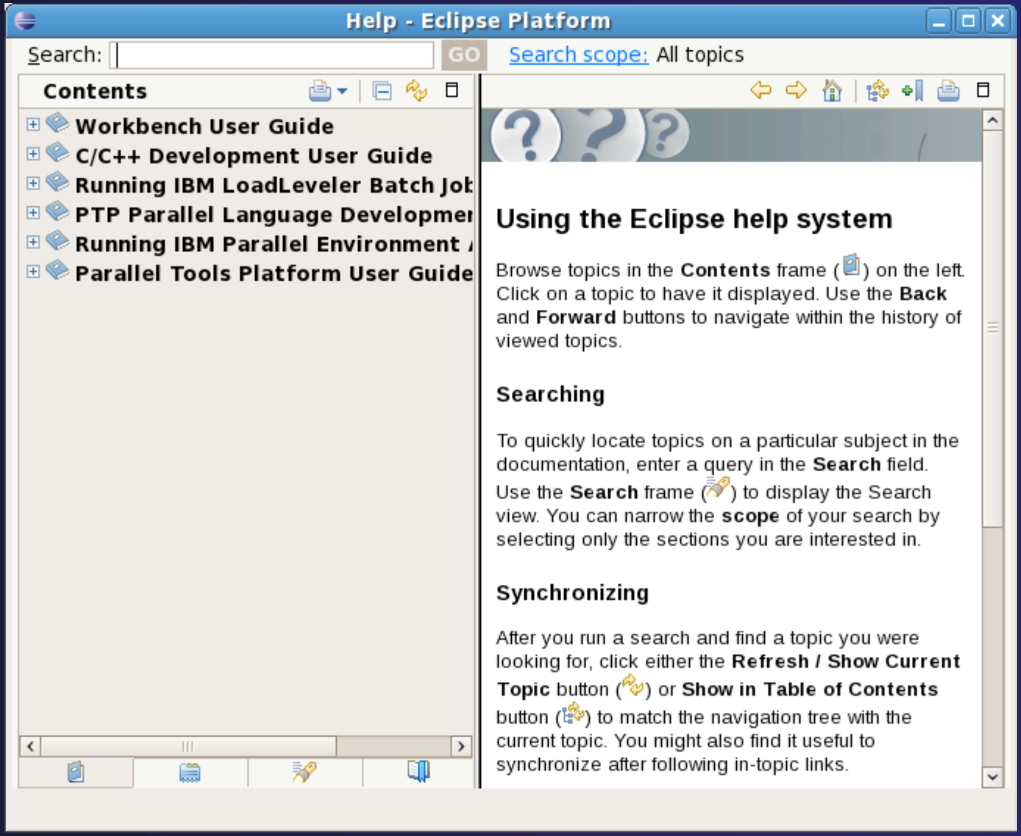
- Stacked views appear as tabs
- Selecting a tab brings that view to the foreground



```
/*
 ============================================================================
 Name        : HelloWorld.c
 Author      :
 Version     :
 Copyright   : Your copyright notice
 Description : Hello World in C, Ansi-style
 ============================================================================
 */

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

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

Help

- Access help
  - **Help▶Help Contents**
  - **Help▶Search**
  - **Help▶Dynamic Help**
- **Help Contents** provides detailed help on different Eclipse features
- **Search** allows you to search for help locally, or using Google or the Eclipse web site
- **Dynamic Help** shows help related to the current context (perspective, view, etc.)


Help - Eclipse Platform
Search:
GO
Search scope: All topics
Contents
Workbench User Guide
C/C++ Development User Guide
Running IBM LoadLeveler Batch Job
PTP Parallel Language Developmen
Running IBM Parallel Environment
Parallel Tools Platform User Guide

Using the Eclipse help system

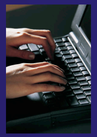
Browse topics in the **Contents** frame ( ) on the left. Click on a topic to have it displayed. Use the **Back** and **Forward** buttons to navigate within the history of viewed topics.

Searching

To quickly locate topics on a particular subject in the documentation, enter a query in the **Search** field. Use the **Search** frame ( ) to display the Search view. You can narrow the **scope** of your search by selecting only the sections you are interested in.

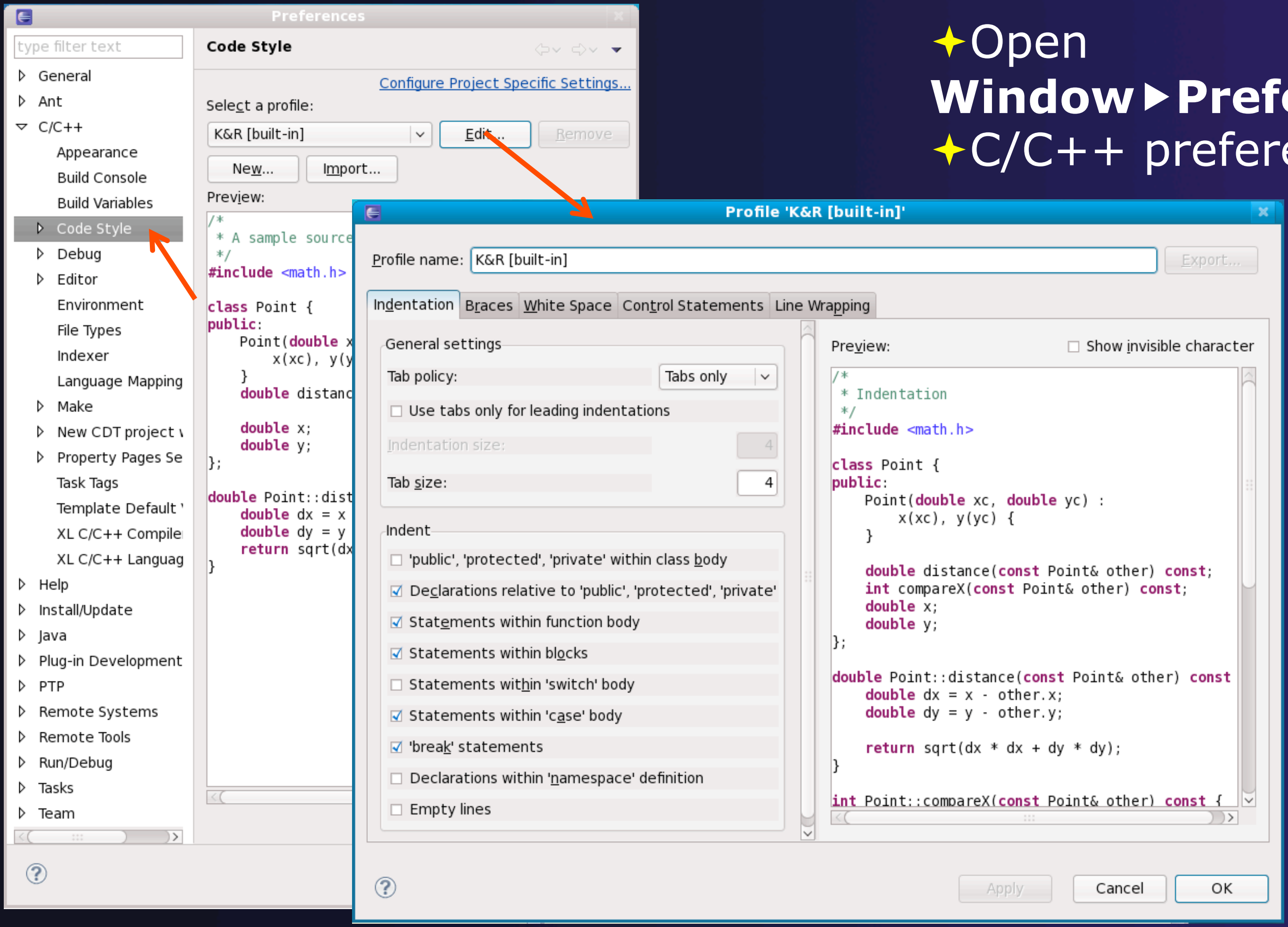
Synchronizing

After you run a search and find a topic you were looking for, click either the **Refresh / Show Current Topic** button ( ) or **Show in Table of Contents** button ( ) to match the navigation tree with the current topic. You might also find it useful to synchronize after following in-topic links.



Preferences

- Eclipse Preferences allow customization of almost everything



Preferences

type filter text

- General
- Ant
- C/C++
  - Appearance
  - Build Console
  - Build Variables
  - Code Style
  - Debug
  - Editor
  - Environment
  - File Types
  - Indexer
  - Language Mapping
  - Make
  - New CDT project w
  - Property Pages Se
  - Task Tags
  - Template Default '
  - XL C/C++ Compile
  - XL C/C++ Languag
- Help
- Install/Update
- Java
- Plug-in Development
- PTP
- Remote Systems
- Remote Tools
- Run/Debug
- Tasks
- Team

Code Style

Configure Project Specific Settings...

Select a profile: K&R [built-in]

Edit... Remove New... Import...

Preview:

```
/*
 * A sample source
 */
#include <math.h>

class Point {
public:
    Point(double x
        x(xc), y(y
    }
    double distanc

    double x;
    double y;
};

double Point::dist
    double dx = x
    double dy = y
    return sqrt(dx
}
```

Profile 'K&R [built-in]'

Profile name: K&R [built-in]   Export...

Indentation | Braces | White Space | Control Statements | Line Wrapping

General settings

Tab policy: Tabs only

- ☐ Use tabs only for leading indentations

Indentation size: 4

Tab size: 4

Indent

- ☐ 'public', 'protected', 'private' within class body
- ☑ Declarations relative to 'public', 'protected', 'private'
- ☑ Statements within function body
- ☑ Statements within blocks
- ☐ Statements within 'switch' body
- ☑ Statements within 'case' body
- ☑ 'break' statements
- ☐ Declarations within 'namespace' definition
- ☐ Empty lines

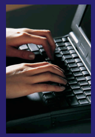
Preview: ☐ Show invisible character

```
/*
 * Indentation
 */
#include <math.h>

class Point {
public:
    Point(double xc, double yc) :
        x(xc), y(yc) {
    }

    double distance(const Point& other) const;
    int compareX(const Point& other) const;
    double x;
    double y;
};

double Point::distance(const Point& other) const
    double dx = x - other.x;
    double dy = y - other.y;

    return sqrt(dx * dx + dy * dy);
}

int Point::compareX(const Point& other) const {
```

Apply   Cancel   OK

- Open **Window ▸ Preferences...**
- C/C++ preferences allow many options

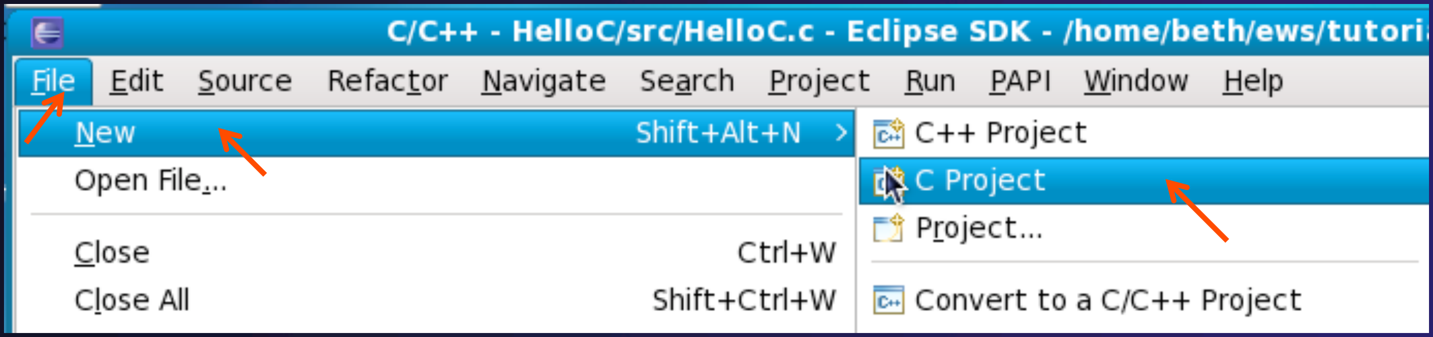
- Code formatting settings (“Code Style”) shown here



Creating a C/C++ Application

Steps:

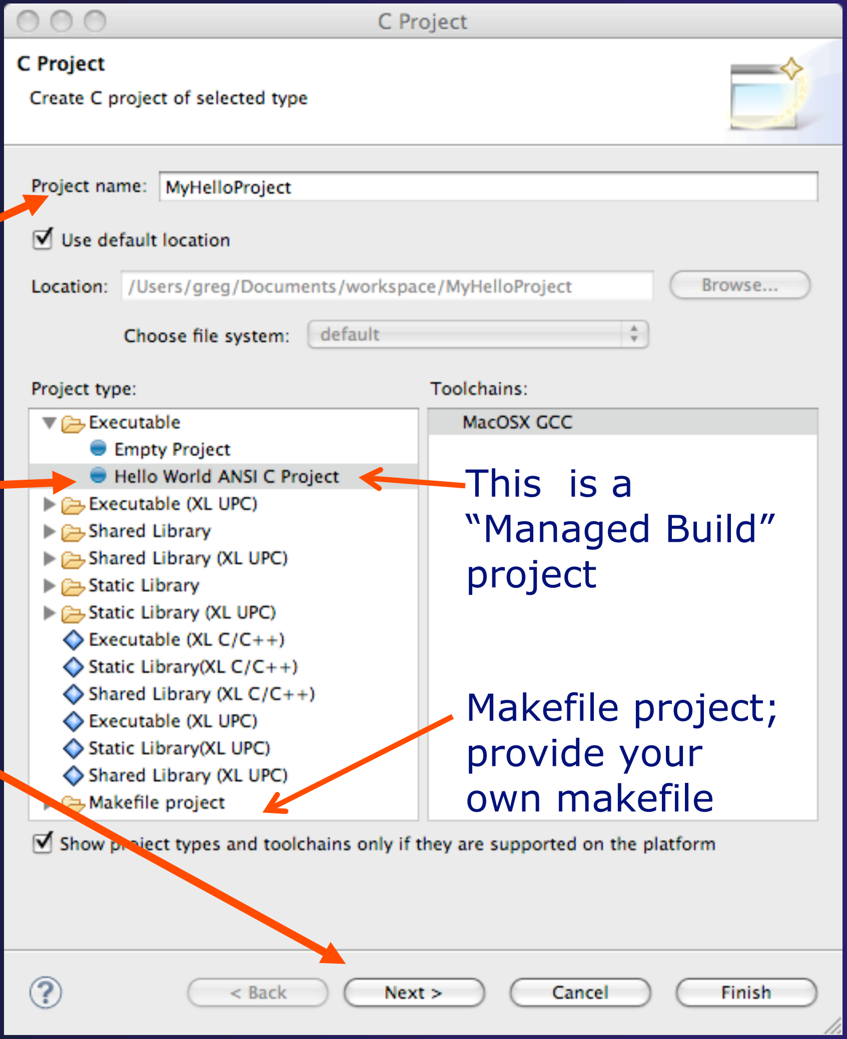
- ✦ Create a new C project
- ✦ Edit source code
- ✦ Save and build


C/C++ - HelloC/src/HelloC.c - Eclipse SDK - /home/beth/ews/tutori
File Edit Source Refactor Navigate Search Project Run PAPI Window Help
New Shift+Alt+N
Open File...
Close Ctrl+W
Close All Shift+Ctrl+W
C++ Project
C Project
Project...
Convert to a C/C++ Project

New C Project Wizard

Create a new C project

- **File▶New▶C Project** (see prev. slide)
- Name the project 'MyHelloProject'
- Under Project types, under Executable, select **Hello World ANSI C Project** (no makefile req'd) and hit **Next**
- On **Basic Settings** page, fill in information for your new project (**Author name** etc.) and hit **Finish**



C Project

C Project

Create C project of selected type

Project name: MyHelloProject

☑ Use default location

Location: /Users/greg/Documents/workspace/MyHelloProject

Choose file system: default

Project type:
- Executable
  - Empty Project
  - Hello World ANSI C Project
- Executable (XL UPC)
- Shared Library
- Shared Library (XL UPC)
- Static Library
- Static Library (XL UPC)
- Executable (XL C/C++)
- Static Library(XL C/C++)
- Shared Library (XL C/C++)
- Executable (XL UPC)
- Static Library(XL UPC)
- Shared Library (XL UPC)
- Makefile project

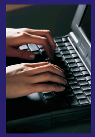
Toolchains: MacOSX GCC

☑ Show project types and toolchains only if they are supported on the platform

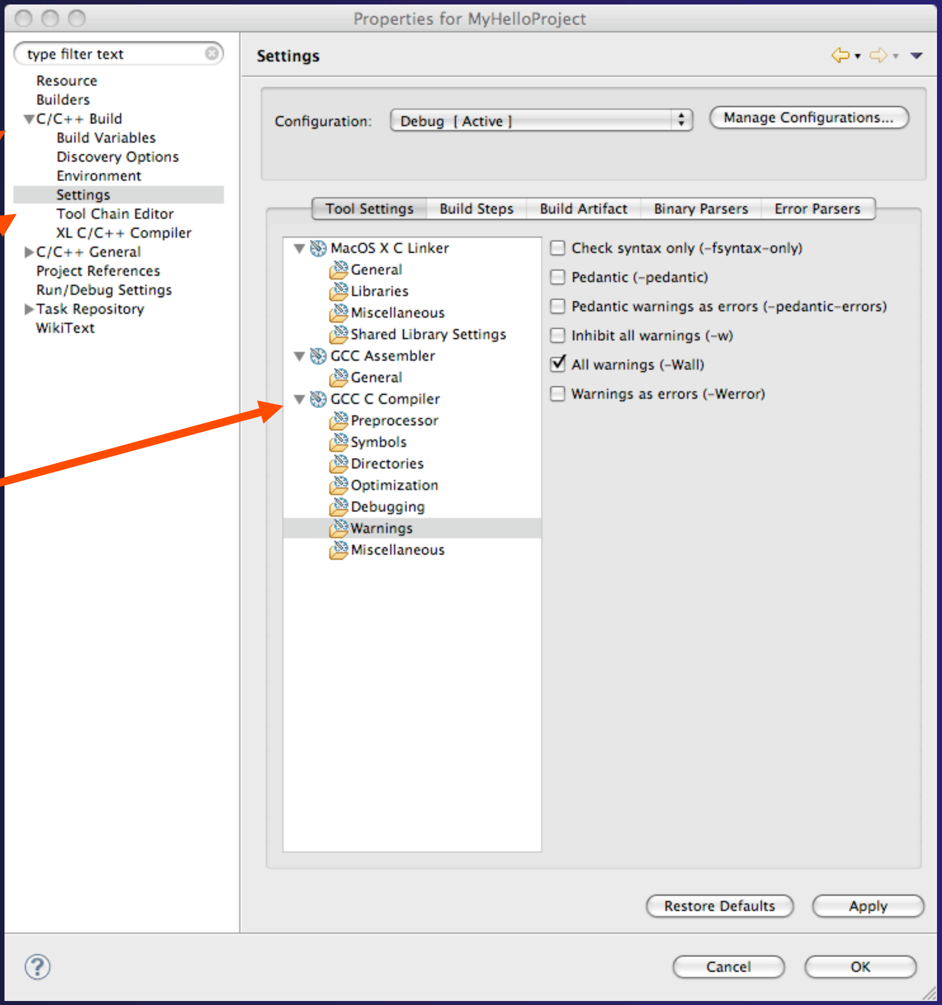
This is a "Managed Build" project

Makefile project; provide your own makefile

Changing the C/C++ Build Settings Manually



- Open the project properties by right-mouse clicking on project and select **Properties**
- Expand **C/C++ Build**
- Select **Settings**
- Select **C Compiler** to change compiler settings
- Select **C Linker** to change linker settings
- It's also possible to change compiler/linker arguments
- Hit **OK** to close



Properties for MyHelloProject

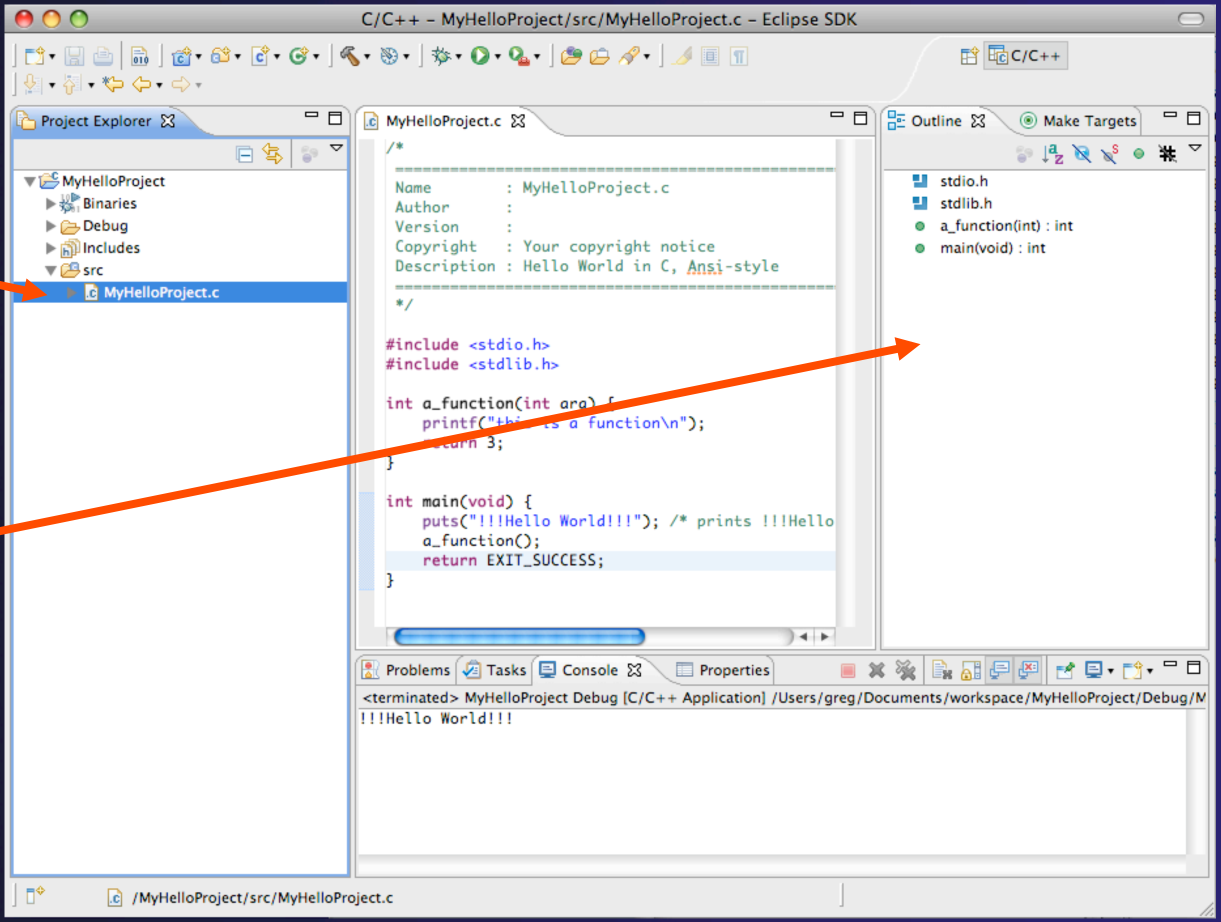
Settings

Configuration: Debug [Active]

- [ ] Check syntax only (-fsyntax-only)
- [ ] Pedantic (-pedantic)
- [ ] Pedantic warnings as errors (-pedantic-errors)
- [ ] Inhibit all warnings (-w)
- [x] All warnings (-Wall)
- [ ] Warnings as errors (-Werror)

Editor and Outline View

- Expand project to see source code
- Double-click on source file in the **Project Explorer** to open C editor
- Outline view is shown for file in editor
- We'll describe the editor in the next few slides…

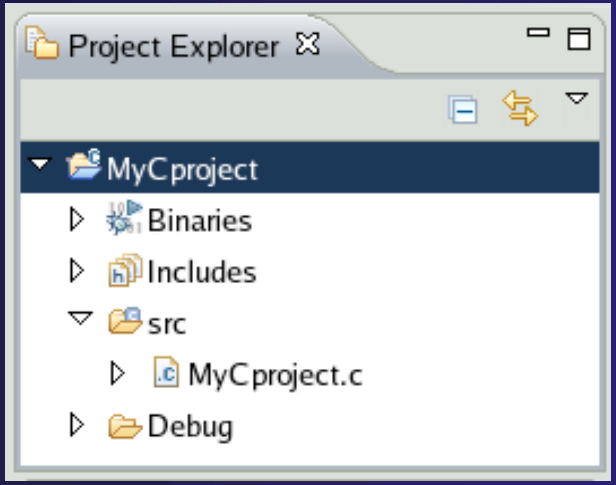


```
/*
 ============================================================
 Name        : MyHelloProject.c
 Author      :
 Version     :
 Copyright   : Your copyright notice
 Description : Hello World in C, Ansi-style
 ============================================================
 */

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

int a_function(int arg) {
    printf("this is a function\n");
    return 3;
}

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

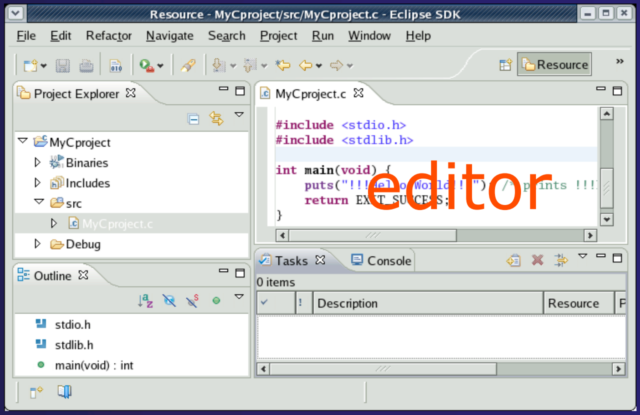
```
<terminated> MyHelloProject Debug [C/C++ Application] /Users/greg/Documents/workspace/MyHelloProject/Debug/M
!!!Hello World!!!
```

Project Explorer View

- Represents user's data
- It is a set of user defined resources
  - Files
  - Folders
  - Projects
    - Collections of files and folders
    - Plus meta-data
- Resources are visible in the Project Explorer View

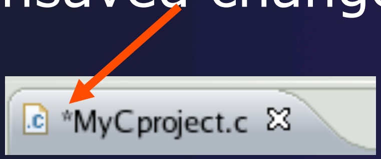


Editors



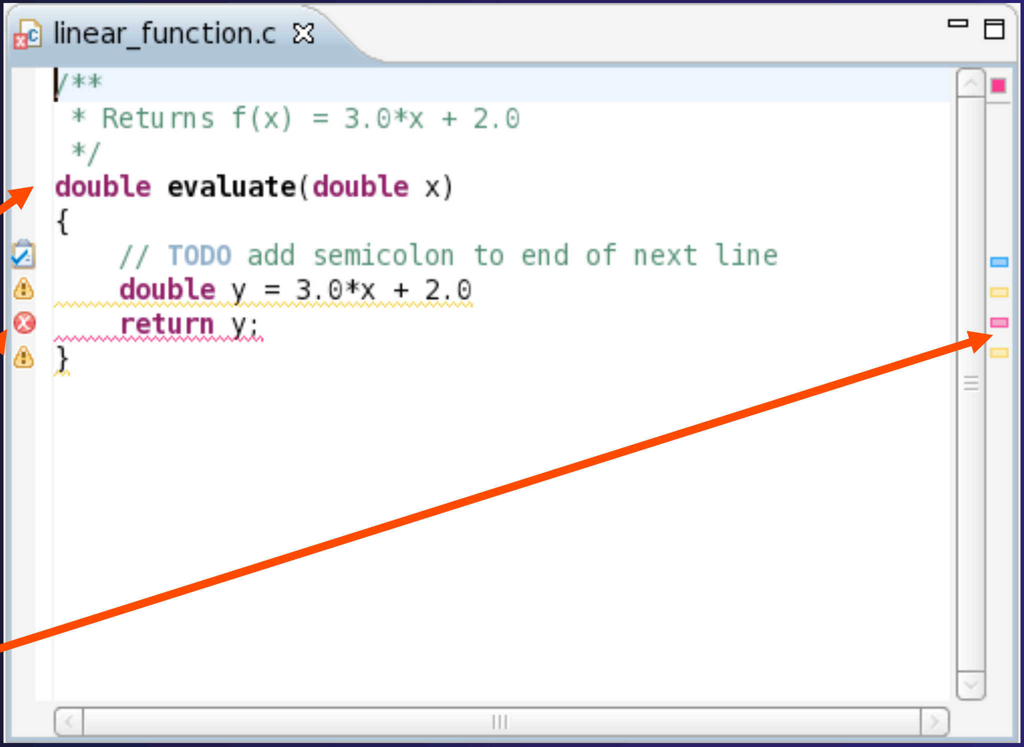
```
#include <stdio.h>
#include <stdlib.h>

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

- An editor for a resource (e.g. a file) opens when you double-click on a resource
- The type of editor depends on the type of the resource
  - .c files are opened with the C/C++ editor
  - Some editors do not just edit raw text
- When an editor opens on a resource, it stays open across different perspectives
- An active editor contains menus and toolbars specific to that editor
- When you change a resource, an asterisk on the editor's title bar indicates unsaved changes
- How to Save



Source Code Editors

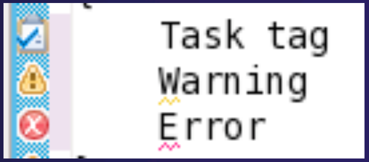
- A source code editor is a special type of editor for manipulating source code
- Language features are highlighted
- Marker bars for showing
  - Breakpoints
  - Errors/warnings
  - Task Tags, Bookmarks
- Location bar for navigating to interesting features in the entire file



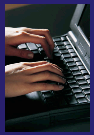
linear_function.c

```
/**
 * Returns f(x) = 3.0*x + 2.0
 */
double evaluate(double x)
{
    // TODO add semicolon to end of next line
    double y = 3.0*x + 2.0
    return y;
}
```

Icons:

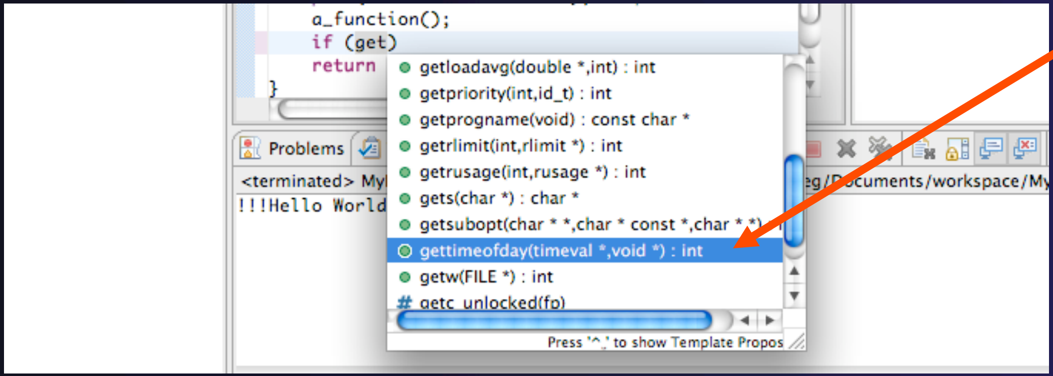


✓	Task tag
⚠	Warning
⊗	Error



Content Assist

- Type an incomplete function name e.g. ”get" into the editor, and hit **ctrl-space**
- Select desired completion value with cursor or mouse



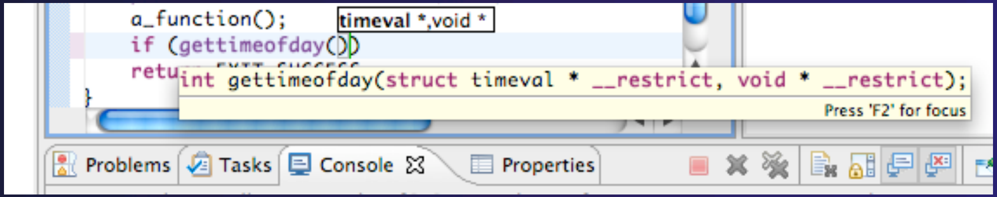
```
a_function();
if (get)
return
}
```

```
getloadavg(double *,int) : int
getpriority(int,id_t) : int
getprogname(void) : const char *
getrlimit(int,rlimit *) : int
getrusage(int,rusage *) : int
gets(char *) : char *
getsubopt(char * *,char * const *,char * *)
gettimeofday(timeval *,void *) : int
getw(FILE *) : int
getc_unlocked(fp)
```

Press '^,' to show Template Propos

```
<terminated> My
!!!Hello World
eg/Documents/workspace/My
```

- Hover over a program element in the source file to see additional information



```
a_function();
if (gettimeofday())
retu
}
```

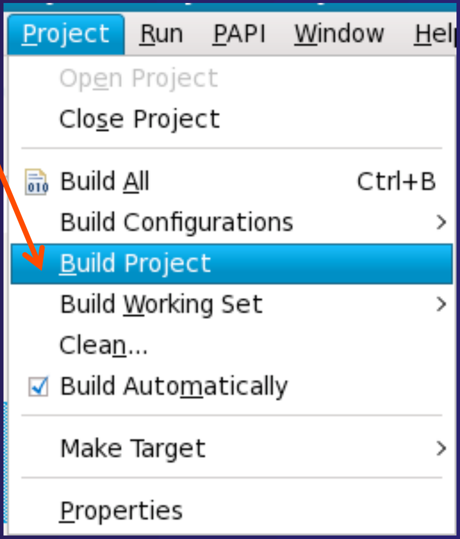
timeval *,void *

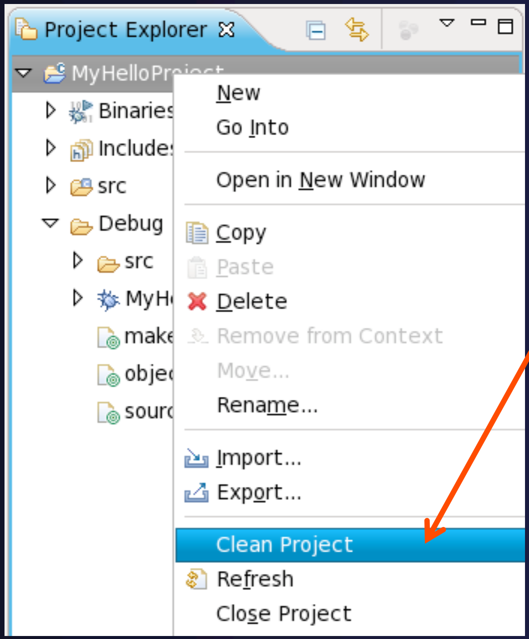
```
int gettimeofday(struct timeval * __restrict, void * __restrict);
```

Press 'F2' for focus

Build

- Your program should build when created.
- To rebuild, many ways include:
  - Select project, Hit hammer icon in toolbar
  - Select project, Project ▶ Build Project
  - Right mouse on project, Clean Project

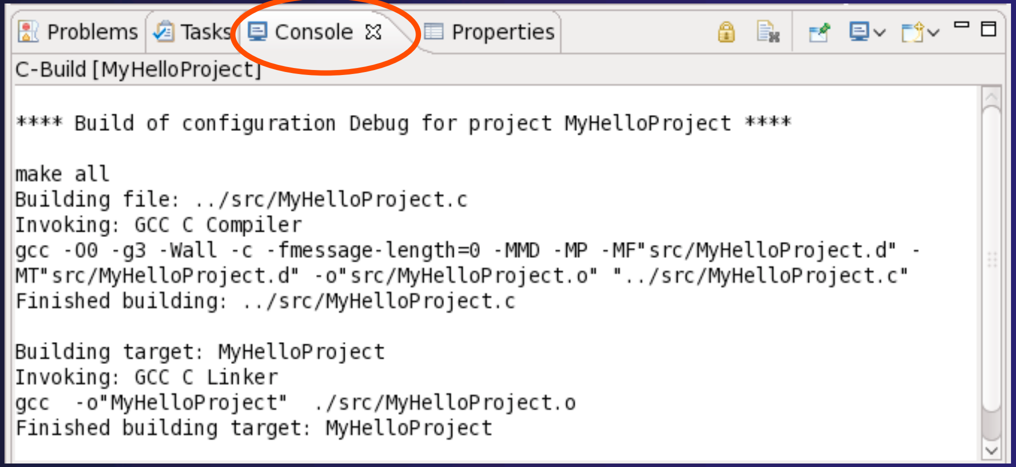




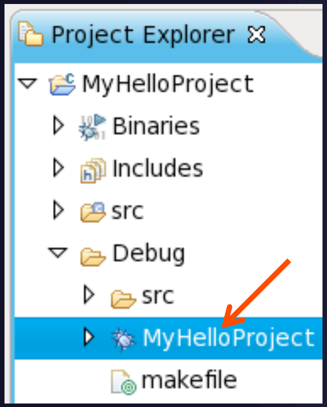
Next: see build output

Build (2)

- See the results of the build in the Console View
- Executable should be in Debug folder:



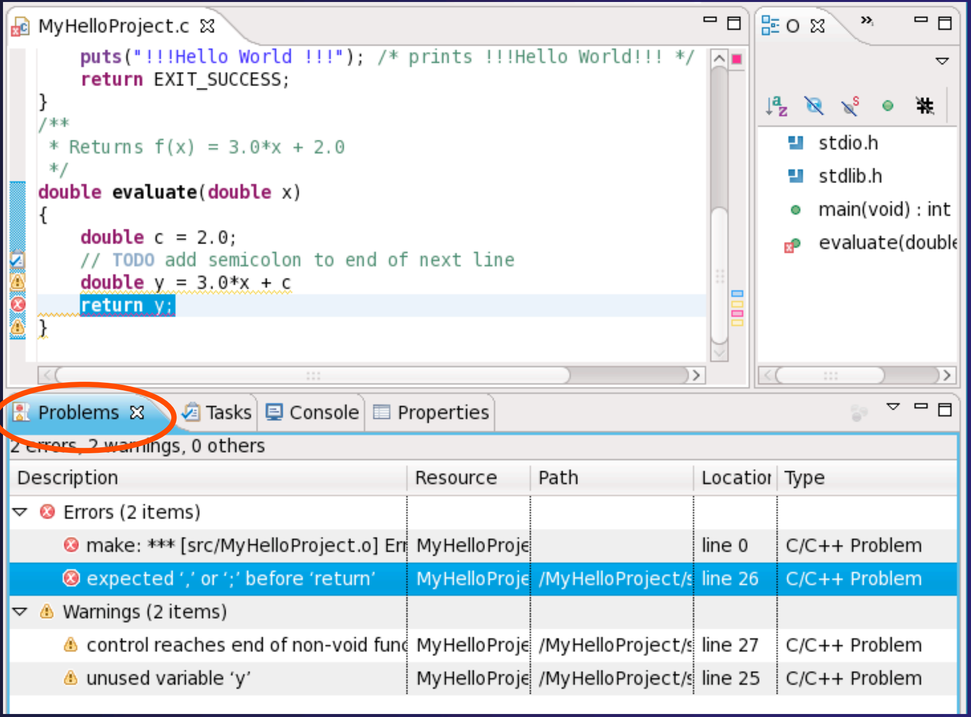
```
**** Build of configuration Debug for project MyHelloProject ****

make all
Building file: ../src/MyHelloProject.c
Invoking: GCC C Compiler
gcc -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"src/MyHelloProject.d" -
MT"src/MyHelloProject.d" -o"src/MyHelloProject.o" "../src/MyHelloProject.c"
Finished building: ../src/MyHelloProject.c

Building target: MyHelloProject
Invoking: GCC C Linker
gcc  -o"MyHelloProject"  ./src/MyHelloProject.o
Finished building target: MyHelloProject
```



Build problems?

- If there are problems, see:
- Marker on editor line
- **Problems view**

- Double-click on line in **Problems** view to go to location of error



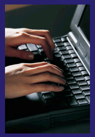
MyHelloProject.c

```
    puts("!!!Hello World !!!"); /* prints !!!Hello World!!! */
    return EXIT_SUCCESS;
}
/**
 * Returns f(x) = 3.0*x + 2.0
 */
double evaluate(double x)
{
    double c = 2.0;
    // TODO add semicolon to end of next line
    double y = 3.0*x + c
    return y;
}
```

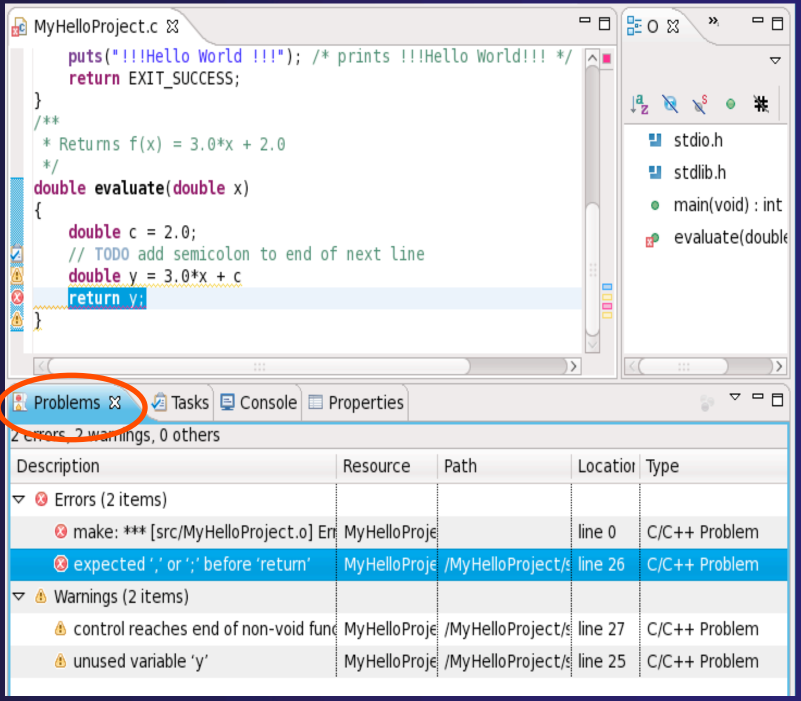
2 errors, 2 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (2 items)				
make: *** [src/MyHelloProject.o] Err	MyHelloProje		line 0	C/C++ Problem
expected ',' or ';' before 'return'	MyHelloProje	/MyHelloProject/s	line 26	C/C++ Problem
Warnings (2 items)				
control reaches end of non-void func	MyHelloProje	/MyHelloProject/s	line 27	C/C++ Problem
unused variable 'y'	MyHelloProje	/MyHelloProject/s	line 25	C/C++ Problem

Build problems? Try it



- Remove a semicolon from a line in your "Hello World" example
- Save file
- Rebuild
- **See the Problems view**
- Double-click on line in **Problems** view to go to location of error
- Fix it and rebuild to continue



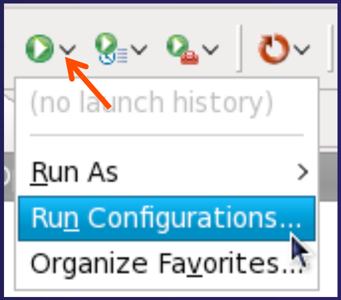
```
    puts("!!!Hello World !!!"); /* prints !!!Hello World!!! */
    return EXIT_SUCCESS;
}
/**
 * Returns f(x) = 3.0*x + 2.0
 */
double evaluate(double x)
{
    double c = 2.0;
    // TODO add semicolon to end of next line
    double y = 3.0*x + c
    return y;
}
```

2 errors, 2 warnings, 0 others

Description	Resource	Path	Locatioı	Type
Errors (2 items)				
make: *** [src/MyHelloProject.o] Err	MyHelloProje		line 0	C/C++ Problem
expected ',' or ';' before 'return'	MyHelloProje	/MyHelloProject/s	line 26	C/C++ Problem
Warnings (2 items)				
control reaches end of non-void func	MyHelloProje	/MyHelloProject/s	line 27	C/C++ Problem
unused variable 'y'	MyHelloProje	/MyHelloProject/s	line 25	C/C++ Problem

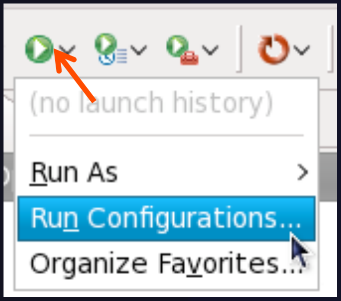
Run

- To run your C program,
- Create a *launch configuration* (see next slide)
- This saves the run/launching information and can be used to quickly run your program each time, with and without debug.

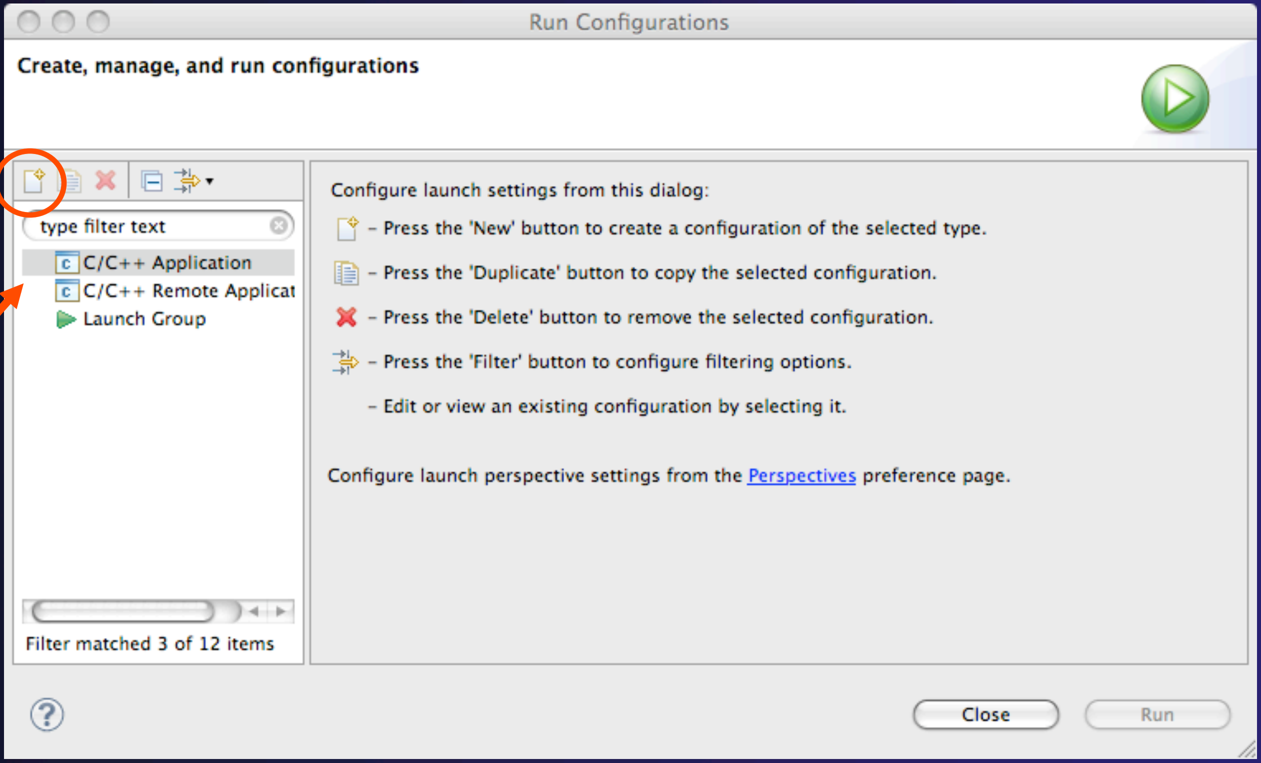


Create a Launch Configuration

a.k.a. Run Configuration



- Open the run configuration dialog **Run ▶ Run Configurations…**
- Select **C/C++ Application**
- Select the **New** button



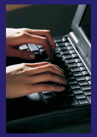
Create, manage, and run configurations

Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

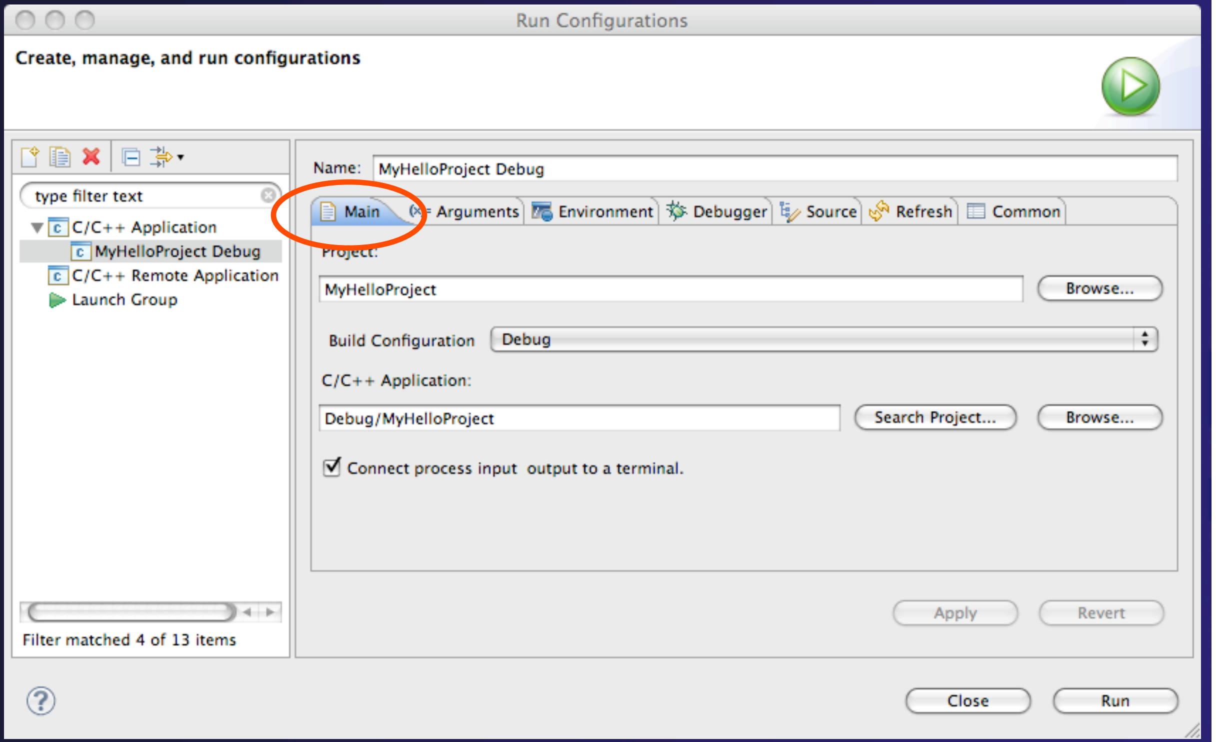
Configure launch perspective settings from the Perspectives preference page.

Depending on which flavor of Eclipse you installed, you might have more choices in Application types.



Complete the Main Tab

- Ensure that the correct project is selected
- Select the **C/C++ Application** (executable) if necessary
  - **Search Project…** will search just within the project
  - **Browse** will search anywhere on the local file system
- Select **Connect process input/output to a terminal** if desired



Run Configurations

Create, manage, and run configurations

Name: MyHelloProject Debug

Project: MyHelloProject

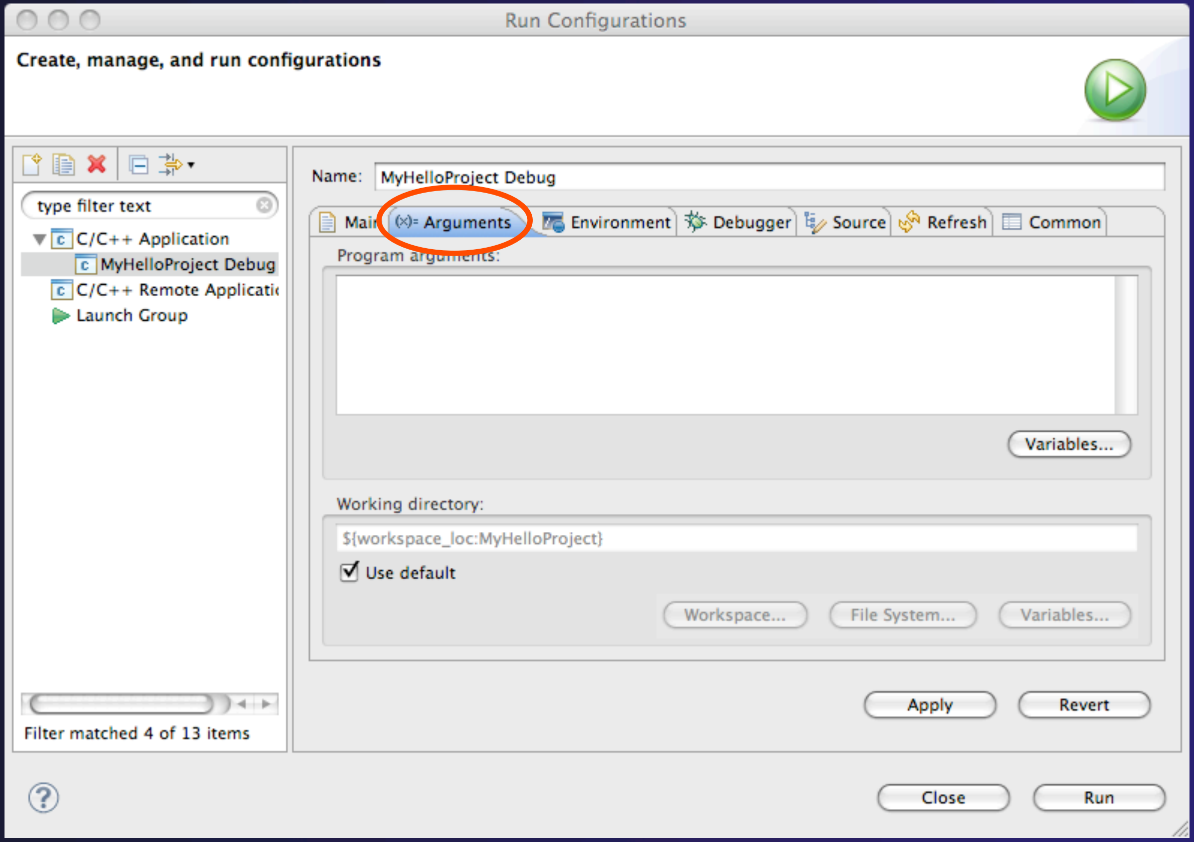
Build Configuration: Debug

C/C++ Application: Debug/MyHelloProject

☑ Connect process input output to a terminal.

Complete the Arguments Tab

- Enter any program arguments into the text box
- Eclipse variables can also be passed using the **Variables…** button
- Select a different working directory if desired

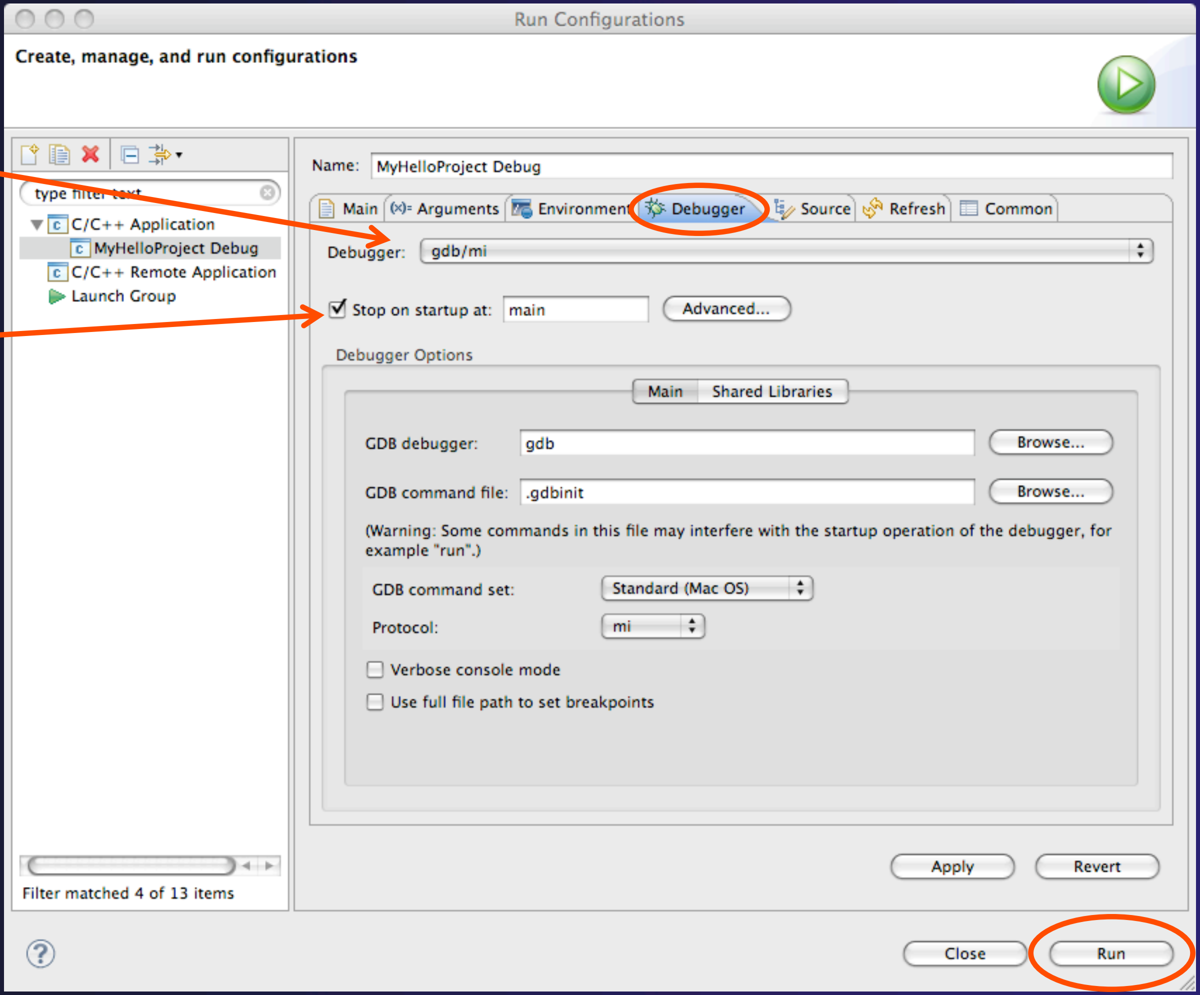


Complete the Debugger Tab

- Select **Debugger** tab
- Make sure **gdb/mi** is selected
- Change where the program should stop if desired
- Change any gdb-specific options if desired (advanced users only)

The information on the debugger tab will only be used for a debug launch

- Hit the Run button to launch your program


Run Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
  - MyHelloProject Debug
- C/C++ Remote Application
- Launch Group

Filter matched 4 of 13 items

Name: MyHelloProject Debug

Main | Arguments | Environment | Debugger | Source | Refresh | Common

Debugger: gdb/mi

☑ Stop on startup at: main [Advanced...]

Debugger Options

Main | Shared Libraries

GDB debugger: gdb [Browse...]

GDB command file: .gdbinit [Browse...]

(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)

GDB command set: Standard (Mac OS)

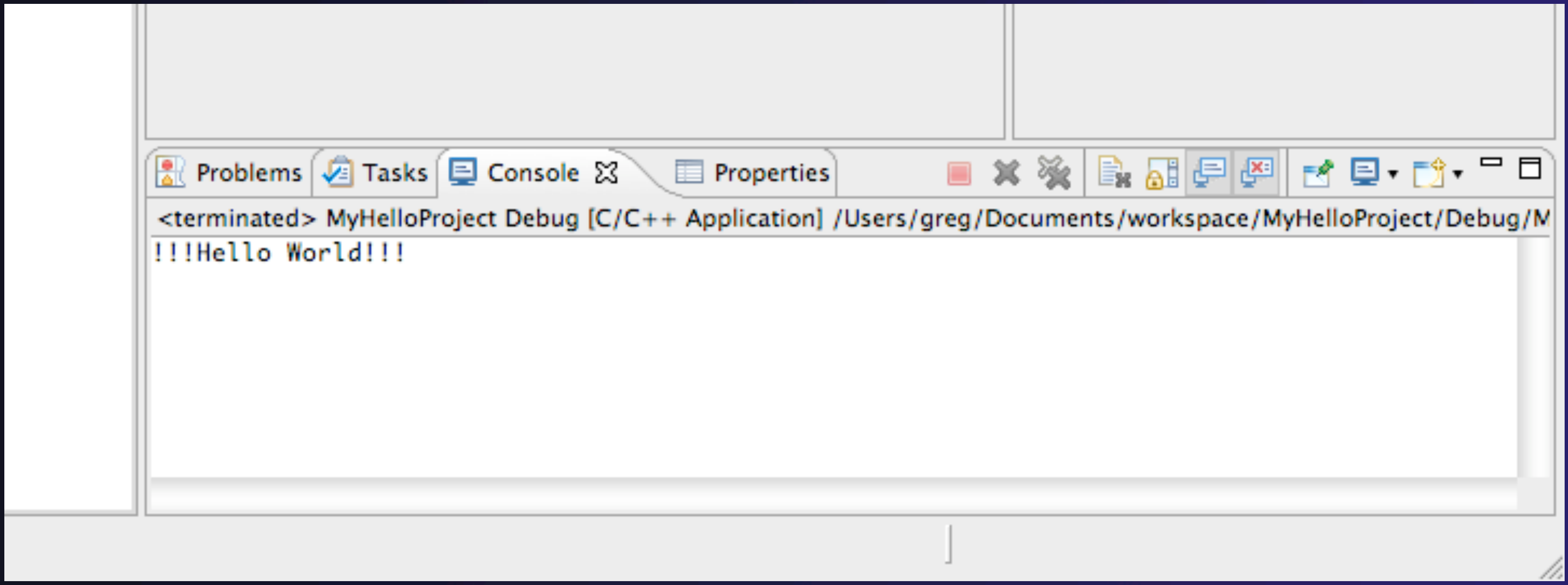
Protocol: mi

☐ Verbose console mode

☐ Use full file path to set breakpoints

Viewing Program Output

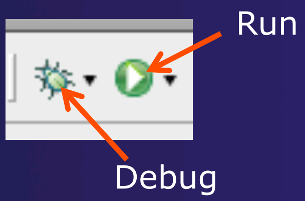
- When the program runs, the **Console** view should automatically become active
- Any output will be displayed in this view (stderr in red)

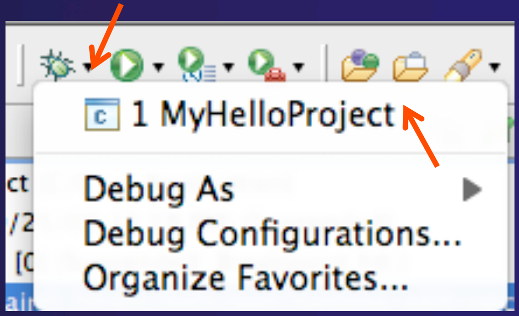


```
<terminated> MyHelloProject Debug [C/C++ Application] /Users/greg/Documents/workspace/MyHelloProject/Debug/M
!!!Hello World!!!
```

Debug your code

- Launch with same config used for Run
- If asked, you can set:
  - Preferred Launcher: Standard
    - Use Config-specific or change Workspace setting
  - Debugger: gdb/mi
- Eclipse asks to switch to Debug Perspective
- Select **Yes** to continue

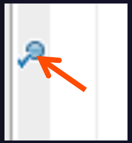




We'll cover debugging in much more detail when we cover parallel debugging

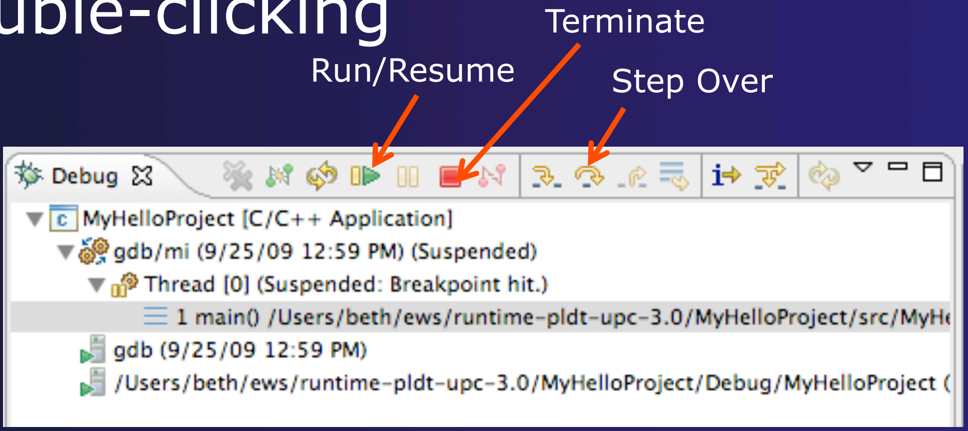
Debug your code (2)

- Upon launch, Eclipse switches to Debug Perspective
- Program stops at main
- Set Breakpoint by double-clicking in editor left margin



Breakpoint Marker

- Step with F5 or
- Run with F8 or
- Hit Breakpoint; inspect variables; inspect stack
- End with Terminate, or run to end of Prog

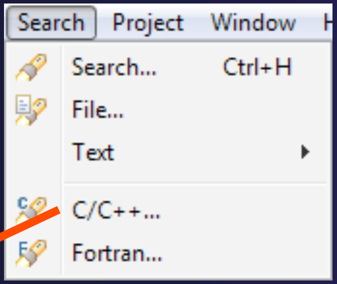


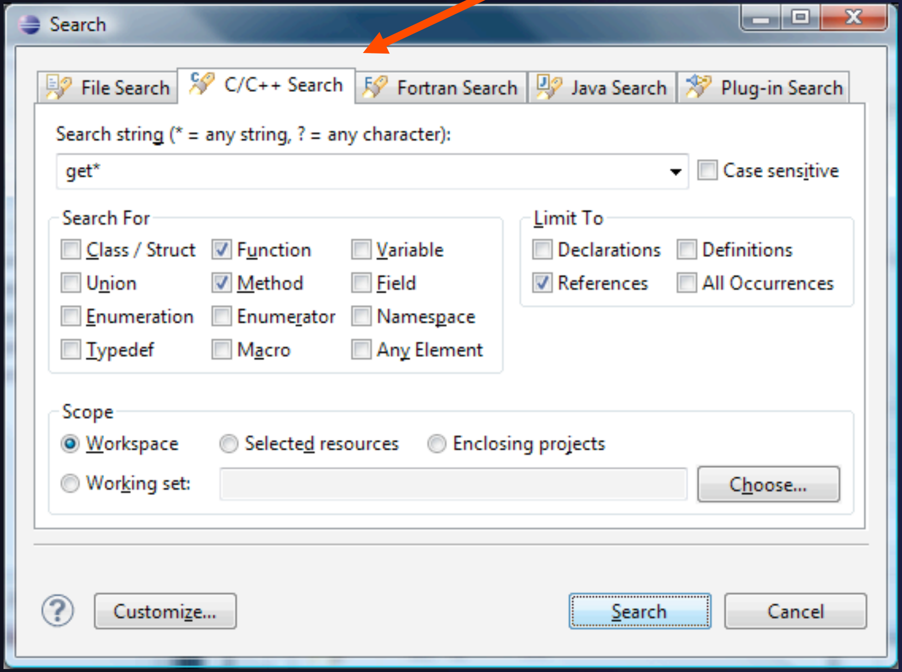
MyHelloProject [C/C++ Application]
gdb/mi (9/25/09 12:59 PM) (Suspended)
Thread [0] (Suspended: Breakpoint hit.)
1 main() /Users/beth/ews/runtime-pldt-upc-3.0/MyHelloProject/src/MyHe
gdb (9/25/09 12:59 PM)
/Users/beth/ews/runtime-pldt-upc-3.0/MyHelloProject/Debug/MyHelloProject (

Other CDT features

- Searching
- Open Declaration / hyperlinking between files in the editor
- Rename in file (in-place in editor)
- Refactoring
  - Rename refactoring / Preview panes
  - Extract Constant refactoring
  - Other refactorings in CDT

Language-Based Searching





Search string (* = any string, ? = any character): get*

[ ] Case sensitive

Search For

[ ] Class / Struct [✓] Function [ ] Variable
[ ] Union [✓] Method [ ] Field
[ ] Enumeration [ ] Enumerator [ ] Namespace
[ ] Typedef [ ] Macro [ ] Any Element

Limit To

[ ] Declarations [ ] Definitions
[✓] References [ ] All Occurrences

Scope

[● Workspace] ○ Selected resources ○ Enclosing projects
○ Working set:

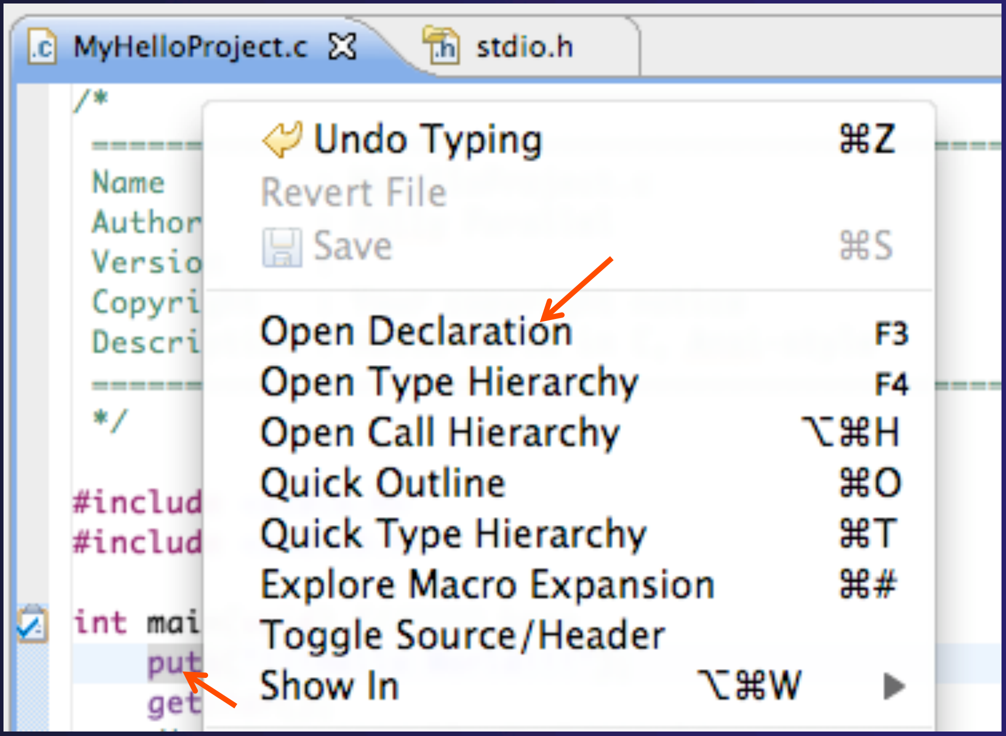
- "Knows" what things can be declared in each language (functions, variables, classes, modules, etc.)
- E.g., search for every call to a function whose name starts with "get"
- Search can be project- or workspace-wide

Open Declaration

- Jumps to the declaration of a variable, function, etc., even if it's in a different file

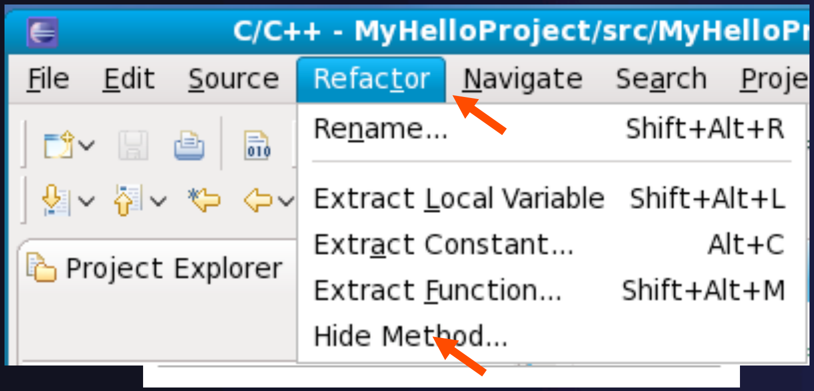
- Right-click on an identifier
- Click **Open Declaration**

- Can also Ctrl-click (Mac: Cmd-click) to "hyperlink" to declaration



Rename Refactoring

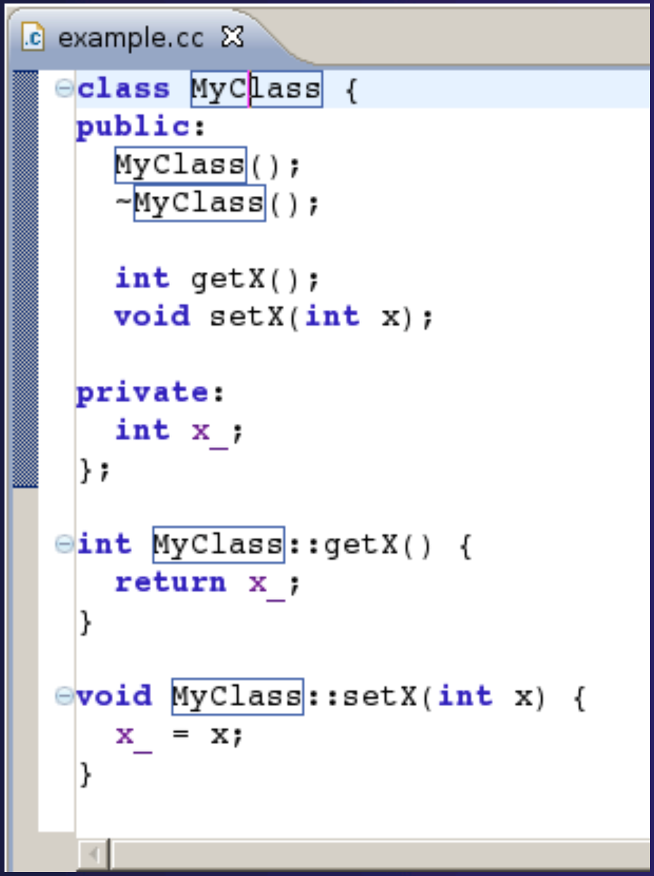
- Changes the name of a variable, function, etc., *including every use*
  (change is semantic, not textual, and can be workspace-wide)
- Only proceeds if the new name will be legal
  (aware of scoping rules, namespaces, etc.)


C/C++ - MyHelloProject/src/MyHelloPr
File
Edit
Source
Refactor
Navigate
Search
Proje
Rename... Shift+Alt+R
Extract Local Variable Shift+Alt+L
Extract Constant... Alt+C
Extract Function... Shift+Alt+M
Hide Method...
Project Explorer

- Select **C/C++ Perspective**
- Open a source file
- Click in editor view on declaration of a variable
- Select menu item **Refactor▸Rename**
  - Or use context menu
- Enter new name

CDT Rename in File

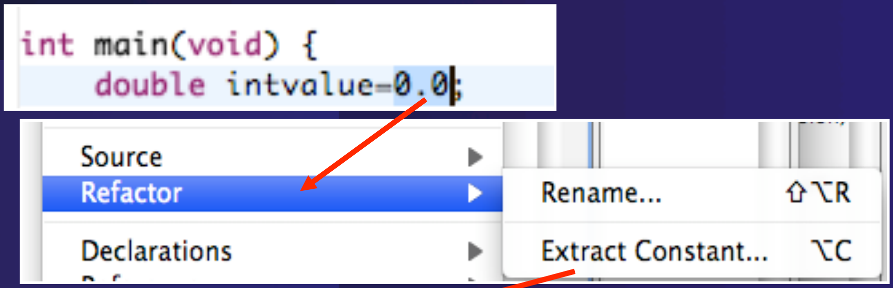
- Position the caret over an identifier.
- Press Ctrl+1 (Command+1 on Mac).
- Enter a new name. Changes are propagated within the file as you type.



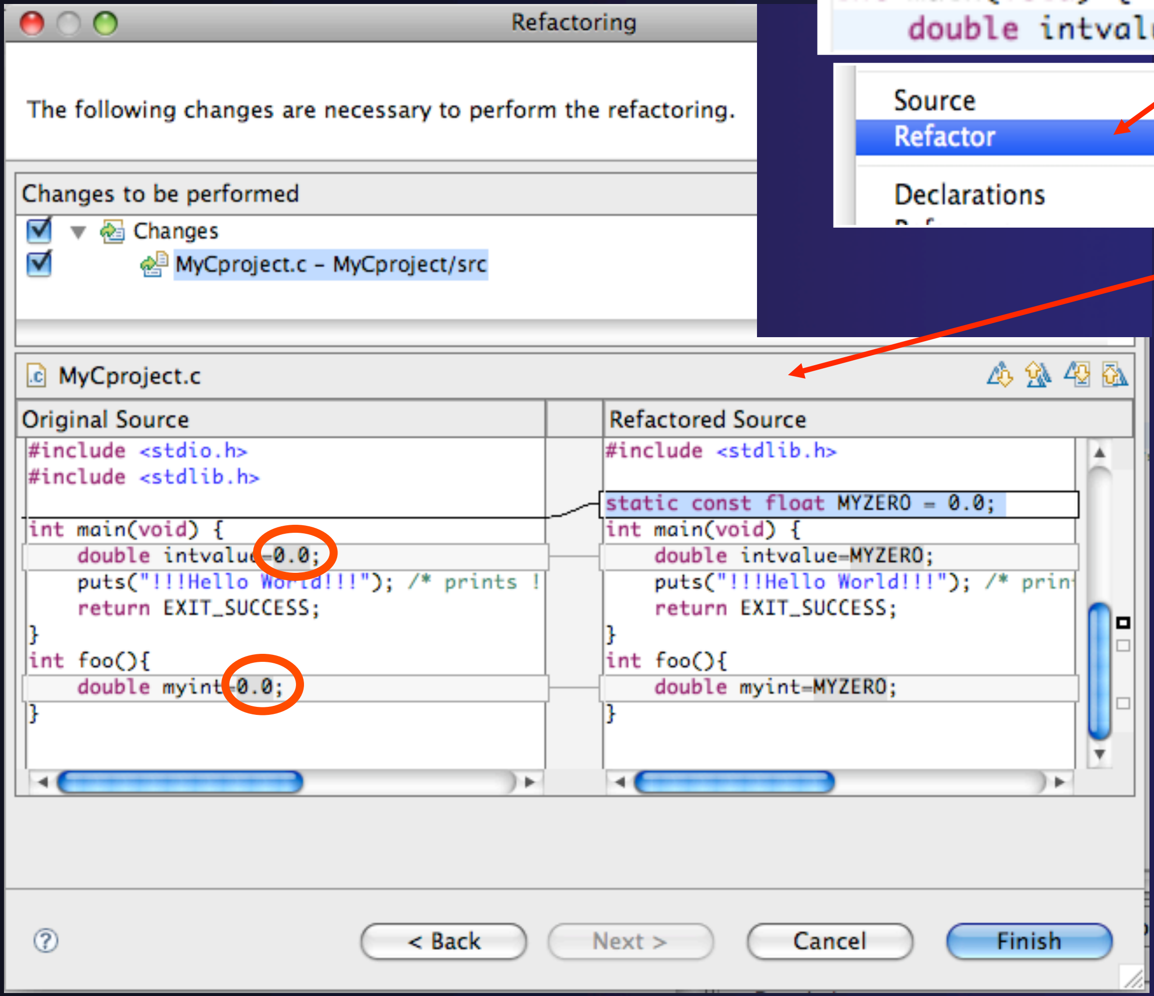
example.cc

```
class MyClass {
public:
   MyClass();
   ~MyClass();

   int getX();
   void setX(int x);

private:
   int x_;
};

int MyClass::getX() {
   return x_;
}

void MyClass::setX(int x) {
   x_ = x;
}
```

CDT Extract Constant Refactoring



```
int main(void) {
    double intvalue=0.0;
```



Refactoring

The following changes are necessary to perform the refactoring.

Changes to be performed

☑ Changes
☑ MyCproject.c – MyCproject/src

MyCproject.c

Original Source

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

int main(void) {
    double intvalue=0.0;
    puts("!!!Hello World!!!"); /* prints !
    return EXIT_SUCCESS;
}
int foo(){
    double myint=0.0;
}
```

Refactored Source

```
#include <stdlib.h>

static const float MYZERO = 0.0;
int main(void) {
    double intvalue=MYZERO;
    puts("!!!Hello World!!!"); /* prin
    return EXIT_SUCCESS;
}
int foo(){
    double myint=MYZERO;
}
```

✦ Other refactorings that are planned:

- Extract Function
- Hide Member Function
- Move Field or Member Function
- Extract Subclass
- Extract Baseclass
- Separate Class
- Implement Function
- Declare Function
- Move Function Definition
- Generate Getters and Setters

Module 4: Working with MPI

- Objective
  - Learn how to use a source code repository (CVS)
  - Learn how to develop, build and launch an MPI program on a remote parallel machine
- Contents
  - Using a version control system (CVS)
  - Remote project setup
  - Building with Makefiles and autoconf
  - MPI assistance features
  - Working with resource managers
  - Launching a parallel application

Userids for PTP Tutorial

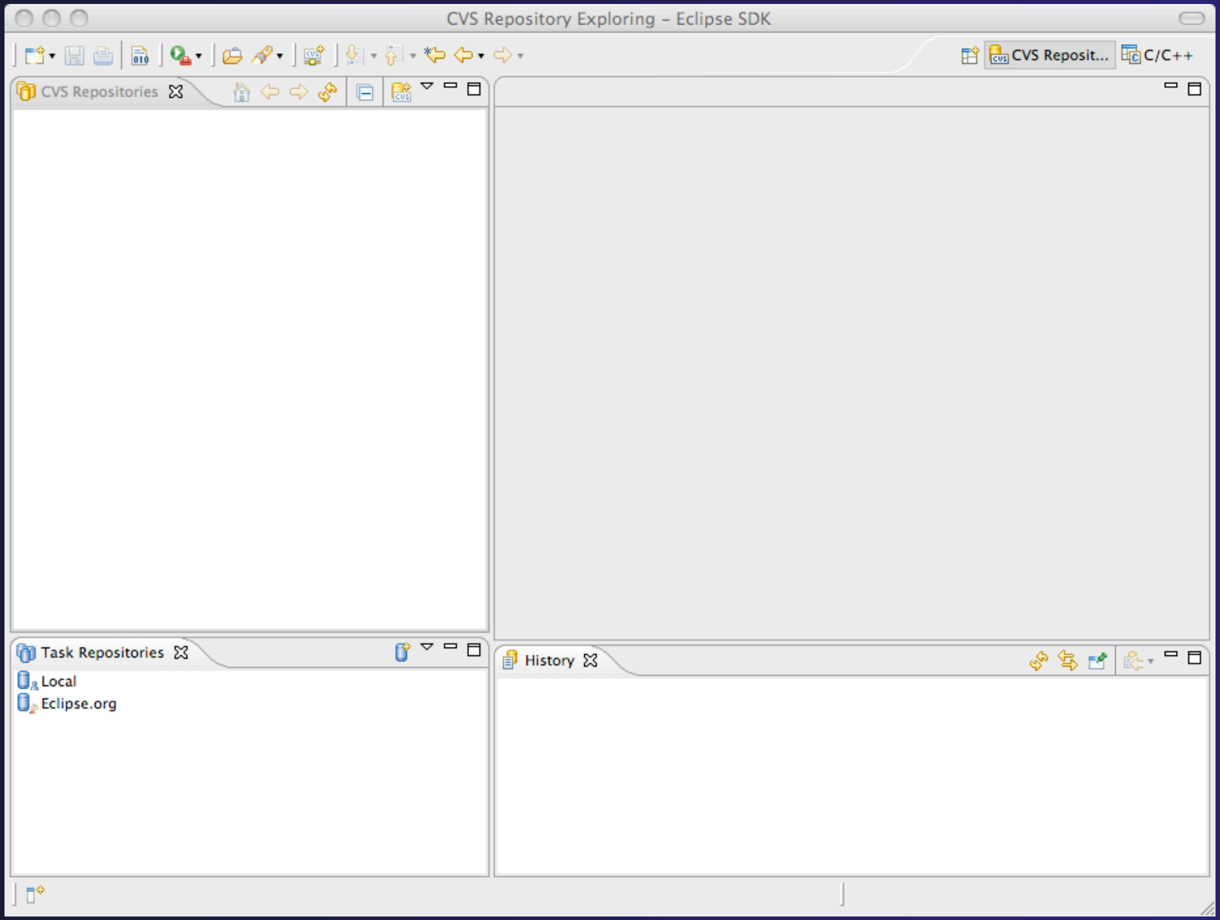
- The hands on portion of this module will be done on a remote system (abe.ncsa.uiuc.edu) at NCSA
- http://www.ncsa.illinois.edu/UserInfo/Resources/Hardware/Intel64Cluster/
- Login information provided at the beginning of the tutorial

Creating the Project

- Configuring version control
- Checking out the source code
- Team support

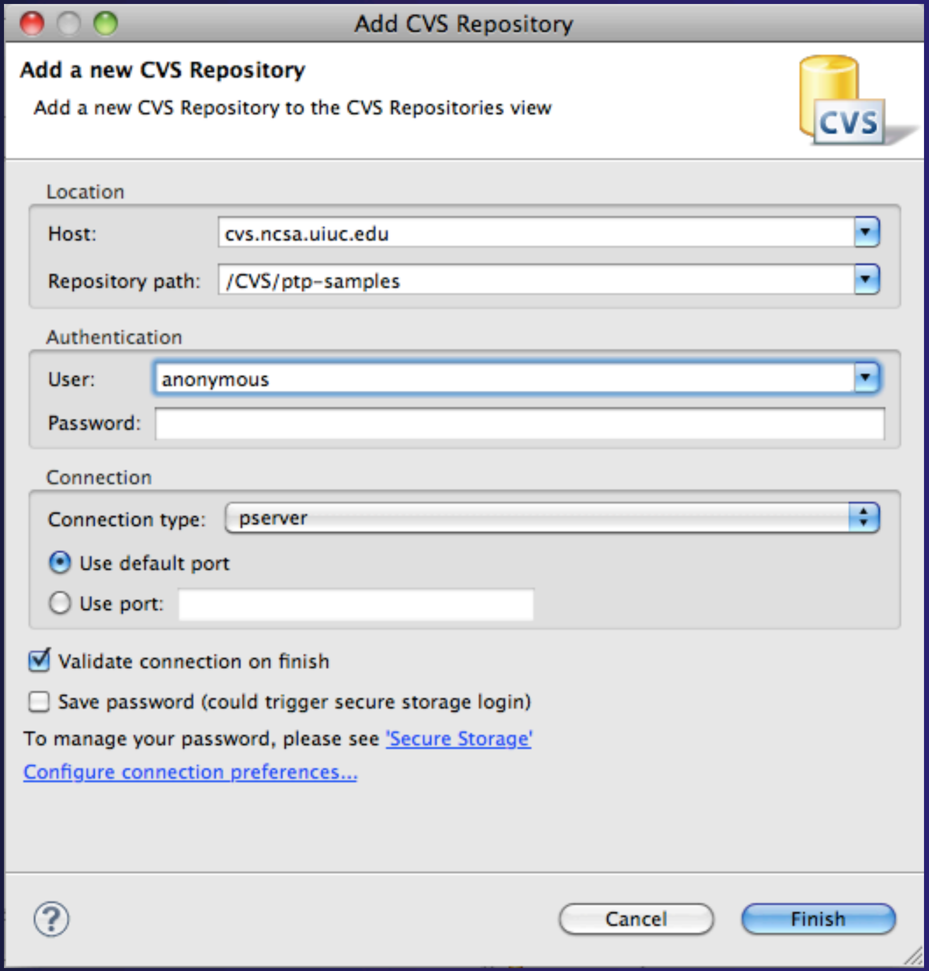
Connecting to a Repository

- Select **Window▶Open Perspective▶Other…**
- Select **CVS Repository Exploring** then **OK**



Specify Repository Location

- Right-click in the **CVS Repositories** view, then select **New▶Repository Location…**
- Set **Host** to the hostname of remote machine
- Set **Repository path** to the CVS repository path
- Fill in **Username** and **Password**
- Set **Connection type** to **extssh** to use an ssh connection
  - For anonymous access, use pserver connection type
- Check **Save password** if you wish to save the password
- Select **Finish**


Add CVS Repository

Add a new CVS Repository

Add a new CVS Repository to the CVS Repositories view

Location

Host: cvs.ncsa.uiuc.edu

Repository path: /CVS/ptp-samples

Authentication

User: anonymous

Password:

Connection

Connection type: pserver

[✓ Use default port]

Use port:

[✓] Validate connection on finish

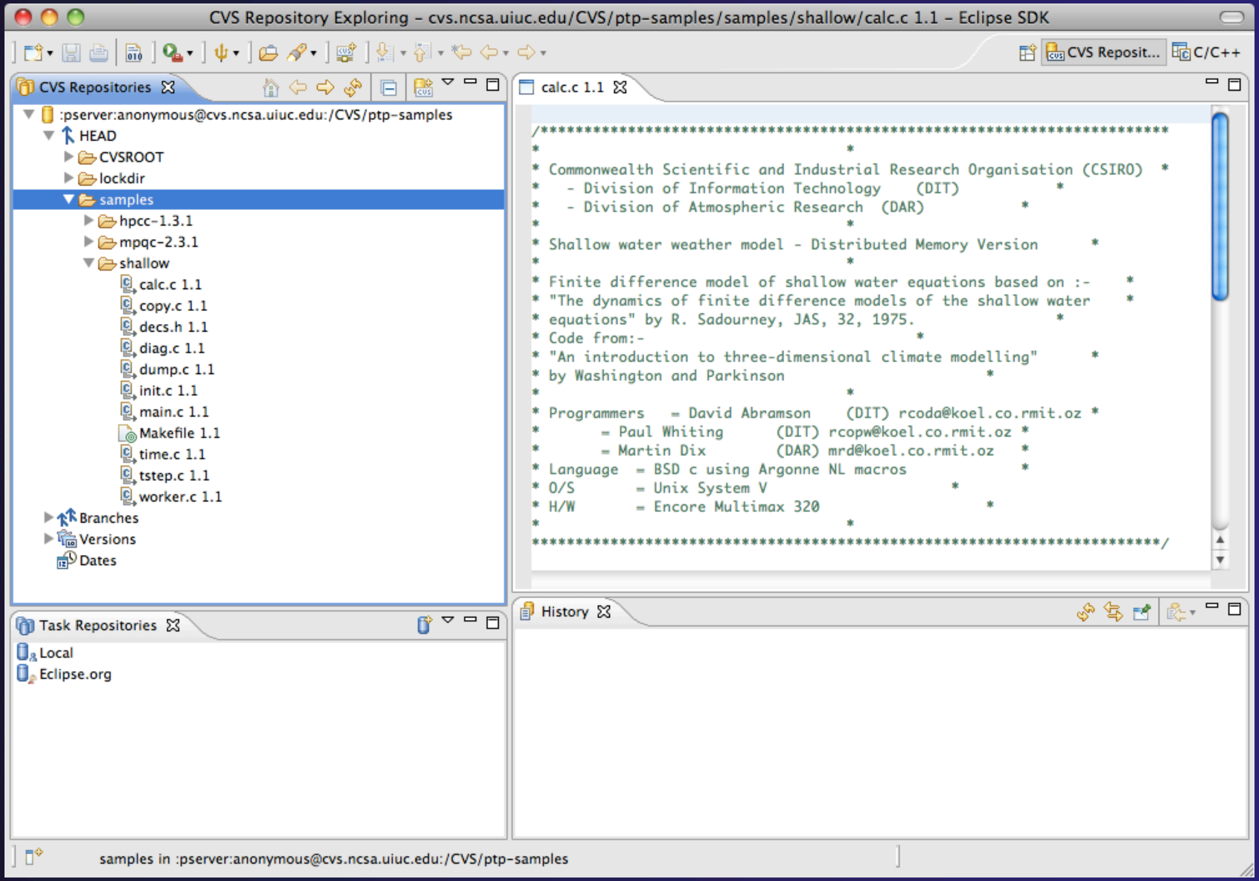
[ ] Save password (could trigger secure storage login)

To manage your password, please see 'Secure Storage'

Configure connection preferences...

CVS Repository Exploring

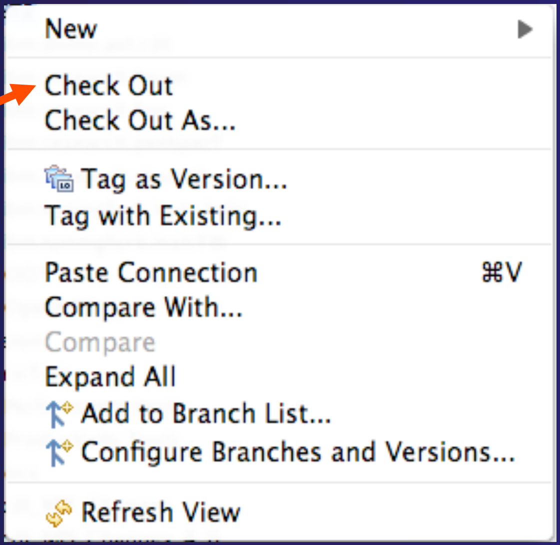
- Open the repository in the **CVS Repository** view
- Open **HEAD** to view files and folders in the CVS head
- Open **Branches** or **Versions** to view CVS branches or versions respectively
- Right-click on the repository and select **Refresh Branches…** to see all branches and versions



```
/***********************************************************************
*                                    *
* Commonwealth Scientific and Industrial Research Organisation (CSIRO)  *
*   - Division of Information Technology    (DIT)           *
*   - Division of Atmospheric Research  (DAR)         *
*                                    *
* Shallow water weather model - Distributed Memory Version      *
*                                    *
* Finite difference model of shallow water equations based on :-    *
* "The dynamics of finite difference models of the shallow water    *
* equations" by R. Sadourney, JAS, 32, 1975.                *
* Code from:-                               *
* "An introduction to three-dimensional climate modelling"      *
* by Washington and Parkinson                     *
*                                    *
* Programmers   = David Abramson    (DIT) rcoda@koel.co.rmit.oz *
*       = Paul Whiting      (DIT) rcopw@koel.co.rmit.oz *
*       = Martin Dix        (DAR) mrd@koel.co.rmit.oz   *
* Language  = BSD c using Argonne NL macros           *
* O/S       = Unix System V                 *
* H/W       = Encore Multimax 320                 *
*                                    *
***********************************************************************/
```

Checking out code in Eclipse

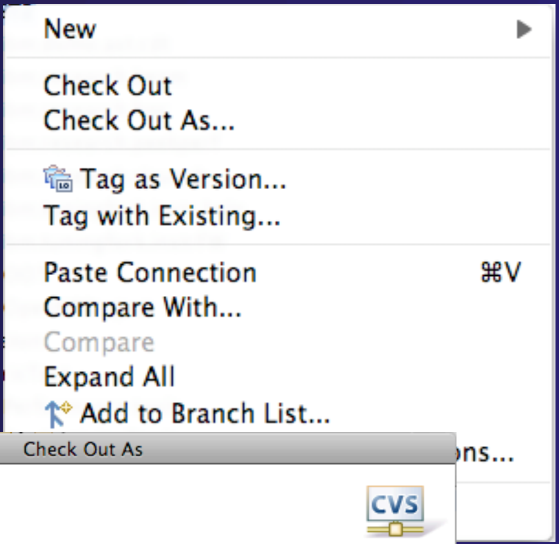
- If the project exists in the repository as an Eclipse Project, then one can simply "Check Out" the code
- In CVS Repositories view, right-click on project and select **Project▶Check Out**
- Our example doesn't have Eclipse Project information – this code was checked in with command line tools.
- Our next slide shows how to add Eclipse Project information automatically as you check out the code.

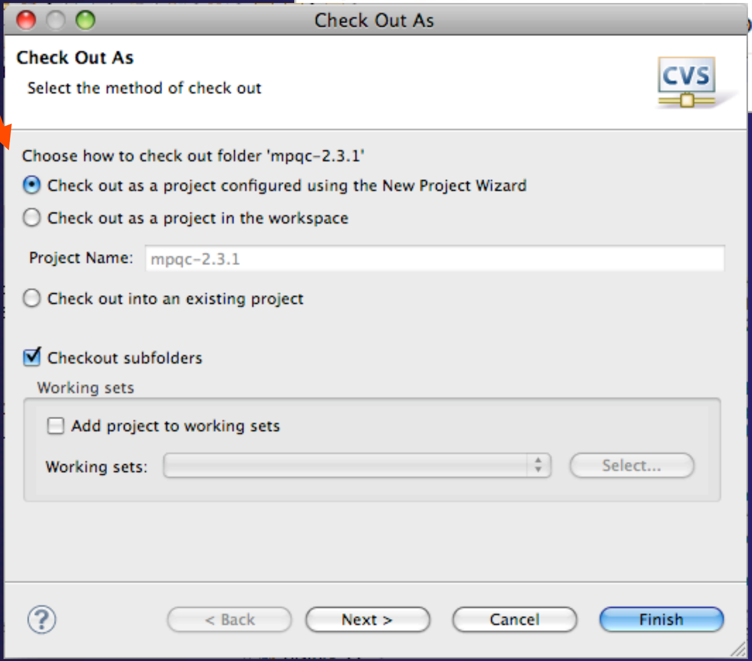


Check out as an Eclipse Project

- In CVS Repositories view, right-click on project and select **Project▶Check Out As…**
- Make sure **Check out as a project configured using the New Project Wizard** is selected
- Leave **Checkout subfolders** checked
- Select **Finish**

The wizard that runs next will add Eclipse information to the project.




Check Out As

Check Out As
Select the method of check out

Choose how to check out folder 'mpqc-2.3.1'

- [✓] Check out as a project configured using the New Project Wizard
- Check out as a project in the workspace

Project Name: mpqc-2.3.1

- Check out into an existing project

[✓] Checkout subfolders

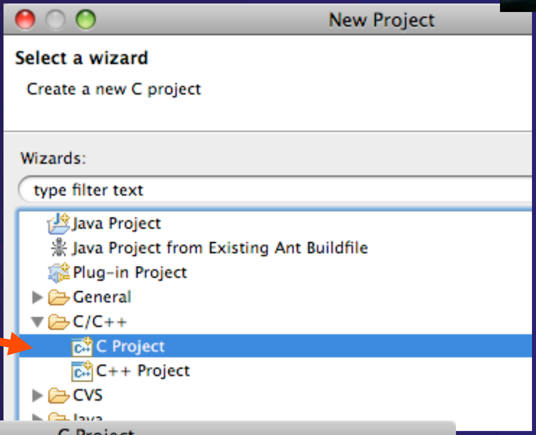
Working sets

[ ] Add project to working sets

Working sets:

New Project Wizard: Create a C Project

- The **New Project Wizard** is used to create a C project
- Enter **Project name**
- Under **Project Types**, select **Makefile project▶Empty Project**
  - Ensures that CDT will use existing makefiles
- Select **Finish**
- When prompted to switch to the **C/C++ Perspective**, select **Yes**


New Project

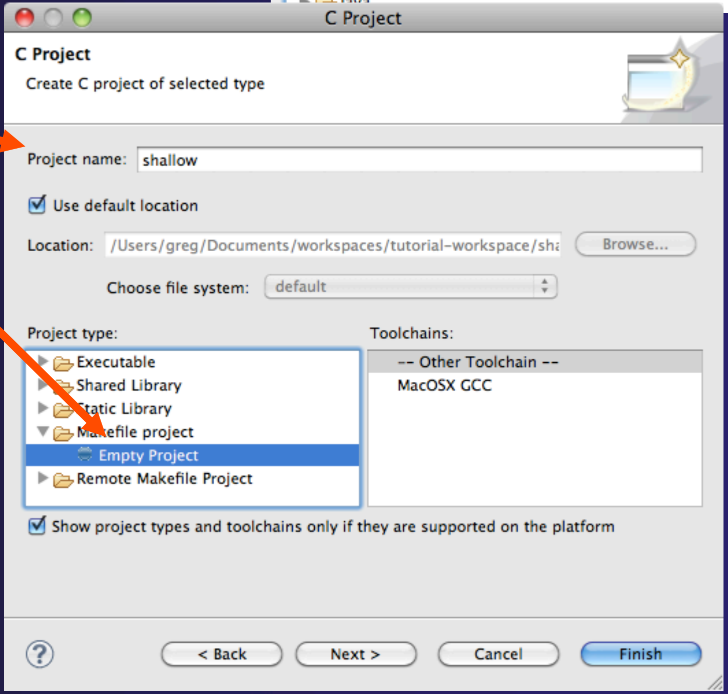
Select a wizard

Create a new C project

Wizards:

type filter text

- Java Project
- Java Project from Existing Ant Buildfile
- Plug-in Project
- General
- C/C++
  - C Project
  - C++ Project
- CVS


C Project

C Project

Create C project of selected type

Project name: shallow

☑ Use default location

Location: /Users/greg/Documents/workspaces/tutorial-workspace/sha

Choose file system: default

Project type:

- Executable
- Shared Library
- Static Library
- Makefile project
  - Empty Project
- Remote Makefile Project

Toolchains:

- -- Other Toolchain --
- MacOSX GCC

☑ Show project types and toolchains only if they are supported on the platform

MPI Assistance Tools

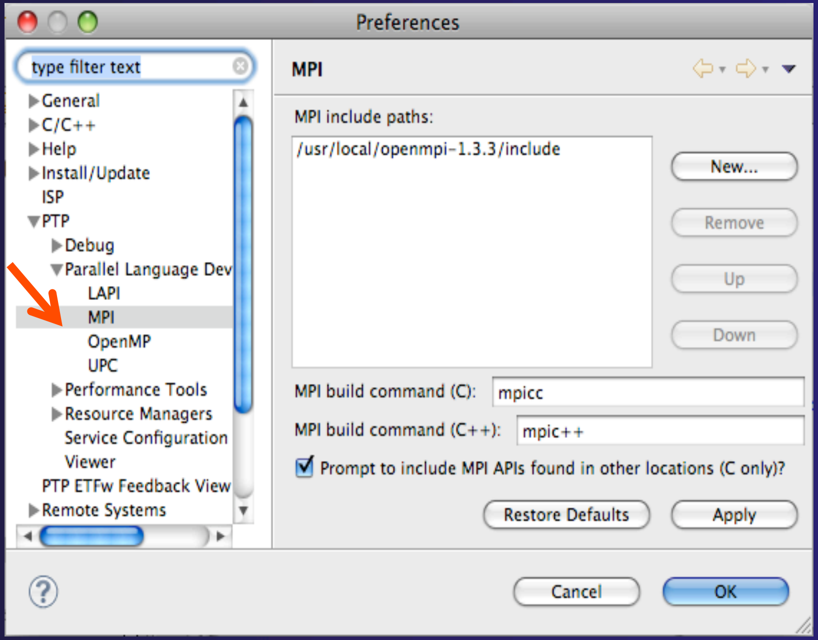
Added by PLDT (Parallel Lang. Dev. Tools) feature of PTP

- MPI Context sensitive help
- MPI artifact locations
- MPI barrier analysis
- MPI templates

- For this part, use the local project that you created from CVS.

Set MPI Preferences

- To run MPI analysis, you first need to tell PLDT *what* an MPI API is, by locating the MPI include files.
- Open Preferences
  Window > Preferences
  Mac: Eclipse > Preferences
- On the MPI Preferences page, add a new MPI include path:
- New … and point to the *directory* containing your MPI header file
- Select OK
- Back on New Project Wizard page, select **Finish.**



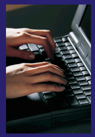
MPI

MPI include paths:
/usr/local/openmpi-1.3.3/include

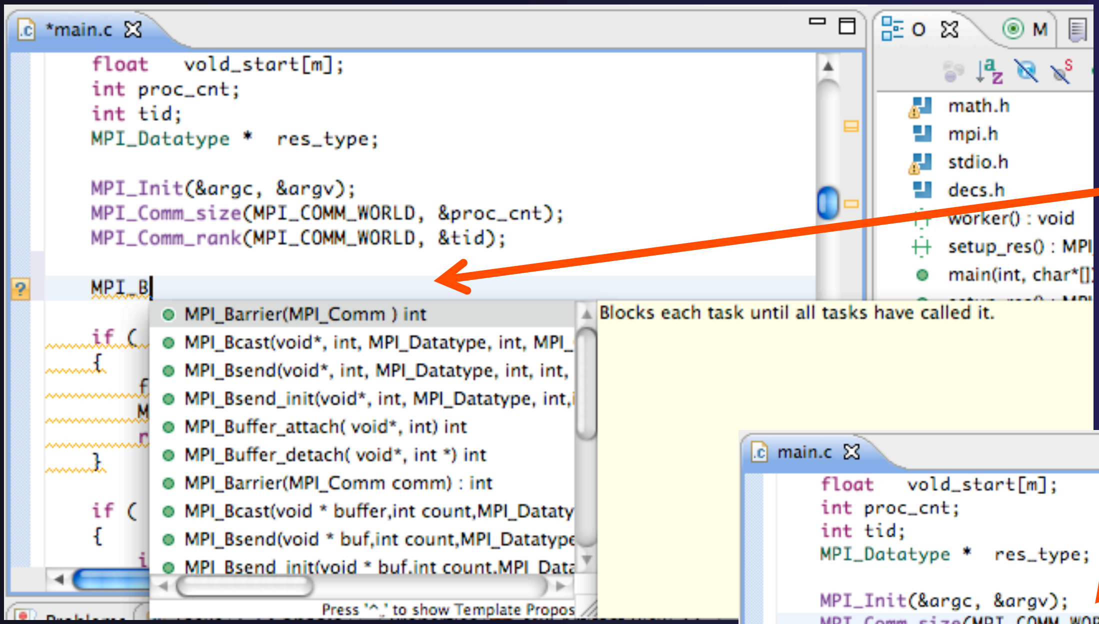
MPI build command (C): mpicc

MPI build command (C++): mpic++

☑ Prompt to include MPI APIs found in other locations (C only)?



Edit aids to MPI development

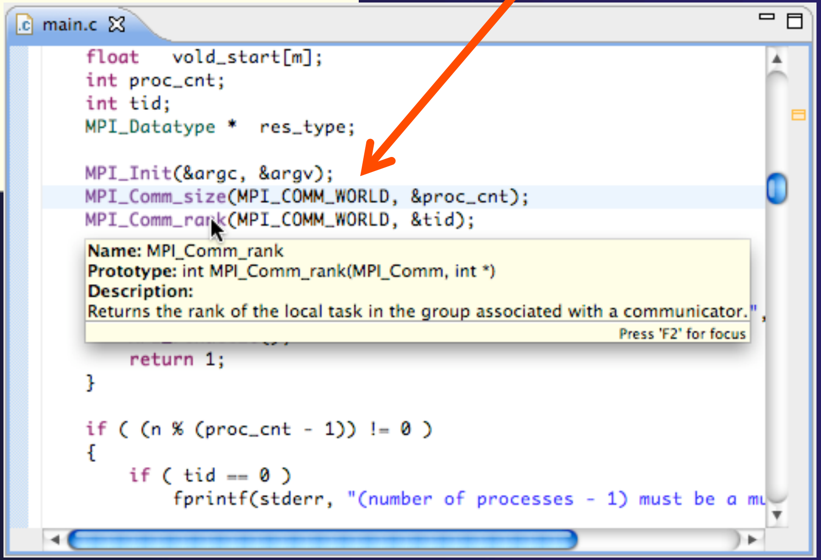


```
    float   vold_start[m];
    int proc_cnt;
    int tid;
    MPI_Datatype *  res_type;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &proc_cnt);
    MPI_Comm_rank(MPI_COMM_WORLD, &tid);

    MPI_B
```

```
MPI_Barrier(MPI_Comm ) int
MPI_Bcast(void*, int, MPI_Datatype, int, MPI_
MPI_Bsend(void*, int, MPI_Datatype, int, int,
MPI_Bsend_init(void*, int, MPI_Datatype, int,i
MPI_Buffer_attach( void*, int) int
MPI_Buffer_detach( void*, int *) int
MPI_Barrier(MPI_Comm comm) : int
MPI_Bcast(void * buffer,int count,MPI_Dataty
MPI_Bsend(void * buf,int count,MPI_Datatype
MPI_Bsend_init(void * buf,int count,MPI_Data
Press '^.' to show Template Propos
```

Blocks each task until all tasks have called it.

- Helpful editor features:
  - Code completion (Ctrl-space)
  - Hover over MPI API
  - Help (see next slide)



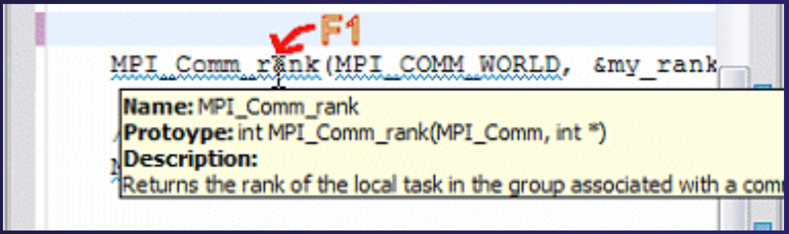
```
    float   vold_start[m];
    int proc_cnt;
    int tid;
    MPI_Datatype *  res_type;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &proc_cnt);
    MPI_Comm_rank(MPI_COMM_WORLD, &tid);
```

Name: MPI_Comm_rank
Prototype: int MPI_Comm_rank(MPI_Comm, int *)
Description:
Returns the rank of the local task in the group associated with a communicator.
Press 'F2' for focus

```
        return 1;
    }

    if ( (n % (proc_cnt - 1)) != 0 )
    {
        if ( tid == 0 )
            fprintf(stderr, "(number of processes - 1) must be a mu
```

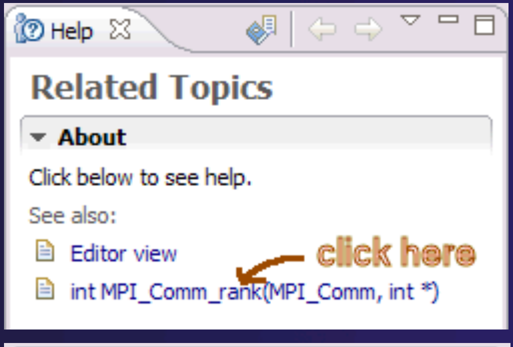
Context Sensitive Help

- Click mouse, then press help key when the cursor is within a function name
  - Windows: **F1** key
  - Linux: **ctrl-F1** key
  - MacOS X: **Help** key or **Help▸Dynamic Help**
- A help view appears (**Related Topics**) which shows additional information (You may need to click on MPI API in editor again, to populate)
- Click on the function name to see more information
- Move the help view within your Eclipse workbench, if you like, by dragging its title tab



```
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank
```

Name: MPI_Comm_rank
Protoype: int MPI_Comm_rank(MPI_Comm, int *)
Description:
Returns the rank of the local task in the group associated with a com



Related Topics

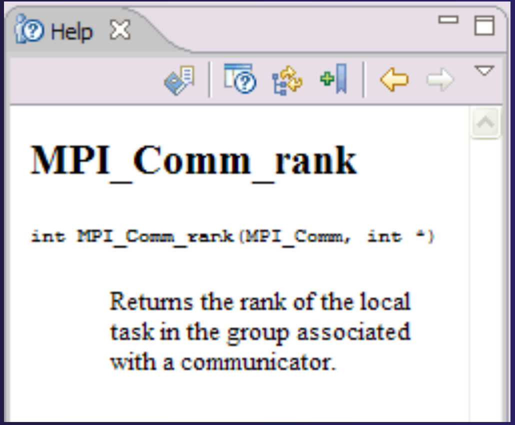
About

Click below to see help.

See also:

Editor view

int MPI_Comm_rank(MPI_Comm, int *)



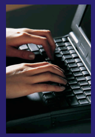
MPI_Comm_rank

```
int MPI_Comm_rank(MPI_Comm, int *)
```

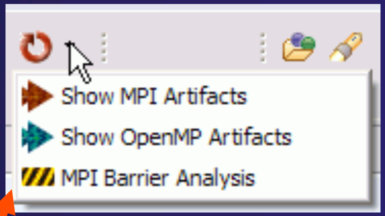
Returns the rank of the local task in the group associated with a communicator.

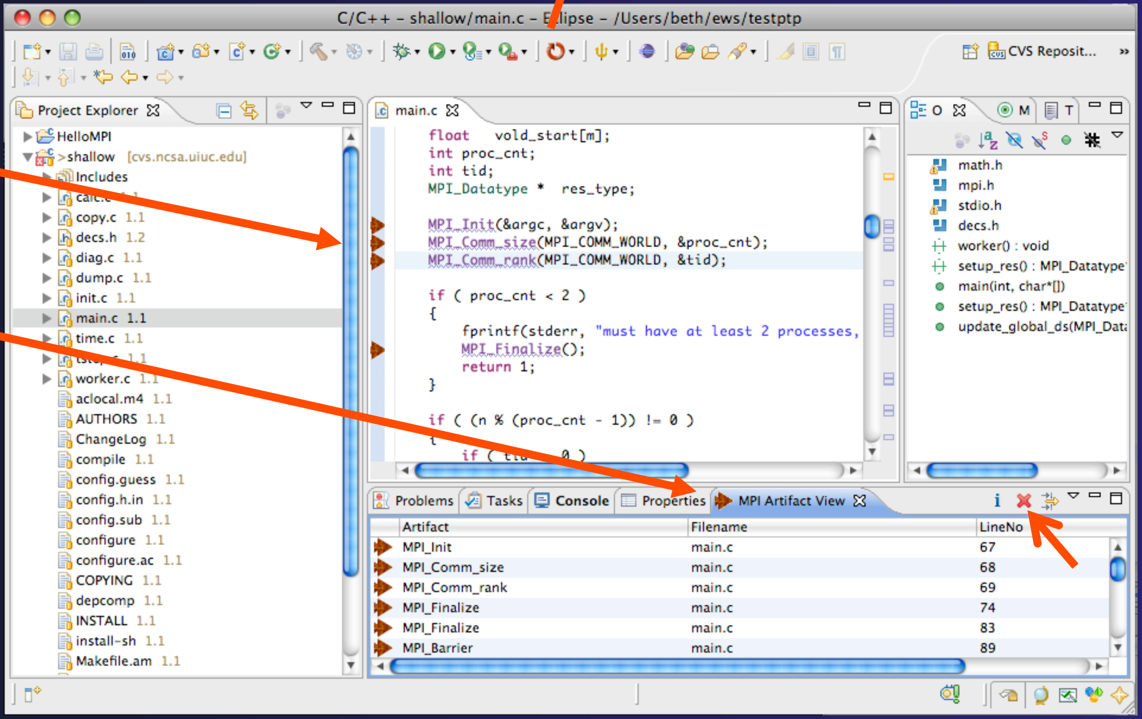
Some special info has been added for MPI APIs.

Show MPI Artifacts



- Select source file; Run analysis by clicking on drop-down menu next to the analysis button and selecting **Show MPI Artifacts**
- Markers indicate the location of artifacts in editor
- In **MPI Artifact View** sort by any column (click on col. heading)
- Navigate to source code line by double-clicking on the artifact
- Run the analysis on another file (or entire project!) and its markers will be added to the view
- Remove markers via ✖



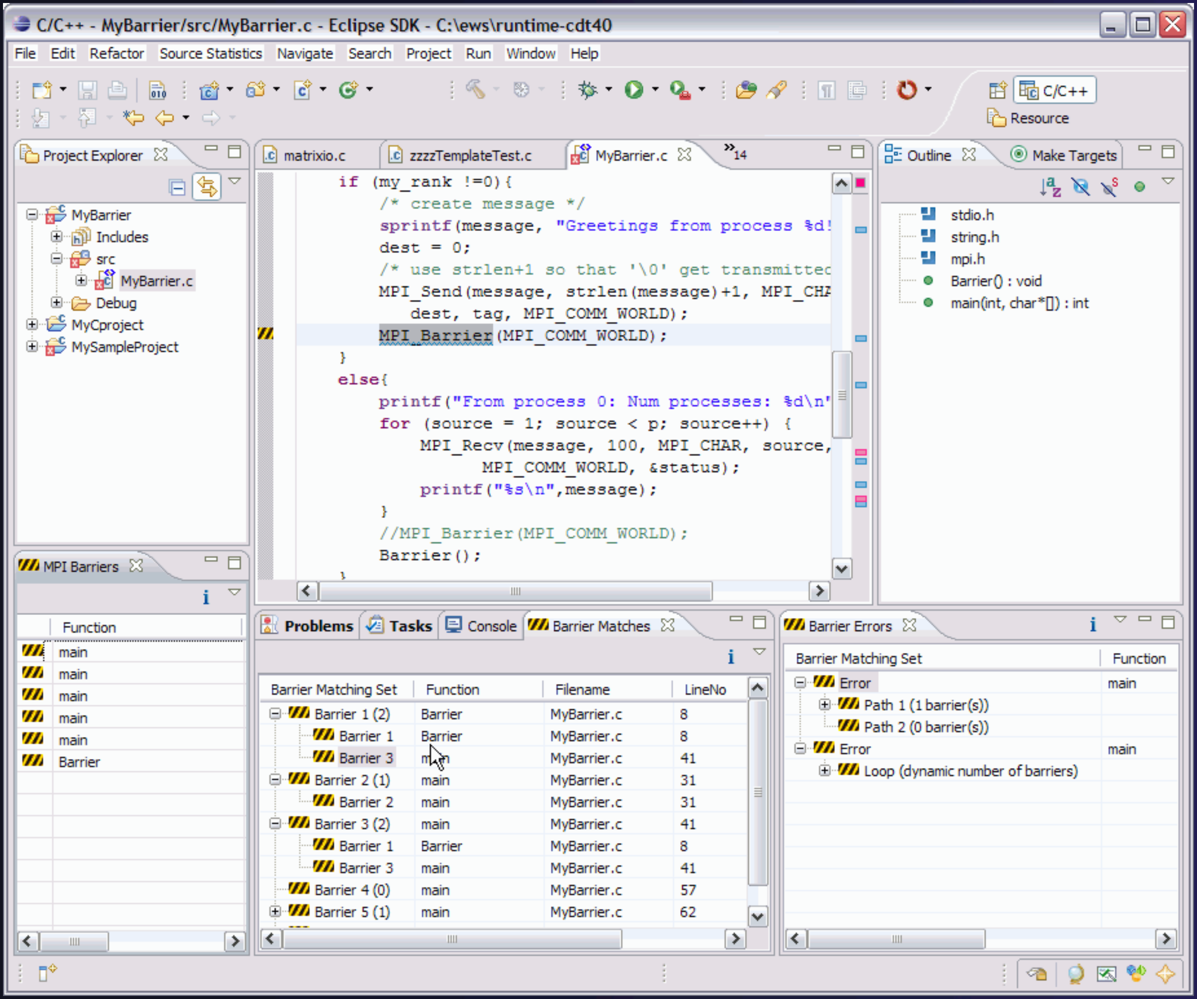


```
    float   vold_start[m];
    int proc_cnt;
    int tid;
    MPI_Datatype *  res_type;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &proc_cnt);
    MPI_Comm_rank(MPI_COMM_WORLD, &tid);

    if ( proc_cnt < 2 )
    {
        fprintf(stderr, "must have at least 2 processes,
        MPI_Finalize();
        return 1;
    }

    if ( (n % (proc_cnt - 1)) != 0 )
    {
        if ( tid == 0 )
```

Artifact	Filename	LineNo
MPI_Init	main.c	67
MPI_Comm_size	main.c	68
MPI_Comm_rank	main.c	69
MPI_Finalize	main.c	74
MPI_Finalize	main.c	83
MPI_Barrier	main.c	89

Click on column headings to sort

MPI Barrier Analysis



```
    if (my_rank !=0){
        /* create message */
        sprintf(message, "Greetings from process %d!
        dest = 0;
        /* use strlen+1 so that '\0' get transmittec
        MPI_Send(message, strlen(message)+1, MPI_CHA
           dest, tag, MPI_COMM_WORLD);
        MPI_Barrier(MPI_COMM_WORLD);
    }
    else{
        printf("From process 0: Num processes: %d\n"
        for (source = 1; source < p; source++) {
            MPI_Recv(message, 100, MPI_CHAR, source,
                  MPI_COMM_WORLD, &status);
            printf("%s\n",message);
        }
        //MPI_Barrier(MPI_COMM_WORLD);
        Barrier();
```

Function
main
main
main
main
main
Barrier

Barrier Matching Set	Function	Filename	LineNo
Barrier 1 (2)	Barrier	MyBarrier.c	8
Barrier 1	Barrier	MyBarrier.c	8
Barrier 3	main	MyBarrier.c	41
Barrier 2 (1)	main	MyBarrier.c	31
Barrier 2	main	MyBarrier.c	31
Barrier 3 (2)	main	MyBarrier.c	41
Barrier 1	Barrier	MyBarrier.c	8
Barrier 3	main	MyBarrier.c	41
Barrier 4 (0)	main	MyBarrier.c	57
Barrier 5 (1)	main	MyBarrier.c	62

Barrier Matching Set	Function
Error	main
Path 1 (1 barrier(s))	
Path 2 (0 barrier(s))	
Error	main
Loop (dynamic number of barriers)	

Verify barrier synchronization in C/MPI programs

Interprocedural static analysis outputs:

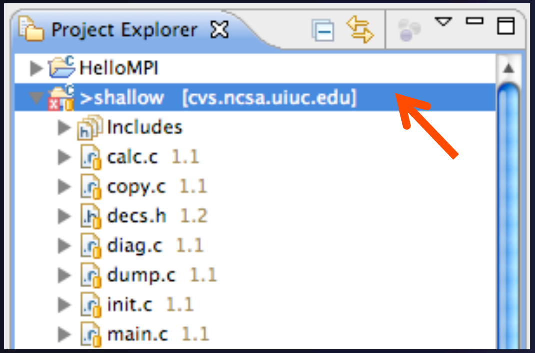
✦For verified programs, lists barrier statements that synchronize together (match)

✦ For synchronization errors, reports counter example that illustrates and explains the error

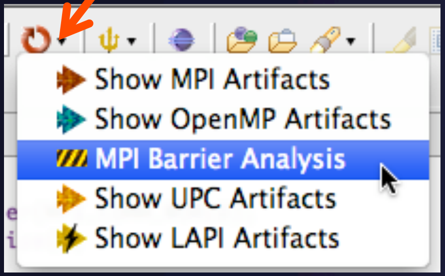
MPI Barrier Analysis – Try it

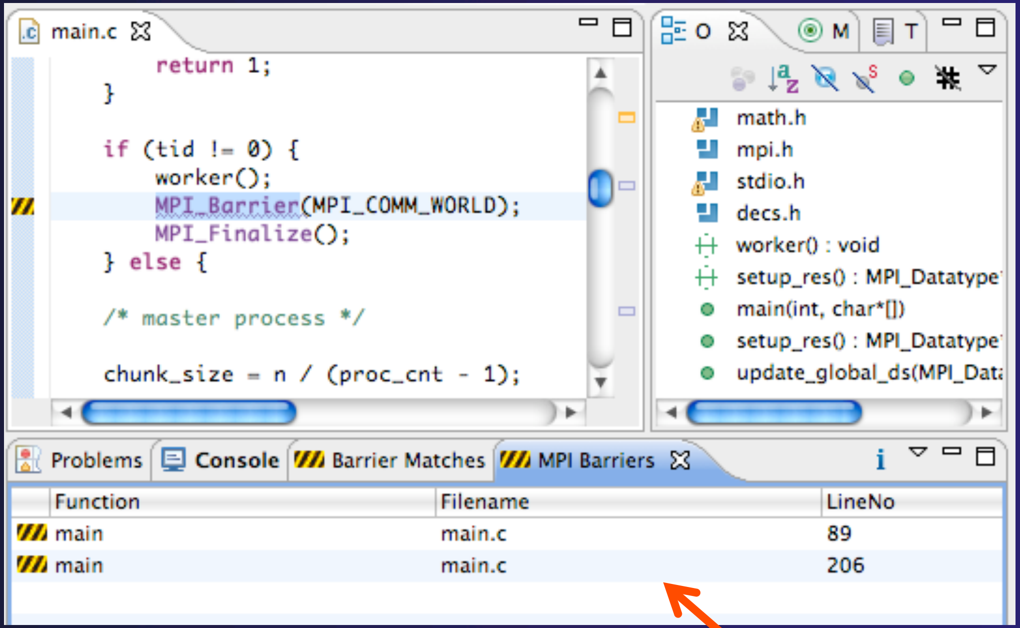
Run the Analysis:

✦In the Project Explorer, Select the shallow project to analyze



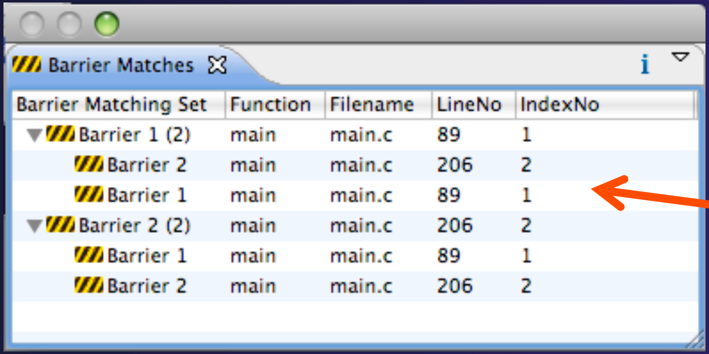
✦ Select the MPI Barrier Analysis action in the menu





```
            return 1;
        }

        if (tid != 0) {
            worker();
            MPI_Barrier(MPI_COMM_WORLD);
            MPI_Finalize();
        } else {

        /* master process */

        chunk_size = n / (proc_cnt - 1);
```

Function	Filename	LineNo
main	main.c	89
main	main.c	206

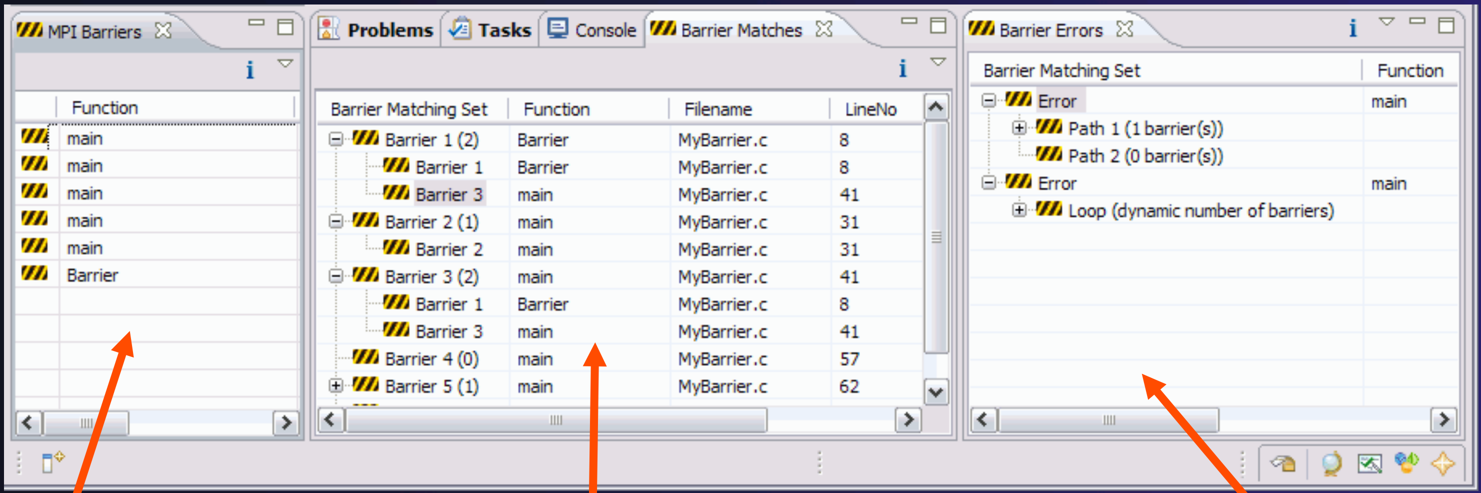


Barrier Matching Set	Function	Filename	LineNo	IndexNo
Barrier 1 (2)	main	main.c	89	1
Barrier 2	main	main.c	206	2
Barrier 1	main	main.c	89	1
Barrier 2 (2)	main	main.c	206	2
Barrier 1	main	main.c	89	1
Barrier 2	main	main.c	206	2

✦Found two barriers

✦Barrier matches

MPI Barrier Analysis - views



Function
main
main
main
main
main
Barrier

Barrier Matching Set	Function	Filename	LineNo
Barrier 1 (2)	Barrier	MyBarrier.c	8
Barrier 1	Barrier	MyBarrier.c	8
Barrier 3	main	MyBarrier.c	41
Barrier 2 (1)	main	MyBarrier.c	31
Barrier 2	main	MyBarrier.c	31
Barrier 3 (2)	main	MyBarrier.c	41
Barrier 1	Barrier	MyBarrier.c	8
Barrier 3	main	MyBarrier.c	41
Barrier 4 (0)	main	MyBarrier.c	57
Barrier 5 (1)	main	MyBarrier.c	62

Barrier Matching Set	Function
Error	main
Path 1 (1 barrier(s))	
Path 2 (0 barrier(s))	
Error	main
Loop (dynamic number of barriers)	

MPI Barriers view

Simply lists the barriers

Like MPI Artifacts view, double-click to navigate to source code line (all 3 views)

Barrier Matches view

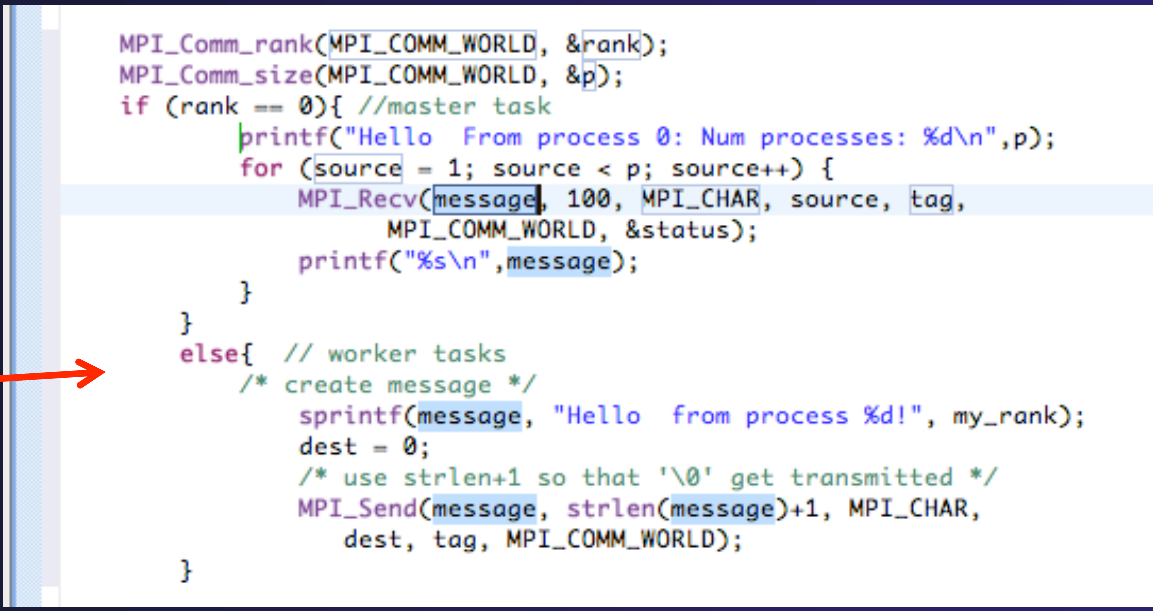
Groups barriers that match together in a barrier set – all processes must go through a barrier in the set to prevent a deadlock

Barrier Errors view

If there are errors, a counter-example shows paths with mismatched number of barriers

MPI Templates

- Allows quick entry of common patterns in MPI programming
- Example: MPI send-receive
- Enter: `mpisr` <ctrl-space>
- Expands to



```
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &p);
if (rank == 0){ //master task
        printf("Hello  From process 0: Num processes: %d\n",p);
        for (source = 1; source < p; source++) {
            MPI_Recv(message, 100, MPI_CHAR, source, tag,
                  MPI_COMM_WORLD, &status);
            printf("%s\n",message);
        }
    }
    else{  // worker tasks
        /* create message */
            sprintf(message, "Hello  from process %d!", my_rank);
            dest = 0;
            /* use strlen+1 so that '\0' get transmitted */
            MPI_Send(message, strlen(message)+1, MPI_CHAR,
               dest, tag, MPI_COMM_WORLD);
    }
```

- Eclipse preferences: add more!
  - C/C++ > Editor > Templates
- Extend to other common patterns

Building the Application

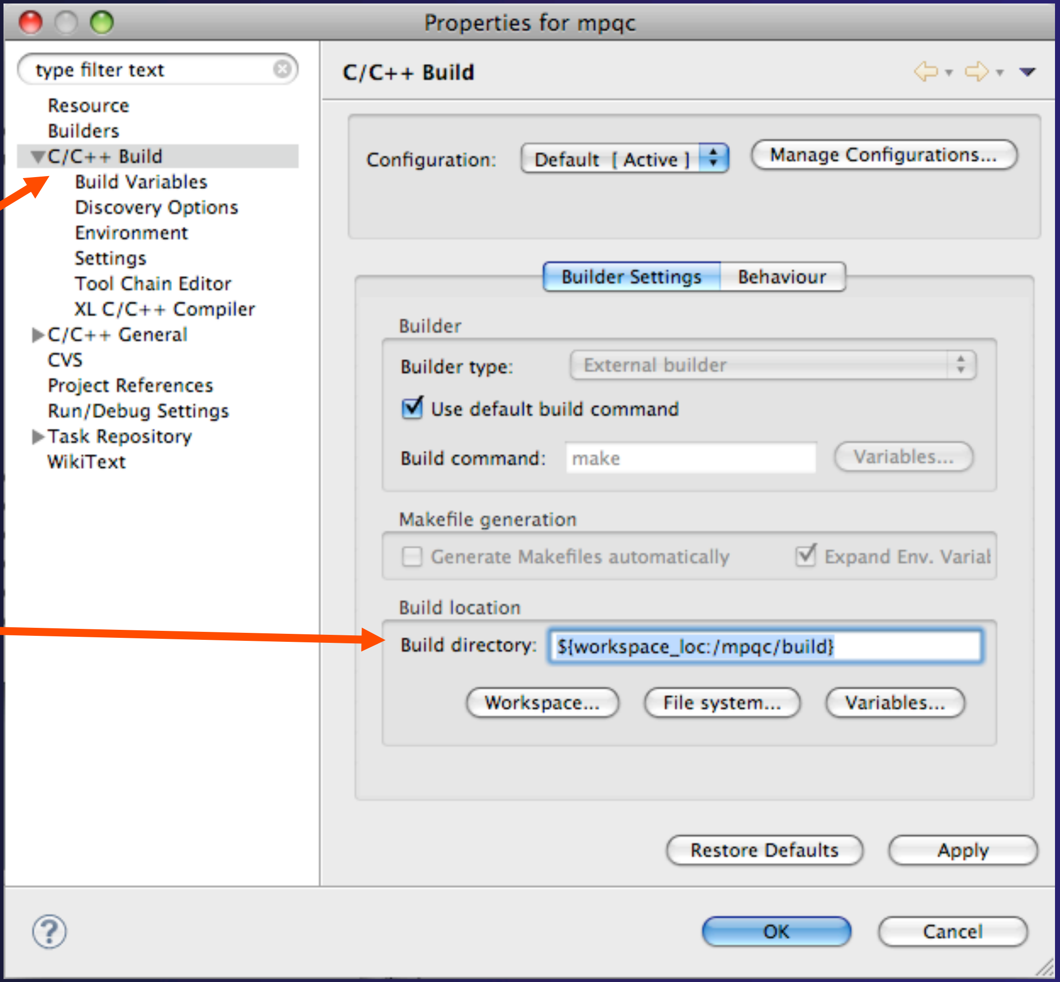
- Configuring the project build directory
- Generating Makefiles
- Creating a Make Target
- Running the build

Makefile Project

- Similar to managed project, but uses custom Makefile (or other script) to control build
- User can specify command that will be used to initiate build
- Can also specify the directory in which the build will take place
- “Make targets” are used to control type of build
- Can switch between managed and un-managed project

Makefile Project Properties

- Right click on project in **Project Explorer** to bring up properties
- Click on **C/C++ Build** for the build settings
- Can change build command if desired
- Can change the **Build location** if it is not the top level



Configuration: Default [Active]

Builder type: External builder

☑ Use default build command

Build command: make

☐ Generate Makefiles automatically ☑ Expand Env. Varial

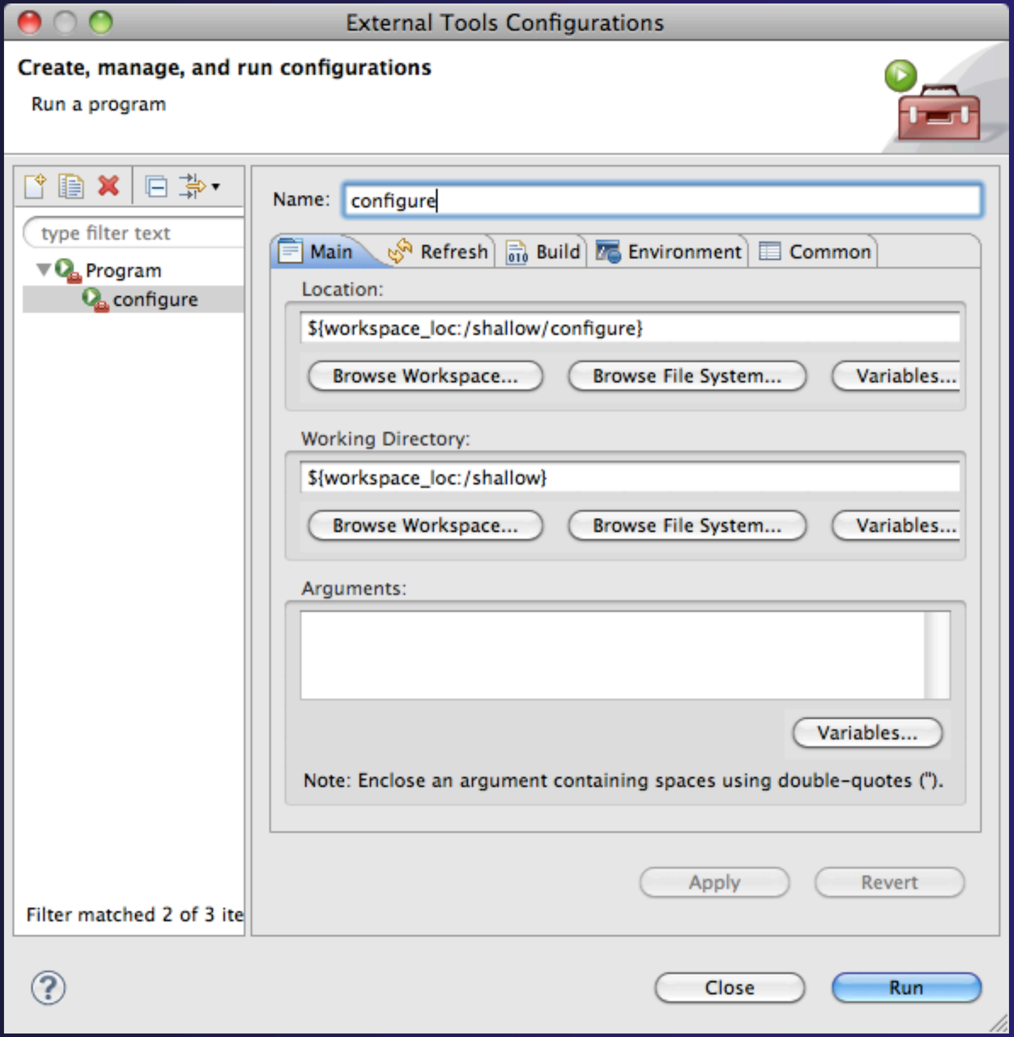
Build directory: ${workspace_loc:/mpqc/build}

About Makefiles and autoconf

- `Autoconf` is a GNU utility often used to create Makefiles for open source projects
  - Used to generate a `configure` script
  - `Configure` is run to generate a Makefile that suits a particular system configuration
  - Normally only needs to be run once, unless the build process needs to be changed
- Run `configure` using two methods:
  - Manually from an external shell
  - By creating an **External Tools Launch Configuration**
- Must refresh **Project Explorer** whenever file system is modified outside of Eclipse, such as after running `configure`

Generate the Makefiles

- From the **Run** menu, select **External Tools▸External Tools Configurations…**
- Create a new **Program**
- For **Location**, click **Browse Workspace…** and find the configure script
- For **Working Directory**, click **Browse Workspace…** and select the project
- Click **Run** and you should see output in the **Console** view
- In **Project Explorer**, right-click and select **Refresh** to see the new files that have been created


External Tools Configurations

Create, manage, and run configurations

Run a program

type filter text

Program

configure

Filter matched 2 of 3 ite

Name: configure

Location:

${workspace_loc:/shallow/configure}

Working Directory:

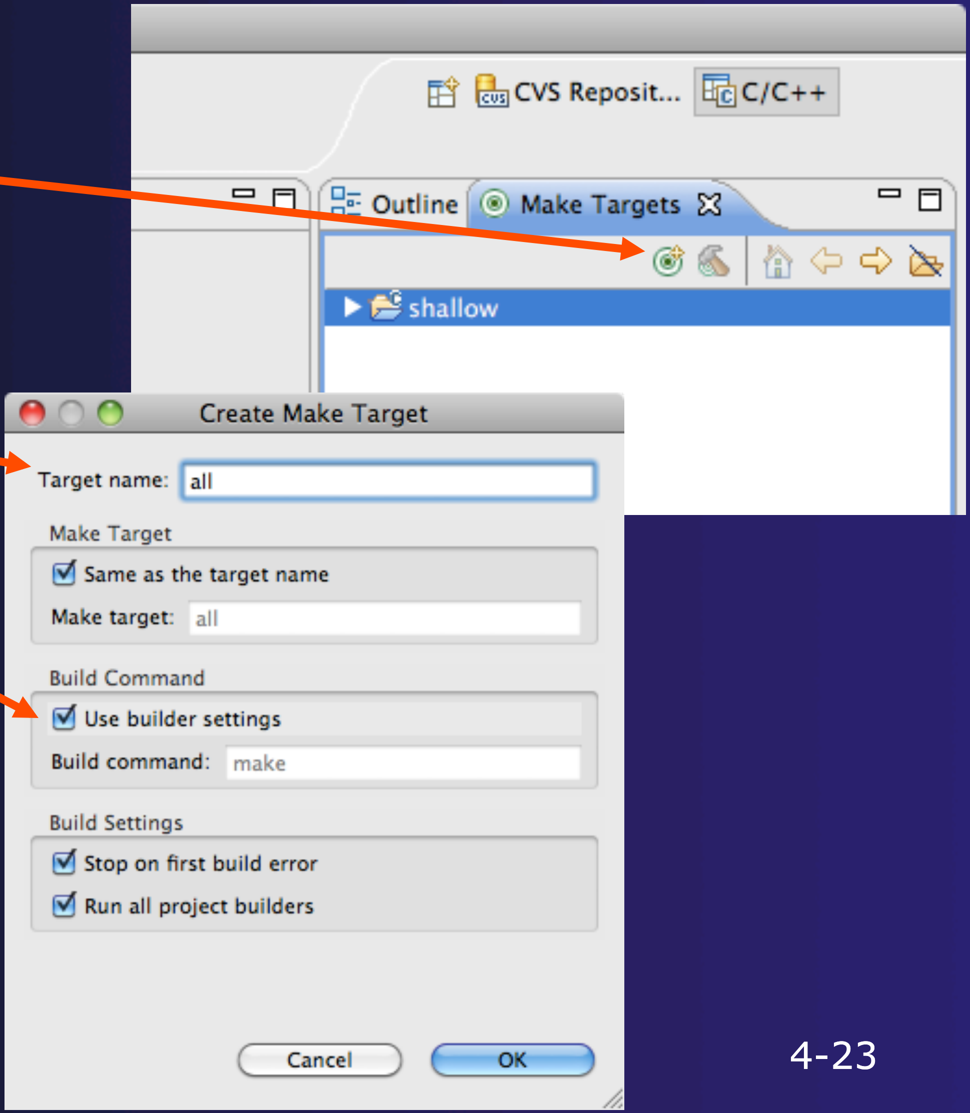
${workspace_loc:/shallow}

Arguments:

Note: Enclose an argument containing spaces using double-quotes (").

Create a Make Target

- Select the project in **Make Targets** view
- Click on **New Make Target** icon
  - This will add a user interface to the targets that already exist in the Makefile
- Enter the name of the target, in this case "all"
- If you need to change the build command, do it here for only this target, or in the build properties for all targets.
- Select **OK**



Target name: all

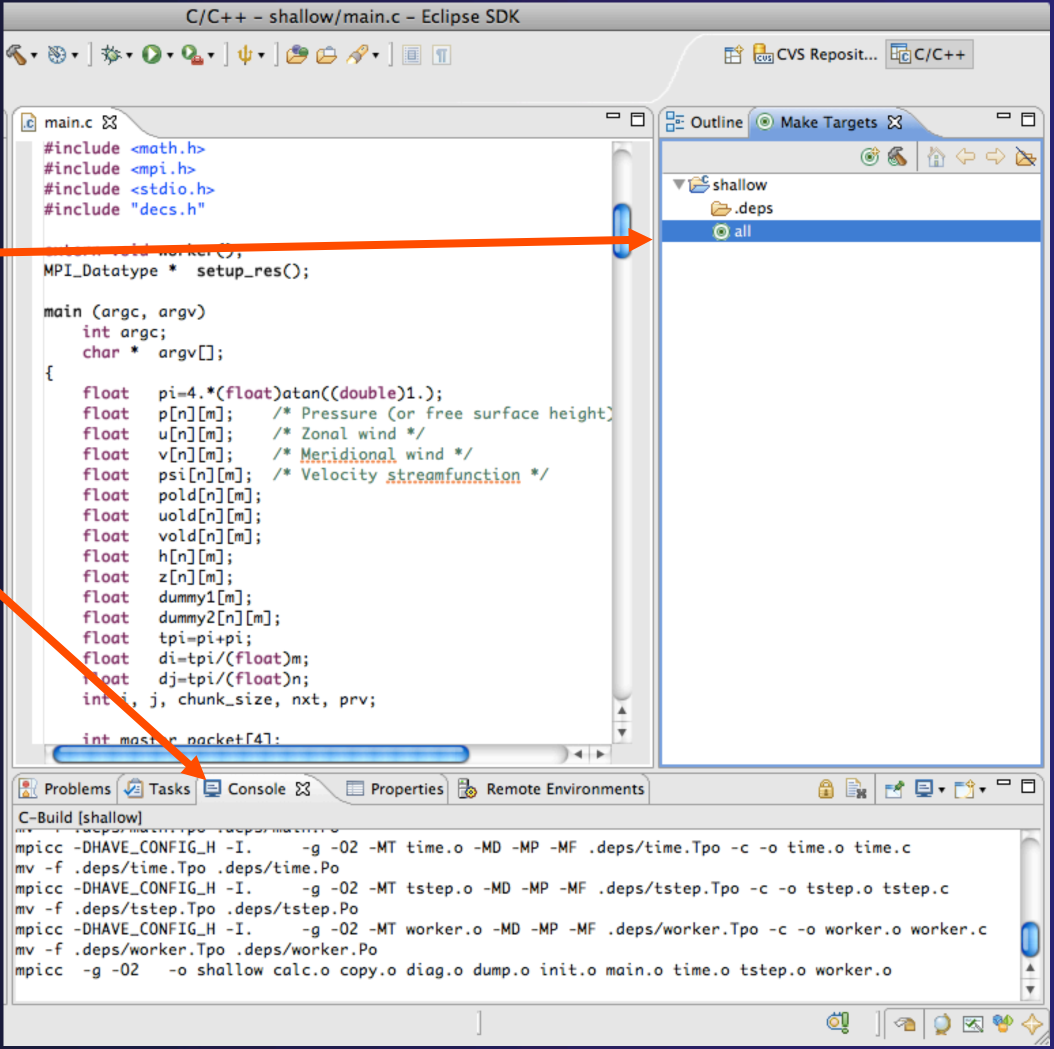
Make Target
[✓] Same as the target name
Make target: all

Build Command
[✓] Use builder settings
Build command: make

Build Settings
[✓] Stop on first build error
[✓] Run all project builders

Running the Build

- Open the project in the **Make Targets** view to see the **all** target
- Double-click on the **all** target to initiate the build
- Output from the build will be visible in the **Console** view



```
#include <math.h>
#include <mpi.h>
#include <stdio.h>
#include "decs.h"

extern void worker();
MPI_Datatype *  setup_res();

main (argc, argv)
    int argc;
    char *  argv[];
{
    float   pi=4.*(float)atan((double)1.);
    float   p[n][m];    /* Pressure (or free surface height)
    float   u[n][m];    /* Zonal wind */
    float   v[n][m];    /* Meridional wind */
    float   psi[n][m];  /* Velocity streamfunction */
    float   pold[n][m];
    float   uold[n][m];
    float   vold[n][m];
    float   h[n][m];
    float   z[n][m];
    float   dummy1[m];
    float   dummy2[n][m];
    float   tpi=pi+pi;
    float   di=tpi/(float)m;
    float   dj=tpi/(float)n;
    int i, j, chunk_size, nxt, prv;

    int master_packet[4];
```

```
mpicc -DHAVE_CONFIG_H -I.     -g -02 -MT time.o -MD -MP -MF .deps/time.Tpo -c -o time.o time.c
mv -f .deps/time.Tpo .deps/time.Po
mpicc -DHAVE_CONFIG_H -I.     -g -02 -MT tstep.o -MD -MP -MF .deps/tstep.Tpo -c -o tstep.o tstep.c
mv -f .deps/tstep.Tpo .deps/tstep.Po
mpicc -DHAVE_CONFIG_H -I.     -g -02 -MT worker.o -MD -MP -MF .deps/worker.Tpo -c -o worker.o worker.c
mv -f .deps/worker.Tpo .deps/worker.Po
mpicc  -g -02   -o shallow calc.o copy.o diag.o dump.o init.o main.o time.o tstep.o worker.o
```

Running the Program

- Creating a resource manager
- Starting the resource manager
- Creating a launch configuration
- Launching the application
- Viewing the application run

Local vs. Remote

- PTP allows the program to be run locally if you have MPI installed
- However we want to run the program on a remote machine
- Need to either cross-compile (hard to configure) or compile remotely
- Remote compiling will be available in the 3.0 release
  - Will be demonstrated later
- For the tutorial, we have pre-compiled the program on the remote machine

Terminology

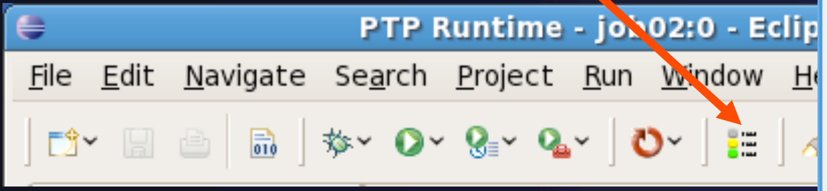
- The **PTP Runtime** perspective is provided for monitoring and controlling applications
- Some terminology
  - **Resource manager** - Corresponds to an instance of a resource management system (e.g. a job scheduler). You can have multiple resource mangers connected to different machines.
  - **Queue** - A queue of pending jobs
  - **Job** – A single run of a parallel application
  - **Machine** - A parallel computer system
  - **Node** - Some form of computational resource
  - **Process** - An execution unit (may be multiple threads of execution)

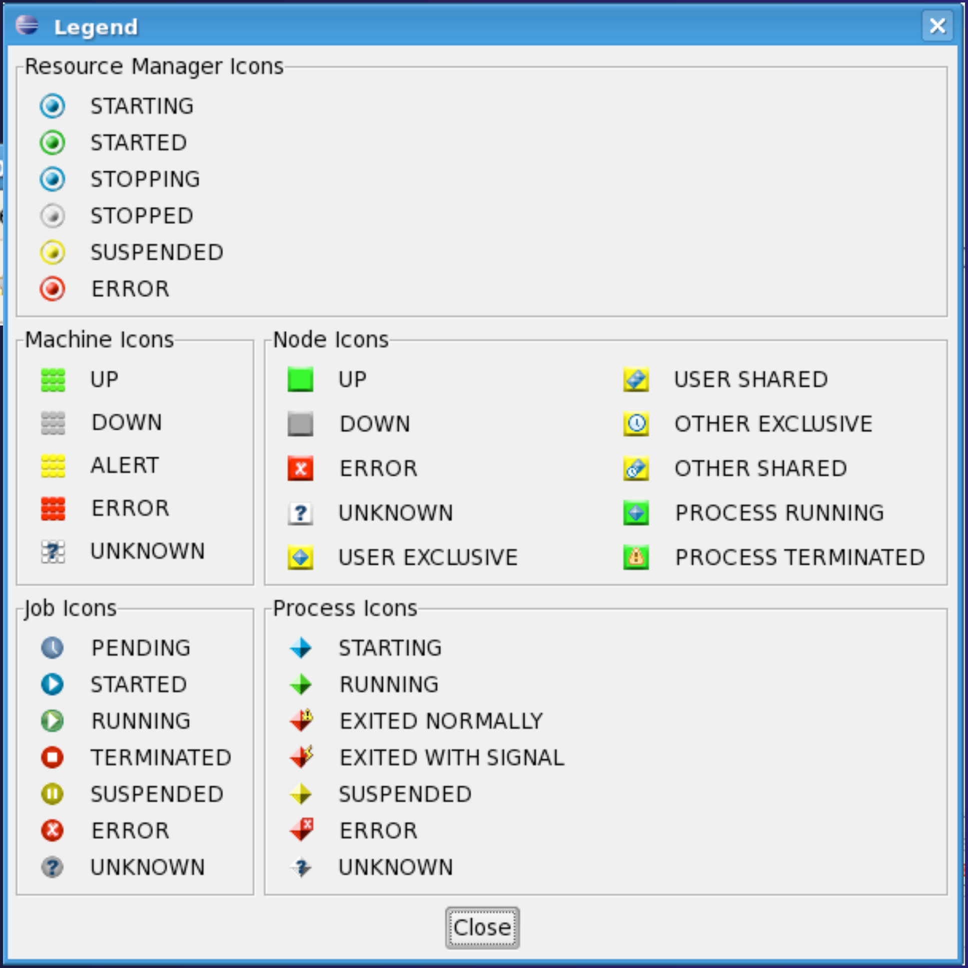
Resource Managers

- PTP uses the term "resource manager" to refer to any subsystem that controls the resources required for launching a parallel job.
- Examples:
  - Job scheduler (e.g. LoadLeveler)
  - Open MPI Runtime Environment (ORTE)
- Each resource manager controls one target system
- Resource Managers can be local or remote

About PTP Icons

- Open using legend icon in toolbar


PTP Runtime - job02:0 - Eclip
File Edit Navigate Search Project Run Window He



Resource Manager Icons

- STARTING
- STARTED
- STOPPING
- STOPPED
- SUSPENDED
- ERROR

Machine Icons

- UP
- DOWN
- ALERT
- ERROR
- UNKNOWN

Node Icons

- UP
- DOWN
- ERROR
- UNKNOWN
- USER EXCLUSIVE
- USER SHARED
- OTHER EXCLUSIVE
- OTHER SHARED
- PROCESS RUNNING
- PROCESS TERMINATED

Job Icons

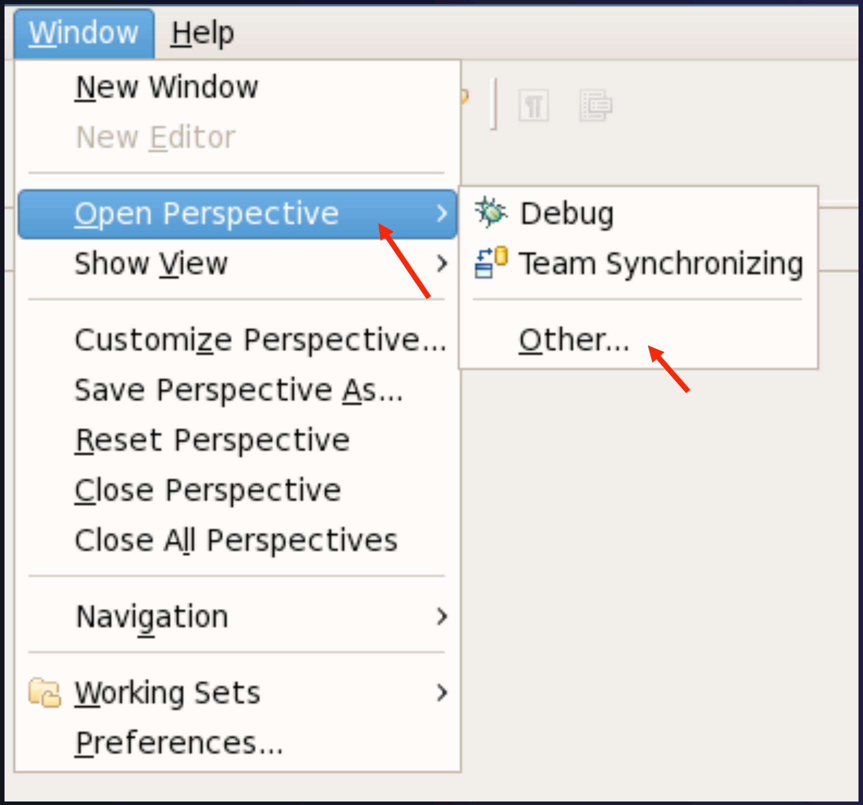
- PENDING
- STARTED
- RUNNING
- TERMINATED
- SUSPENDED
- ERROR
- UNKNOWN

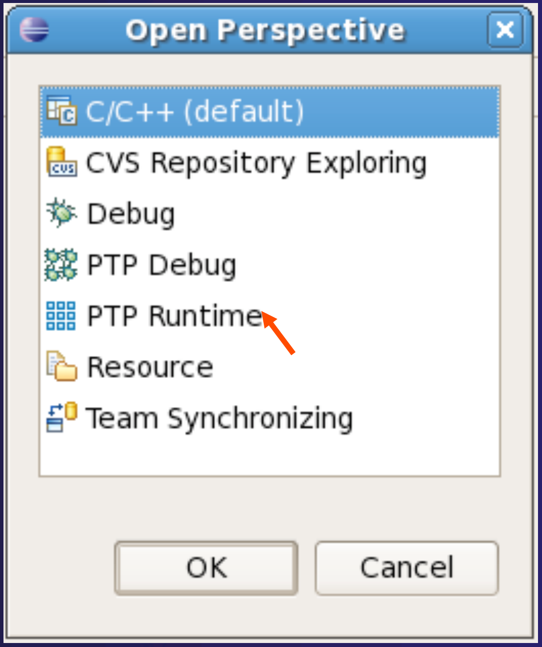
Process Icons

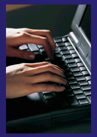
- STARTING
- RUNNING
- EXITED NORMALLY
- EXITED WITH SIGNAL
- SUSPENDED
- ERROR
- UNKNOWN

Open PTP Runtime Perspective

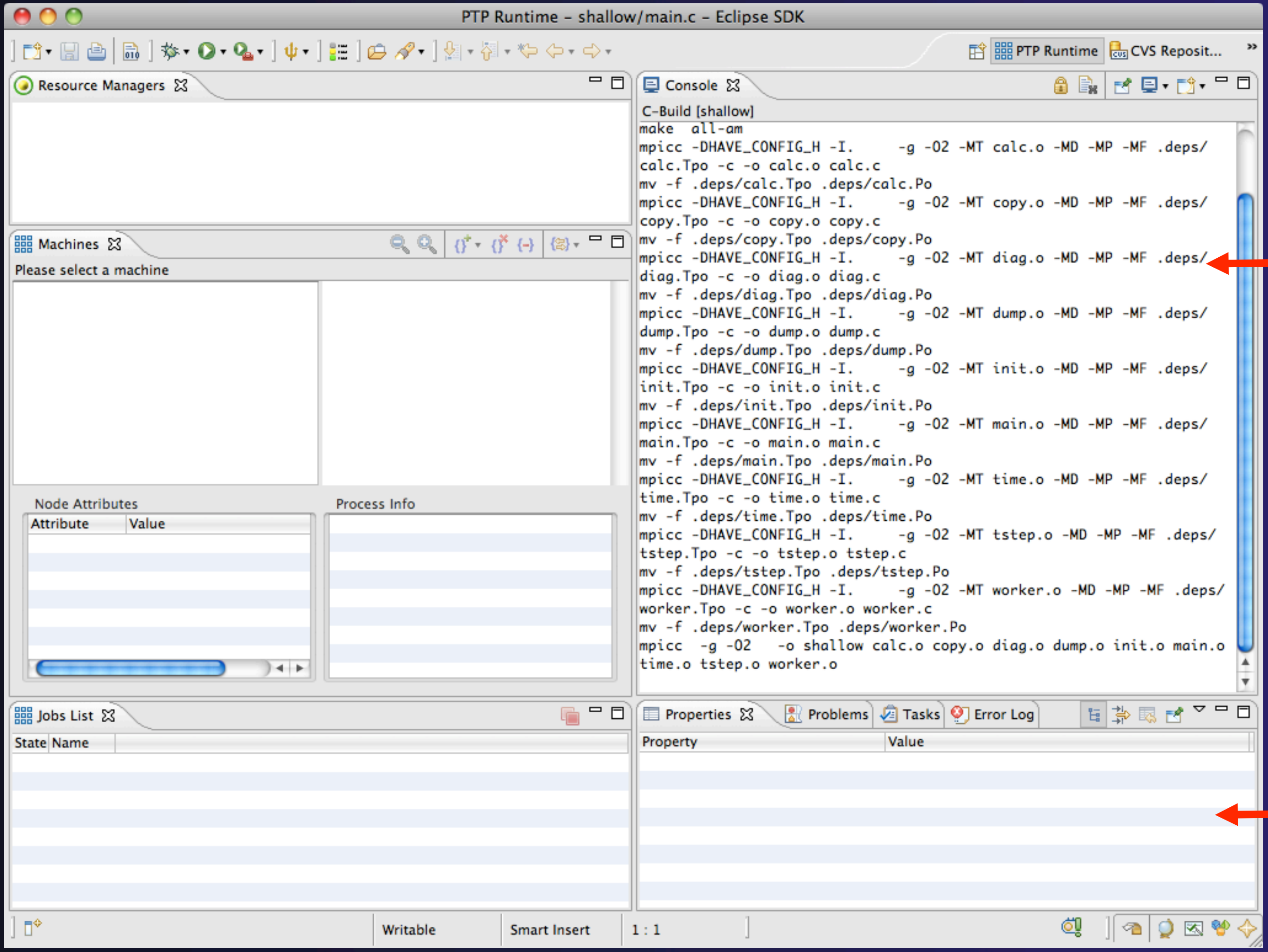
Window > Open Perspective > Other…







PTP Runtime Perspective

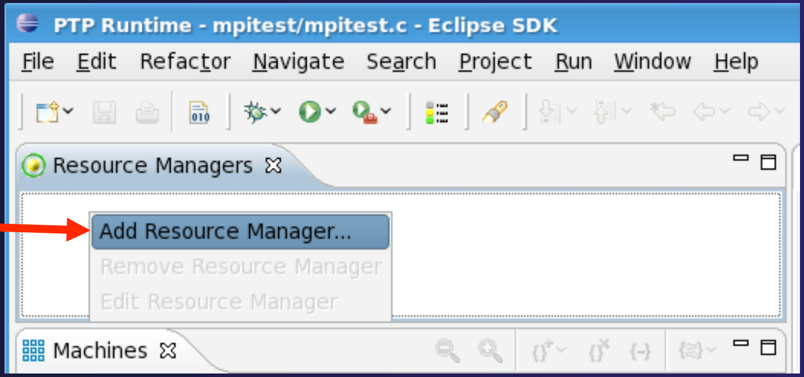


C-Build [shallow]

```
make  all-am
mpicc -DHAVE_CONFIG_H -I.     -g -O2 -MT calc.o -MD -MP -MF .deps/
calc.Tpo -c -o calc.o calc.c
mv -f .deps/calc.Tpo .deps/calc.Po
mpicc -DHAVE_CONFIG_H -I.     -g -O2 -MT copy.o -MD -MP -MF .deps/
copy.Tpo -c -o copy.o copy.c
mv -f .deps/copy.Tpo .deps/copy.Po
mpicc -DHAVE_CONFIG_H -I.     -g -O2 -MT diag.o -MD -MP -MF .deps/
diag.Tpo -c -o diag.o diag.c
mv -f .deps/diag.Tpo .deps/diag.Po
mpicc -DHAVE_CONFIG_H -I.     -g -O2 -MT dump.o -MD -MP -MF .deps/
dump.Tpo -c -o dump.o dump.c
mv -f .deps/dump.Tpo .deps/dump.Po
mpicc -DHAVE_CONFIG_H -I.     -g -O2 -MT init.o -MD -MP -MF .deps/
init.Tpo -c -o init.o init.c
mv -f .deps/init.Tpo .deps/init.Po
mpicc -DHAVE_CONFIG_H -I.     -g -O2 -MT main.o -MD -MP -MF .deps/
main.Tpo -c -o main.o main.c
mv -f .deps/main.Tpo .deps/main.Po
mpicc -DHAVE_CONFIG_H -I.     -g -O2 -MT time.o -MD -MP -MF .deps/
time.Tpo -c -o time.o time.c
mv -f .deps/time.Tpo .deps/time.Po
mpicc -DHAVE_CONFIG_H -I.     -g -O2 -MT tstep.o -MD -MP -MF .deps/
tstep.Tpo -c -o tstep.o tstep.c
mv -f .deps/tstep.Tpo .deps/tstep.Po
mpicc -DHAVE_CONFIG_H -I.     -g -O2 -MT worker.o -MD -MP -MF .deps/
worker.Tpo -c -o worker.o worker.c
mv -f .deps/worker.Tpo .deps/worker.Po
mpicc  -g -O2   -o shallow calc.o copy.o diag.o dump.o init.o main.o
time.o tstep.o worker.o
```

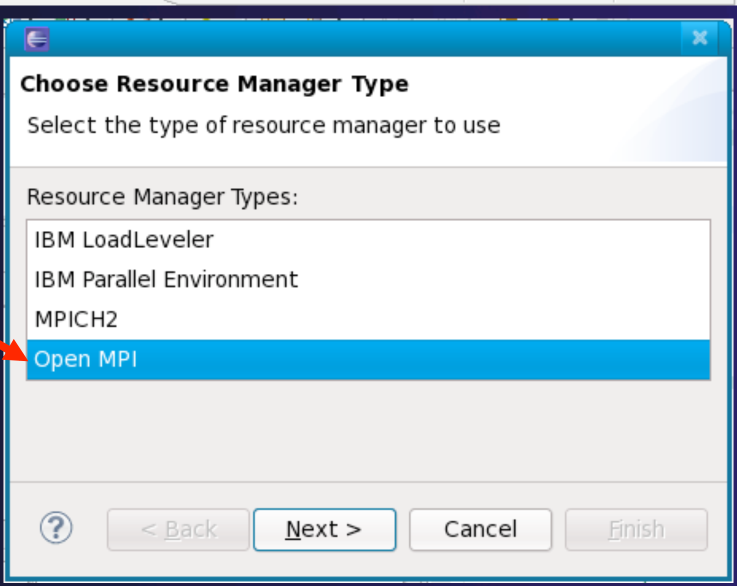
Adding a Resource Manager

- Right-click in Resource Managers view and select **Add Resource Manager**


PTP Runtime - mpitest/mpitest.c - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
Resource Managers
Add Resource Manager...
Remove Resource Manager
Edit Resource Manager
Machines

- Choose the **Open MPI Resource Manager Type**

- Select **Next>**

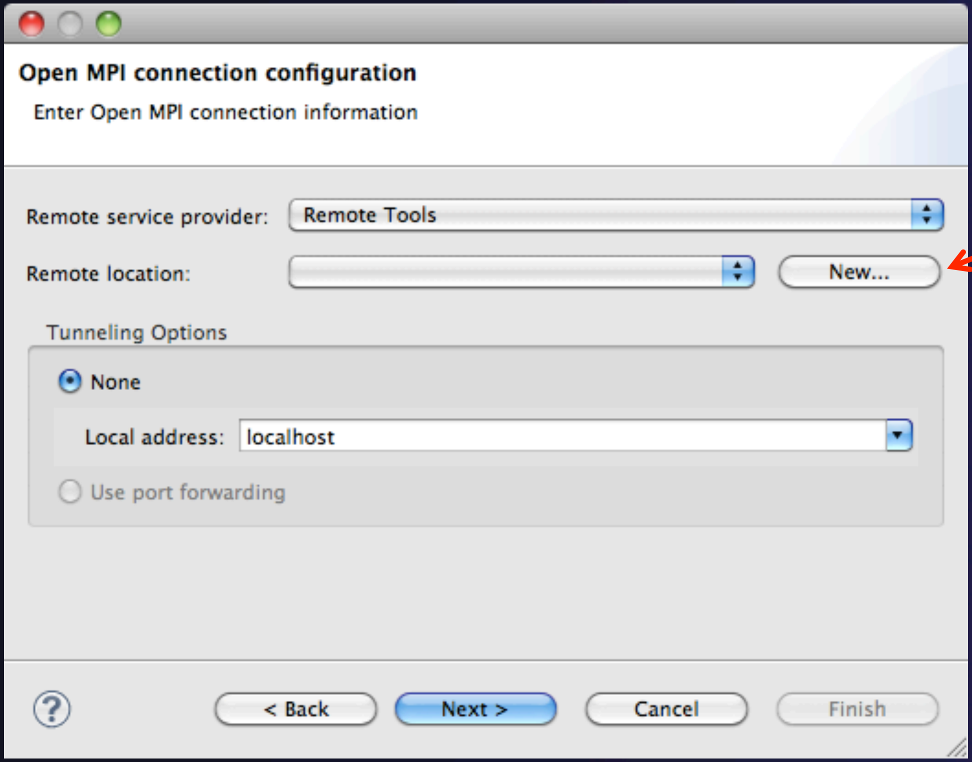

Choose Resource Manager Type

Select the type of resource manager to use

Resource Manager Types:

IBM LoadLeveler
IBM Parallel Environment
MPICH2
Open MPI

Configure the Remote Location



Open MPI connection configuration

Enter Open MPI connection information

Remote service provider: Remote Tools

Remote location:

Tunneling Options

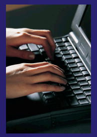
(● None)

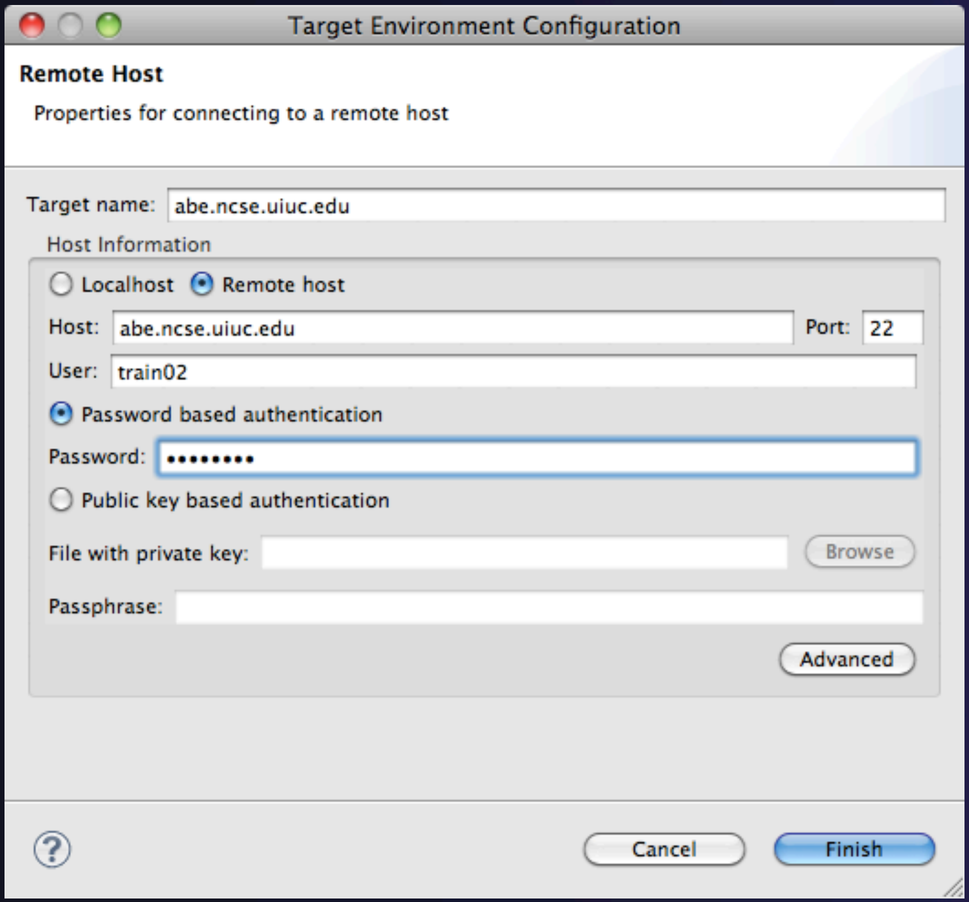
Local address: localhost

○ Use port forwarding

- Choose **Remote Tools** for **Remote service provider**
- Choose **Remote location** or click **New...** to create a new location

Create a New Location





Remote Host

Properties for connecting to a remote host

Target name: abe.ncse.uiuc.edu

Host Information

( ) Localhost [✓ Remote host]

Host: abe.ncse.uiuc.edu Port: 22

User: train02

[✓ Password based authentication]

Password: ••••••••

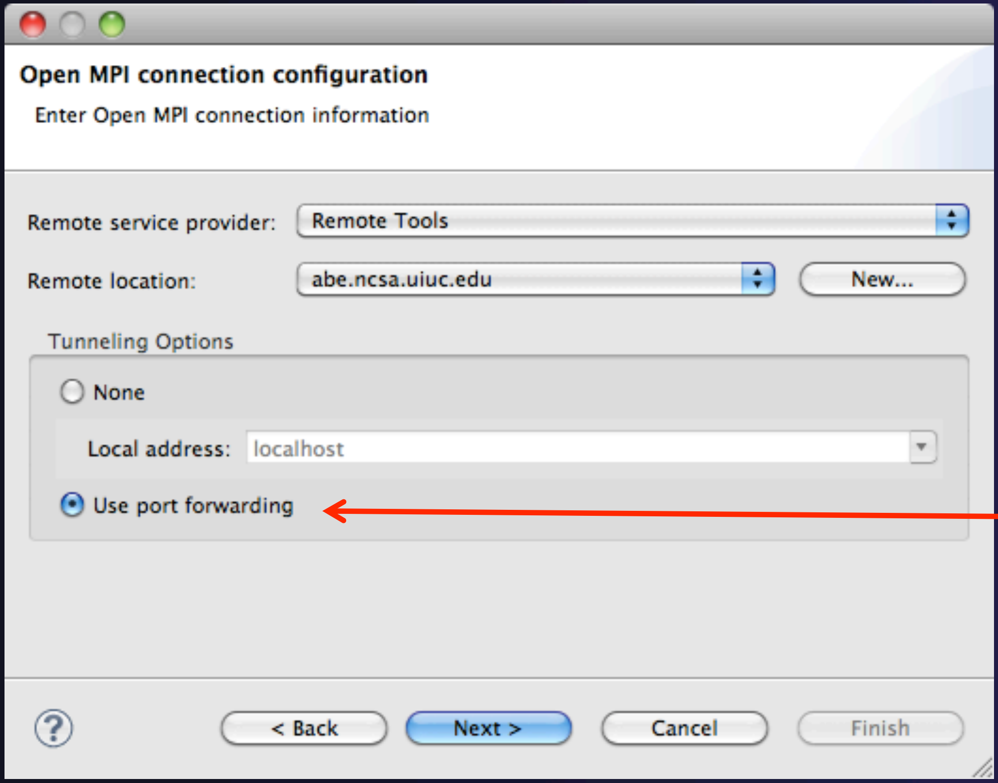
( ) Public key based authentication

File with private key:

Passphrase:

- Enter a name for the connection
  - Can be any string
- Select **Remote host**
- Enter the **Host** name
- Enter the **User** name
- Select **Password based authentication**
- Enter the **Password**
- Click **Finish**

Configure Tunneling



Open MPI connection configuration

Enter Open MPI connection information

Remote service provider: Remote Tools

Remote location: abe.ncsa.uiuc.edu

Tunneling Options

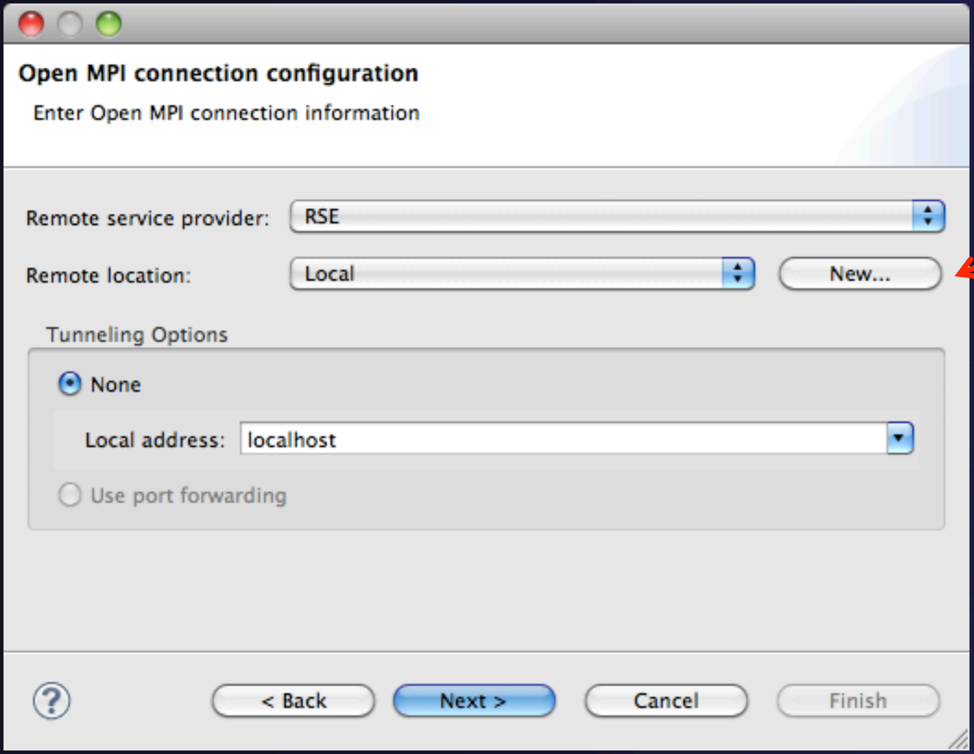
None

Local address: localhost

[✓ Use port forwarding]

- Some remote service providers support tunneling over ssh connections (e.g. Remote Tools)
- The port forwarding option would be enabled this if it was available
- Select **Use port forwarding** so all communication will be sent over the tunnel

Configure the Remote Location (RSE)


Open MPI connection configuration

Enter Open MPI connection information

Remote service provider: RSE

Remote location: Local

Tunneling Options

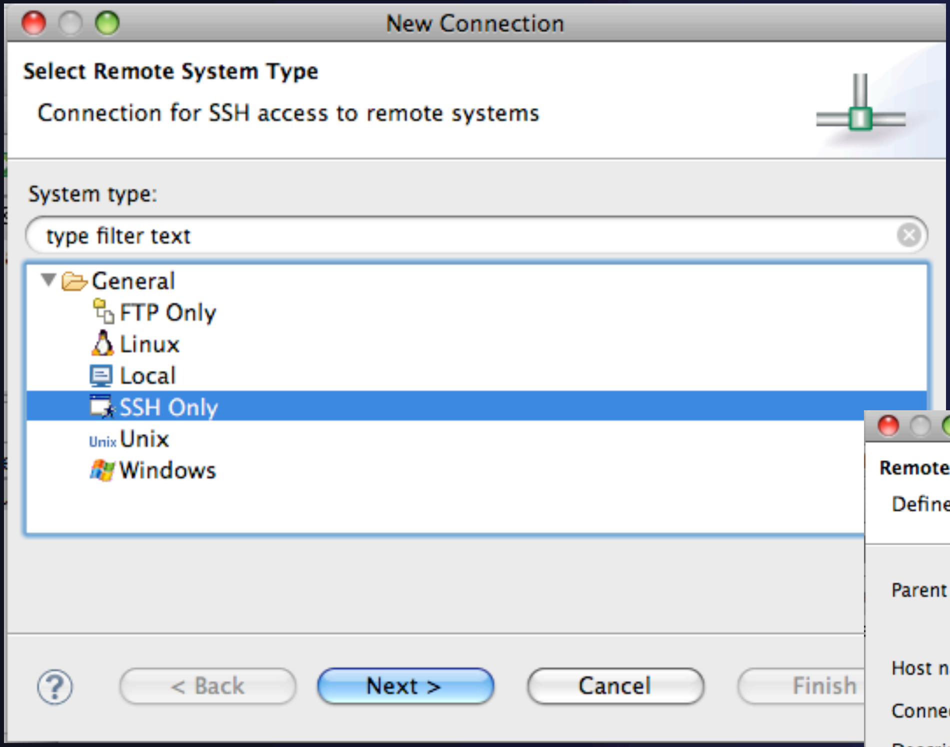
[✓ None]

Local address: localhost

Use port forwarding

- Choose **RSE** for **Remote service provider**
- Choose **Remote location** or click **New...** to create a new location
  - **Local** can be used to run applications locally

Create a New Location (RSE)



New Connection

Select Remote System Type

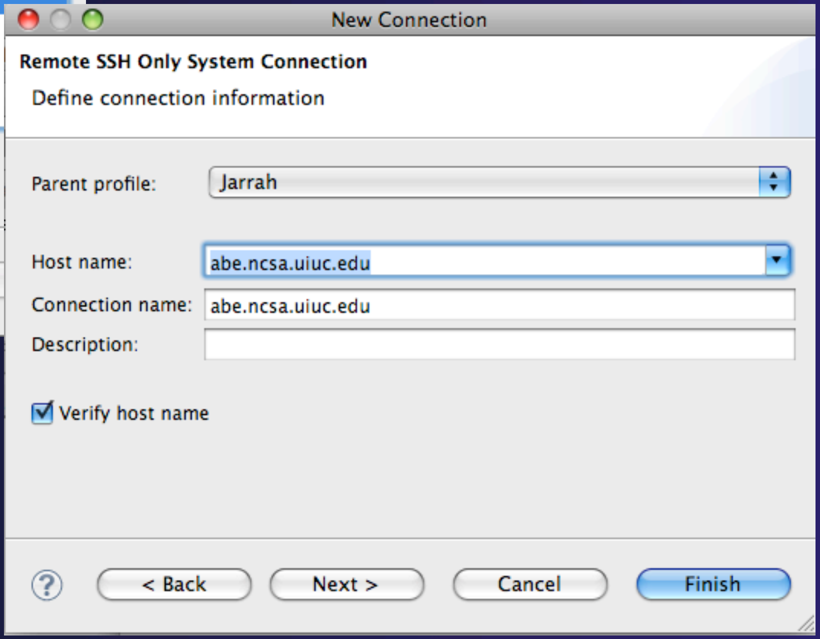
Connection for SSH access to remote systems

System type:

type filter text

- General
  - FTP Only
  - Linux
  - Local
  - SSH Only
  - Unix
  - Windows

- Choose **SSH Only** for this connection
- Click **Next>**
- Enter **Host name** of remote system
- Click **Finish**



New Connection

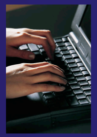
Remote SSH Only System Connection

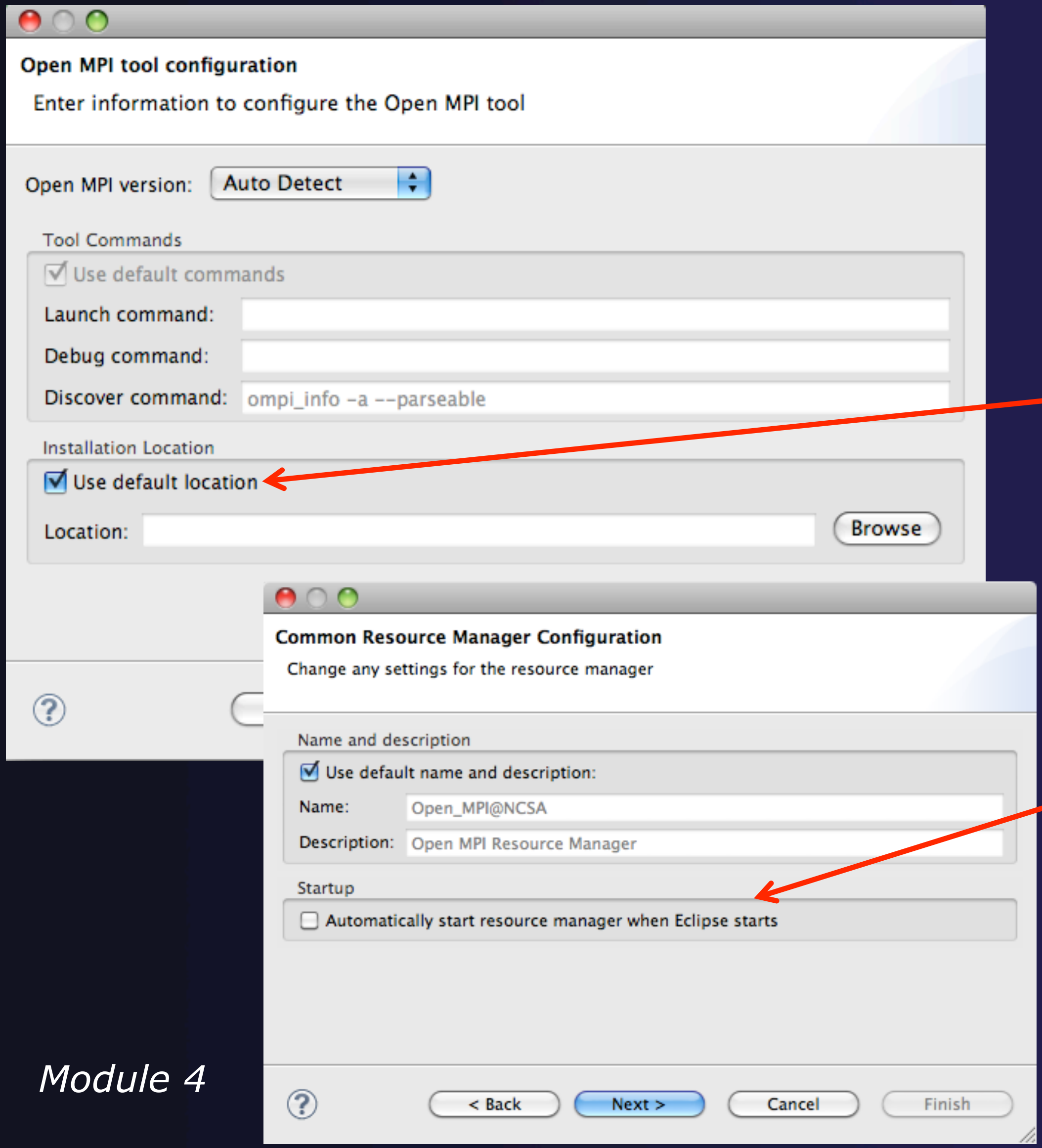
Define connection information

Parent profile:	Jarrah
Host name:	abe.ncsa.uiuc.edu
Connection name:	abe.ncsa.uiuc.edu
Description:	

[✓] Verify host name

Configure the Resource Manager





Open MPI tool configuration

Enter information to configure the Open MPI tool

Open MPI version: Auto Detect

Tool Commands

☑ Use default commands

Launch command:

Debug command:

Discover command: ompi_info -a --parseable

Installation Location

☑ Use default location

Location:

Common Resource Manager Configuration

Change any settings for the resource manager

Name and description

☑ Use default name and description:

Name: Open_MPI@NCSA

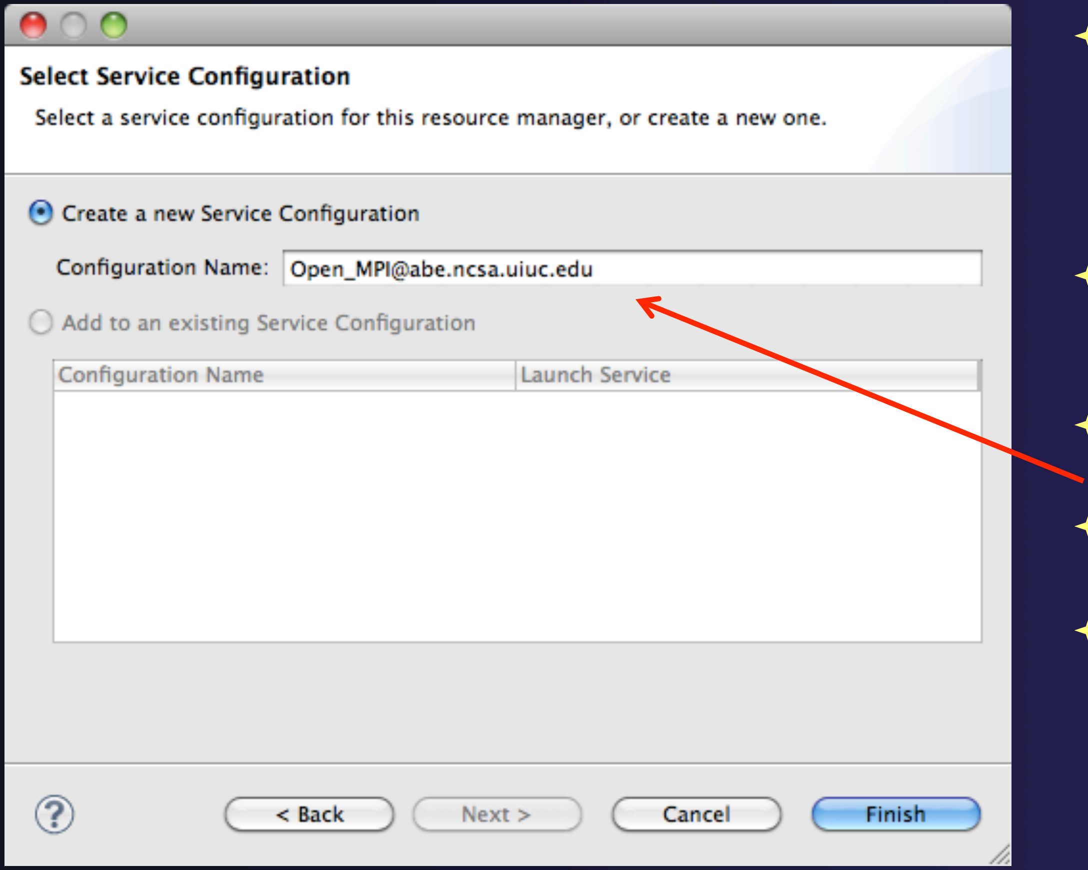
Description: Open MPI Resource Manager

Startup

☐ Automatically start resource manager when Eclipse starts

- The Open MPI resource manager will auto detect the version and use the appropriate commands
  - Change only if you're an expert
- Set the location of the "mpirun" command if it is not in your path
- Click **Next>**
- Change the **Name** or **Description** of the resource manager if you wish
- You can also set the resource manager to automatically start
- Click **Next>**

Save the Service Configuration


Select Service Configuration

Select a service configuration for this resource manager, or create a new one.

[✓ Create a new Service Configuration]

Configuration Name: Open_MPI@abe.ncsa.uiuc.edu

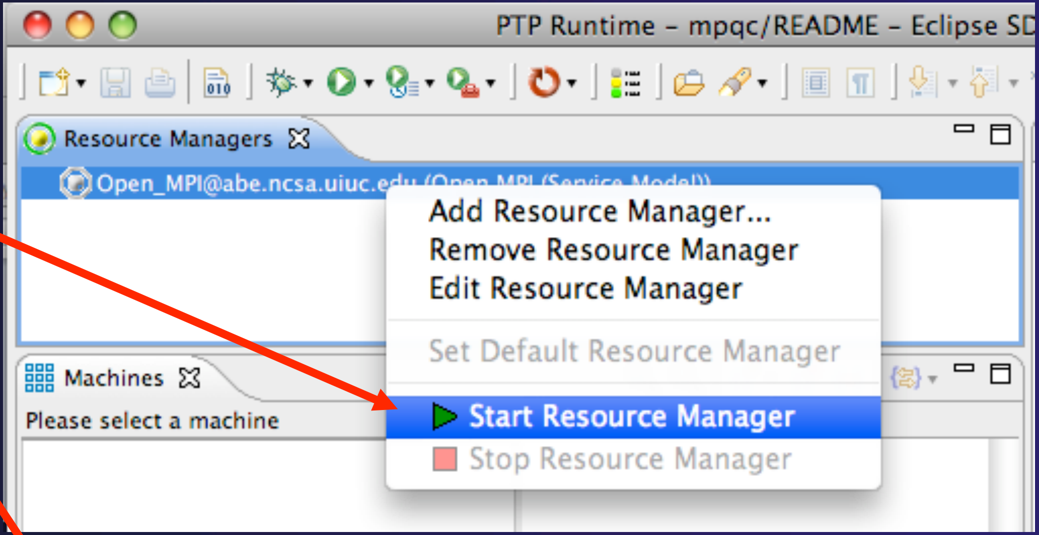
Add to an existing Service Configuration

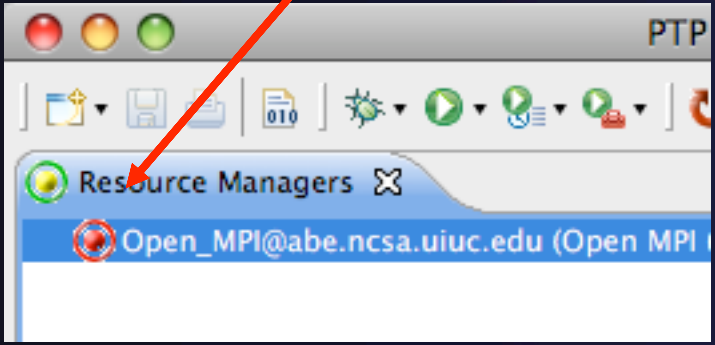
Configuration Name	Launch Service

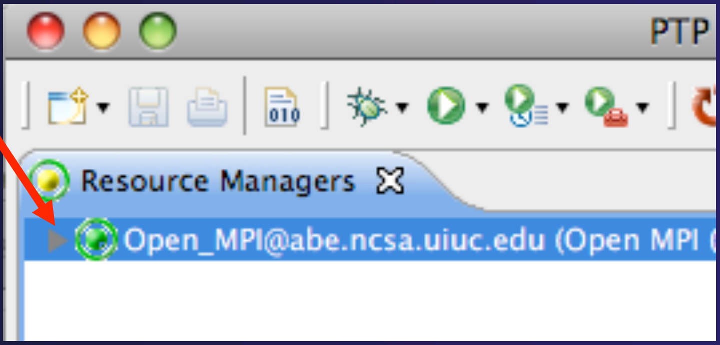
- Resource manager configuration details are kept in a "service configuration" along with other configuration information
- The details can be kept in a new configuration, or added to an existing configuration
- In this example, we create a new service configuration
- Use a name that describes the purpose of the configuration
- Click **Finish**

Starting the Resource Manager

- Right click on new resource manager and select **Start resource manager**
- If everything is ok, you should see the resource manager change to green
- If something goes wrong, it will change to red

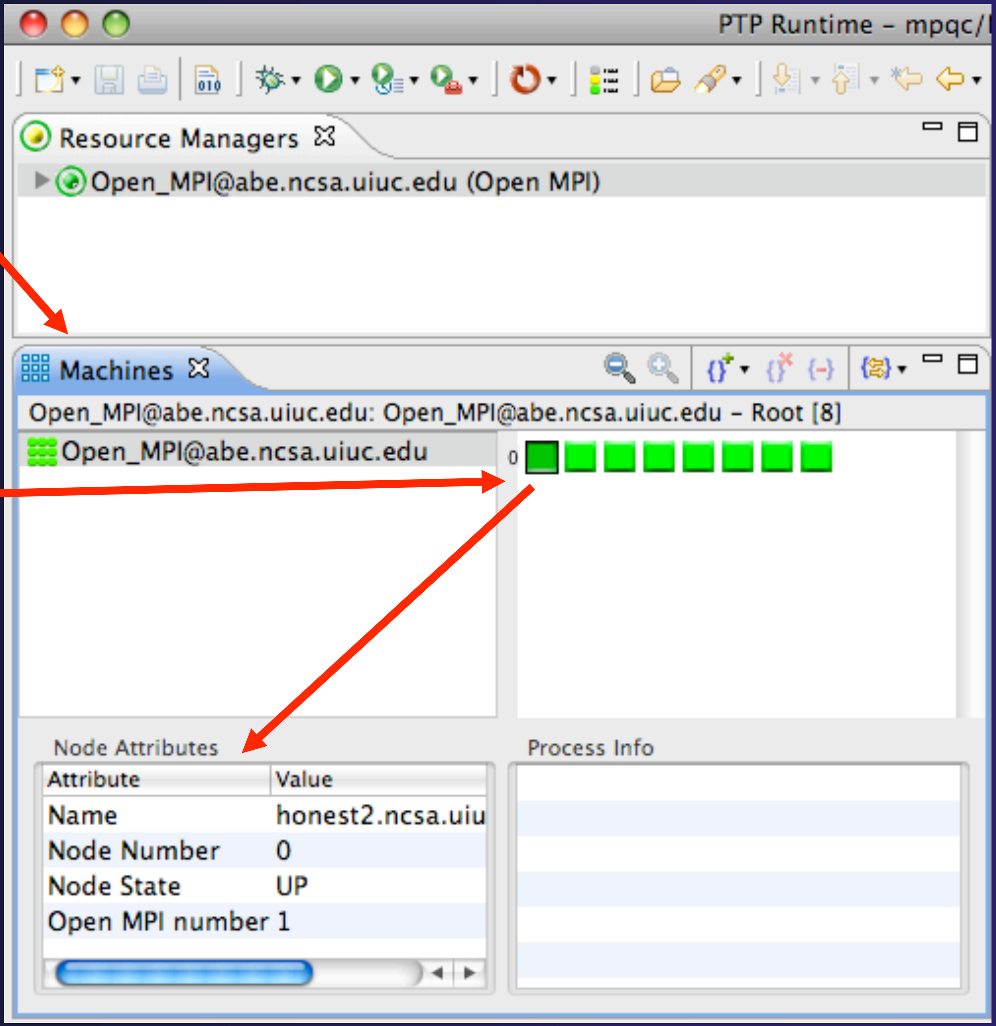






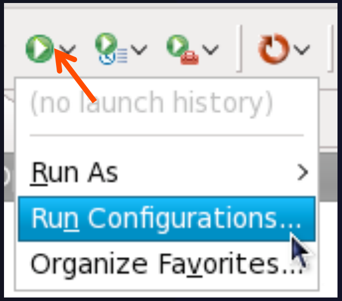
System Monitoring

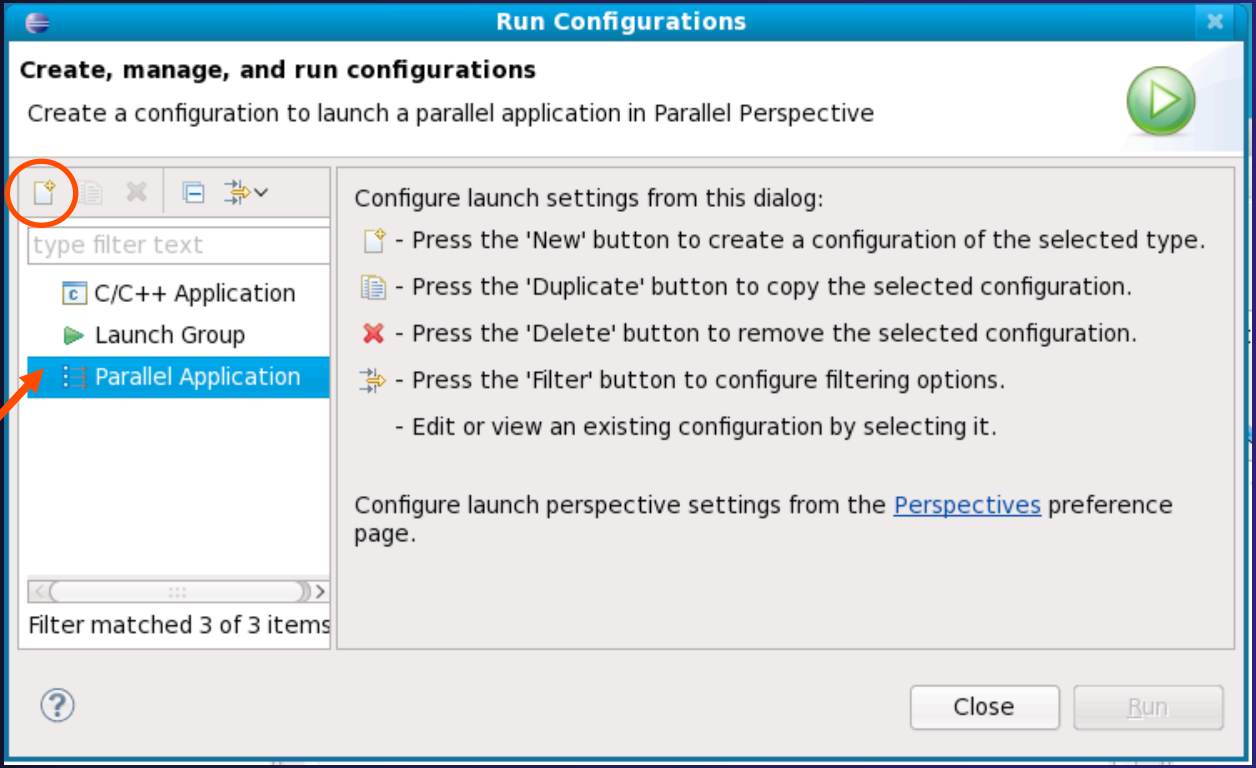
- Machine status shown in **Machines** view
- Node status also shown **Machines** view
- Hover over node to see node name
- Double-click on node to show attributes


PTP Runtime – mpqc/
Resource Managers
Open_MPI@abe.ncsa.uiuc.edu (Open MPI)
Machines
Open_MPI@abe.ncsa.uiuc.edu: Open_MPI@abe.ncsa.uiuc.edu – Root [8]
Open_MPI@abe.ncsa.uiuc.edu
0
Node Attributes
Process Info

Attribute	Value
Name	honest2.ncsa.uiu
Node Number	0
Node State	UP
Open MPI number	1

Create a Launch Configuration





Run Configurations

Create, manage, and run configurations

Create a configuration to launch a parallel application in Parallel Perspective

type filter text

- C/C++ Application
- Launch Group
- Parallel Application

Filter matched 3 of 3 items

Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the Perspectives preference page.

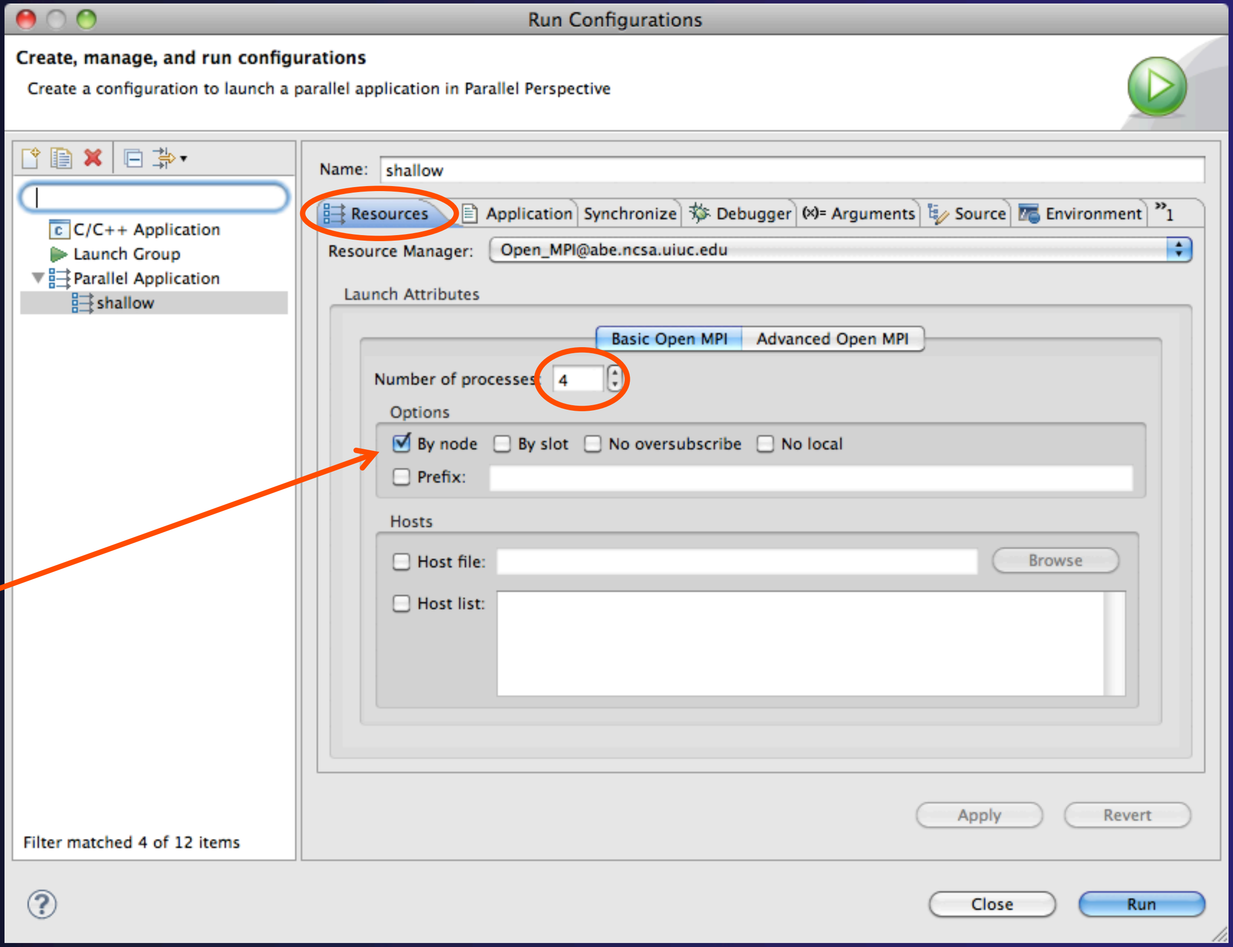
Close Run

- Open the run configuration dialog **Run ▶ Run Configurations…**
- Select **Parallel Application**
- Select the **New** button

Depending on which flavor of Eclipse you installed, you might have more choices in Application types.

Complete the Resources Tab

- In **Resources** tab, select the resource manager you want to use to launch this job
- Enter a value in the **Number of processes** field
- Other fields can be used to specify resource manager-specific information
  - E.g. specify **By node** to allocate each process to a different node



Run Configurations

Create, manage, and run configurations

Create a configuration to launch a parallel application in Parallel Perspective

Name: shallow

Resource Manager: Open_MPI@abe.ncsa.uiuc.edu

Launch Attributes

Number of processes: 4

Options

[✓ By node] By slot No oversubscribe No local

Prefix:

Hosts

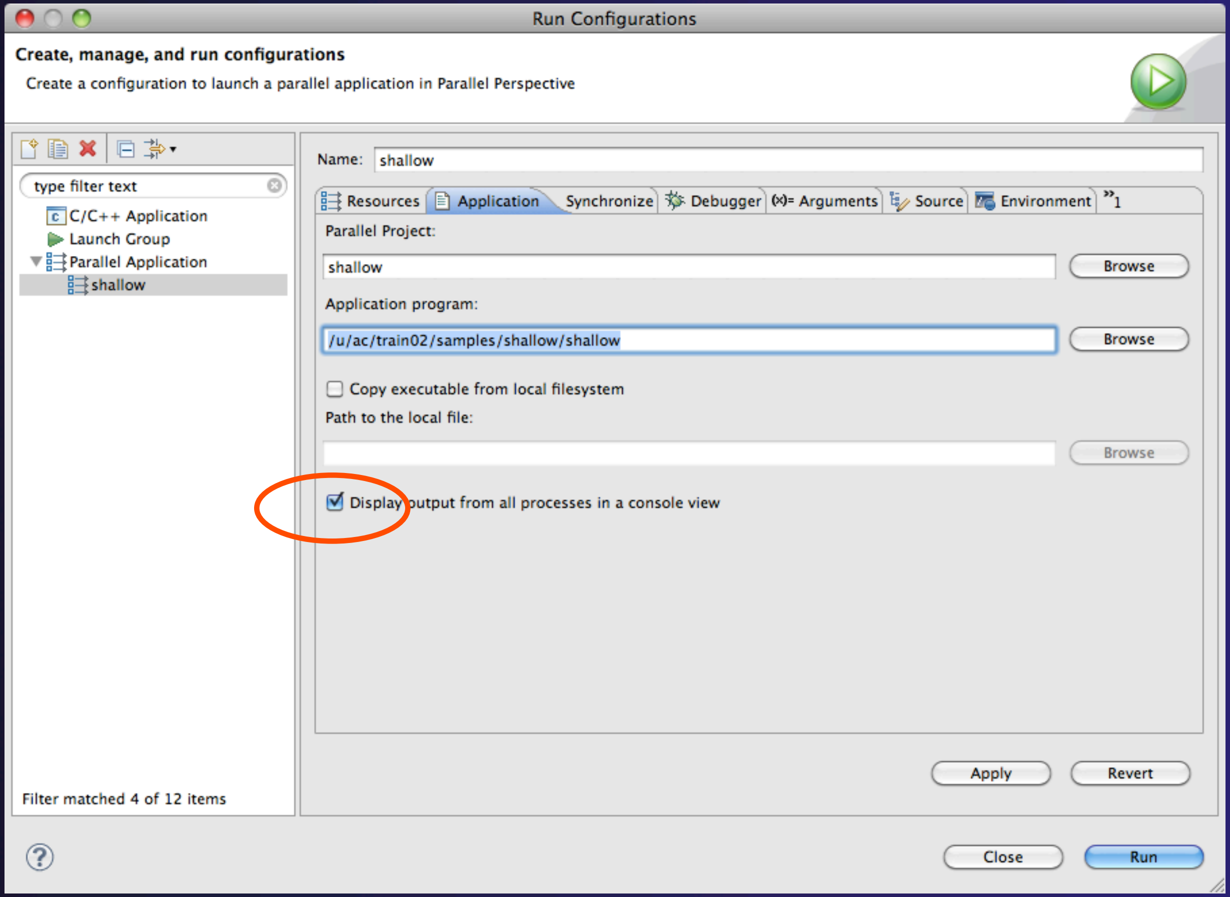
Host file:

Host list:

Filter matched 4 of 12 items

Complete the Application Tab

- Select the **Application** tab
- Choose the **Application program** (executable) by clicking the **Browse** button
- Navigate to the executable location (in this case on the remote machine)
- **Display combined output in a console view** will show program output in a console view


Run Configurations

Create, manage, and run configurations

Create a configuration to launch a parallel application in Parallel Perspective

type filter text

- C/C++ Application
- Launch Group
- Parallel Application
  - shallow

Filter matched 4 of 12 items

Name: shallow

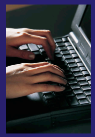
Parallel Project: shallow

Application program: /u/ac/train02/samples/shallow/shallow

☐ Copy executable from local filesystem

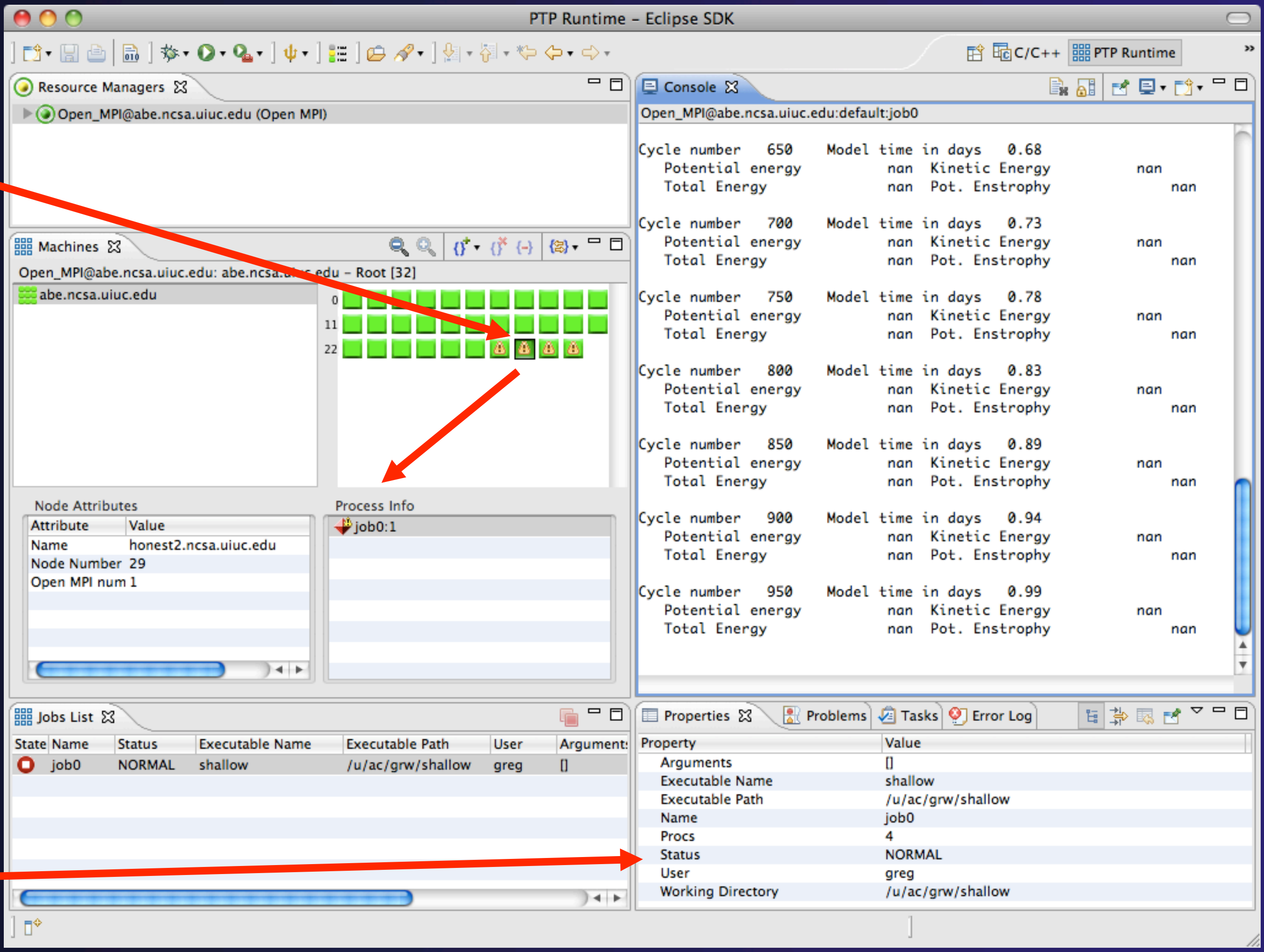
Path to the local file:

☑ Display output from all processes in a console view



Viewing The Run

- Double-click a node in machines view to see which processes ran on the node
- Hover over a process for tooltip popup
- Job status and information

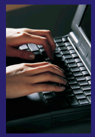


Attribute	Value
Name	honest2.ncsa.uiuc.edu
Node Number	29
Open MPI num	1

```
Open_MPI@abe.ncsa.uiuc.edu:default:job0

Cycle number    650    Model time in days   0.68
   Potential energy          nan  Kinetic Energy           nan
   Total Energy              nan  Pot. Enstrophy                nan

Cycle number    700    Model time in days   0.73
   Potential energy          nan  Kinetic Energy           nan
   Total Energy              nan  Pot. Enstrophy                nan

Cycle number    750    Model time in days   0.78
   Potential energy          nan  Kinetic Energy           nan
   Total Energy              nan  Pot. Enstrophy                nan

Cycle number    800    Model time in days   0.83
   Potential energy          nan  Kinetic Energy           nan
   Total Energy              nan  Pot. Enstrophy                nan

Cycle number    850    Model time in days   0.89
   Potential energy          nan  Kinetic Energy           nan
   Total Energy              nan  Pot. Enstrophy                nan

Cycle number    900    Model time in days   0.94
   Potential energy          nan  Kinetic Energy           nan
   Total Energy              nan  Pot. Enstrophy                nan

Cycle number    950    Model time in days   0.99
   Potential energy          nan  Kinetic Energy           nan
   Total Energy              nan  Pot. Enstrophy                nan
```

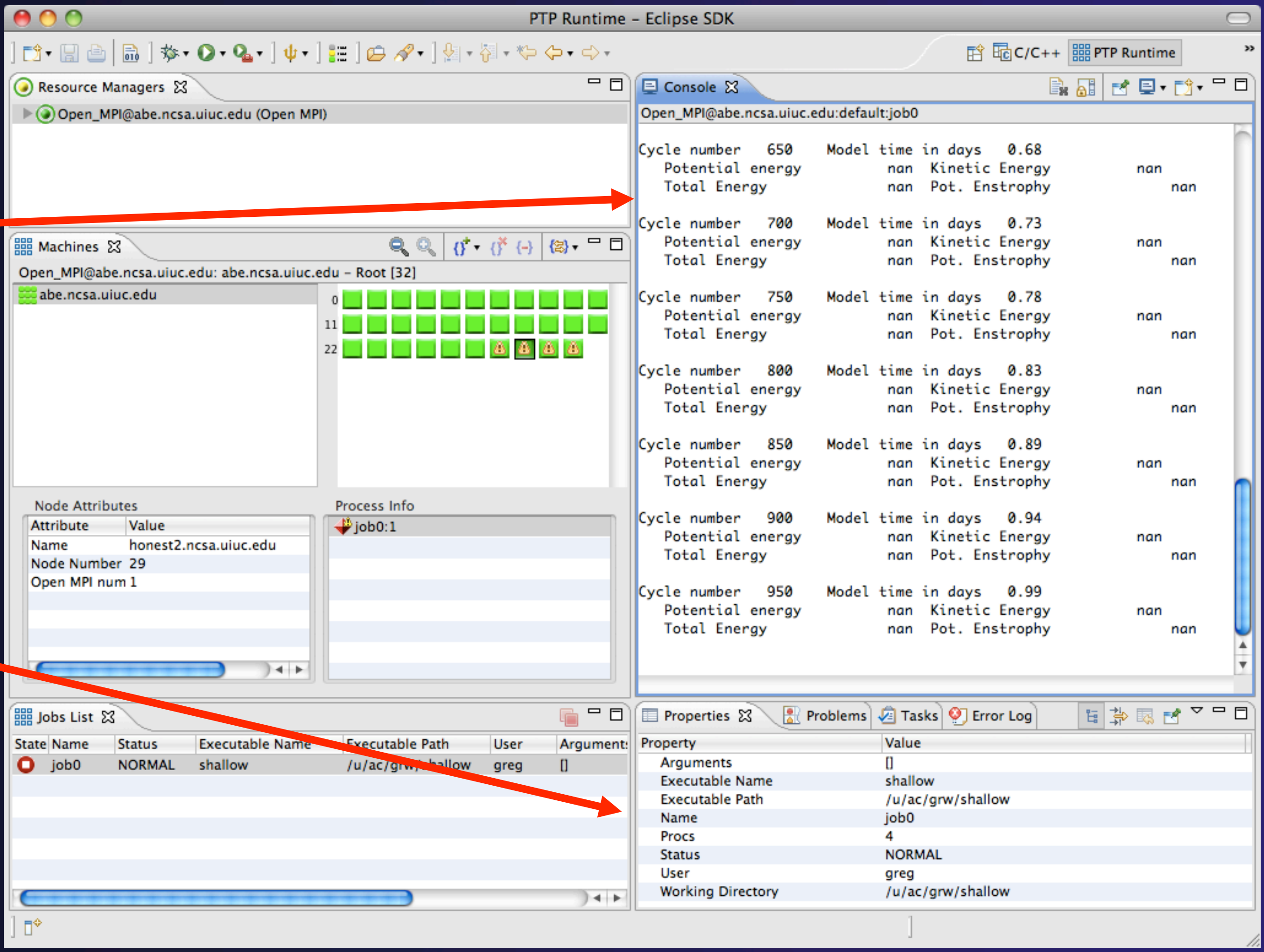
State	Name	Status	Executable Name	Executable Path	User	Arguments
●	job0	NORMAL	shallow	/u/ac/grw/shallow	greg	[]

Property	Value
Arguments	[]
Executable Name	shallow
Executable Path	/u/ac/grw/shallow
Name	job0
Procs	4
Status	NORMAL
User	greg
Working Directory	/u/ac/grw/shallow



Viewing Program Output

- Console displays combined output from all processes
- Properties view shows job details


PTP Runtime – Eclipse SDK
C/C++
PTP Runtime
Resource Managers
Open_MPI@abe.ncsa.uiuc.edu (Open MPI)
Machines
Open_MPI@abe.ncsa.uiuc.edu: abe.ncsa.uiuc.edu – Root [32]
abe.ncsa.uiuc.edu
0
11
22
Console
Problems
Tasks
Error Log

Open_MPI@abe.ncsa.uiuc.edu:default:job0

```
Cycle number   650    Model time in days   0.68
   Potential energy          nan  Kinetic Energy           nan
   Total Energy              nan  Pot. Enstrophy               nan

Cycle number   700    Model time in days   0.73
   Potential energy          nan  Kinetic Energy           nan
   Total Energy              nan  Pot. Enstrophy               nan

Cycle number   750    Model time in days   0.78
   Potential energy          nan  Kinetic Energy           nan
   Total Energy              nan  Pot. Enstrophy               nan

Cycle number   800    Model time in days   0.83
   Potential energy          nan  Kinetic Energy           nan
   Total Energy              nan  Pot. Enstrophy               nan

Cycle number   850    Model time in days   0.89
   Potential energy          nan  Kinetic Energy           nan
   Total Energy              nan  Pot. Enstrophy               nan

Cycle number   900    Model time in days   0.94
   Potential energy          nan  Kinetic Energy           nan
   Total Energy              nan  Pot. Enstrophy               nan

Cycle number   950    Model time in days   0.99
   Potential energy          nan  Kinetic Energy           nan
   Total Energy              nan  Pot. Enstrophy               nan
```

Node Attributes

Attribute	Value
Name	honest2.ncsa.uiuc.edu
Node Number	29
Open MPI num	1

Process Info

job0:1

Jobs List

State	Name	Status	Executable Name	Executable Path	User	Arguments
●	job0	NORMAL	shallow	/u/ac/grw/shallow	greg	[]

Properties

Property	Value
Arguments	[]
Executable Name	shallow
Executable Path	/u/ac/grw/shallow
Name	job0
Procs	4
Status	NORMAL
User	greg
Working Directory	/u/ac/grw/shallow

Remote Projects (RDT)

- Source located on remote machine
- Local Eclipse installation is used for:
  - Editing
  - Building
  - Running
  - Debugging
- Source indexing is performed on remote machine
  - Enables call hierarchy, type hierarchy, include browser, search, outline view, and more…
- Builds are performed on remote machine
  - Supports both managed and unmanaged projects
- Application is run and debugged remotely using the PTP resource managers

Remote Project Demo…

- Create a remote project from existing source
- Show editing remote files
- Show outline and include browser
- Show remote build

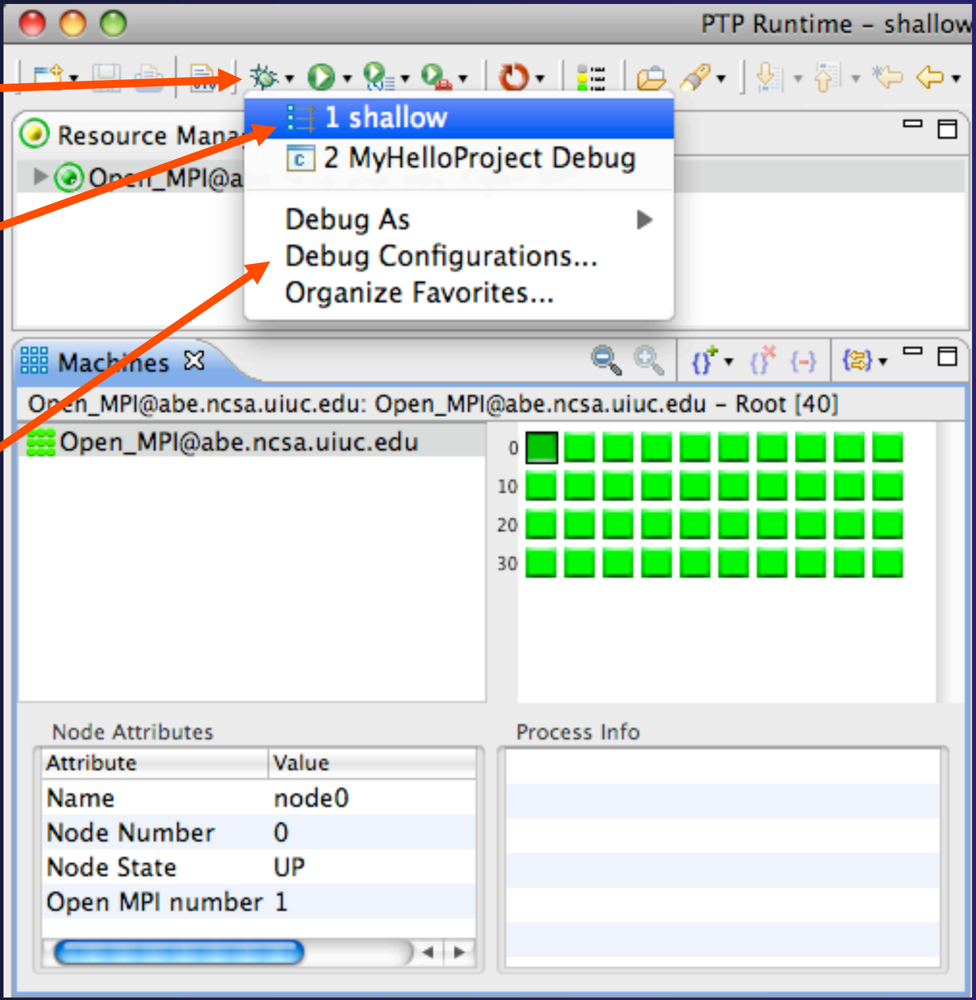
- Limitations:
  - Can’t be used with CVS
  - Only supports fully remote source
  - Partial remote (synchronize) may be added in the future

Module 5: Parallel Debugging

- Objective
  - Learn the basics of debugging parallel programs with PTP
- Contents
  - Launching a parallel debug session
  - The PTP Debug Perspective
  - Controlling sets of processes
  - Controlling individual processes
  - Parallel Breakpoints
  - Terminating processes

Launching A Debug Session

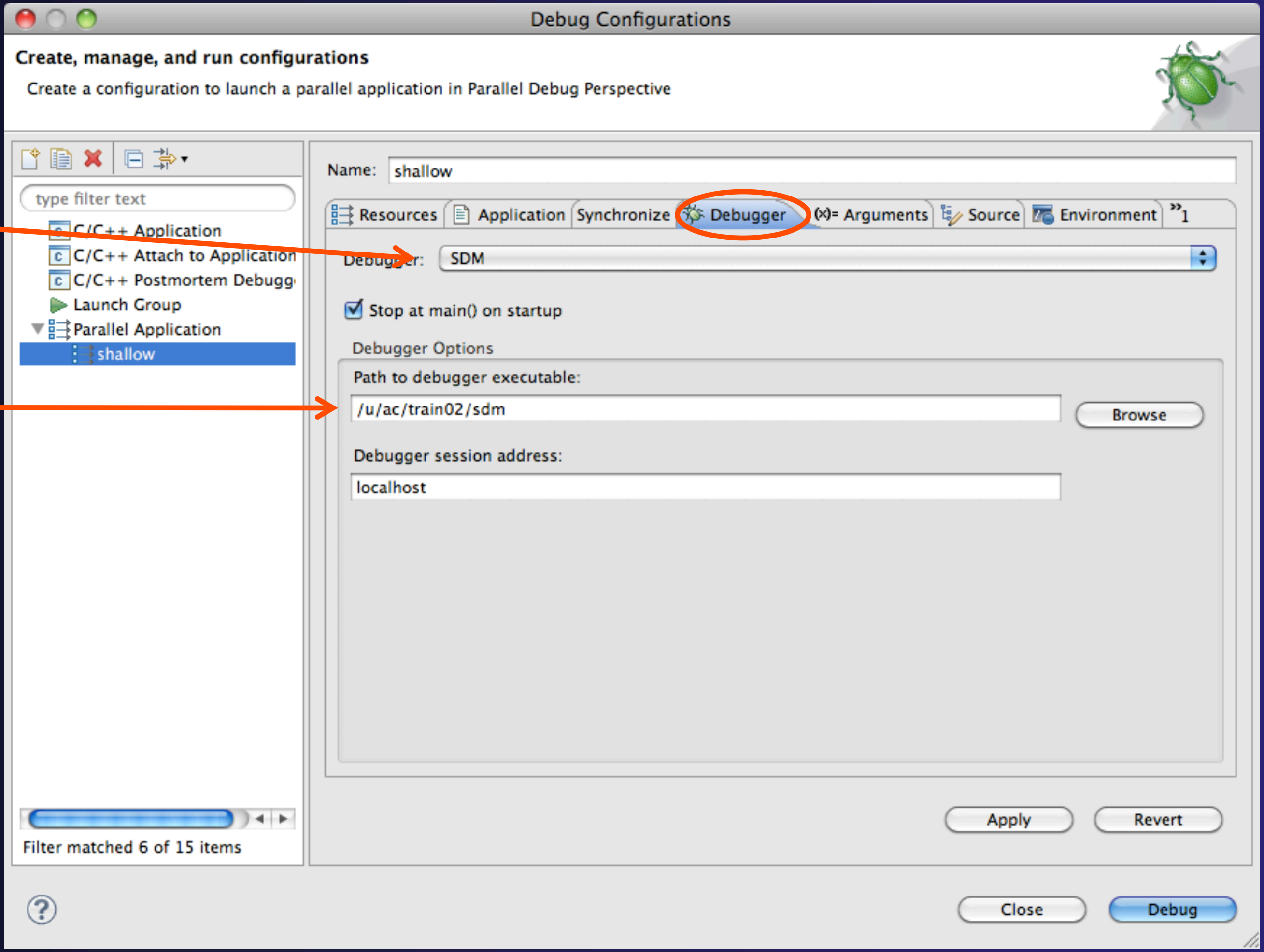
- Use the drop-down next to the debug button (bug icon) instead of run button
- Select the project to launch
- The debug launch will use the same number of processes that the normal launch used
- First, select **Debug Configurations…** to verify the debugger settings



Attribute	Value
Name	node0
Node Number	0
Node State	UP
Open MPI number	1

Verify the Debugger Tab

- Select **Debugger** tab
- Make sure **SDM** is selected in the **Debugger** dropdown
- Use the **Browse** button to select the debugger executable if required
  - If launching remotely, the debugger executable must also be located remotely
- Debugger session address should not need to be changed
- Click on **Debug** to launch the program



Debug Configurations

Create, manage, and run configurations

Create a configuration to launch a parallel application in Parallel Debug Perspective

Name: shallow

Debugger: SDM

☑ Stop at main() on startup

Debugger Options

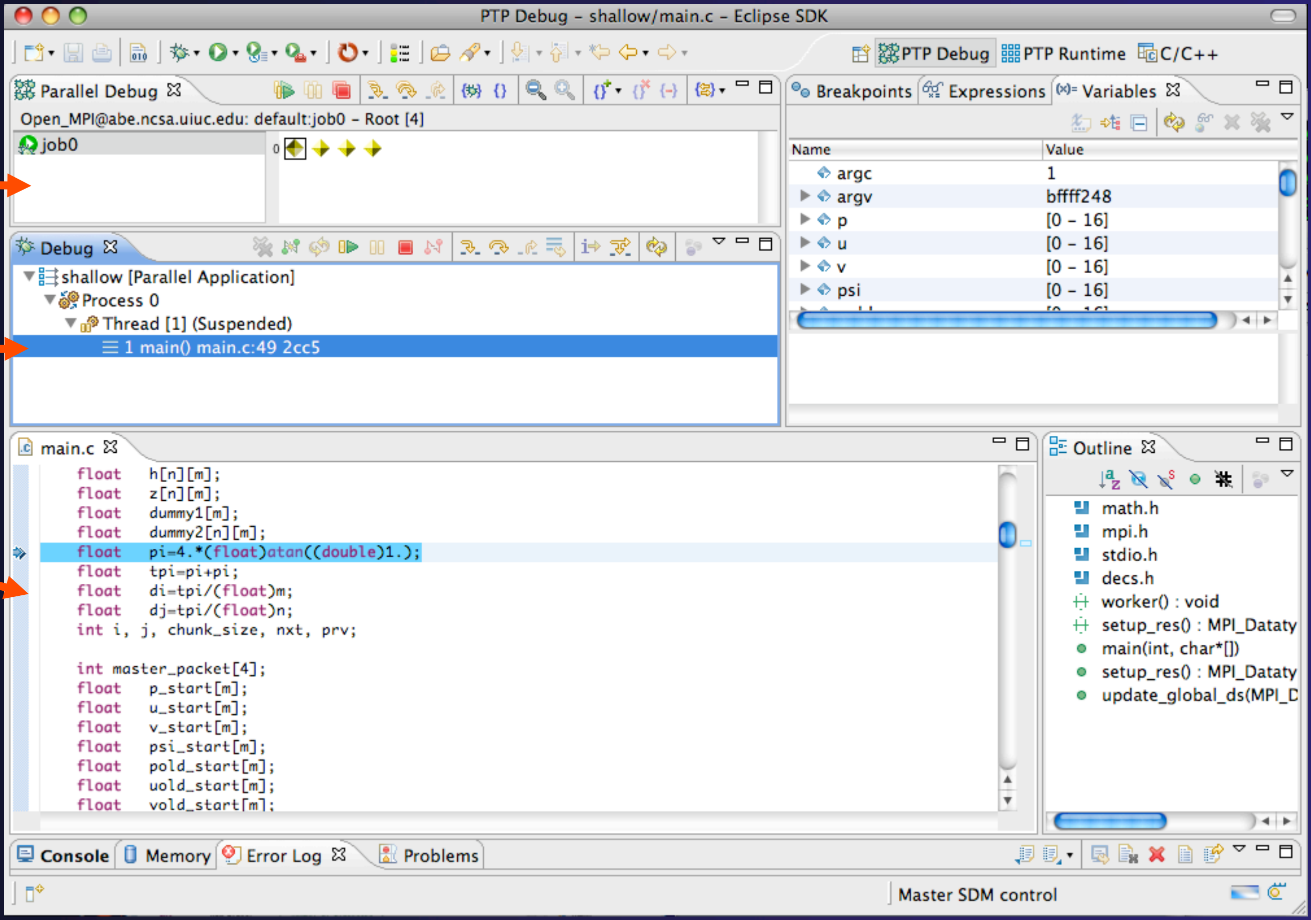
Path to debugger executable: /u/ac/train02/sdm

Debugger session address: localhost

Filter matched 6 of 15 items

The PTP Debug Perspective (1)

- **Parallel Debug view** shows job and processes being debugged
- **Debug** view shows threads and call stack for individual processes
- **Source** view shows a **current line marker** for all processes



Open_MPI@abe.ncsa.uiuc.edu: default:job0 – Root [4]

job0

```
shallow [Parallel Application]
  Process 0
    Thread [1] (Suspended)
      1 main() main.c:49 2cc5
```

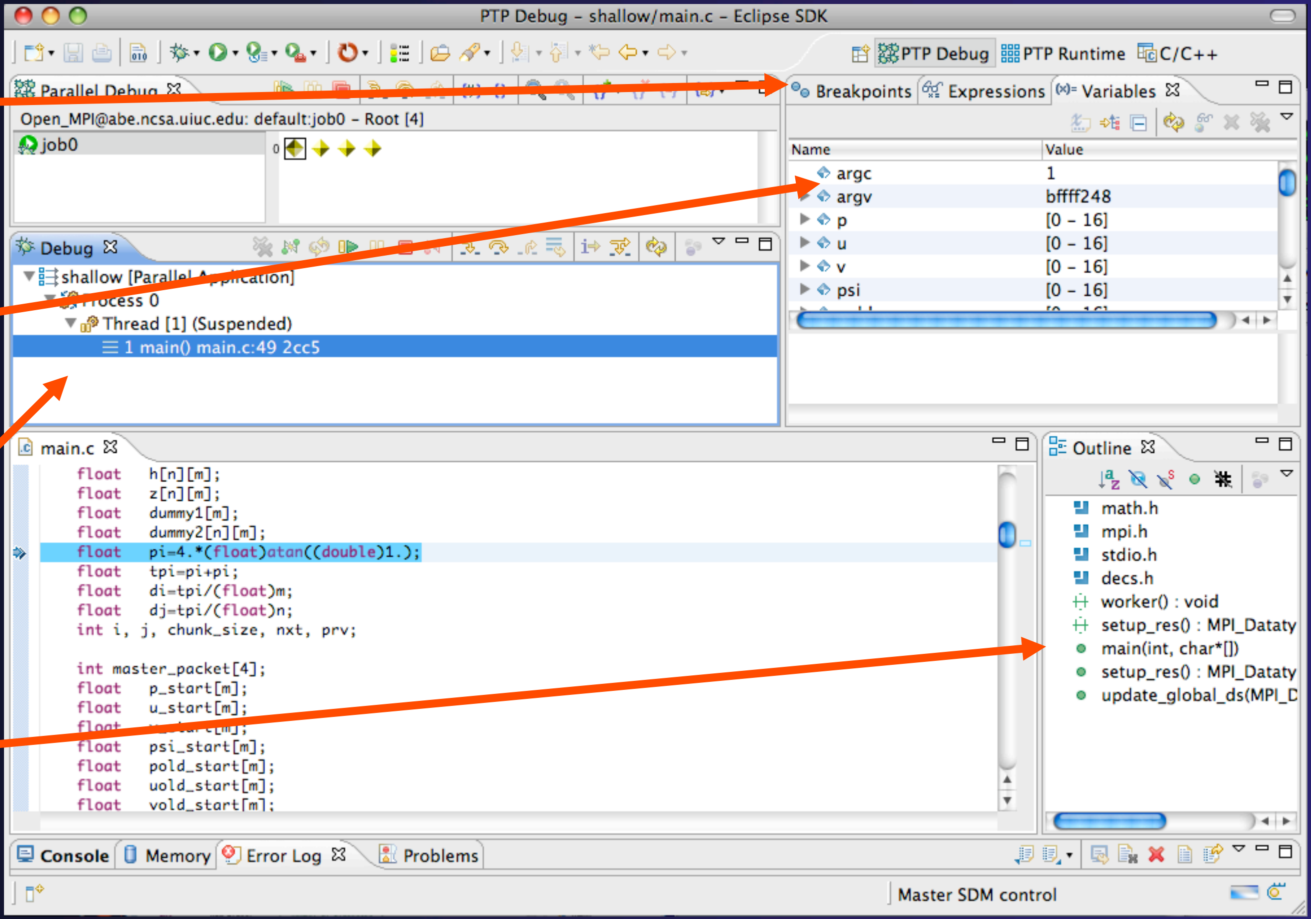
Name	Value
argc	1
argv	bffff248
p	[0 – 16]
u	[0 – 16]
v	[0 – 16]
psi	[0 – 16]

```
    float   h[n][m];
    float   z[n][m];
    float   dummy1[m];
    float   dummy2[n][m];
    float   pi=4.*(float)atan((double)1.);
    float   tpi=pi+pi;
    float   di=tpi/(float)m;
    float   dj=tpi/(float)n;
    int i, j, chunk_size, nxt, prv;

    int master_packet[4];
    float   p_start[m];
    float   u_start[m];
    float   v_start[m];
    float   psi_start[m];
    float   pold_start[m];
    float   uold_start[m];
    float   vold_start[m];
```

```
math.h
mpi.h
stdio.h
decs.h
worker() : void
setup_res() : MPI_Dataty
main(int, char*[])
setup_res() : MPI_Dataty
update_global_ds(MPI_D
```

The PTP Debug Perspective (2)

- **Breakpoints** view shows breakpoints that have been set (more on this later)
- **Variables** view shows the current values of variables for the currently selected process in the **Debug** view
- **Outline** view (from CDT) of source code



Open_MPI@abe.ncsa.uiuc.edu: default:job0 – Root [4]

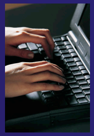
job0

Name	Value
argc	1
argv	bffff248
p	[0 – 16]
u	[0 – 16]
v	[0 – 16]
psi	[0 – 16]

```
shallow [Parallel Application]
  Process 0
    Thread [1] (Suspended)
      1 main() main.c:49 2cc5
```

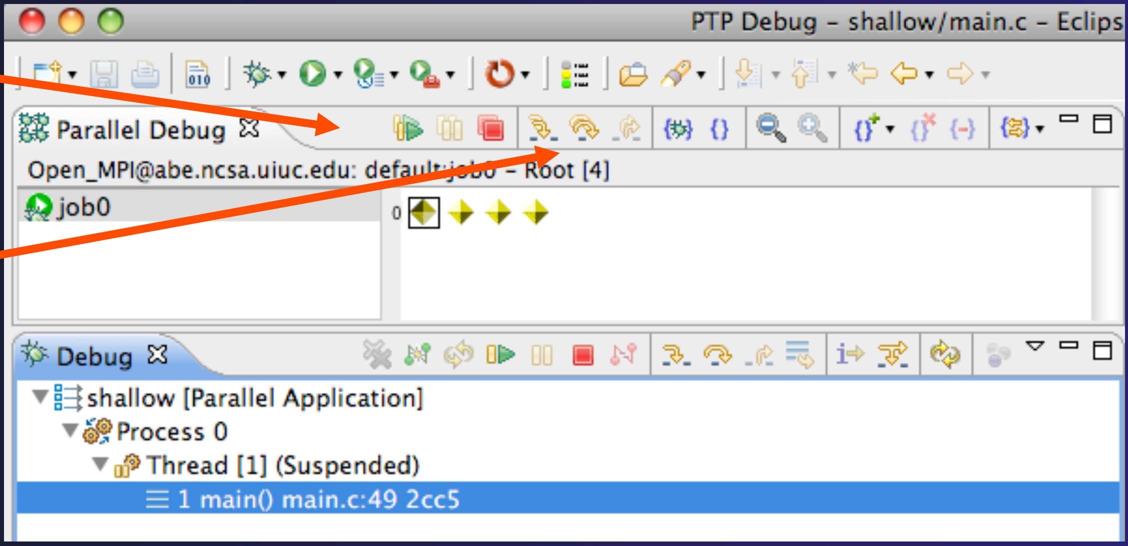
```
float   h[n][m];
float   z[n][m];
float   dummy1[m];
float   dummy2[n][m];
float   pi=4.*(float)atan((double)1.);
float   tpi=pi+pi;
float   di=tpi/(float)m;
float   dj=tpi/(float)n;
int i, j, chunk_size, nxt, prv;

int master_packet[4];
float   p_start[m];
float   u_start[m];
float   v_start[m];
float   psi_start[m];
float   pold_start[m];
float   uold_start[m];
float   vold_start[m];
```

```
math.h
mpi.h
stdio.h
decs.h
worker() : void
setup_res() : MPI_Dataty
main(int, char*[])
setup_res() : MPI_Dataty
update_global_ds(MPI_D
```



Stepping All Processes

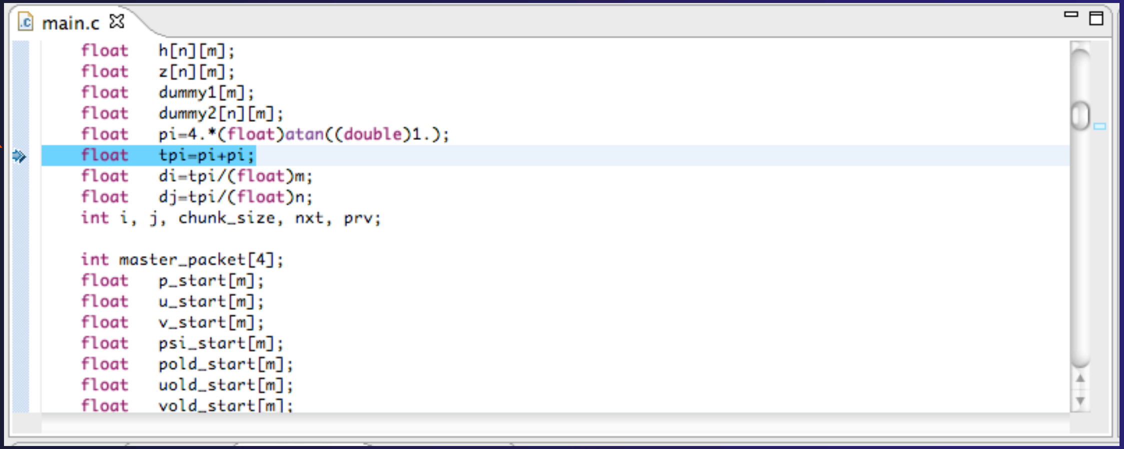
- The buttons in the **Parallel Debug View** control groups of processes
- Click on the **Step Over** button
- Observe that all process icons change to green, then back to yellow
- Notice that the current line marker has moved to the next source line



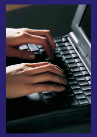
PTP Debug – shallow/main.c – Eclips

Open_MPI@abe.ncsa.uiuc.edu: default:job0 – Root [4]

```
▼ shallow [Parallel Application]
  ▼ Process 0
    ▼ Thread [1] (Suspended)
        1 main() main.c:49 2cc5
```

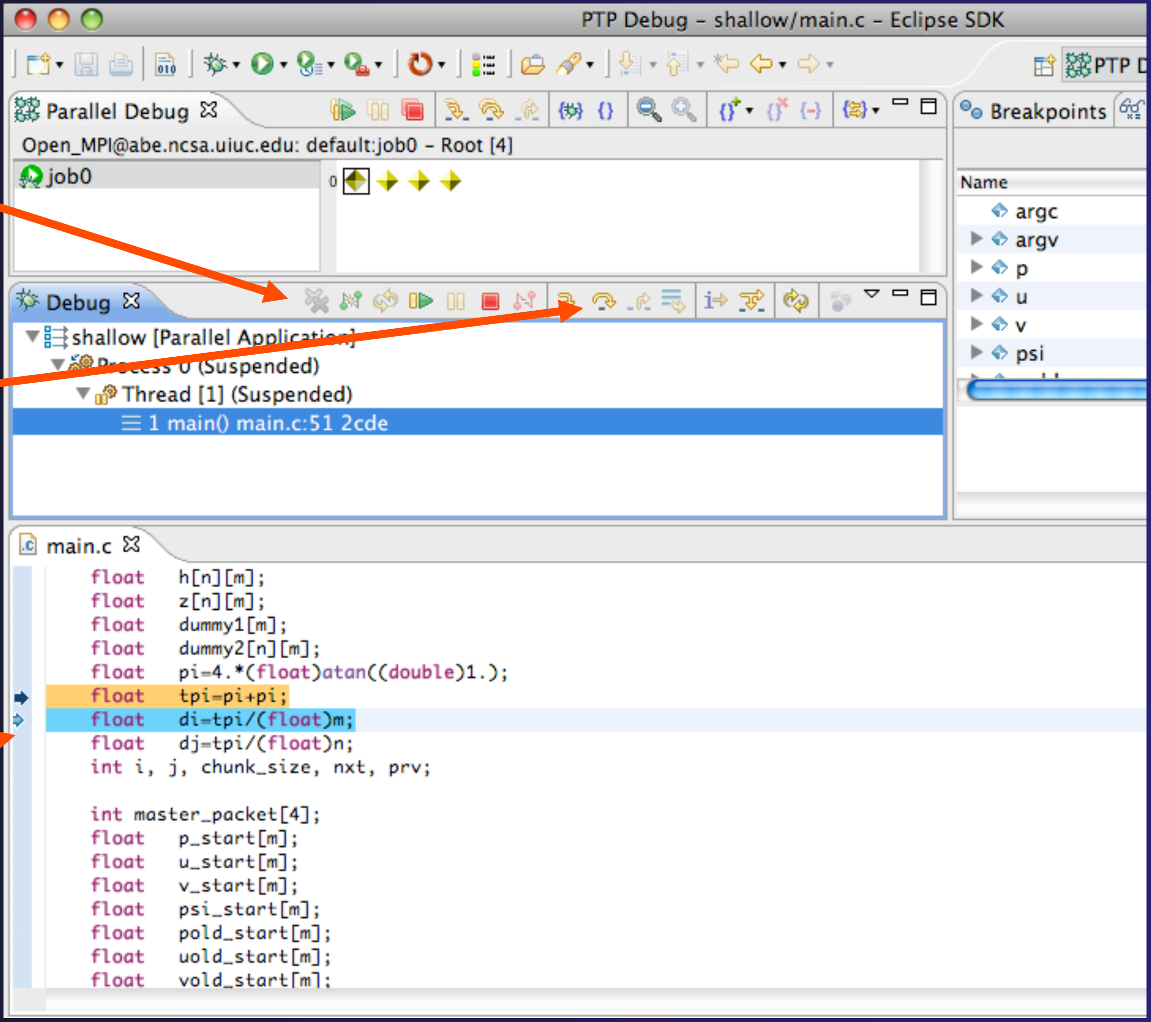


```
    float    h[n][m];
    float    z[n][m];
    float    dummy1[m];
    float    dummy2[n][m];
    float    pi=4.*(float)atan((double)1.);
    float    tpi=pi+pi;
    float    di=tpi/(float)m;
    float    dj=tpi/(float)n;
    int i, j, chunk_size, nxt, prv;

    int master_packet[4];
    float    p_start[m];
    float    u_start[m];
    float    v_start[m];
    float    psi_start[m];
    float    pold_start[m];
    float    uold_start[m];
    float    vold_start[m];
```



Stepping An Individual Process

- The buttons in the **Debug view** are used to control an individual process, in this case process 0
- Click the **Step Over** button
- You will now see two current line markers, the first shows the position of process 0, the second shows the positions of processes 1-3



```
Open_MPI@abe.ncsa.uiuc.edu: default:job0 – Root [4]
job0
0
```

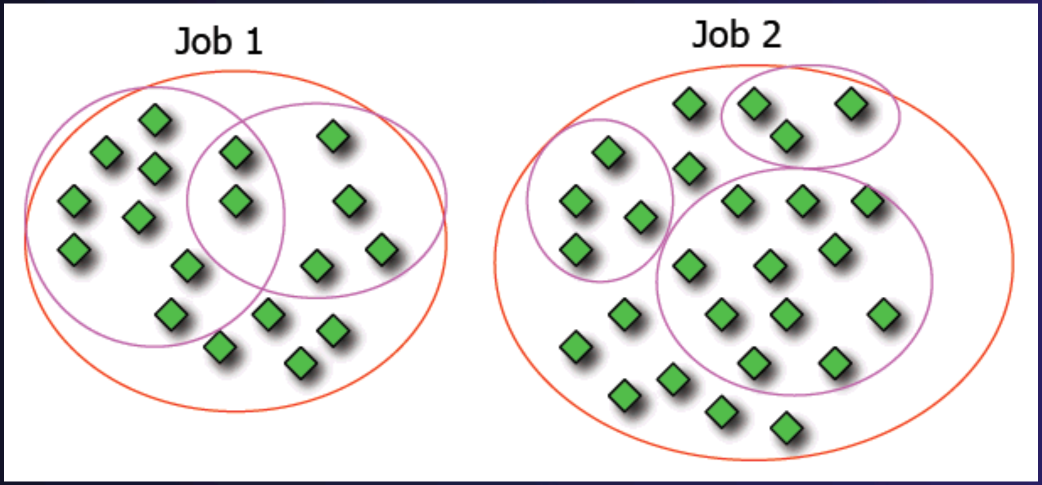
```
shallow [Parallel Application]
  Process 0 (Suspended)
    Thread [1] (Suspended)
      1 main() main.c:51 2cde
```

```
Name
  argc
  argv
  p
  u
  v
  psi
```

```
    float   h[n][m];
    float   z[n][m];
    float   dummy1[m];
    float   dummy2[n][m];
    float   pi=4.*(float)atan((double)1.);
    float   tpi=pi+pi;
    float   di=tpi/(float)m;
    float   dj=tpi/(float)n;
    int i, j, chunk_size, nxt, prv;

    int master_packet[4];
    float   p_start[m];
    float   u_start[m];
    float   v_start[m];
    float   psi_start[m];
    float   pold_start[m];
    float   uold_start[m];
    float   vold_start[m];
```

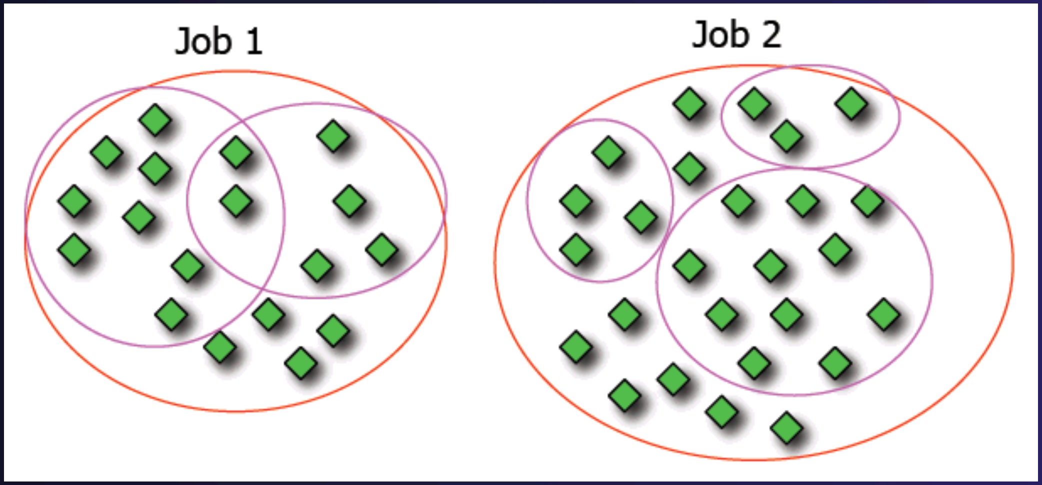
Process Sets (1)

- Traditional debuggers apply operations to a single process
- Parallel debugging operations apply to a single process or to arbitrary collections of processes
- A process set is a means of simultaneously referring to one or more processes



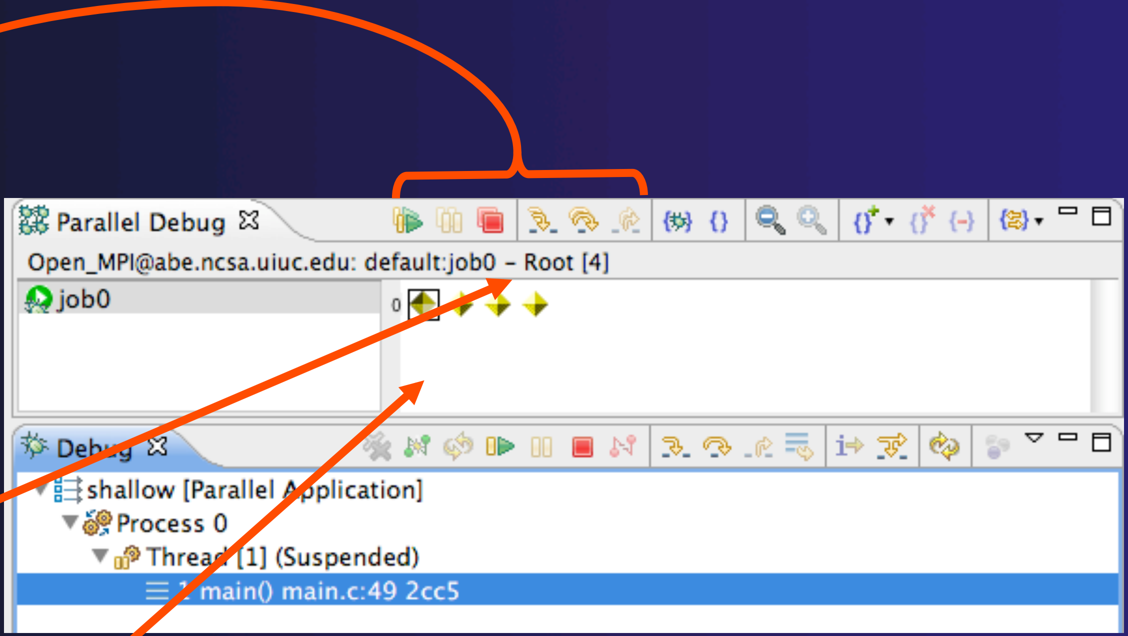
Process Sets (2)

- When a parallel debug session is first started, all processes are placed in a set, called the **Root** set
- Sets are always associated with a single job
- A job can have any number of process sets
- A set can contain from 1 to the number of processes in a job



Operations On Process Sets

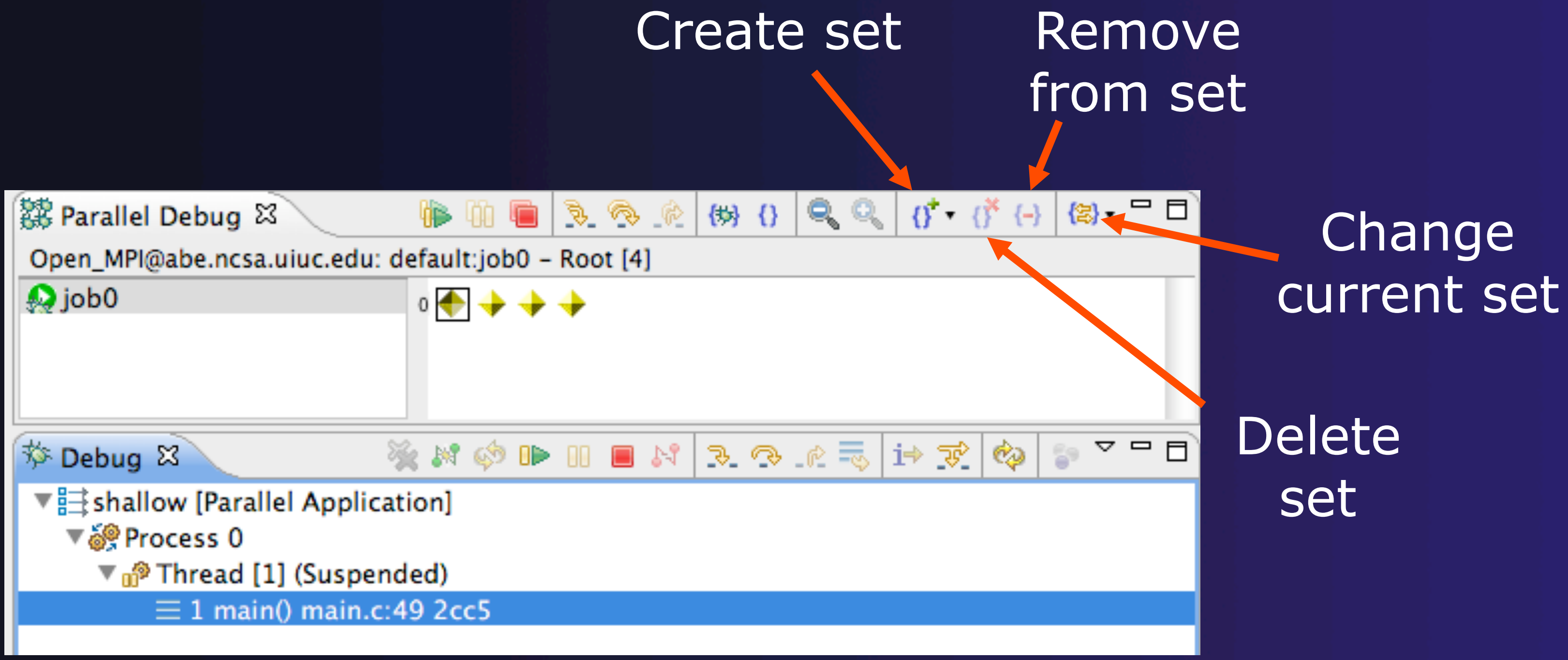
- Debug operations on the **Parallel Debug view** toolbar always apply to the current set:
  - Resume, suspend, stop, step into, step over, step return
- The current process set is listed next to job name along with number of processes in the set
- The processes in process set are visible in right hand part of the view


Parallel Debug
Open_MPI@abe.ncsa.uiuc.edu: default:job0 - Root [4]
job0
0
Debug
shallow [Parallel Application]
Process 0
Thread [1] (Suspended)
1 main() main.c:49 2cc5

Root set = all processes

Managing Process Sets

✦ The remaining icons in the toolbar of the **Parallel Debug view** allow you to create, modify, and delete process sets, and to change the current process set



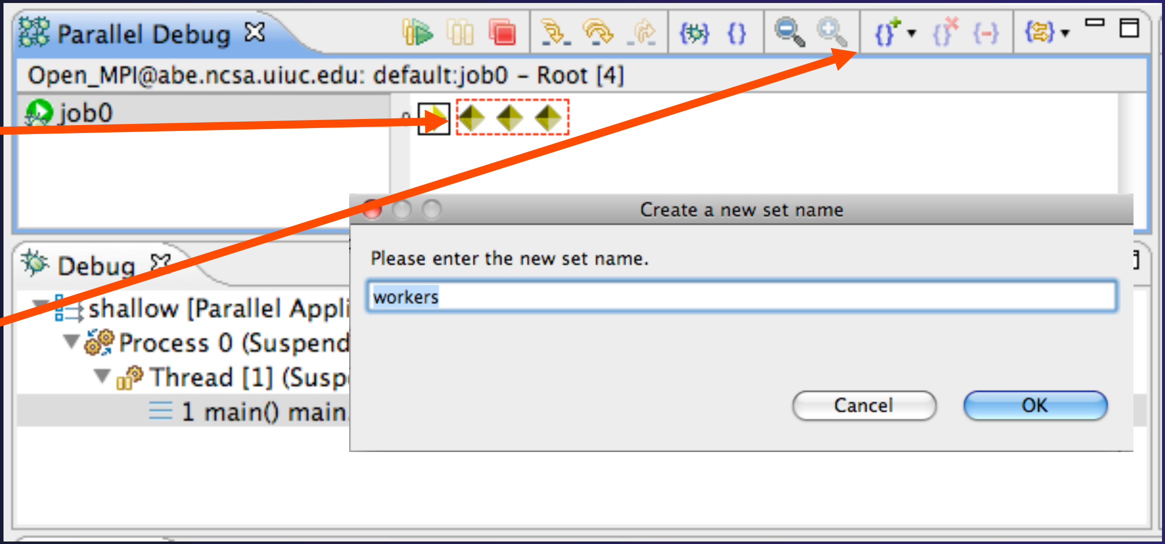
Open_MPI@abe.ncsa.uiuc.edu: default:job0 – Root [4]

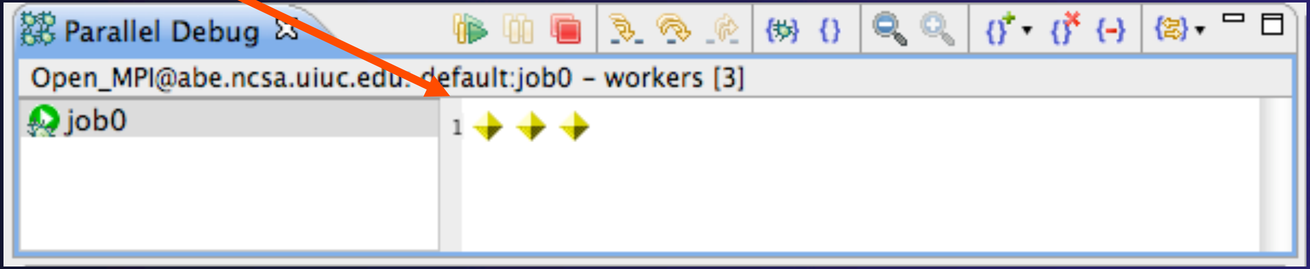
job0

shallow [Parallel Application]
Process 0
Thread [1] (Suspended)
1 main() main.c:49 2cc5

Creating A New Process Set

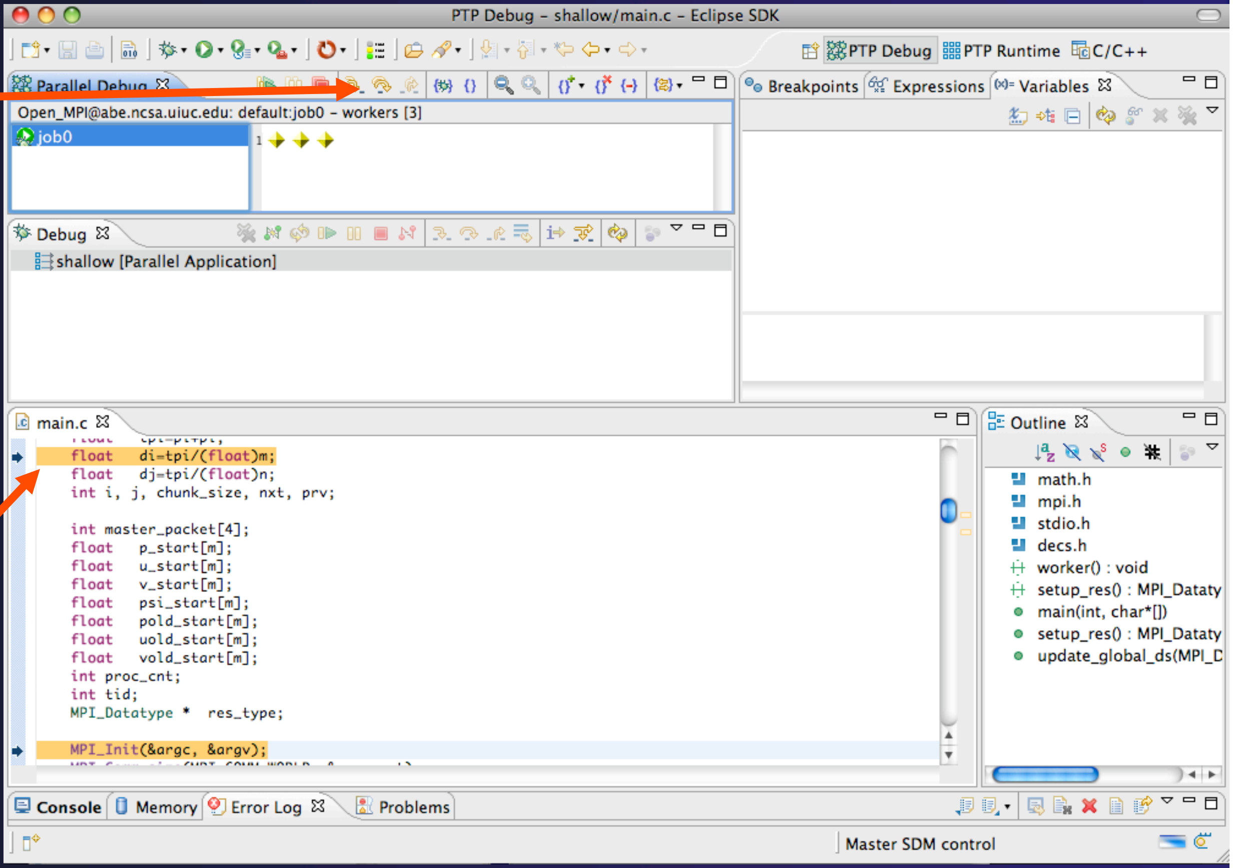
- Select the processes you want in the set by clicking and dragging, in this case, the last three
- Click on the **Create Set** button
- Enter a name for the set, in this case **workers**, and click **OK**
- You will see the view change to display only the selected processes





Stepping Using New Process Set

- With the **workers** set active, click the **Step Over** button
- You will see only the first current line marker move
- Step a couple more times
- You should see two line markers, one for the single master process, and one for the 3 worker processes



PTP Debug - shallow/main.c - Eclipse SDK

Open_MPI@abe.ncsa.uiuc.edu: default:job0 - workers [3]

job0

shallow [Parallel Application]

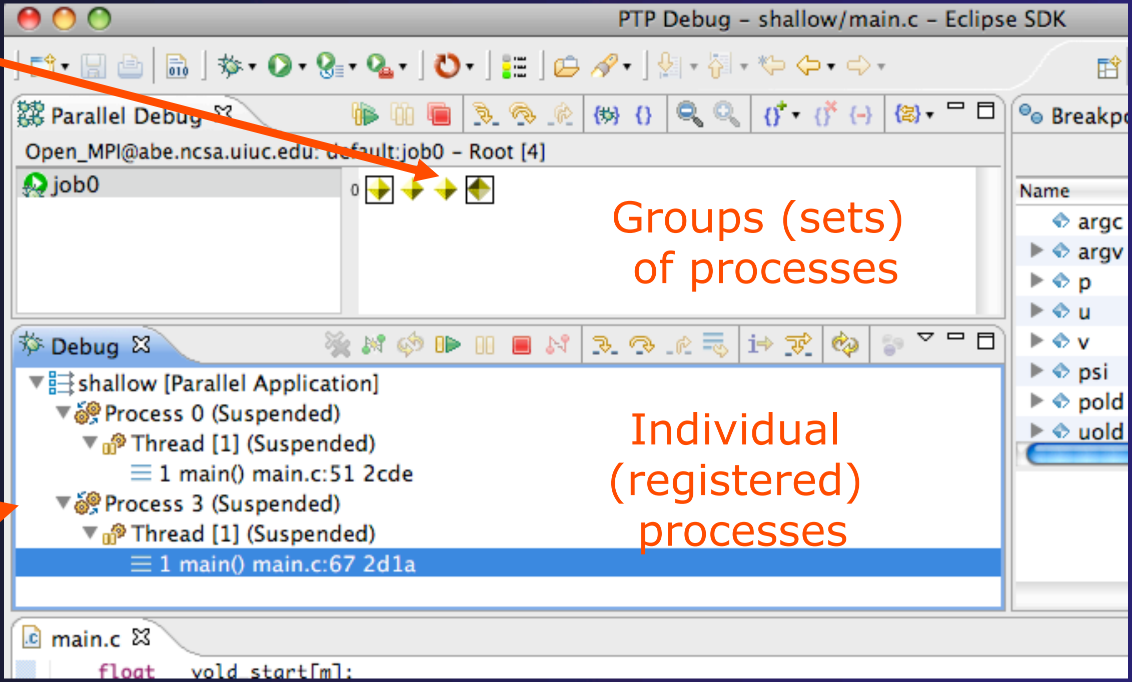
```
    float   di=tpi/(float)m;
    float   dj=tpi/(float)n;
    int i, j, chunk_size, nxt, prv;

    int master_packet[4];
    float   p_start[m];
    float   u_start[m];
    float   v_start[m];
    float   psi_start[m];
    float   pold_start[m];
    float   uold_start[m];
    float   vold_start[m];
    int proc_cnt;
    int tid;
    MPI_Datatype *  res_type;

    MPI_Init(&argc, &argv);
```

Process Registration

- Process set commands apply to groups of processes
- For finer control and more detailed information, a process can be registered and isolated in the **Debug view**
- Registered processes, including their stack traces and threads, appear in the **Debug view**
- Any number of processes can be registered, and processes can be registered or un-registered at any time

Registering A Process

- To register a process, double-click its process icon in the **Parallel Debug view** or select a number of processes and click on the **register** button
- The process icon will be surrounded by a box and the process appears in the **Debug view**
- To un-register a process, double-click on the process icon or select a number of processes and click on the **unregister** button


PTP Debug – shallow/main.c – Eclipse SDK
Parallel Debug
Open_MPI@abe.ncsa.uiuc.edu: default:job0 – Root [4]
job0
0
Groups (sets) of processes
Debug
Individual (registered) processes
Breakpoints
Name
argc
argv
p
u
v
psi
pold
uold
main.c
float void start[m];

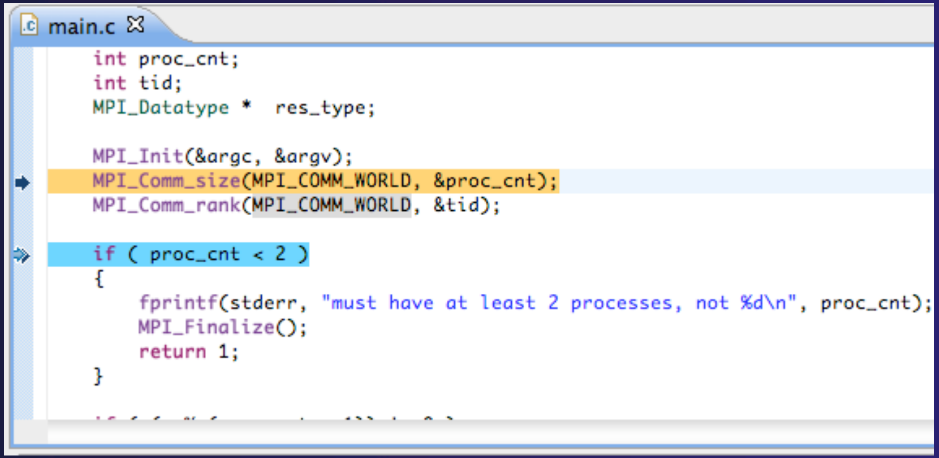
shallow [Parallel Application]
Process 0 (Suspended)
Thread [1] (Suspended)
1 main() main.c:51 2cde
Process 3 (Suspended)
Thread [1] (Suspended)
1 main() main.c:67 2d1a

Current Line Marker

- The current line marker is used to show the current location of suspended processes
- In traditional programs, there is a single current line marker (the exception to this is multi-threaded programs)
- In parallel programs, there is a current line marker for every process
- The PTP debugger shows one current line marker for every group of processes at the same location

Colors And Markers

- The highlight color depends on the processes suspended at that line:
  - **Blue:** All registered process(es)
  - **Orange:** All unregistered process(es)
  - **Green:** Registered or unregistered process with no source line (e.g. suspended in a library routine)
- The marker depends on the type of process stopped at that location
- Hover over marker for more details about the processes suspend at that location

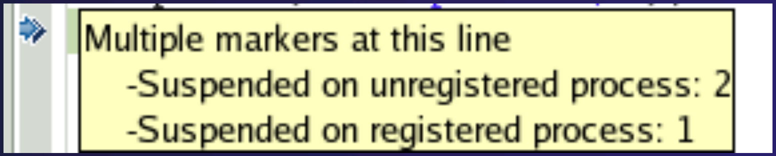


```
main.c
    int proc_cnt;
    int tid;
    MPI_Datatype *  res_type;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &proc_cnt);
    MPI_Comm_rank(MPI_COMM_WORLD, &tid);

    if ( proc_cnt < 2 )
    {
        fprintf(stderr, "must have at least 2 processes, not %d\n", proc_cnt);
        MPI_Finalize();
        return 1;
    }
```

Multiple processes marker

Registered process marker

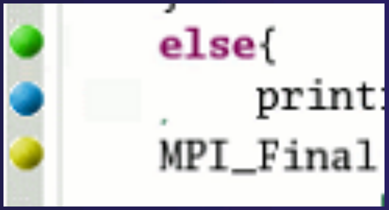
Un-registered process marker



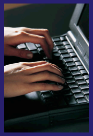
Multiple markers at this line
-Suspended on unregistered process: 2
-Suspended on registered process: 1

Breakpoints

- Apply only to processes in the particular set that is active in the **Parallel Debug view** when the breakpoint is created
- Breakpoints are colored depending on the active process set and the set the breakpoint applies to:
  - Green indicates the breakpoint set is the same as the active set.
  - Blue indicates some processes in the breakpoint set are also in the active set (i.e. the process sets overlap)
  - Yellow indicates the breakpoint set is different from the active set (i.e. the process sets are disjoint)
- When the job completes, the breakpoints are automatically removed

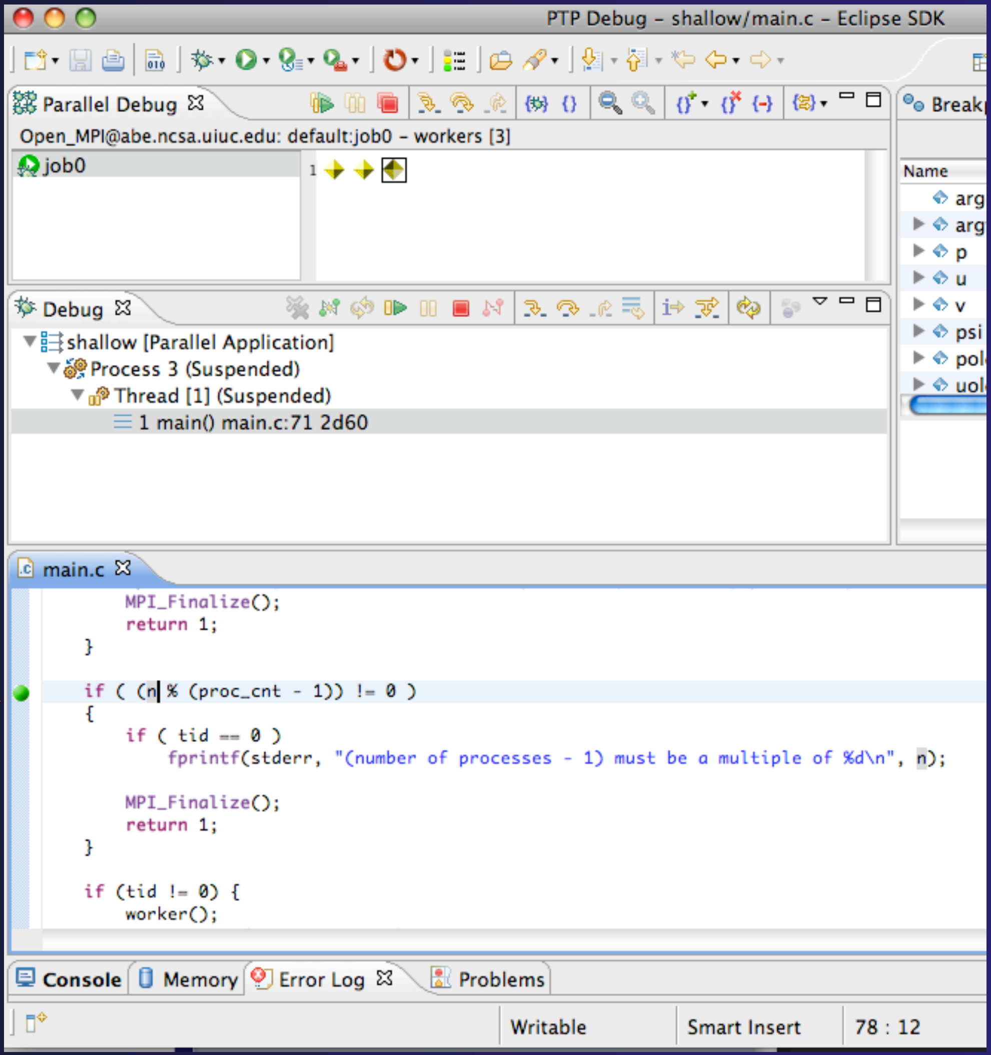


```
else{
    print
MPI_Final
```



Creating A Breakpoint

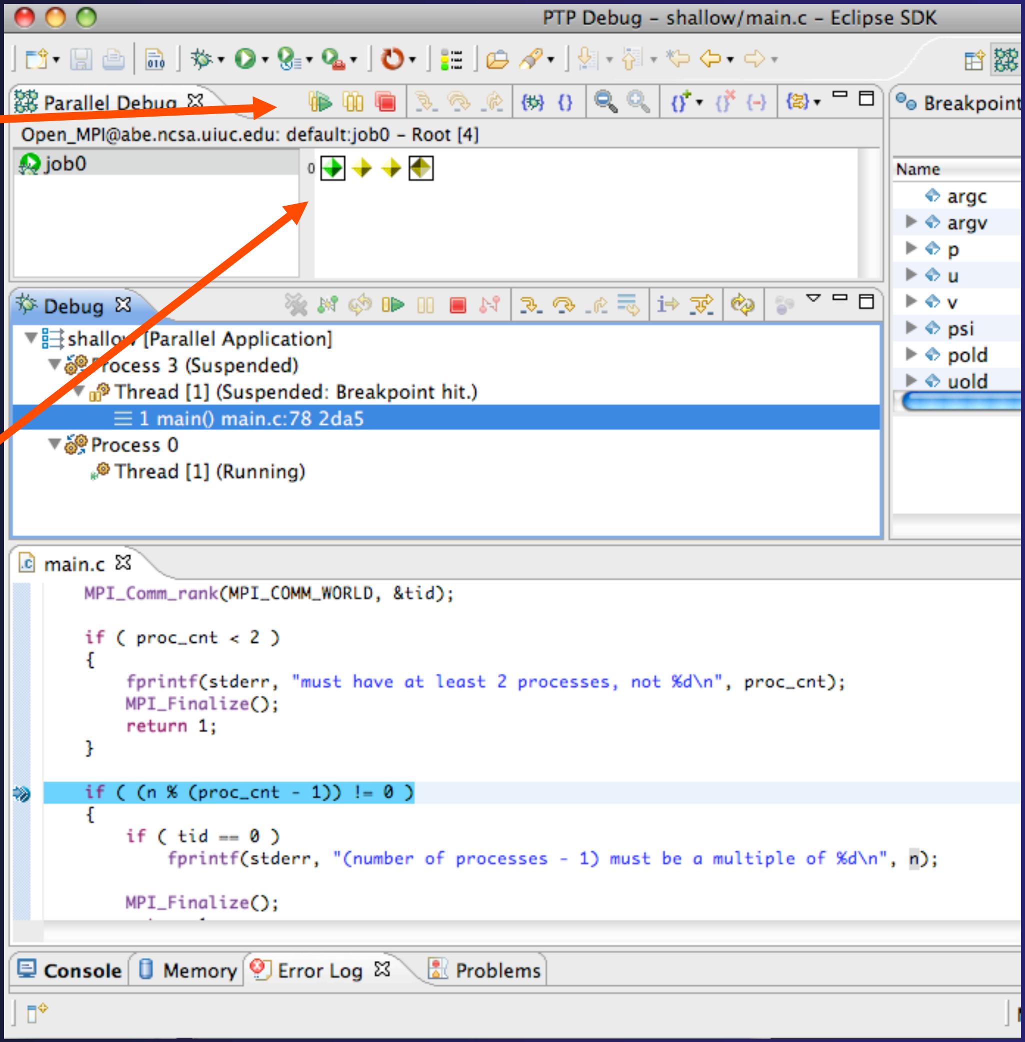
- Select the process set that the breakpoint should apply to, in this case, the **workers** set
- Double-click on the left edge of an editor window, at the line on which you want to set the breakpoint, or right click and use the **Parallel Breakpoint▶Toggle Breakpoint** context menu
- The breakpoint is displayed on the marker bar



```
        MPI_Finalize();
        return 1;
    }

    if ( (n % (proc_cnt - 1)) != 0 )
    {
        if ( tid == 0 )
            fprintf(stderr, "(number of processes - 1) must be a multiple of %d\n", n);

        MPI_Finalize();
        return 1;
    }

    if (tid != 0) {
        worker();
```

Hitting the Breakpoint

- Click on the **Resume** button in the **Parallel Debug view**
- In this example, the three worker processes have hit the breakpoint, as indicated by the yellow process icons and the current line marker
- Process 0 is still running as its icon is green
- Processes 1-3 are suspended on the breakpoint



Open_MPI@abe.ncsa.uiuc.edu: default:job0 – Root [4]

job0

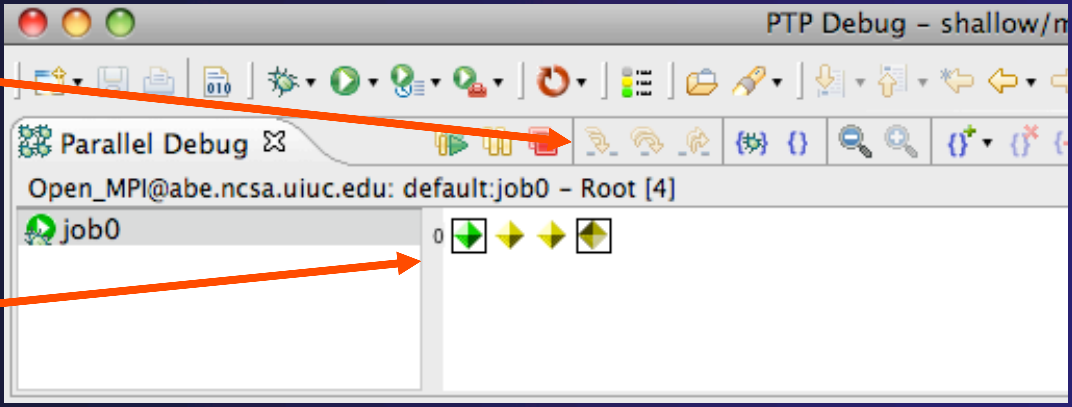
Name
argc
argv
p
u
v
psi
pold
uold

```
shallow [Parallel Application]
  Process 3 (Suspended)
    Thread [1] (Suspended: Breakpoint hit.)
      1 main() main.c:78 2da5
  Process 0
    Thread [1] (Running)
```

```
    MPI_Comm_rank(MPI_COMM_WORLD, &tid);

    if ( proc_cnt < 2 )
    {
        fprintf(stderr, "must have at least 2 processes, not %d\n", proc_cnt);
        MPI_Finalize();
        return 1;
    }

    if ( (n % (proc_cnt - 1)) != 0 )
    {
        if ( tid == 0 )
            fprintf(stderr, "(number of processes - 1) must be a multiple of %d\n", n);

        MPI_Finalize();
```

More On Stepping

- The **Step** buttons are only enabled when all processes in the active set are **suspended** (yellow icon)
- In this case, process 0 is still running

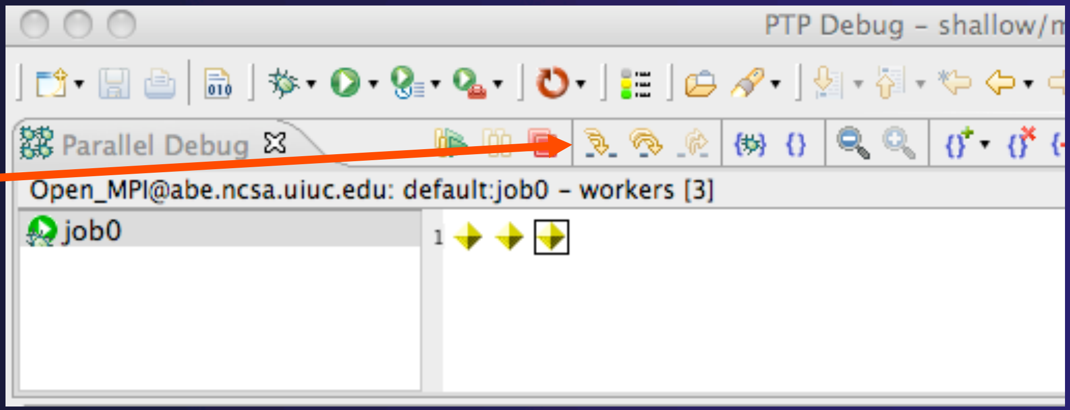

PTP Debug – shallow/m

Parallel Debug

Open_MPI@abe.ncsa.uiuc.edu: default:job0 – Root [4]

job0

- Switch to the set of suspended processes (the **workers** set)
- You will now see the **Step** buttons become enabled


PTP Debug – shallow/m

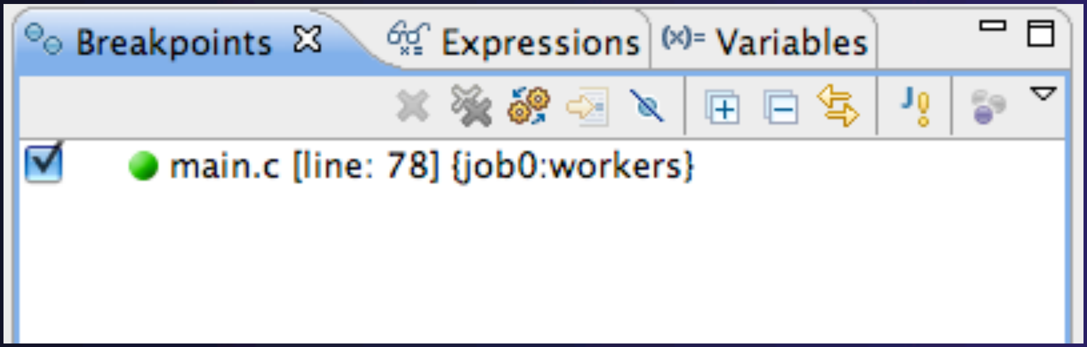
Parallel Debug

Open_MPI@abe.ncsa.uiuc.edu: default:job0 – workers [3]

job0

Breakpoint Information

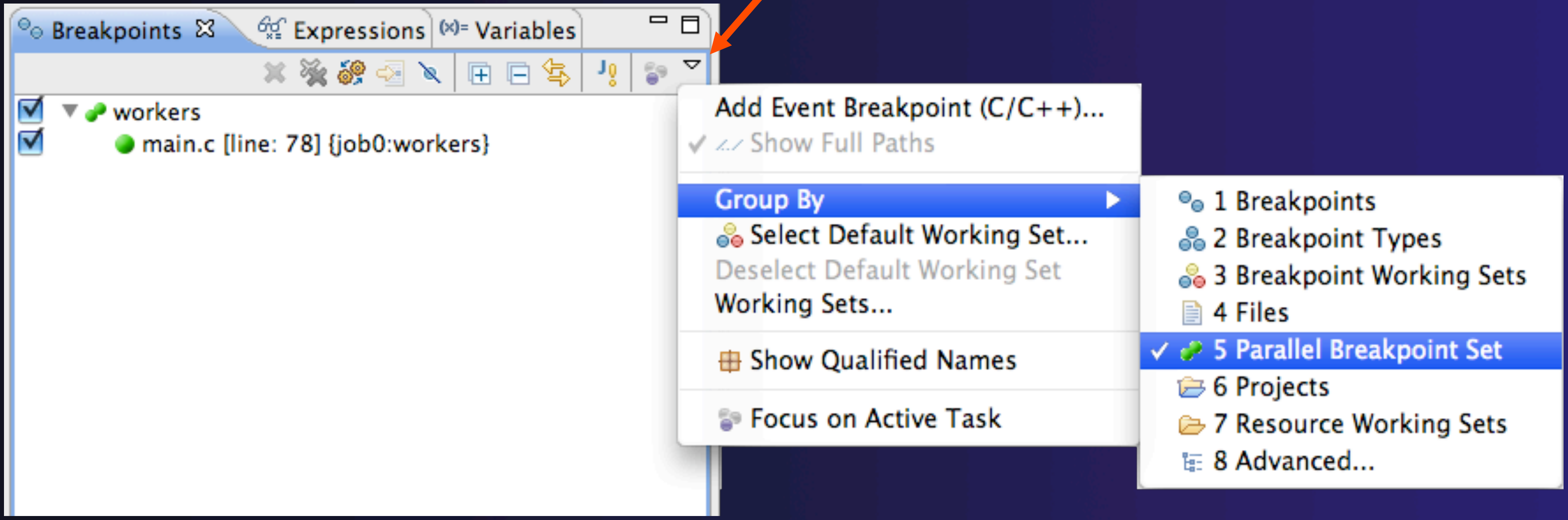
- Hover over breakpoint icon
  - Will show the sets this breakpoint applies to
- Select **Breakpoints** view
  - Will show all breakpoints in all projects



main.c [line: 78] {job0:workers}

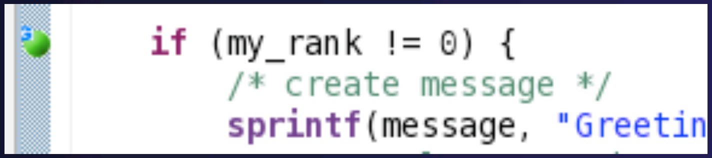
Breakpoints View

- Use the menu in the breakpoints view to group breakpoints by type
- Breakpoints sorted by breakpoint set (process set)



Global Breakpoints

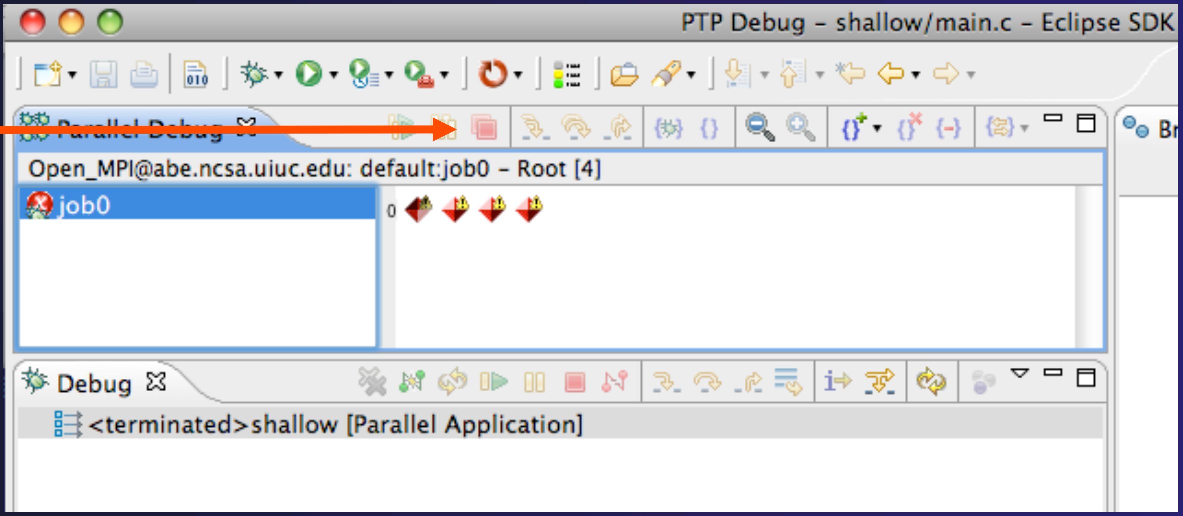
- Apply to all processes and all jobs
- Used for gaining control at debugger startup
- To create a global breakpoint
    - First make sure that no jobs are selected (click in white part of jobs view if necessary)
    - Double-click on the left edge of an editor window
    - Note that if a job is selected, the breakpoint will apply to the current set



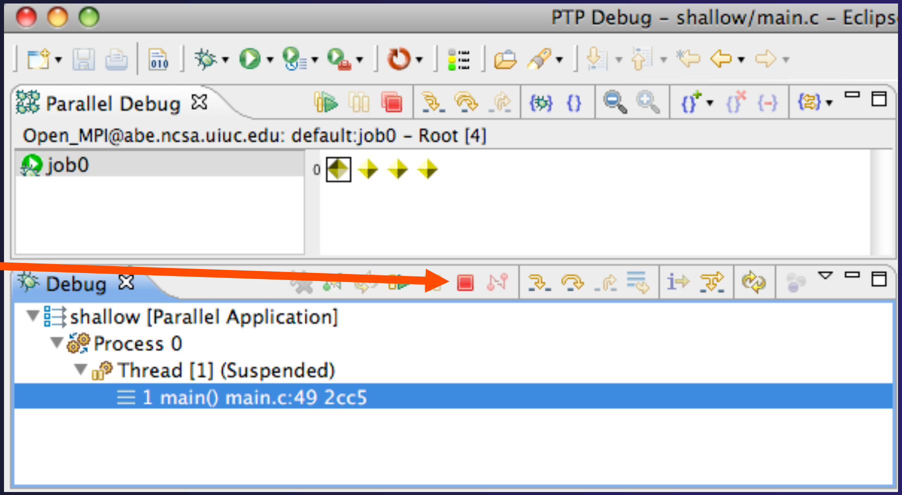
```
    if (my_rank != 0) {
        /* create message */
        sprintf(message, "Greetin
```

Terminating A Debug Session

- Click on the **Terminate** icon in the **Parallel Debug view** to terminate all processes in the active set
- Make sure the **Root** set is active if you want to terminate all processes


PTP Debug – shallow/main.c – Eclipse SDK
Parallel Debug
Open_MPI@abe.ncsa.uiuc.edu: default:job0 – Root [4]
job0
0
Debug
<terminated>shallow [Parallel Application]

- You can also use the terminate icon in the **Debug** view to terminate the currently selected process

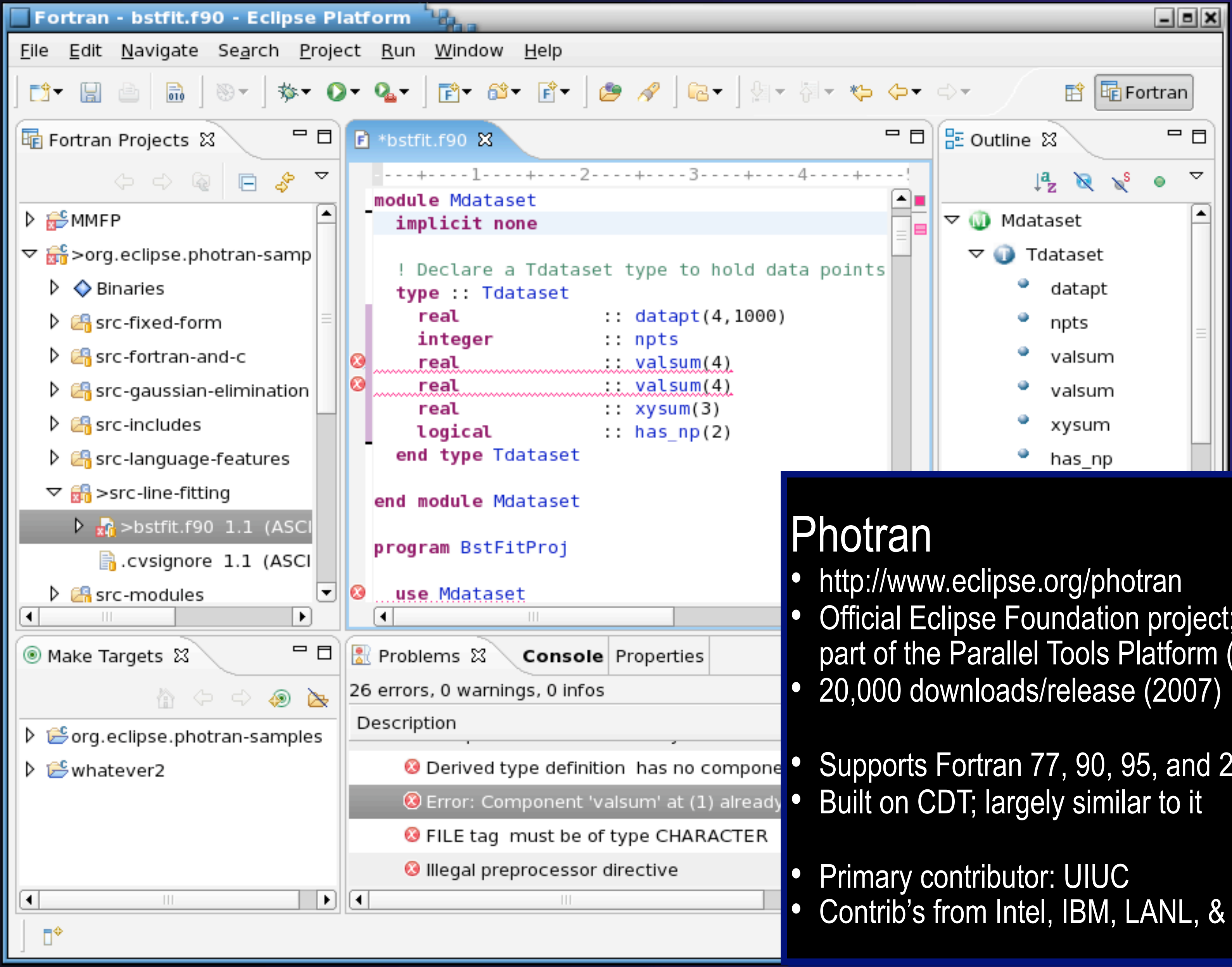

PTP Debug – shallow/main.c – Eclips
Parallel Debug
Open_MPI@abe.ncsa.uiuc.edu: default:job0 – Root [4]
job0
0
Debug
shallow [Parallel Application]
Process 0
Thread [1] (Suspended)
1 main() main.c:49 2cc5

Module 6: Fortran

- Objective
  - Learn what Photran is and how it compares to CDT
  - Learn how to create a Fortran MPI application
  - Learn about refactoring support
- Contents
  - Overview of Photran
  - Module 3 redux (in Fortran)
  - Differences between Photran and CDT
  - Pointers to online documentation for Photran
  - Refactoring support

Ralph Johnson's research group at UIUC used to meet at Pho-Tran…

…which became the name of their Fortran IDE.



```
module Mdataset
  implicit none

  ! Declare a Tdataset type to hold data points
  type :: Tdataset
    real              :: datapt(4,1000)
    integer           :: npts
    real              :: valsum(4)
    real              :: valsum(4)
    real              :: xysum(3)
    logical           :: has_np(2)
  end type Tdataset

end module Mdataset

program BstFitProj

  use Mdataset
```

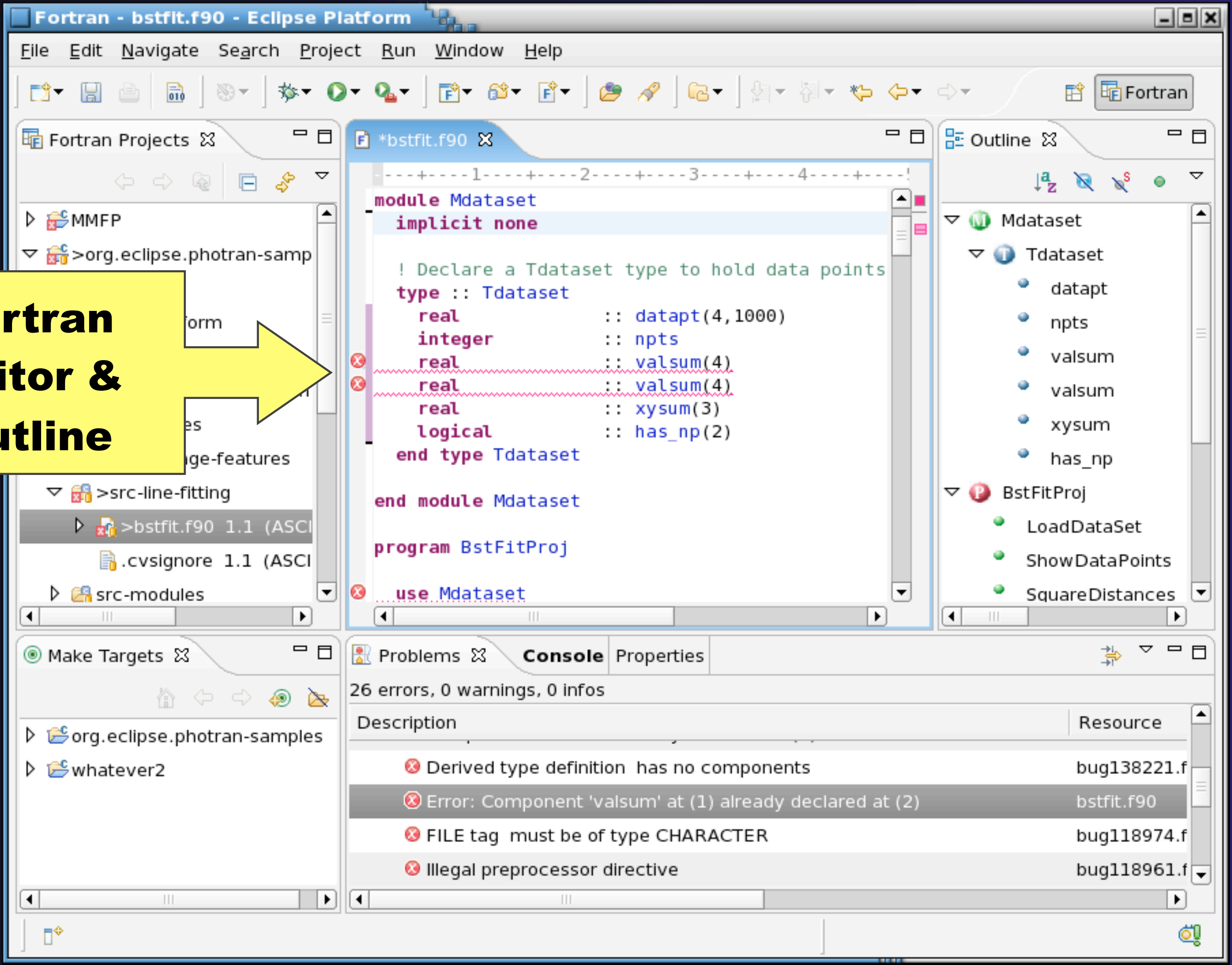
```
26 errors, 0 warnings, 0 infos
Derived type definition  has no compone
Error: Component 'valsum' at (1) already
FILE tag  must be of type CHARACTER
Illegal preprocessor directive
```

Photran

- http://www.eclipse.org/photran
- Official Eclipse Foundation project; part of the Parallel Tools Platform (PTP)
- 20,000 downloads/release (2007)

- Supports Fortran 77, 90, 95, and 2003
- Built on CDT; largely similar to it

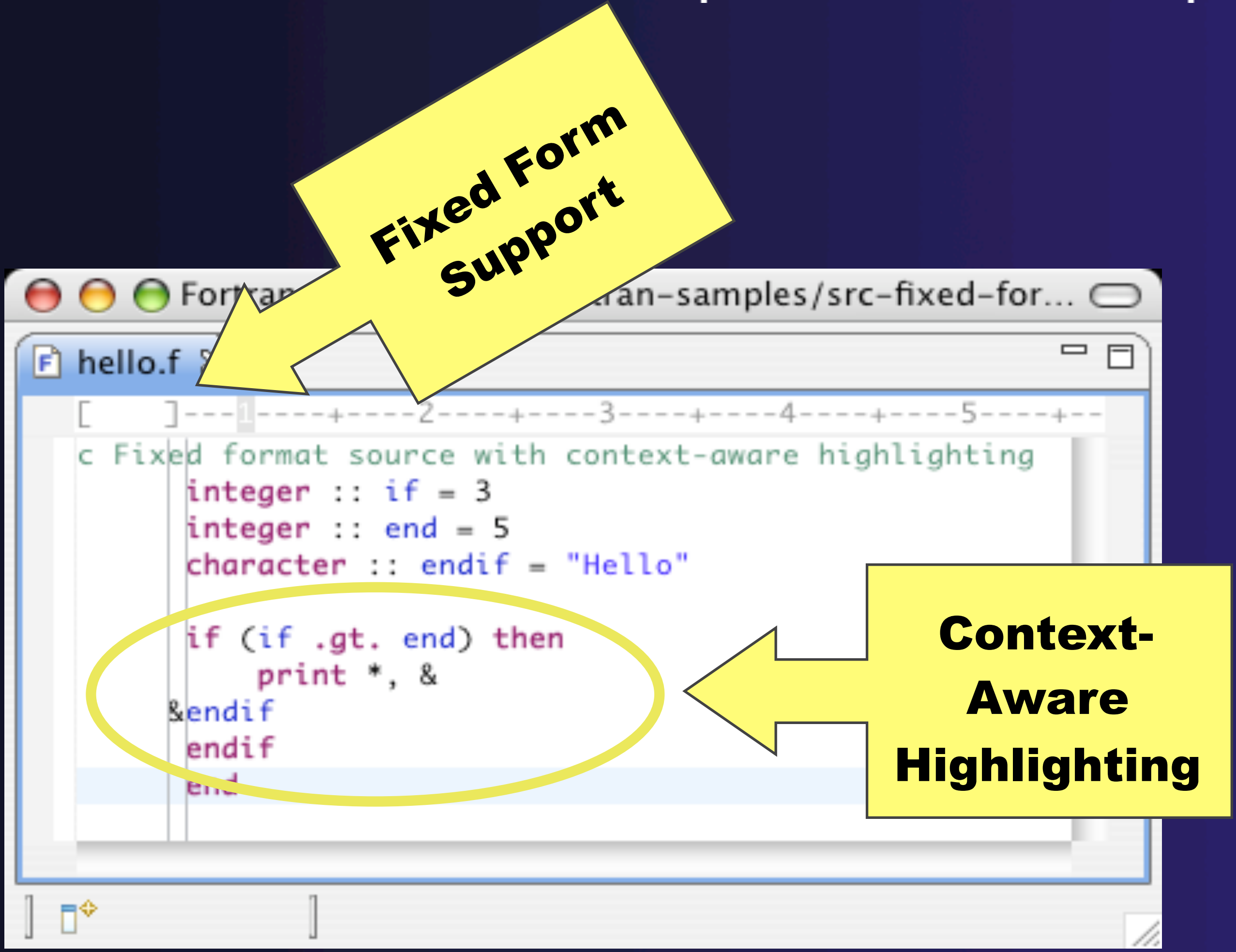
- Primary contributor: UIUC
- Contrib’s from Intel, IBM, LANL, & others



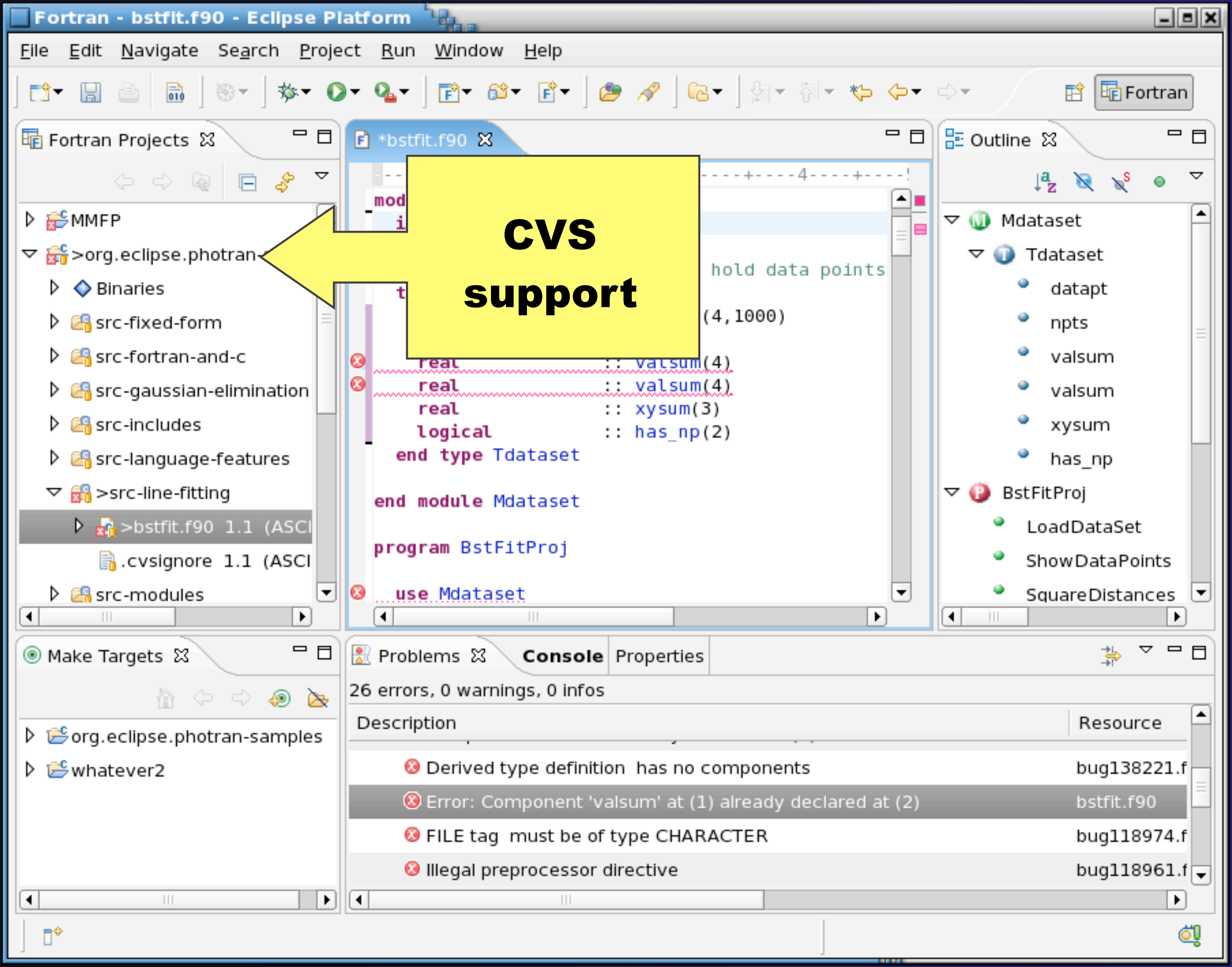
```
module Mdataset
  implicit none

  ! Declare a Tdataset type to hold data points
  type :: Tdataset
    real              :: datapt(4,1000)
    integer           :: npts
    real              :: valsum(4)
    real              :: valsum(4)
    real              :: xysum(3)
    logical           :: has_np(2)
  end type Tdataset

end module Mdataset

program BstFitProj

  use Mdataset
```

26 errors, 0 warnings, 0 infos

Description	Resource
Derived type definition has no components	bug138221.f
Error: Component 'valsum' at (1) already declared at (2)	bstfit.f90
FILE tag must be of type CHARACTER	bug118974.f
Illegal preprocessor directive	bug118961.f



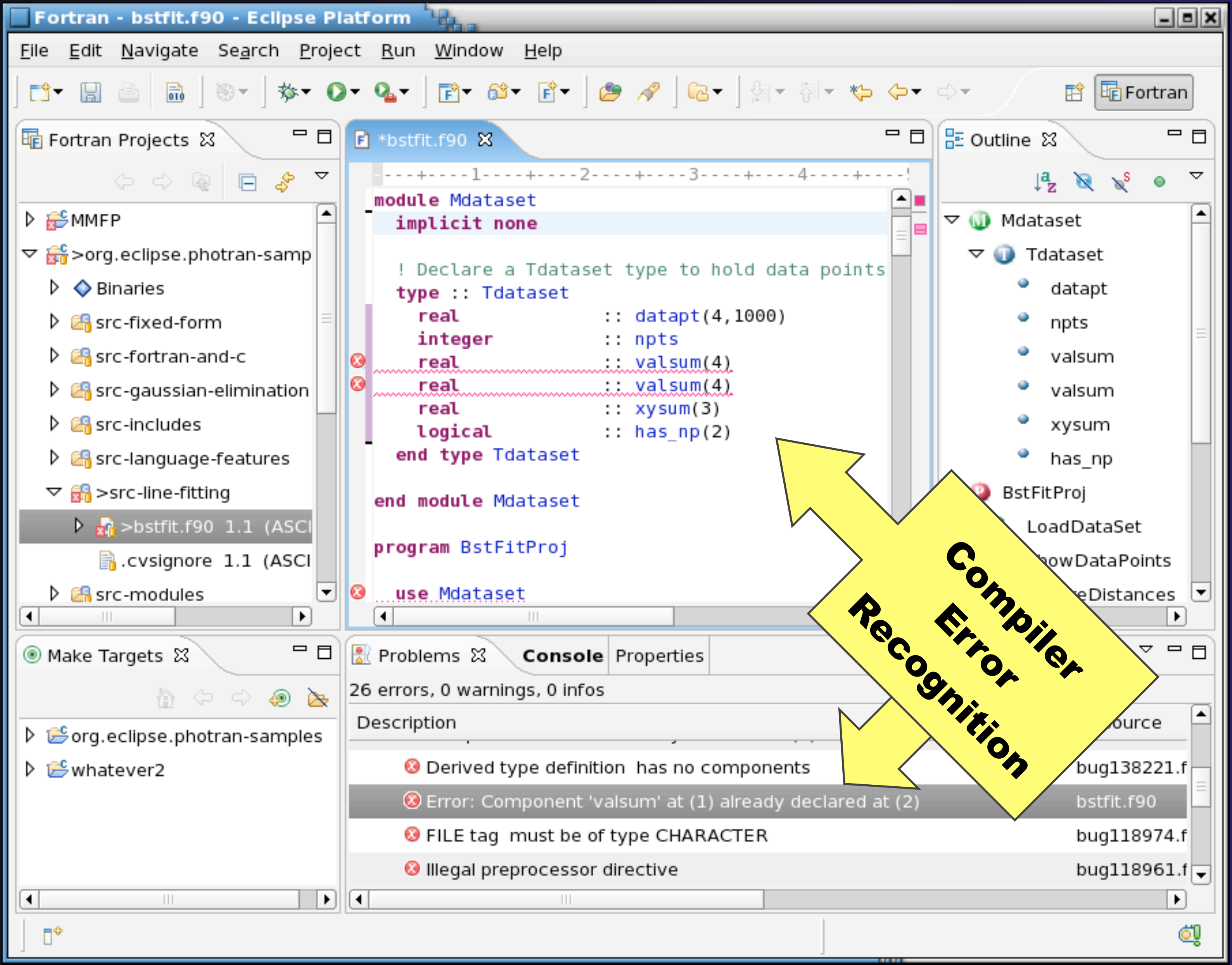
```
c Fixed format source with context-aware highlighting
      integer :: if = 3
      integer :: end = 5
      character :: endif = "Hello"

      if (if .gt. end) then
          print *, &
     &endif
      endif
      end
```



```
mod
  i
                                hold data points
  t
                               (4,1000)
    real                    :: valsum(4)
    real                    :: valsum(4)
    real                    :: xysum(3)
    logical                 :: has_np(2)
  end type Tdataset

end module Mdataset

program BstFitProj

  use Mdataset
```

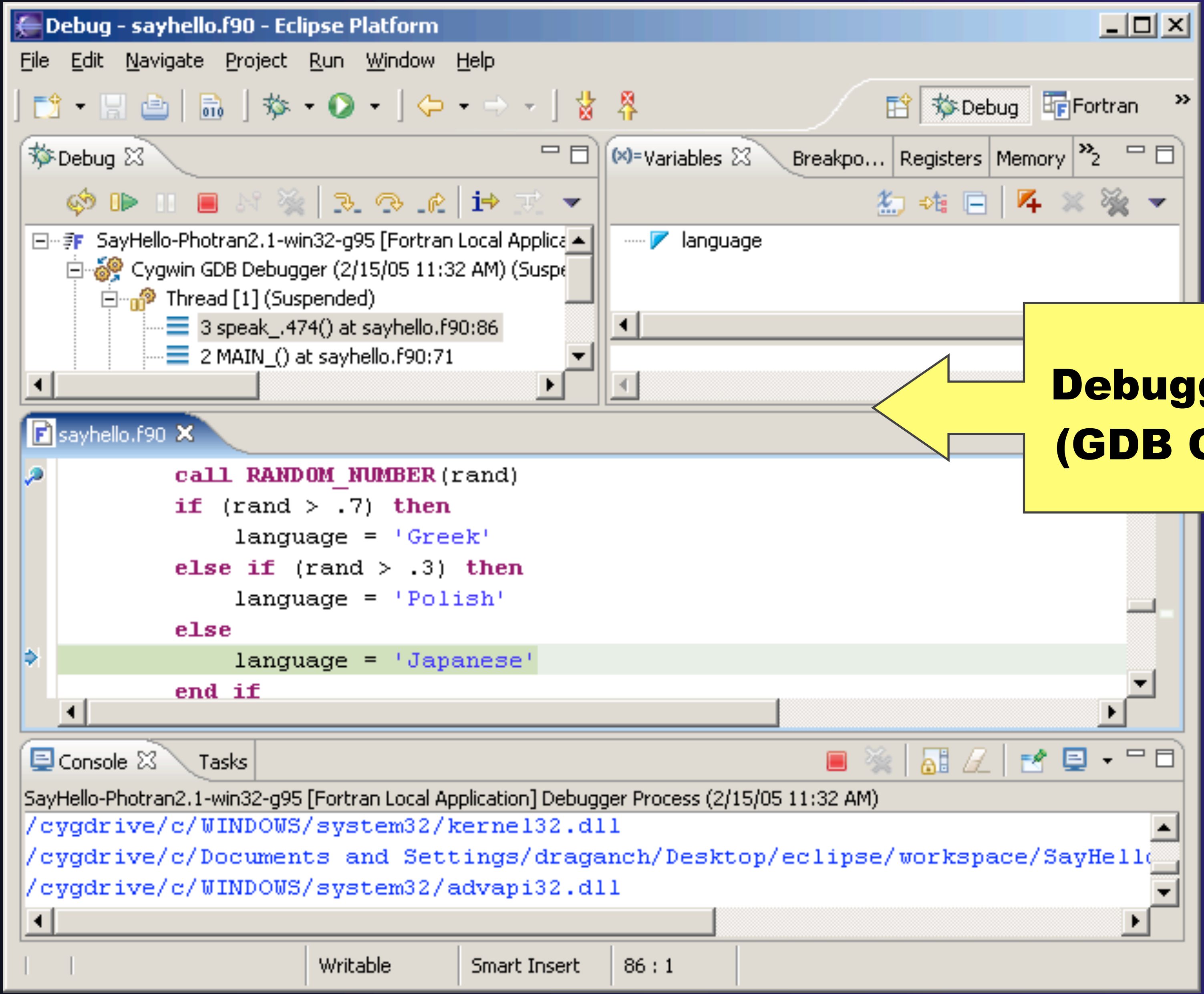
26 errors, 0 warnings, 0 infos

Description	Resource
Derived type definition has no components	bug138221.f
Error: Component 'valsum' at (1) already declared at (2)	bstfit.f90
FILE tag must be of type CHARACTER	bug118974.f
Illegal preprocessor directive	bug118961.f



```
module Mdataset
  implicit none

  ! Declare a Tdataset type to hold data points
  type :: Tdataset
    real              :: datapt(4,1000)
    integer           :: npts
    real              :: valsum(4)
    real              :: valsum(4)
    real              :: xysum(3)
    logical           :: has_np(2)
  end type Tdataset

end module Mdataset

program BstFitProj

  use Mdataset
```

```
26 errors, 0 warnings, 0 infos
Description                                                        ource
Derived type definition  has no components                         bug138221.f
Error: Component 'valsum' at (1) already declared at (2)           bstfit.f90
FILE tag  must be of type CHARACTER                                bug118974.f
Illegal preprocessor directive                                     bug118961.f
```


Debug - sayhello.f90 - Eclipse Platform
File Edit Navigate Project Run Window Help
Debug
Fortran
Debug
Variables
Breakpo...
Registers
Memory
SayHello-Photran2.1-win32-g95 [Fortran Local Applica
Cygwin GDB Debugger (2/15/05 11:32 AM) (Suspe
Thread [1] (Suspended)
3 speak_.474() at sayhello.f90:86
2 MAIN_() at sayhello.f90:71
language
sayhello.f90
Console
Tasks
Writable
Smart Insert
86 : 1

```
        call RANDOM_NUMBER(rand)
        if (rand > .7) then
            language = 'Greek'
        else if (rand > .3) then
            language = 'Polish'
        else
            language = 'Japanese'
        end if
```

```
SayHello-Photran2.1-win32-g95 [Fortran Local Application] Debugger Process (2/15/05 11:32 AM)
/cygdrive/c/WINDOWS/system32/kernel32.dll
/cygdrive/c/Documents and Settings/draganch/Desktop/eclipse/workspace/SayHell
/cygdrive/c/WINDOWS/system32/advapi32.dll
```

Debugging (GDB GUI)

Installing Photran

1. Download the lastest photran-master-xxxxx.zip from http://wiki.eclipse.org/PTP/builds/photran/5.0.0
@SC09 Tutorial: use file provided
2. In Eclipse, click on Help > Install New Software...
3. Click on the "Add..." button.
4. Click on the "Archive..." button.
5. Choose the zip file you downloaded in step 1.
6. Click OK to close the Add Site dialog. This will return you to the Install dialog to complete the installation.

Installing Photran (2)

7. Expand "Photran (Fortran Development Tools)" and check the box next to "Photran End-User Runtime."”
8. Click on the "Next" button.
9. If you get an error message, see Photran's online documentation for troubleshooting information.
10. Click the Finish button and agree to the license to complete the installation.

    http://wiki.eclipse.org/PTP/photran/documentation/photran5#Installation_Procedure

Using Photran

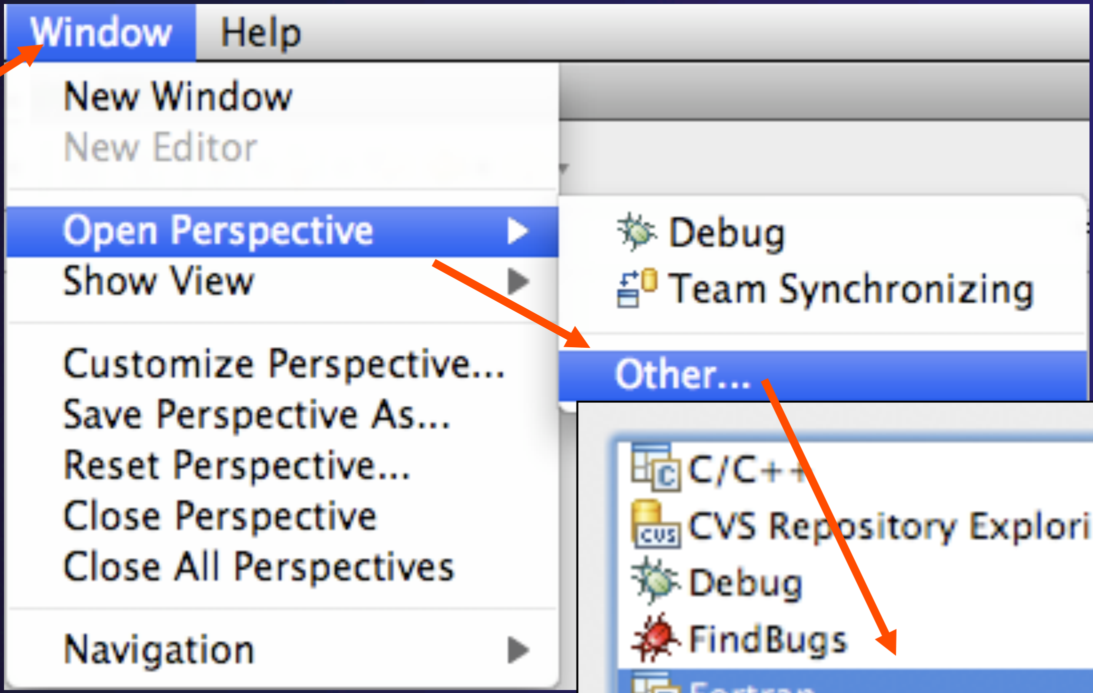
- It’s just like using CDT…
  - Similar New Project wizards
  - Similar build procedure
  - Similar launch/debug procedure
- …but not exactly
  - Configuring fixed vs. free form file extensions
  - Different editor features
  - Different advanced features (Module 7)

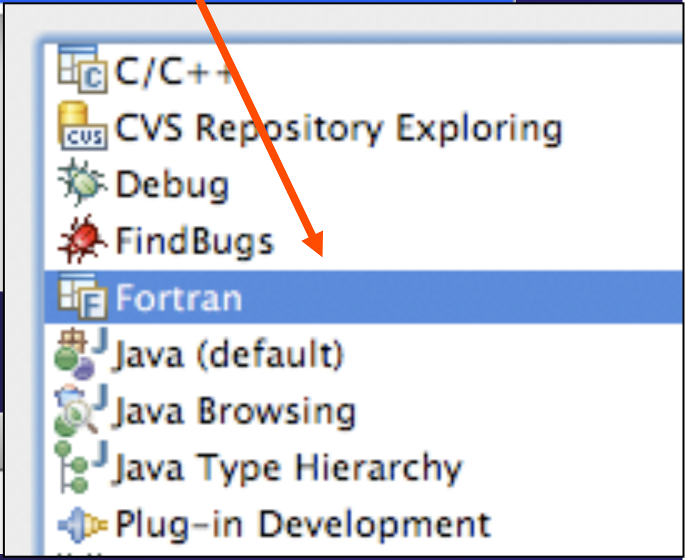
Switch to ~~C/C++~~ Fortran Perspective

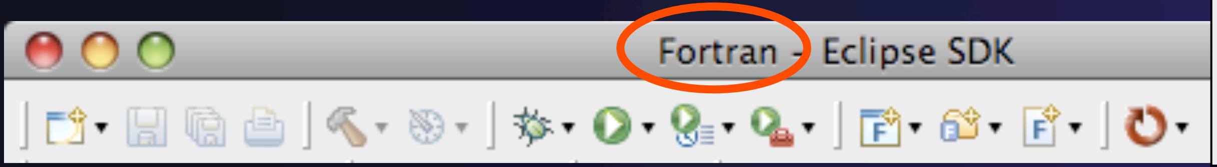
(same as for C/C++)

- Only needed if you're not already in the perspective

- What Perspective am in in? See Title Bar



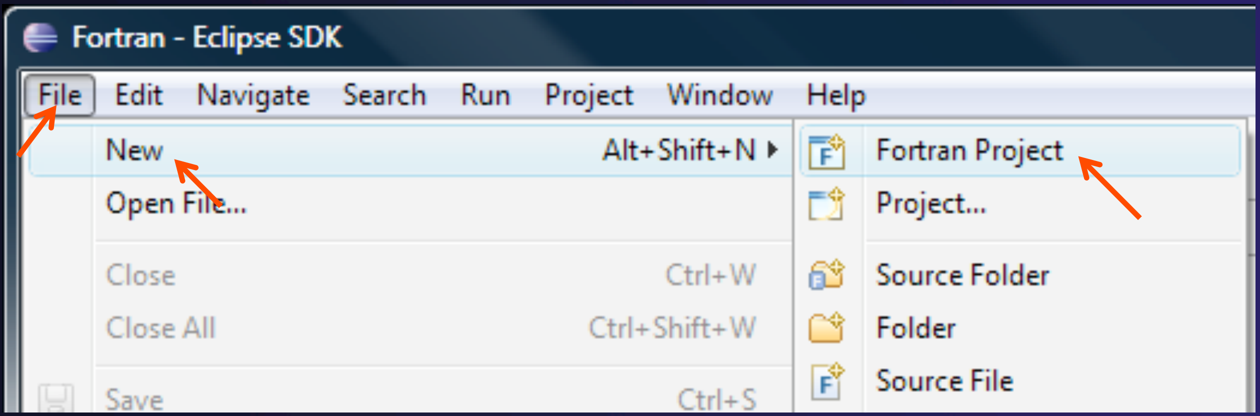


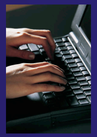


Creating a Fortran Application
(same as Creating a C/C++ Application)

Steps:

- Create a new Fortran project
- Edit source code
- Save and build



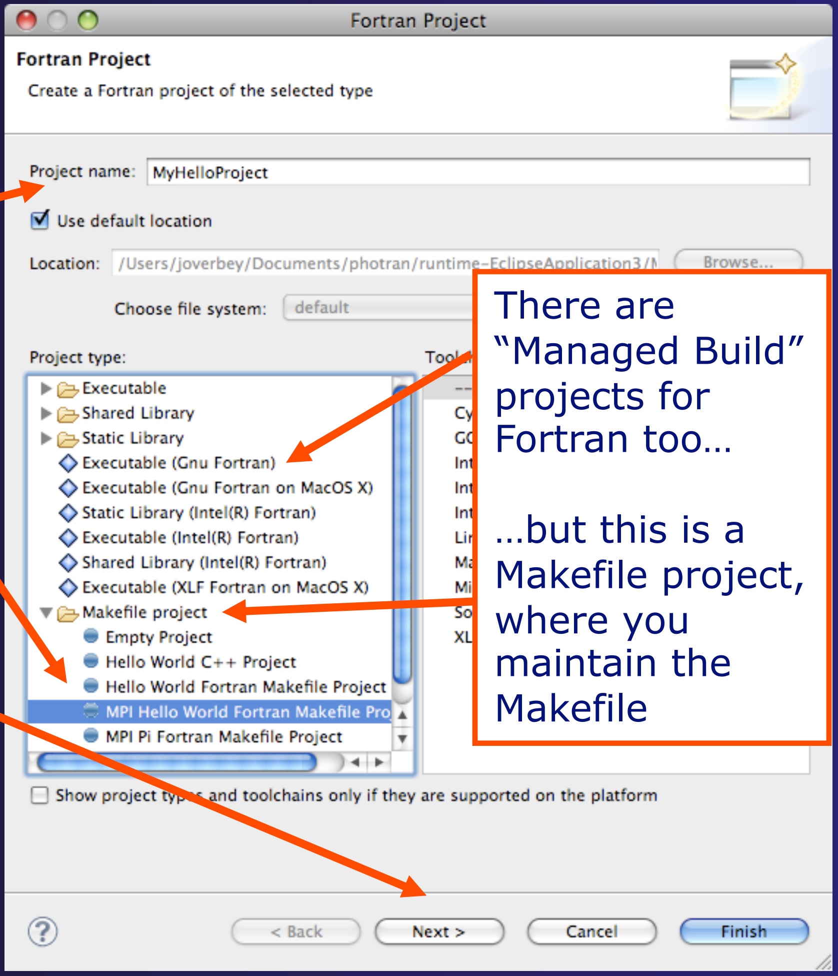


New Fortran Project Wizard

(similar to New C/C++ Project Wizard)

Create a new MPI project

- **File▶New▶Fortran Project** (see prev. slide)
- Name the project 'MyHelloProject'
- Under Project types, under Makefile Project, select **MPI Hello World Fortran Project** and hit **Next**
- On **Basic Settings** page, fill in information for your new project (**Author name** etc.) and hit **Finish**



Fortran Project

Fortran Project

Create a Fortran project of the selected type

Project name: MyHelloProject

☑ Use default location

Location: /Users/joverbey/Documents/photran/runtime-EclipseApplication3/N

Choose file system: default

Project type:

- Executable
- Shared Library
- Static Library
- Executable (Gnu Fortran)
- Executable (Gnu Fortran on MacOS X)
- Static Library (Intel(R) Fortran)
- Executable (Intel(R) Fortran)
- Shared Library (Intel(R) Fortran)
- Executable (XLF Fortran on MacOS X)
- Makefile project
  - Empty Project
  - Hello World C++ Project
  - Hello World Fortran Makefile Project
  - MPI Hello World Fortran Makefile Pro
  - MPI Pi Fortran Makefile Project

☐ Show project types and toolchains only if they are supported on the platform

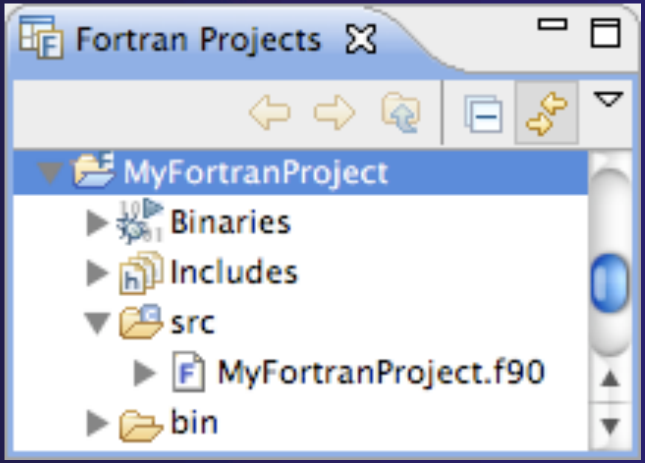
There are "Managed Build" projects for Fortran too…

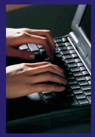
…but this is a Makefile project, where you maintain the Makefile

Fortran Projects View

(similar to C/C++ Project Explorer view)

- Represents user's data
- It is a set of user defined resources
  - Files
  - Folders
  - Projects
    - Collections of files and folders
    - Plus meta-data
- Resources are visible in the Fortran Projects View

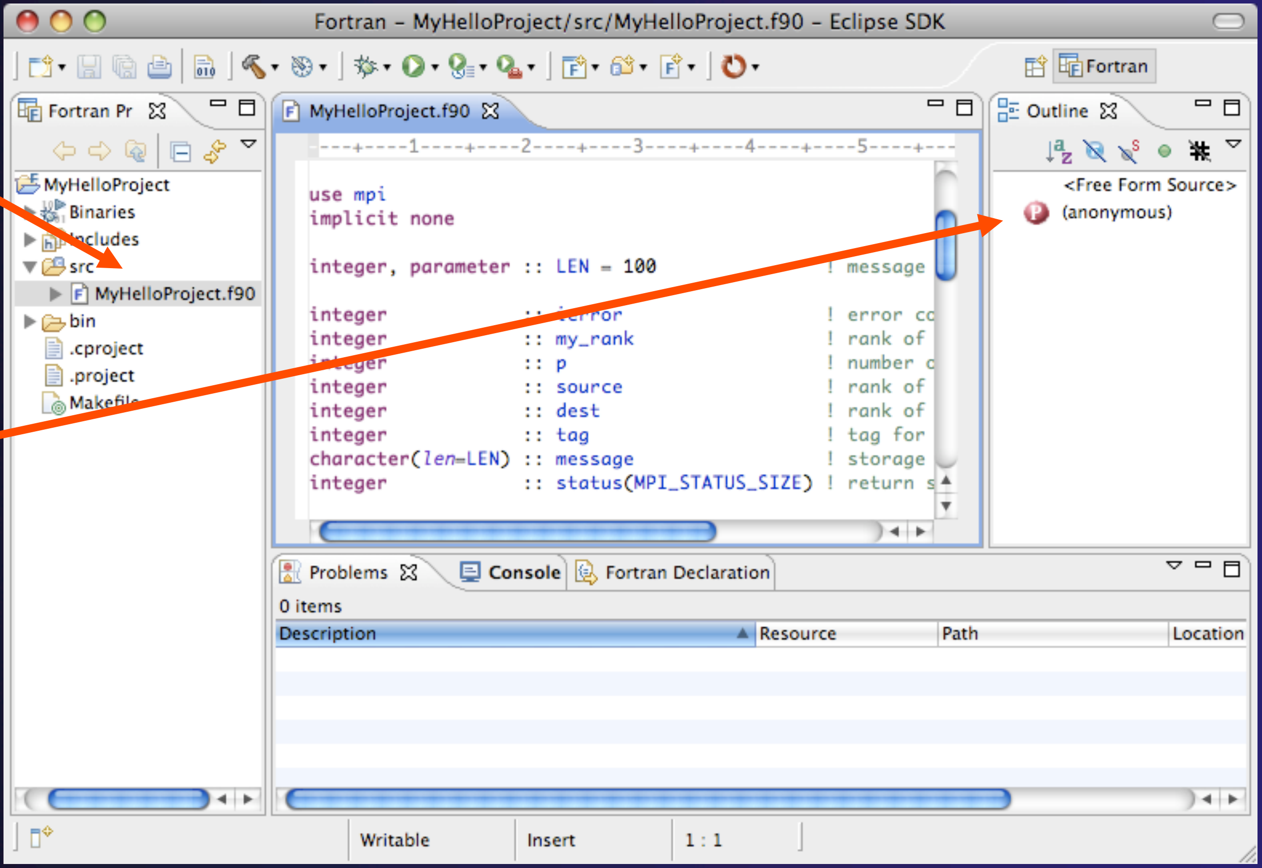




Editor and Outline View

(similar to C/C++)

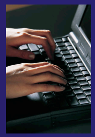
- Double-click on source file to open Fortran editor
- Outline view is shown for file in editor



```
---+----1----+----2----+----3----+----4----+----5----+---

use mpi
implicit none

integer, parameter :: LEN = 100                   ! message

integer            :: ierror                      ! error co
integer            :: my_rank                     ! rank of
integer            :: p                           ! number o
integer            :: source                      ! rank of
integer            :: dest                        ! rank of
integer            :: tag                         ! tag for
character(len=LEN) :: message                     ! storage
integer            :: status(MPI_STATUS_SIZE)     ! return s
```

0 items

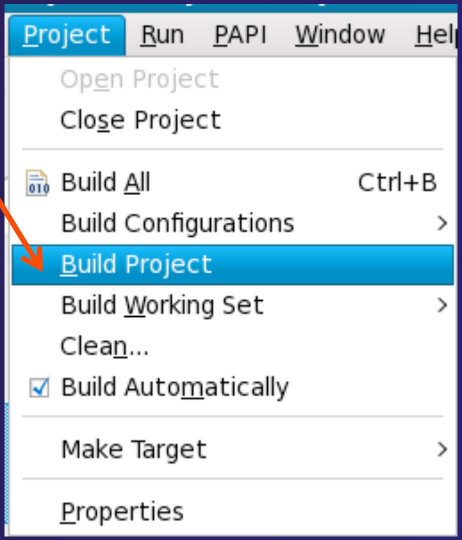
Description	Resource	Path	Location

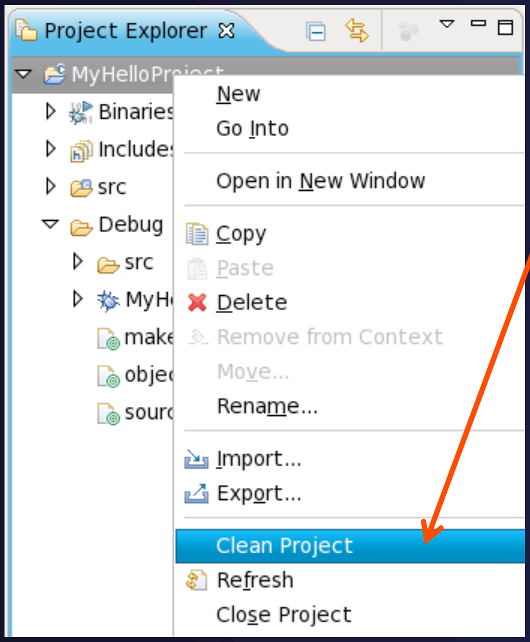


Build

(same as C/C++)

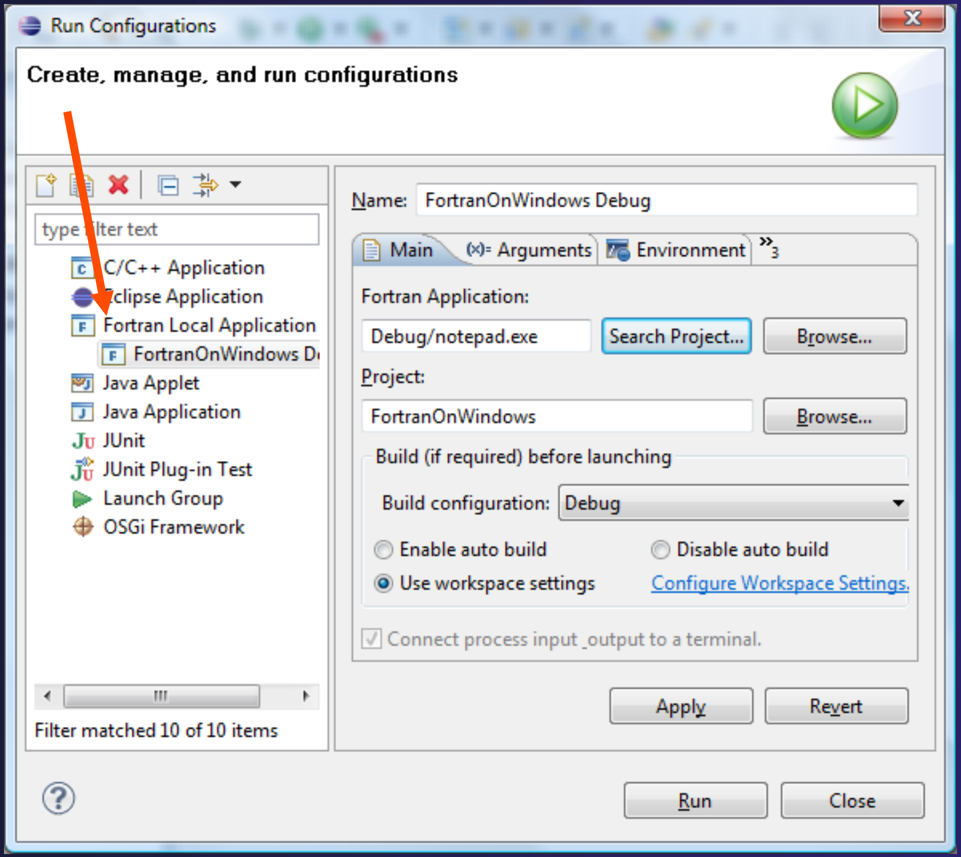
- Your program should build when created.
- To rebuild, many ways include:
  - Select project, Hit hammer icon in toolbar
  - Select project, **Project ▶ Build Project**
  - Right mouse on project, **Clean Project**





Et Cetera

- Creating a launch configuration is identical
  (Suggestion: Uncheck **Stop on startup at main** in the Debugger tab)



Run Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- Eclipse Application
- Fortran Local Application
  - FortranOnWindows D
- Java Applet
- Java Application
- JUnit
- JUnit Plug-in Test
- Launch Group
- OSGi Framework

Filter matched 10 of 10 items

Name: FortranOnWindows Debug

Main | Arguments | Environment | »3

Fortran Application: Debug/notepad.exe

Project: FortranOnWindows

Build (if required) before launching

Build configuration: Debug

Enable auto build | Disable auto build

[✓ Use workspace settings] | Configure Workspace Settings...

[✓] Connect process input _output to a terminal.

Et Cetera

- Debugging is identical
- Launching a parallel application is identical
- Debugging a parallel application is identical

Diagnosing Common Problems

(also true for C/C++)

Building: *Are compile errors not shown in the Problems view?*

- Right-click on the project in the Fortran Projects view, and choose **Properties**
- Expand **Fortran Build▸Settings**
- Switch to the **Error Parsers** tab
- Are Photran's error parsers checked? If not, click **Check all**
- Click **OK** and re-build

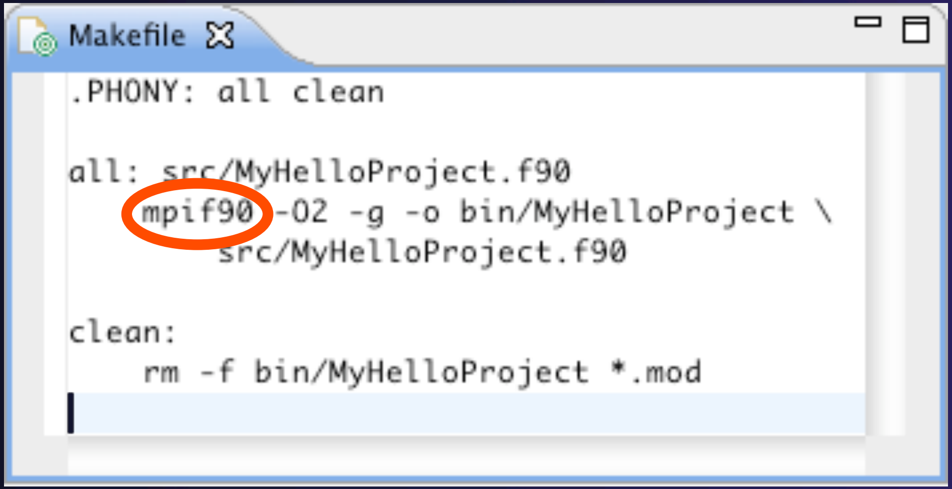
Launching: *Is a binary not listed when creating a launch configuration?*

- Right-click on the project in the Fortran Projects view, and choose **Properties**
- Expand **Fortran Build▸Settings**
- Switch to the **Binary Parsers** tab
- Make sure the parser for your platform is checked
  - PE = Windows
  - Elf = Linux
  - Mach-O = Mac OS X
- Click **OK**

Differences (1): MPI Project Wizard

- In the MPI Hello World C Project,
  the MPI compiler is set in the project settings…
  (See "Changing the C/C++ Build Settings Manually" in Module 3)

- …but in the MPI Hello World Fortran Project,
  the MPI compiler is set in a Makefile.

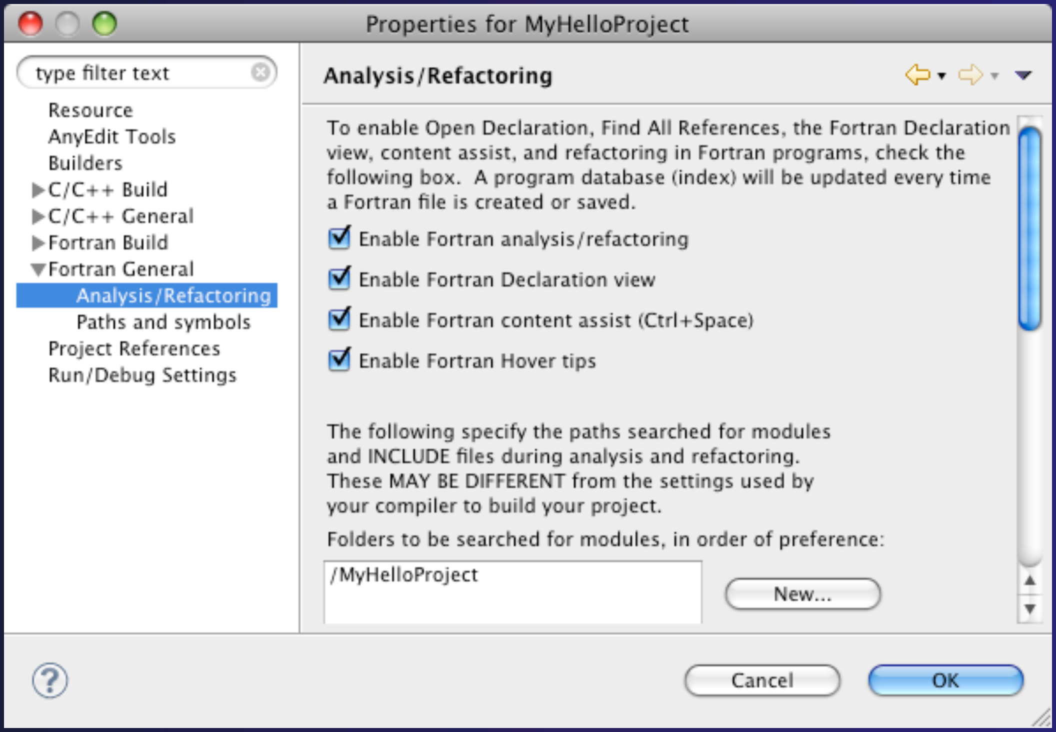


```
.PHONY: all clean

all: src/MyHelloProject.f90
	mpif90 -O2 -g -o bin/MyHelloProject \
		src/MyHelloProject.f90

clean:
	rm -f bin/MyHelloProject *.mod
```

Differences (2): Content Assist

- Content assist is *disabled* by default.
  (So are Declaration View, Hover Tips, Fortran Search, and refactorings.)
  You must specifically enable it for your project.

  - Right-click on the project in the Fortran Projects view, and choose **Properties**
  - Expand **Fortran ▸ Analysis/Refactoring**
  - Check **Enable Fortran analysis/refactoring**
  - Click **OK**
  - Close and re-open any Fortran editors



To enable Open Declaration, Find All References, the Fortran Declaration view, content assist, and refactoring in Fortran programs, check the following box. A program database (index) will be updated every time a Fortran file is created or saved.

- ✓ Enable Fortran analysis/refactoring
- ✓ Enable Fortran Declaration view
- ✓ Enable Fortran content assist (Ctrl+Space)
- ✓ Enable Fortran Hover tips

The following specify the paths searched for modules and INCLUDE files during analysis and refactoring. These MAY BE DIFFERENT from the settings used by your compiler to build your project.

Folders to be searched for modules, in order of preference:

/MyHelloProject

Differences (3): Source Form

- Fortran files are either *free form* or *fixed form*
  - Determined by filename extension
  - Extensions are set in the workspace preferences
  - Defaults:

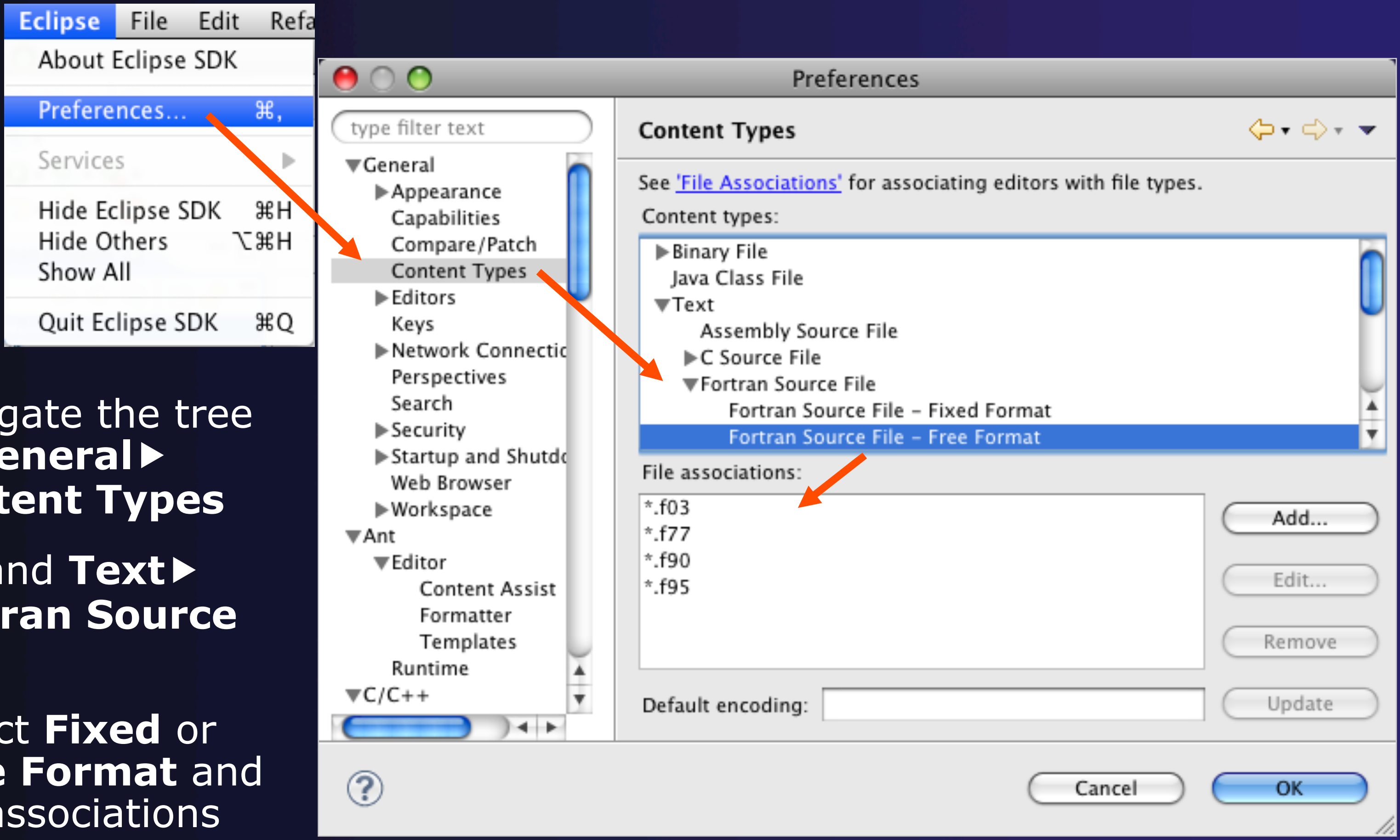
Fixed form:	.f	.fix	.for	.fpp	.ftn
Free form:	.f03	.f95	.f90	.f77	

- Many features *will not work* if filename extensions are associated incorrectly

  (Outline view, content assist, Fortran Search, refactorings, Open Declaration, …)

Differences (3): Source Form

Set fixed/free form filename extensions in the preferences



Content Types

See 'File Associations' for associating editors with file types.

Content types:

- Binary File
- Java Class File
- Text
  - Assembly Source File
  - C Source File
  - Fortran Source File
    - Fortran Source File – Fixed Format
    - Fortran Source File – Free Format

File associations:

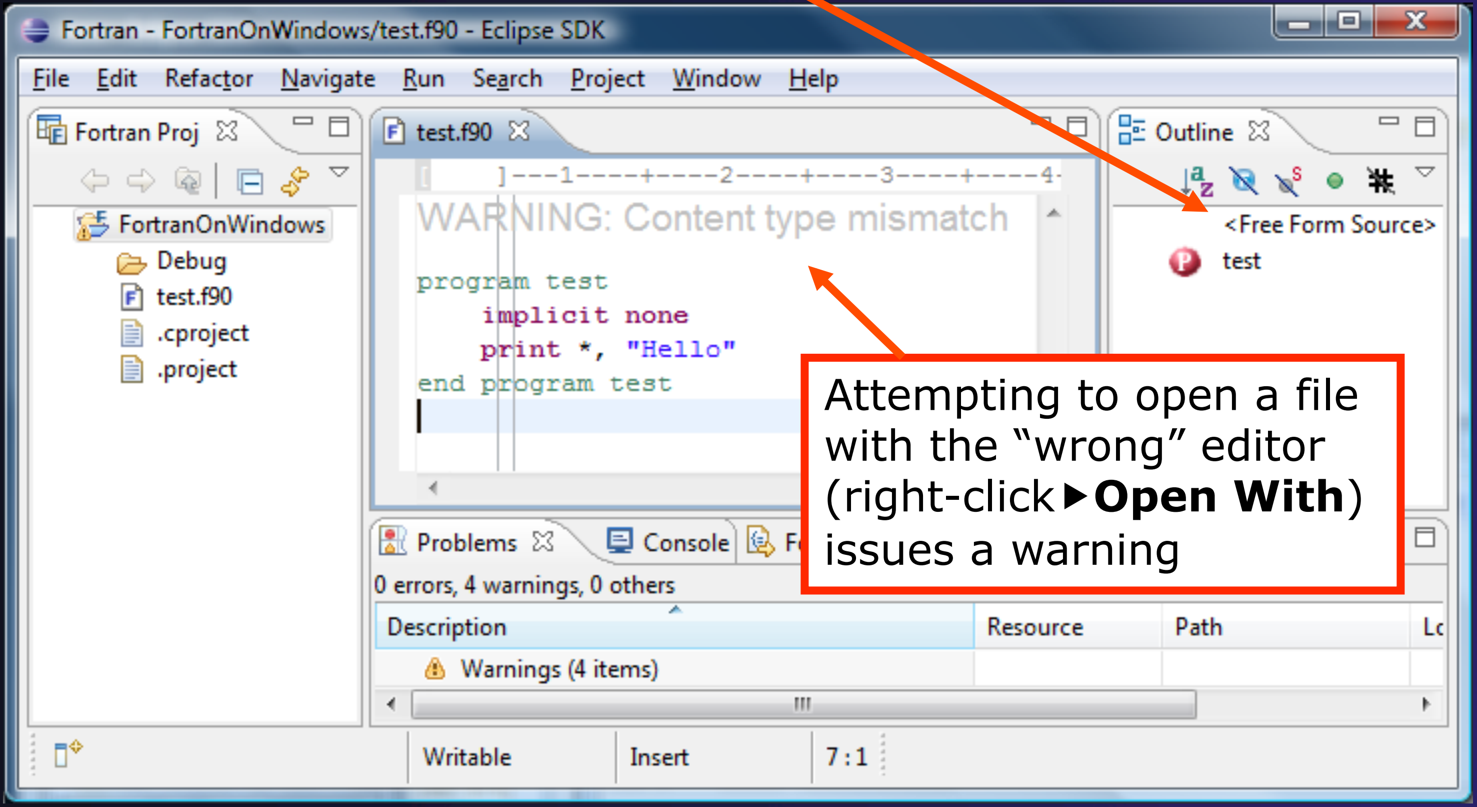
```
*.f03
*.f77
*.f90
*.f95
```

Default encoding:

- Navigate the tree to **General▶ Content Types**
- Expand **Text▶ Fortran Source File**
- Select **Fixed** or **Free Format** and set associations

Differences (3): Source Form

Outline view displays expected source form of file in editor
(according to the workspace preferences)



```
    ]---1----+----2----+----3----+----4-
WARNING: Content type mismatch

program test
    implicit none
    print *, "Hello"
end program test
```

0 errors, 4 warnings, 0 others

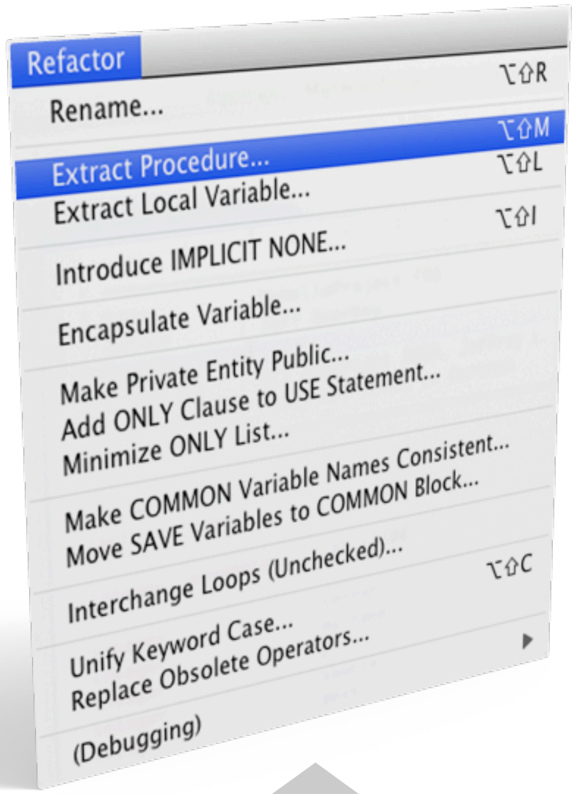
Attempting to open a file with the "wrong" editor (right-click▶**Open With**) issues a warning

For More Information

- **Photran online documentation**
  linked from http://www.eclipse.org/photran
  - **User's Guide**
    General introduction, basic features
  - **Advanced Features Guide**
    Features requiring analysis/refactoring to be enabled
- **Online tutorial:** Compiling and running the Parallel Ocean Program using Photran and PTP
  linked from http://wiki.eclipse.org/PTP/photran/tutorials

Refactoring

(making changes to source code that don't affect the behavior of the program)

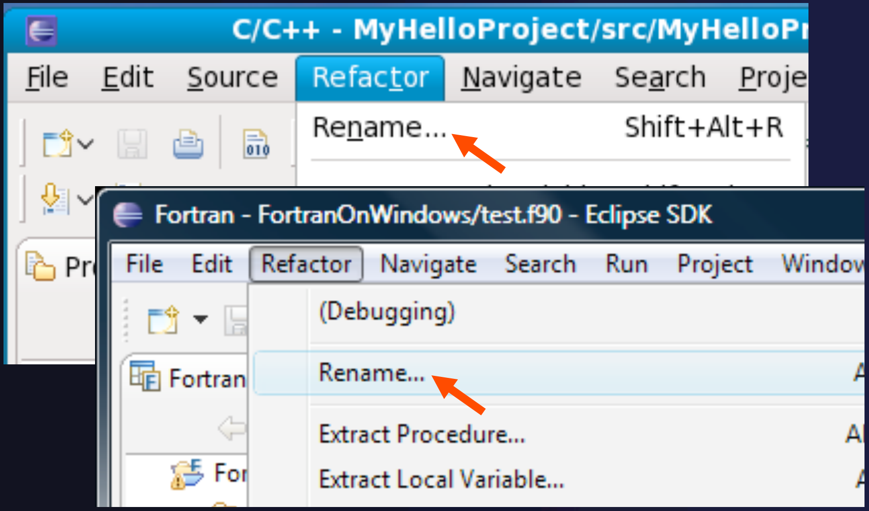

Refactor
Rename... ⌥⇧R
Extract Procedure... ⌥⇧M
Extract Local Variable... ⌥⇧L
Introduce IMPLICIT NONE... ⌥⇧I
Encapsulate Variable...
Make Private Entity Public...
Add ONLY Clause to USE Statement...
Minimize ONLY List...
Make COMMON Variable Names Consistent...
Move SAVE Variables to COMMON Block...
Interchange Loops (Unchecked)... ⌥⇧C
Unify Keyword Case...
Replace Obsolete Operators...
(Debugging)

- Refactoring is the research motivation for Photran @ Illinois
  - Illinois is a leader in refactoring research
  - "Refactoring" was coined in our group (Opdyke & Johnson, 1990)
  - We had the first dissertation… (Opdyke, 1992)
  - …and built the first refactoring tool… (Roberts, Brant, & Johnson, 1997)
  - …and first supported the C preprocessor (Garrido, 2005)
  - Photran's agenda: refactorings for HPC, language evolution, refactoring framework
- Photran 5.0: 10-15 refactorings

Rename Refactoring

(also available in C/C++)

- Changes the name of a variable, function, etc., *including every use*
  (change is semantic, not textual, and can be workspace-wide)
- Only proceeds if the new name will be legal
  (aware of scoping rules, namespaces, etc.)

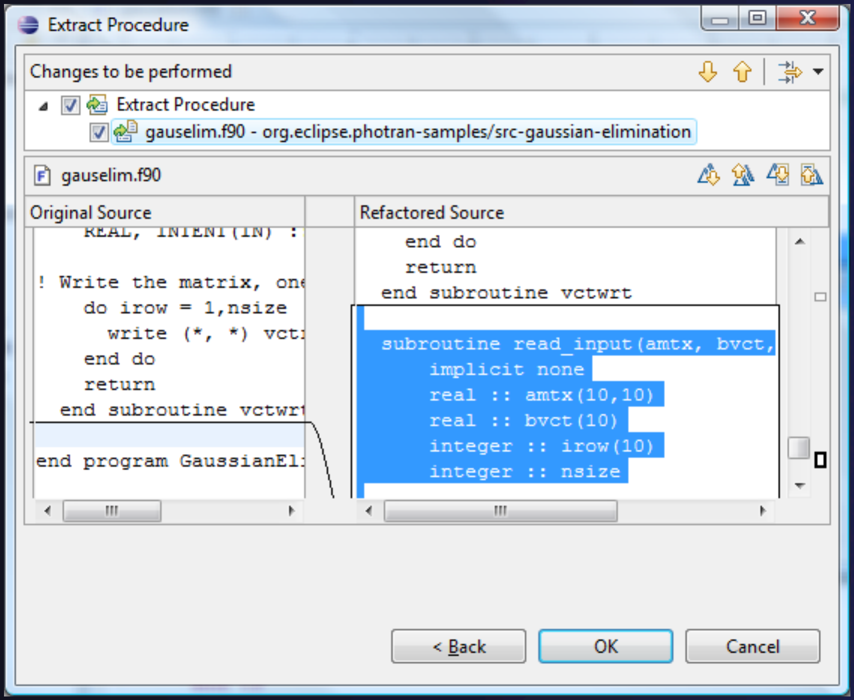

C/C++ - MyHelloProject/src/MyHelloPr
File Edit Source Refactor Navigate Search Proje
Rename... Shift+Alt+R
Fortran - FortranOnWindows/test.f90 - Eclipse SDK
File Edit Refactor Navigate Search Run Project Window
(Debugging)
Rename...
Extract Procedure...
Extract Local Variable...

- Select **Fortran Perspective**
- Open a source file
- Click in editor view on declaration of a variable
- Select menu item **Refactor▶Rename**
  - Or use context menu
- Enter new name

Extract Procedure Refactoring

(also available in C/C++ - “Extract Function”)

- Moves statements into a new subroutine, replacing the statements with a call to that subroutine
- Local variables are passed as arguments



Original Source:

```
   REAL, INTENT(IN) :

! Write the matrix, one
   do irow = 1,nsize
     write (*, *) vct
   end do
   return
 end subroutine vctwrt

end program GaussianEl
```

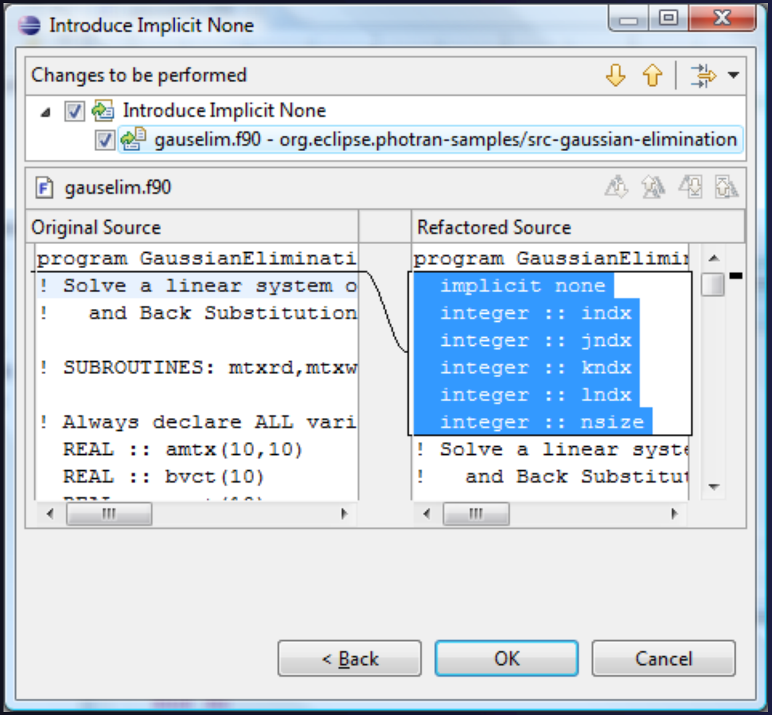
Refactored Source:

```
     end do
     return
   end subroutine vctwrt

   subroutine read_input(amtx, bvct,
       implicit none
       real :: amtx(10,10)
       real :: bvct(10)
       integer :: irow(10)
       integer :: nsize
```

- Select a sequence of statements
- Select menu item **Refactor▶Extract Procedure…**
  - Or use context menu
- Enter new name

Introduce IMPLICIT NONE Refactoring

- Fortran does not require variable declarations
  (by default, names starting with I-N are integer variables; others are reals)
- This adds an IMPLICIT NONE statement and adds explicit variable declarations for all implicitly declared variables



Introduce Implicit None

Changes to be performed

Introduce Implicit None
gauselim.f90 - org.eclipse.photran-samples/src-gaussian-elimination

gauselim.f90

Original Source

```
program GaussianEliminati
! Solve a linear system o
!    and Back Substitution

! SUBROUTINES: mtxrd,mtxw

! Always declare ALL vari
  REAL :: amtx(10,10)
  REAL :: bvct(10)
```

Refactored Source

```
program GaussianElimi
  implicit none
  integer :: indx
  integer :: jndx
  integer :: kndx
  integer :: lndx
  integer :: nsize
! Solve a linear syste
!    and Back Substitut
```

- Introduce in a single file by opening the file and selecting **Refactor▶Introduce IMPLICIT NONE…**
- Introduce in multiple files by selecting them in the Fortran Projects view, right-clicking on the selection, and choosing **Refactor▶Introduce IMPLICIT NONE…**

Module 7: Advanced Development

- Objective
  - Become familiar with other tools that help parallel application development
- Contents
  - Parallel Language Development Tools: MPI, OpenMP, UPC
    - Special Tools for parallel development
  - Performance Tuning and other external tools:
    - PTP ETFw, TAU, PPW
  - MPI Analysis: ISP

Parallel Lang. Dev. Tools

- PLDT Features
  - Analysis of C and C++ code to determine the location of MPI, OpenMP, and UPC Artifacts
  - Content assist via **ctrl+space** ("completion")
  - Hover help
  - Reference information about the API calls via Dynamic Help
  - New project wizard automatically configures managed build projects for MPI & OpenMP
  - OpenMP problems view of common errors
  - OpenMP "show #pragma region" , "show concurrency"
  - MPI Barrier analysis - detects potential deadlocks

Some MPI features were covered in Module 4

Note: Most PLDT features in 3.0 don't work on remote (RDT) projects

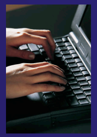
MPI Assistance Tools

Added by PLDT (Parallel Lang. Dev. Tools) feature of PTP

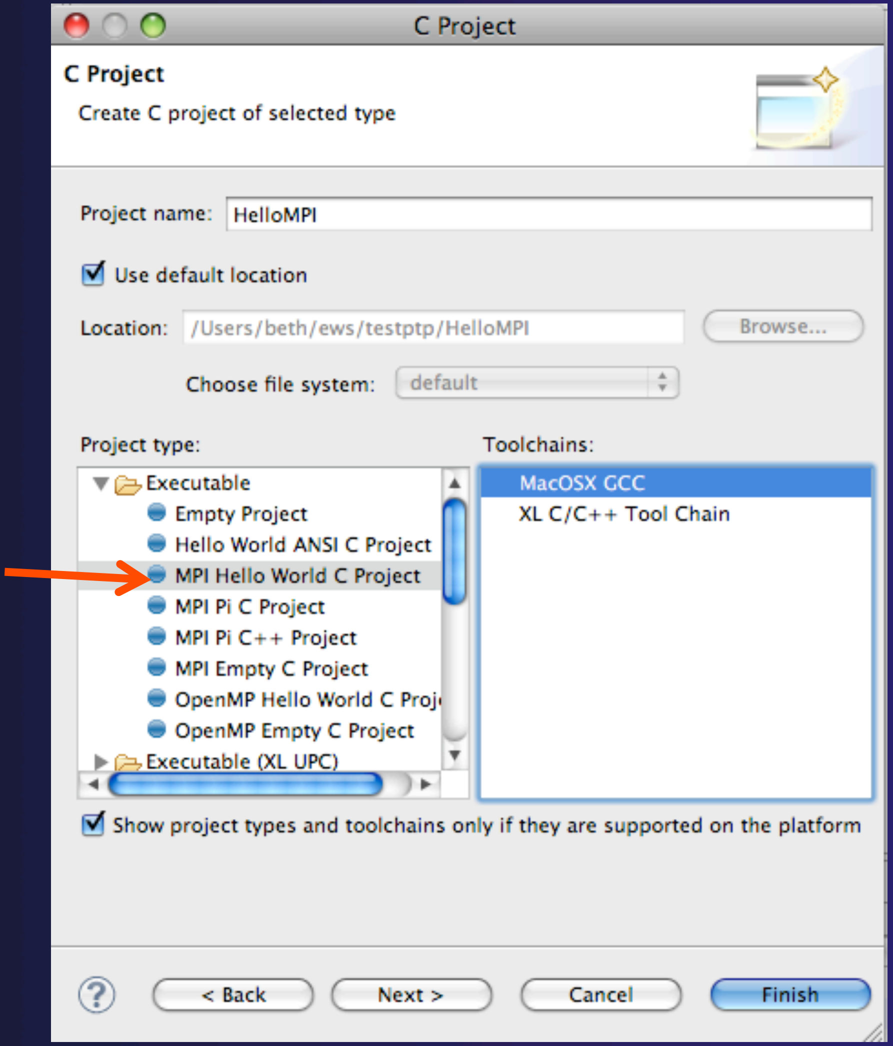
- MPI Context sensitive help
- MPI artifact locations
- MPI barrier analysis
- MPI templates

- For this part, we will use the MPI New Project Wizard and the “MPI Hello World” project

Create MPI Project



- File > New > C Project
- Give Project a name, e.g. HelloMPI
- Select MPI Hello World C Project
- Click Finish



C Project

C Project

Create C project of selected type

Project name: HelloMPI

☑ Use default location

Location: /Users/beth/ews/testptp/HelloMPI

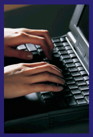
Choose file system: default

Project type:
- Executable
  - Empty Project
  - Hello World ANSI C Project
  - MPI Hello World C Project
  - MPI Pi C Project
  - MPI Pi C++ Project
  - MPI Empty C Project
  - OpenMP Hello World C Proj
  - OpenMP Empty C Project
- Executable (XL UPC)

Toolchains:
- MacOSX GCC
- XL C/C++ Tool Chain

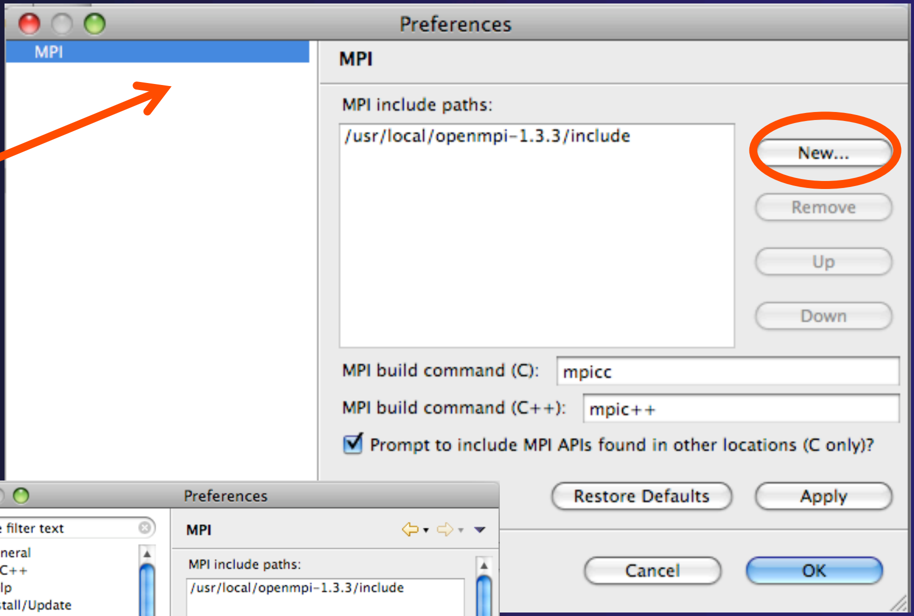
☑ Show project types and toolchains only if they are supported on the platform

Set MPI Preferences



- When creating MPI project for the first time, you will be asked to set MPI Preferences
- Select Yes.
- On the MPI Preferences page, add a new MPI include path.
- New … and point to the *directory* containing your MPI header file
- Select OK
- Back on New Project Wizard page, select **Finish.**

- You can also set preferences at any time



Preferences

MPI

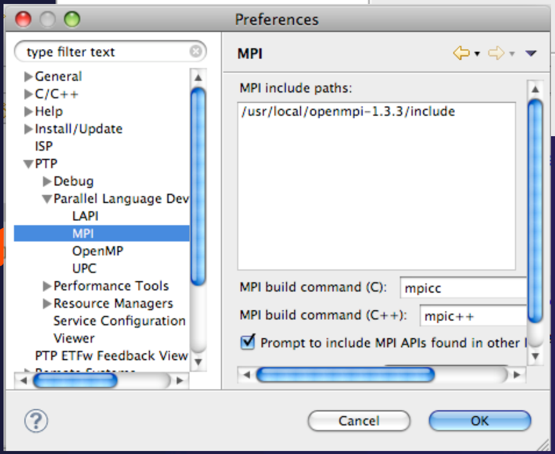
MPI include paths:

/usr/local/openmpi-1.3.3/include

MPI build command (C): mpicc

MPI build command (C++): mpic++

✓ Prompt to include MPI APIs found in other locations (C only)?



Preferences

MPI

MPI include paths:

/usr/local/openmpi-1.3.3/include

MPI build command (C): mpicc

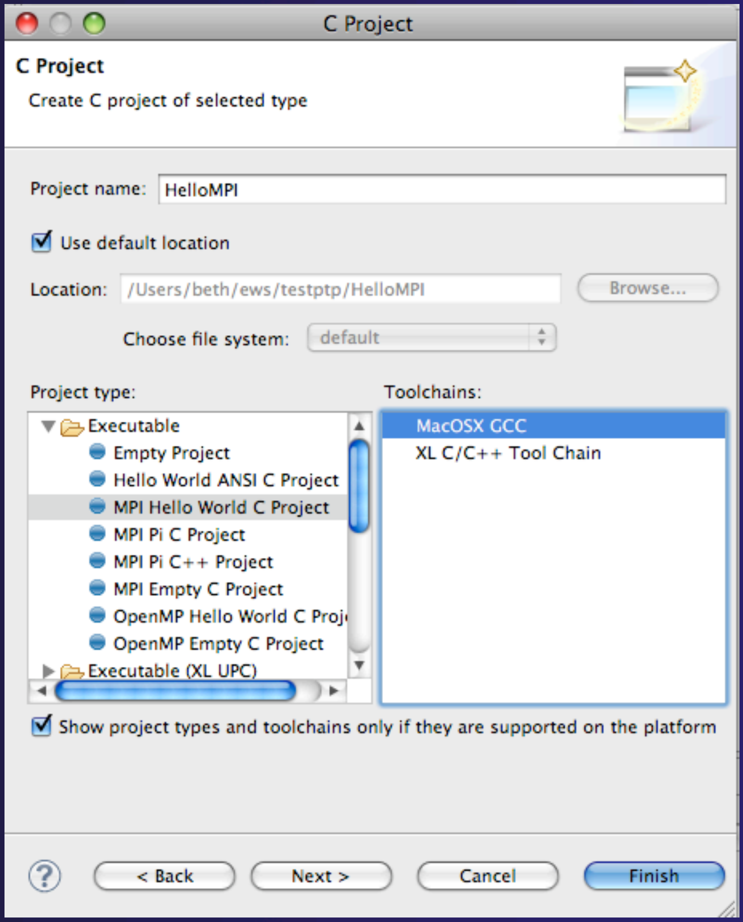
MPI build command (C++): mpic++

✓ Prompt to include MPI APIs found in other l

Create MPI Project

- File > New > C Project
- Give Project a name, e.g. HelloMPI
- Select MPI Hello World Project
- Click Finish

- Helpful editor features:
  - Hover over MPI API
  - Code completion (Ctrl-space)
  - Help (see next slide)



C Project

C Project
Create C project of selected type

Project name: HelloMPI

☑ Use default location

Location: /Users/beth/ews/testptp/HelloMPI

Choose file system: default

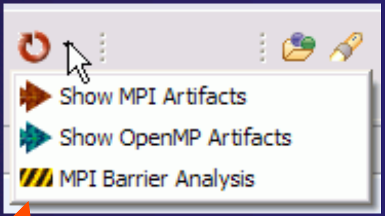
Project type:
- Executable
  - Empty Project
  - Hello World ANSI C Project
  - MPI Hello World C Project
  - MPI Pi C Project
  - MPI Pi C++ Project
  - MPI Empty C Project
  - OpenMP Hello World C Proj
  - OpenMP Empty C Project
- Executable (XL UPC)

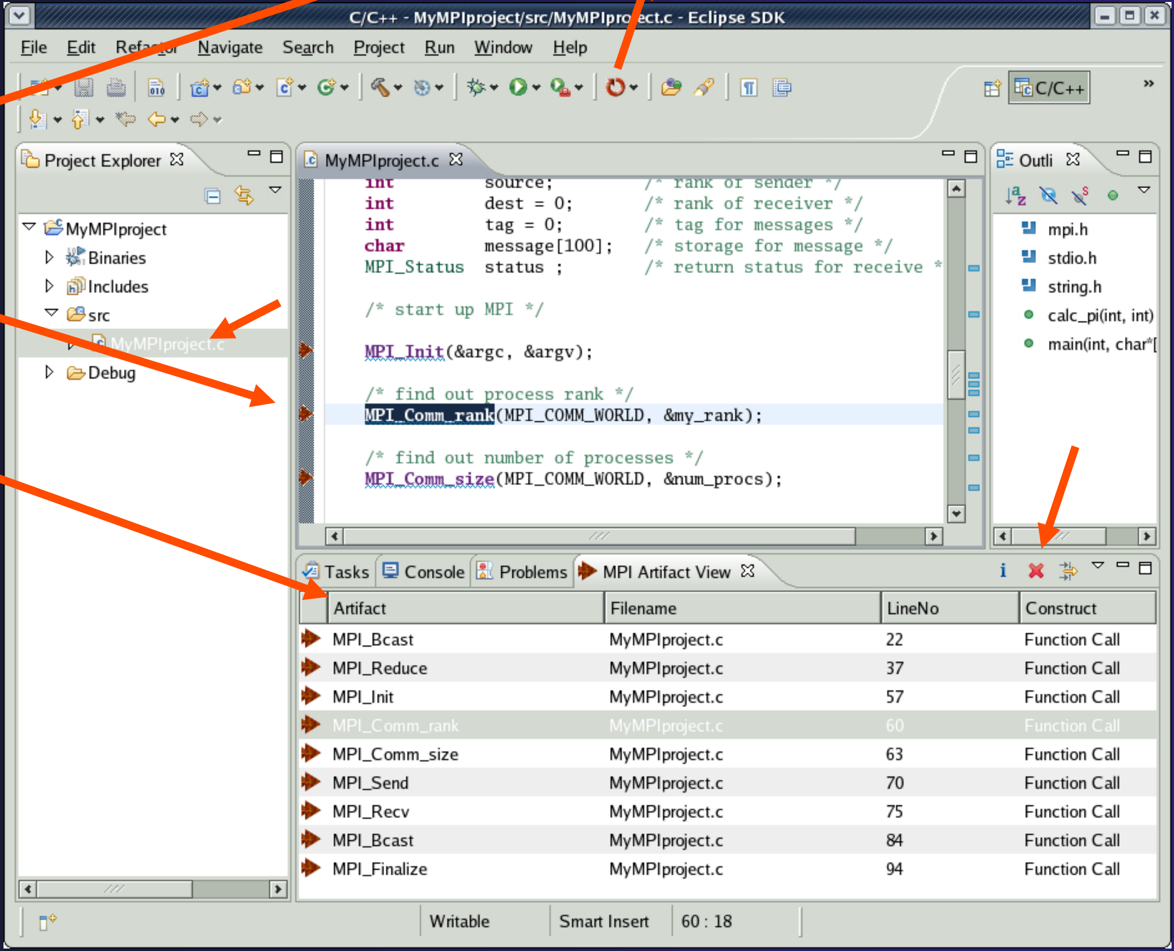
Toolchains:
- MacOSX GCC
- XL C/C++ Tool Chain

☑ Show project types and toolchains only if they are supported on the platform

Show MPI Artifacts

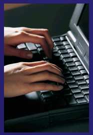
- Select source file; Run analysis by clicking on drop-down menu next to the analysis button and selecting **Show MPI Artifacts**
- Markers indicate the location of artifacts in editor
- In **MPI Artifact View** sort by any column (click on col. heading)
- Navigate to source code line by double-clicking on the artifact
- Run the analysis on another file and its markers will be added to the view
- Remove markers via ✖



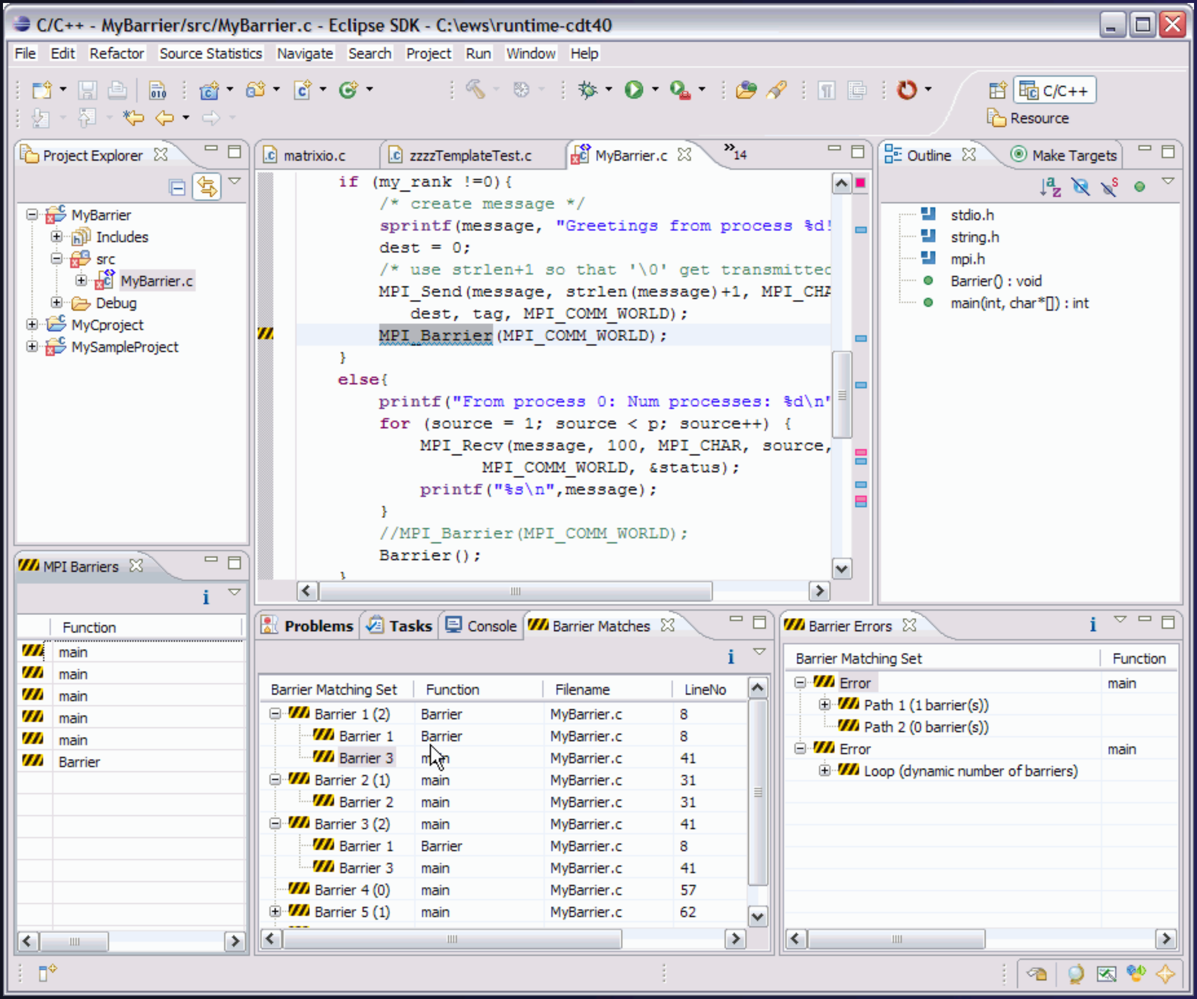

C/C++ - MyMPIproject/src/MyMPIproject.c - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
C/C++
Project Explorer
MyMPIproject
Binaries
Includes
src
MyMPIproject.c
Debug
MyMPIproject.c
Outli
mpi.h
stdio.h
string.h
calc_pi(int, int)
main(int, char*[
Tasks Console Problems MPI Artifact View
Writable Smart Insert 60 : 18

```
int         source;         /* rank of sender */
int         dest = 0;       /* rank of receiver */
int         tag = 0;        /* tag for messages */
char        message[100];   /* storage for message */
MPI_Status  status ;        /* return status for receive *

/* start up MPI */

MPI_Init(&argc, &argv);

/* find out process rank */
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

/* find out number of processes */
MPI_Comm_size(MPI_COMM_WORLD, &num_procs);
```

Artifact	Filename	LineNo	Construct
MPI_Bcast	MyMPIproject.c	22	Function Call
MPI_Reduce	MyMPIproject.c	37	Function Call
MPI_Init	MyMPIproject.c	57	Function Call
MPI_Comm_rank	MyMPIproject.c	60	Function Call
MPI_Comm_size	MyMPIproject.c	63	Function Call
MPI_Send	MyMPIproject.c	70	Function Call
MPI_Recv	MyMPIproject.c	75	Function Call
MPI_Bcast	MyMPIproject.c	84	Function Call
MPI_Finalize	MyMPIproject.c	94	Function Call



MPI Barrier Analysis


C/C++ - MyBarrier/src/MyBarrier.c - Eclipse SDK - C:\ews\runtime-cdt40

```
    if (my_rank !=0){
        /* create message */
        sprintf(message, "Greetings from process %d!
        dest = 0;
        /* use strlen+1 so that '\0' get transmittec
        MPI_Send(message, strlen(message)+1, MPI_CHA
            dest, tag, MPI_COMM_WORLD);
        MPI_Barrier(MPI_COMM_WORLD);
    }
    else{
        printf("From process 0: Num processes: %d\n"
        for (source = 1; source < p; source++) {
            MPI_Recv(message, 100, MPI_CHAR, source,
                    MPI_COMM_WORLD, &status);
            printf("%s\n",message);
        }
        //MPI_Barrier(MPI_COMM_WORLD);
        Barrier();
```

MPI Barriers

Function
main
main
main
main
main
Barrier

Barrier Matches

Barrier Matching Set	Function	Filename	LineNo
Barrier 1 (2)	Barrier	MyBarrier.c	8
Barrier 1	Barrier	MyBarrier.c	8
Barrier 3	main	MyBarrier.c	41
Barrier 2 (1)	main	MyBarrier.c	31
Barrier 2	main	MyBarrier.c	31
Barrier 3 (2)	main	MyBarrier.c	41
Barrier 1	Barrier	MyBarrier.c	8
Barrier 3	main	MyBarrier.c	41
Barrier 4 (0)	main	MyBarrier.c	57
Barrier 5 (1)	main	MyBarrier.c	62

Barrier Errors

Barrier Matching Set	Function
Error	main
Path 1 (1 barrier(s))	
Path 2 (0 barrier(s))	
Error	main
Loop (dynamic number of barriers)	

Verify barrier synchronization in C/ MPI programs

Interprocedural static analysis outputs:

- For verified programs, lists barrier statements that synchronize together (match)
- For synchronization errors, reports counter example that illustrates and explains the error

MPI Barrier Analysis – Try it

Add some barriers:

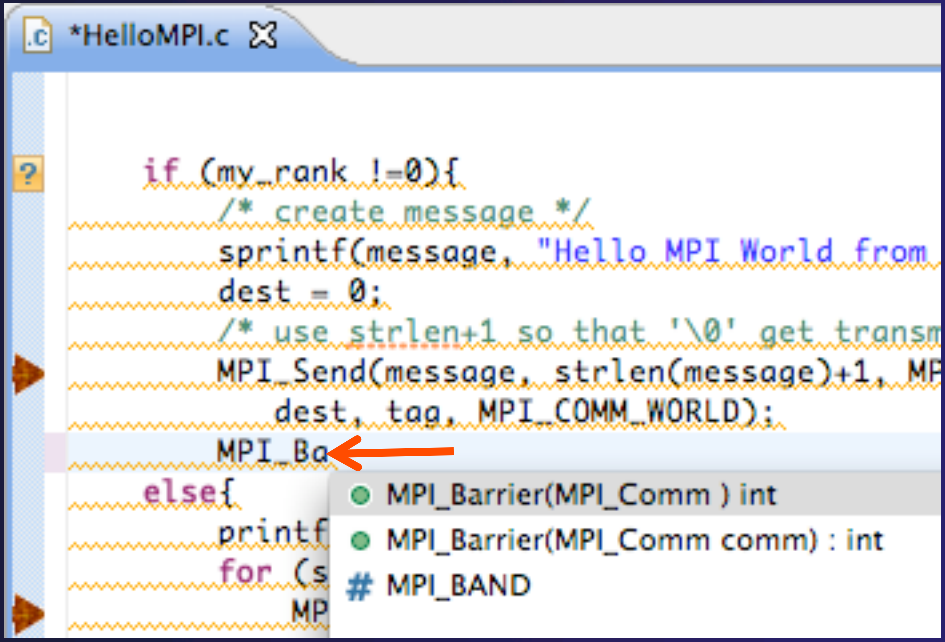
✦ Inside the sample if(rank…) add a barrier:

✦Use Content Assist to help you type

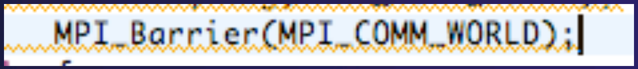
✦Type: MPI_ and press Ctrl-space. See completion alternatives. Keep typing until you see MPI_Barrier and hit enter.

✦For args, start typing MPI_Comm_ etc and it will also complete MPI_COMM_WORLD

✦Add the same barrier statement at the end of the **else** as well.



```
*HelloMPI.c

    if (my_rank !=0){
        /* create message */
        sprintf(message, "Hello MPI World from
        dest = 0;
        /* use strlen+1 so that '\0' get transm
        MPI_Send(message, strlen(message)+1, MP
            dest, tag, MPI_COMM_WORLD);
        MPI_Ba
    else{
        printf
        for (s
            MP
```



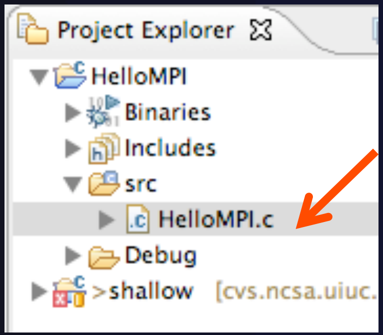
```
MPI_Barrier(MPI_COMM_WORLD);
```

Resulting statement

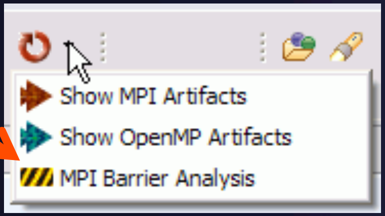
MPI Barrier Analysis – Try it (2)

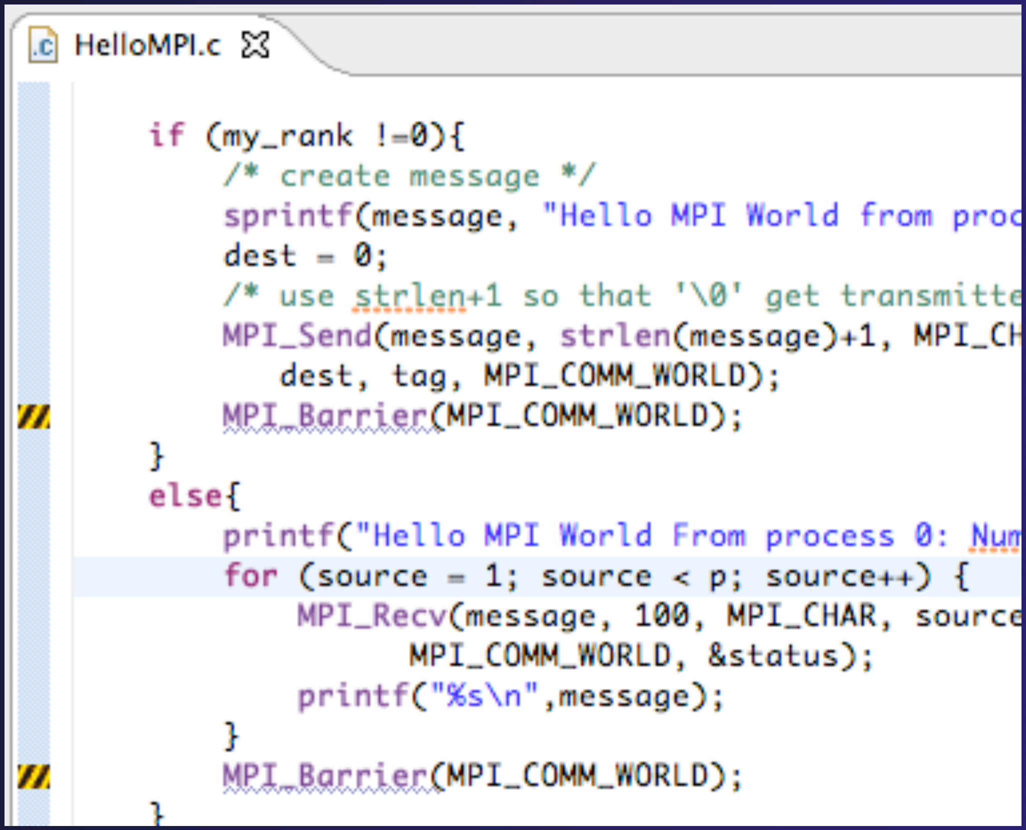
Run the Analysis:

✦In the Project Explorer, Select the source file (or directory, or project) of file(s) to analyze



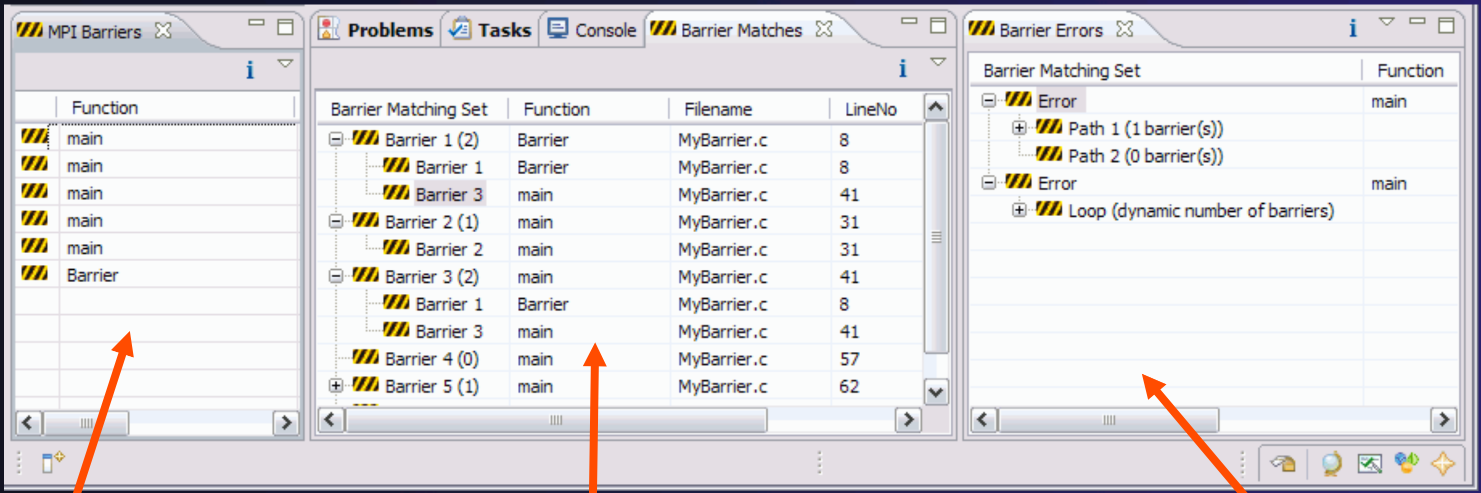
✦ Select the MPI Barrier Analysis action in the menu





```
    if (my_rank !=0){
        /* create message */
        sprintf(message, "Hello MPI World from proc
        dest = 0;
        /* use strlen+1 so that '\0' get transmitte
        MPI_Send(message, strlen(message)+1, MPI_CH
           dest, tag, MPI_COMM_WORLD);
        MPI_Barrier(MPI_COMM_WORLD);
    }
    else{
        printf("Hello MPI World From process 0: Num
        for (source = 1; source < p; source++) {
            MPI_Recv(message, 100, MPI_CHAR, source
                  MPI_COMM_WORLD, &status);
            printf("%s\n",message);
        }
        MPI_Barrier(MPI_COMM_WORLD);
    }
```

MPI Barrier Analysis - views



	Function
🟨	main
🟨	main
🟨	main
🟨	main
🟨	main
🟨	Barrier

Barrier Matching Set	Function	Filename	LineNo
⊟ Barrier 1 (2)	Barrier	MyBarrier.c	8
Barrier 1	Barrier	MyBarrier.c	8
Barrier 3	main	MyBarrier.c	41
⊟ Barrier 2 (1)	main	MyBarrier.c	31
Barrier 2	main	MyBarrier.c	31
⊟ Barrier 3 (2)	main	MyBarrier.c	41
Barrier 1	Barrier	MyBarrier.c	8
Barrier 3	main	MyBarrier.c	41
Barrier 4 (0)	main	MyBarrier.c	57
⊞ Barrier 5 (1)	main	MyBarrier.c	62

Barrier Matching Set	Function
⊟ Error	main
⊞ Path 1 (1 barrier(s))	
Path 2 (0 barrier(s))	
⊟ Error	main
⊞ Loop (dynamic number of barriers)	

MPI Barriers view

Simply lists the barriers

Like MPI Artifacts view, double-click to navigate to source code line (all 3 views)

Barrier Matches view

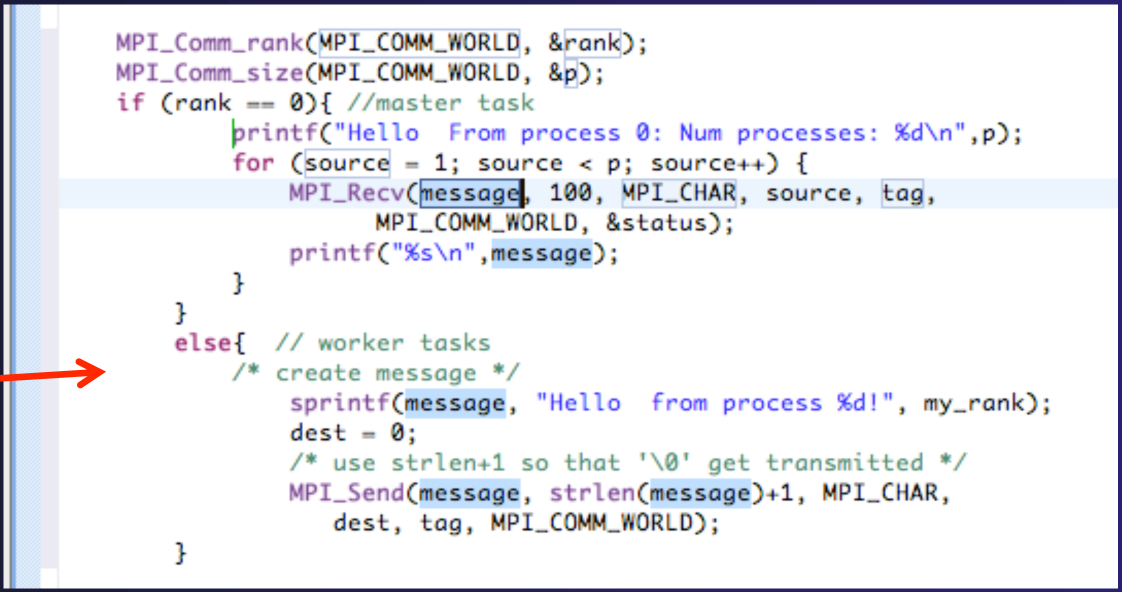
Groups barriers that match together in a barrier set – all processes must go through a barrier in the set to prevent a deadlock

Barrier Errors view

If there are errors, a counter-example shows paths with mismatched number of barriers

MPI Templates

- Allows quick entry of common patterns in MPI programming
- Example: MPI send-receive
- Enter: mpisr <ctrl-space>
- Expands to

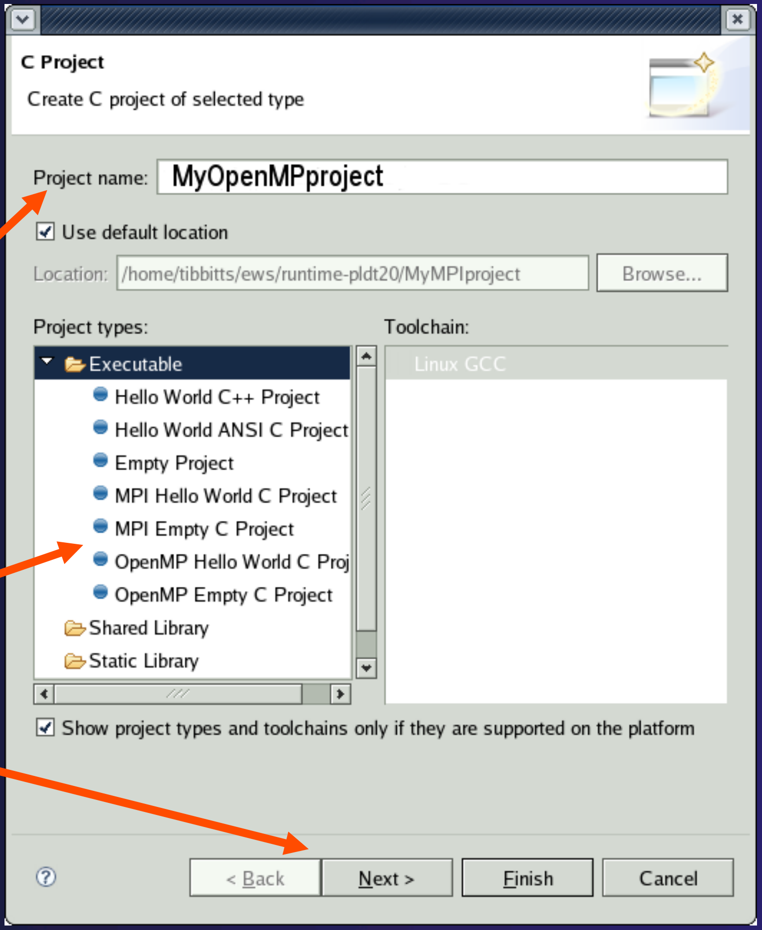


```
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &p);
if (rank == 0){ //master task
        printf("Hello  From process 0: Num processes: %d\n",p);
        for (source = 1; source < p; source++) {
            MPI_Recv(message, 100, MPI_CHAR, source, tag,
                  MPI_COMM_WORLD, &status);
            printf("%s\n",message);
        }
    }
    else{  // worker tasks
        /* create message */
            sprintf(message, "Hello  from process %d!", my_rank);
            dest = 0;
            /* use strlen+1 so that '\0' get transmitted */
            MPI_Send(message, strlen(message)+1, MPI_CHAR,
               dest, tag, MPI_COMM_WORLD);
    }
```

- Eclipse preferences: add more!
    - C/C++ > Editor > Templates
- Extend to other common patterns

OpenMP Managed Build Project

- If you haven't set up OpenMP preferences e.g. include file location, do it now
- Create a new OpenMP project
  - **File▶New▶C Project**
  - Name the project e.g. 'MyOpenMPproject'
  - Select **OpenMP Hello World C Project**
  - Select **Next,** then fill in other info like MPI project


C Project

Create C project of selected type

Project name: MyOpenMPproject

☑ Use default location

Location: /home/tibbitts/ews/runtime-pldt20/MyMPIproject

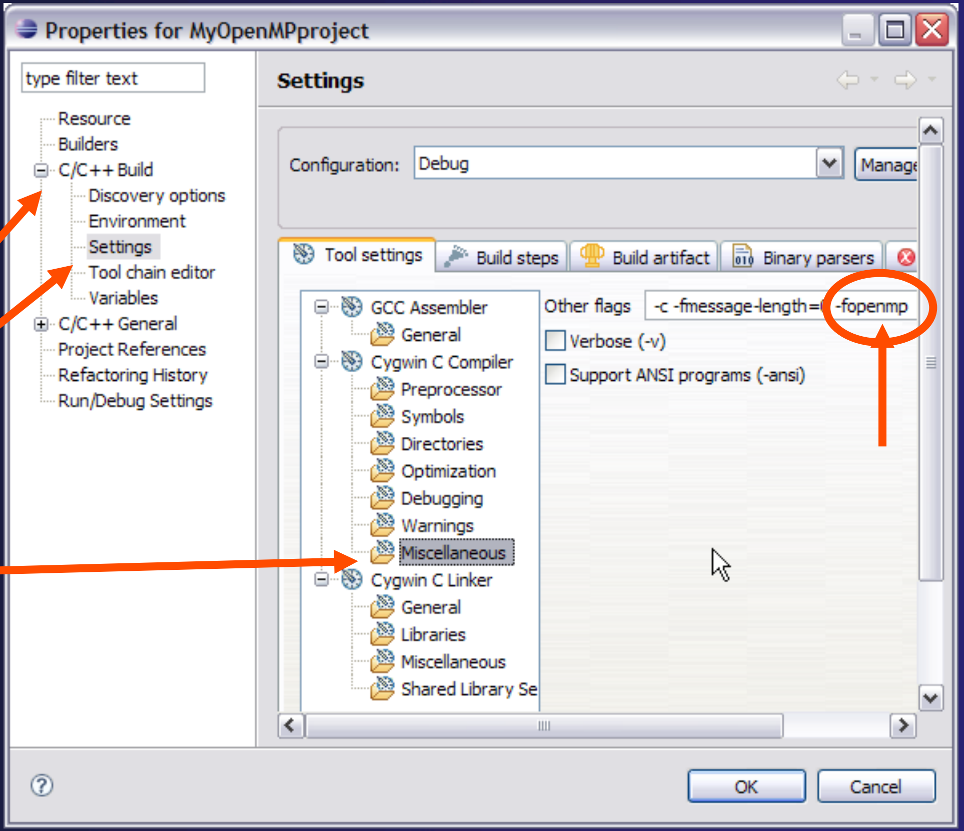
Project types:
- Executable
  - Hello World C++ Project
  - Hello World ANSI C Project
  - Empty Project
  - MPI Hello World C Project
  - MPI Empty C Project
  - OpenMP Hello World C Proj
  - OpenMP Empty C Project
- Shared Library
- Static Library

Toolchain: Linux GCC

☑ Show project types and toolchains only if they are supported on the platform

Setting OpenMP Special Build Options

- OpenMP typically requires special compiler options.
  - Open the project properties
  - Select **C/C++ Build**
  - Select **Settings**
  - Select **C Compiler**
    - In Miscellaneous, add option(s).



Configuration: Debug

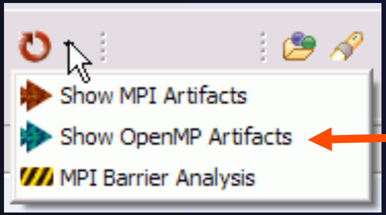
Other flags: -c -fmessage-length=0 -fopenmp

☐ Verbose (-v)

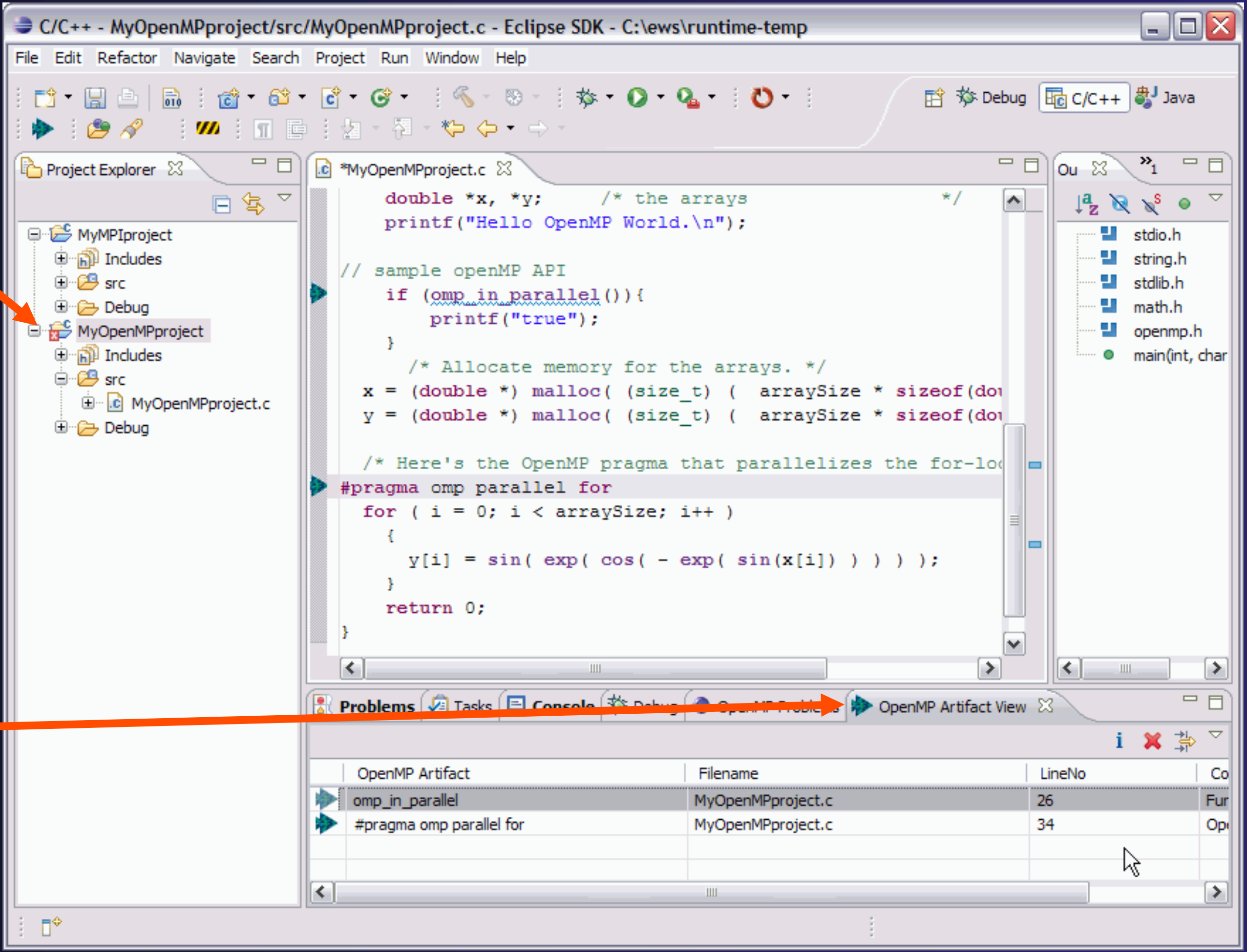
☐ Support ANSI programs (-ansi)

Show OpenMP Artifacts

- Select source file, folder, or project
- Run analysis



- See artifacts in **OpenMP Artifact view**

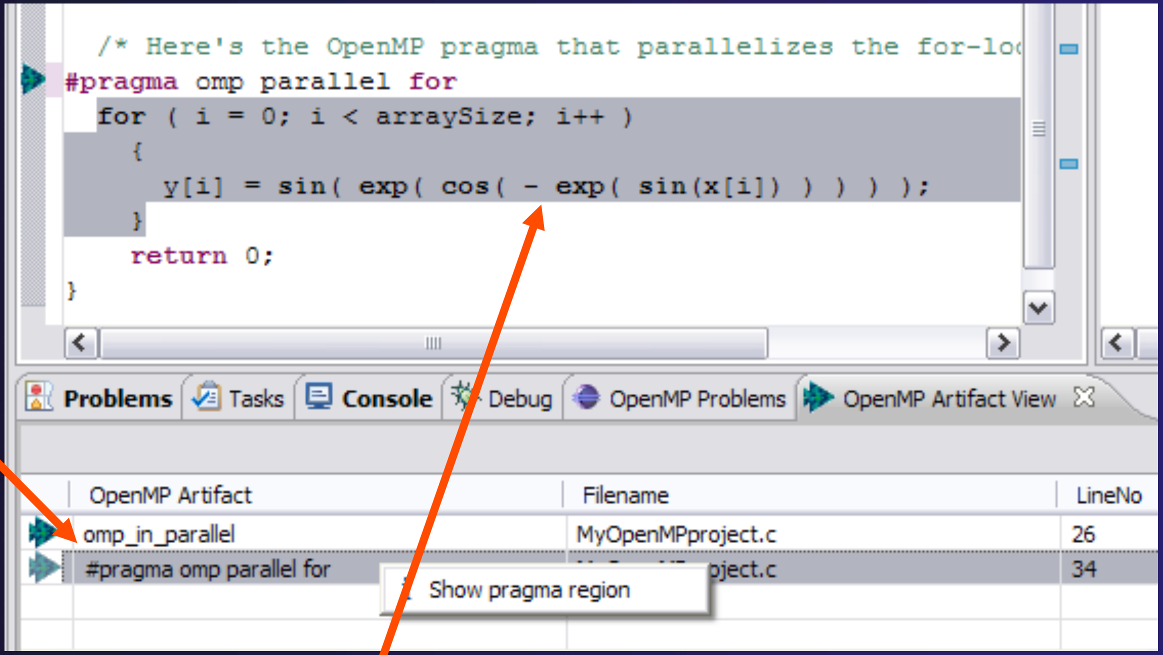


```
    double *x, *y;      /* the arrays                                */
    printf("Hello OpenMP World.\n");

// sample openMP API
    if (omp_in_parallel()){
        printf("true");
    }
      /* Allocate memory for the arrays. */
  x = (double *) malloc( (size_t) (  arraySize * sizeof(dou
  y = (double *) malloc( (size_t) (  arraySize * sizeof(dou

  /* Here's the OpenMP pragma that parallelizes the for-loo
#pragma omp parallel for
  for ( i = 0; i < arraySize; i++ )
    {
      y[i] = sin( exp( cos( - exp( sin(x[i]) ) ) ) );
    }
    return 0;
}
```

OpenMP Artifact	Filename	LineNo	Co
omp_in_parallel	MyOpenMPproject.c	26	Fur
#pragma omp parallel for	MyOpenMPproject.c	34	Op

Show Pragma Region

- Run OpenMP analysis
- Right click on pragma in artifact view
- Select **Show pragma region**
- See highlighted region in C editor

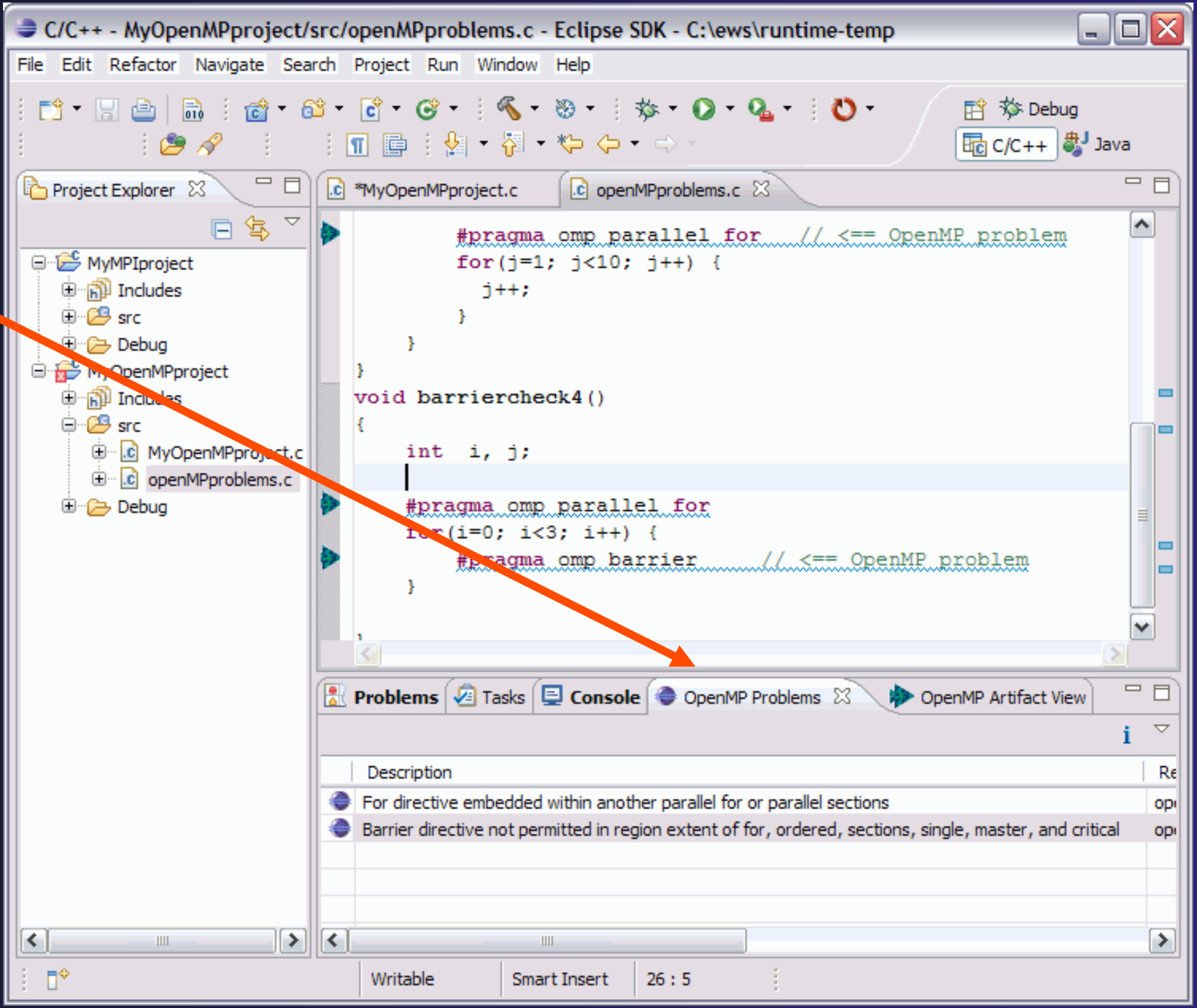


```
  /* Here's the OpenMP pragma that parallelizes the for-loo
#pragma omp parallel for
  for ( i = 0; i < arraySize; i++ )
    {
      y[i] = sin( exp( cos( - exp( sin(x[i]) ) ) ) );
    }
    return 0;
}
```

OpenMP Artifact	Filename	LineNo
omp_in_parallel	MyOpenMPproject.c	26
#pragma omp parallel for	...oject.c	34

Show OpenMP Problems

- Select **OpenMP problems view**
- Will identify standard OpenMP restrictions

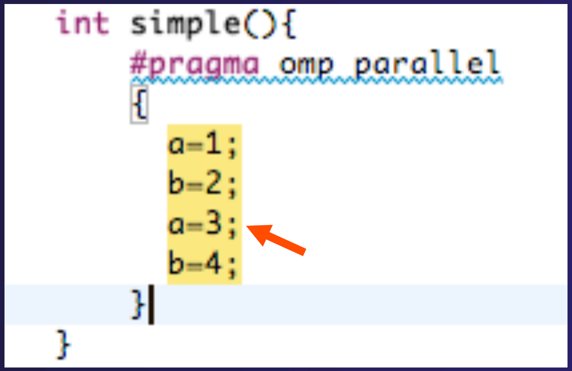


```
        #pragma omp parallel for    // <== OpenMP problem
        for(j=1; j<10; j++) {
          j++;
        }
    }
}
void barriercheck4()
{
    int  i, j;

    #pragma omp parallel for
    for(i=0; i<3; i++) {
        #pragma omp barrier     // <== OpenMP problem
    }
```

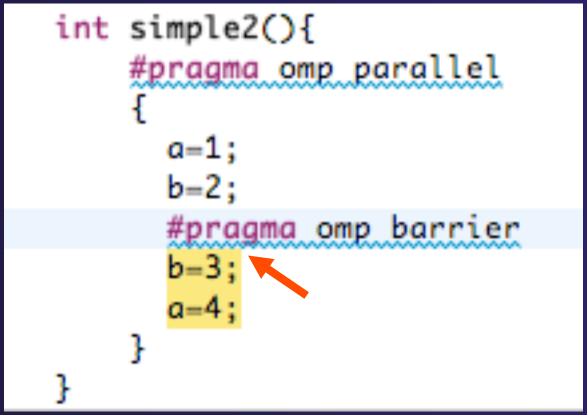
Description	Re
For directive embedded within another parallel for or parallel sections	op
Barrier directive not permitted in region extent of for, ordered, sections, single, master, and critical	op

OpenMP: Show Concurrency

- Highlight a statement
- Select the context menu on the highlighted statement, and click **Show concurrency**
- Other statements will be highlighted in yellow
- The yellow highlighted statements *might* execute concurrently to the selected statement



```
int simple(){
    #pragma omp parallel
    {
        a=1;
        b=2;
        a=3;
        b=4;
    }
}
```

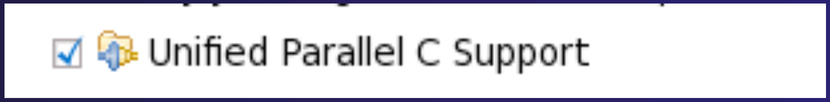


```
int simple2(){
    #pragma omp parallel
    {
        a=1;
        b=2;
        #pragma omp barrier
        b=3;
        a=4;
    }
}
```

UPC Support

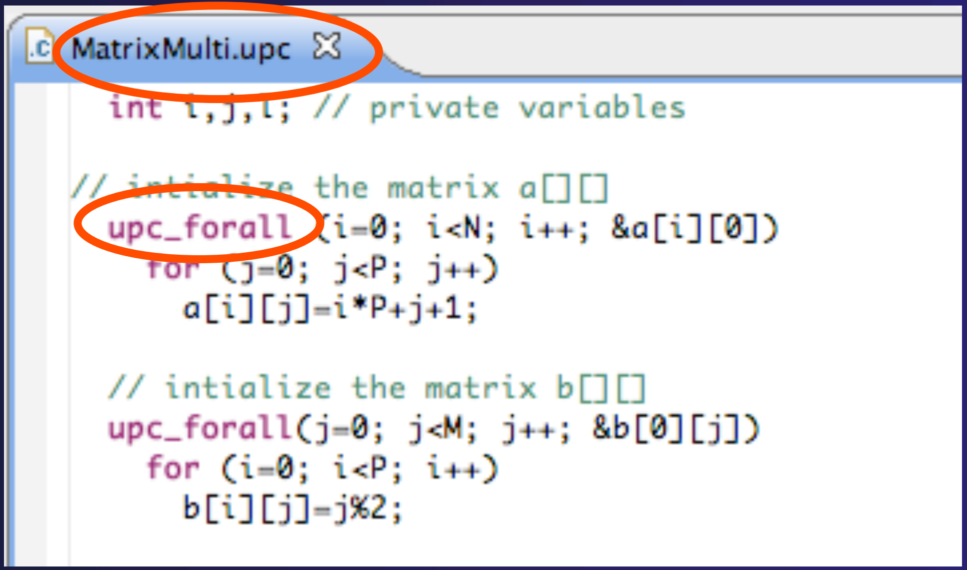
- To see UPC support in C editor, install the optional feature from CDT
- See Also: http://wiki.eclipse.org/PTP/other_tools_setup#Using_UPC_features

Under Optional Features



☑ Unified Parallel C Support

- Filetypes of "upc" will get UPC syntax high-lighting, content assist, etc.
- Use preferences to change default for *.c if you like



MatrixMulti.upc

```
  int i,j,l; // private variables

// intialize the matrix a[][]
  upc_forall(i=0; i<N; i++; &a[i][0])
    for (j=0; j<P; j++)
      a[i][j]=i*P+j+1;

  // intialize the matrix b[][]
  upc_forall(j=0; j<M; j++; &b[0][j])
    for (i=0; i<P; i++)
      b[i][j]=j%2;
```

More Advanced Features

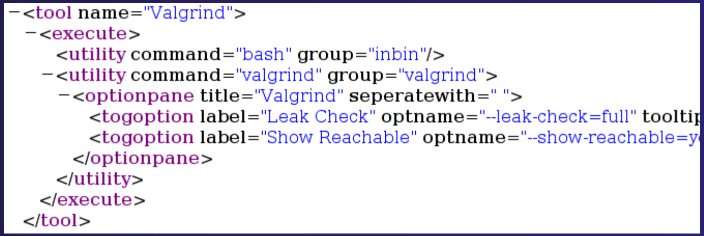
- ETFw – External Tools Framework and TAU, Tuning and Analysis Utilities
  - Wyatt Spear, U. Oregon
- PPW – Parallel Performance Wizard
  - Max Billingsley III, U. Florida
- ISP – In-situ Partial Order: Dynamic Formal Verification for MPI
  - Alan Humphrey, U. Utah

PTP/External Tools Framework

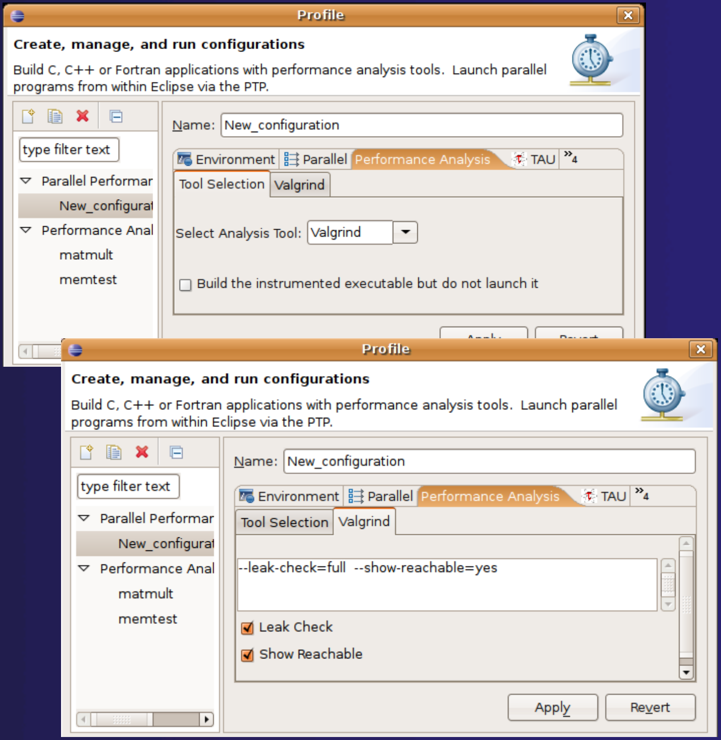
formerly "Performance Tools Framework"

Goal:

- **Reduce the "eclipse plumbing" necessary to integrate tools**
- Provide integration for instrumentation, measurement, and analysis for a variety of performance tools
  - Dynamic Tool Definitions: Workflows & UI
  - Tools and tool workflows are specified in an XML file
  - Tools are selected and configured in the launch configuration window
  - Output is generated, managed and analyzed as specified in the workflow



```
-<tool name="Valgrind">
  -<execute>
     <utility command="bash" group="inbin"/>
    -<utility command="valgrind" group="valgrind">
      -<optionpane title="Valgrind" seperatewith=" ">
          <togoption label="Leak Check" optname="--leak-check=full" tooltip
          <togoption label="Show Reachable" optname="--show-reachable=y
        </optionpane>
     </utility>
   </execute>
 </tool>
```



Profile

Create, manage, and run configurations

Build C, C++ or Fortran applications with performance analysis tools. Launch parallel programs from within Eclipse via the PTP.

Name: New_configuration

Select Analysis Tool: Valgrind

Build the instrumented executable but do not launch it

Profile

Create, manage, and run configurations

Build C, C++ or Fortran applications with performance analysis tools. Launch parallel programs from within Eclipse via the PTP.

Name: New_configuration

--leak-check=full --show-reachable=yes

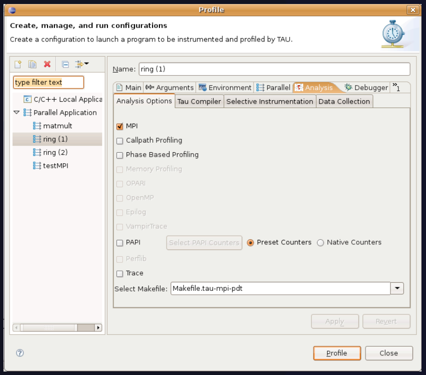
☑ Leak Check

☑ Show Reachable

PTP TAU plug-ins

http://www.cs.uoregon.edu/research/tau

- TAU (Tuning and Analysis Utilities)
- First implementation of External Tools Framework
- Eclipse plug-ins wrap TAU functions, make them available from Eclipse
- Compatible with Photran and CDT projects and with PTP parallel application launching
- Other plug-ins launch Paraprof from Eclipse too


Profile

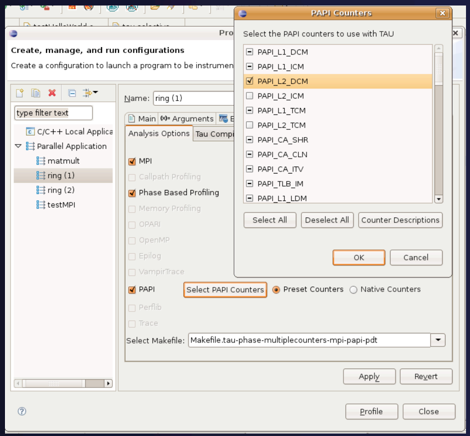
Create, manage, and run configurations

Create a configuration to launch a program to be instrumented and profiled by TAU.

Name: ring (1)

Analysis Options:
- [✓] MPI
- [ ] Callpath Profiling
- [ ] Phase Based Profiling
- [ ] Memory Profiling
- [ ] OPARI
- [ ] OpenMP
- [ ] Epilog
- [ ] VampirTrace
- [ ] PAPI — (● Preset Counters ○ Native Counters)
- [ ] Perflib
- [ ] Trace

Select Makefile: Makefile.tau-mpi-pdt


PAPI Counters

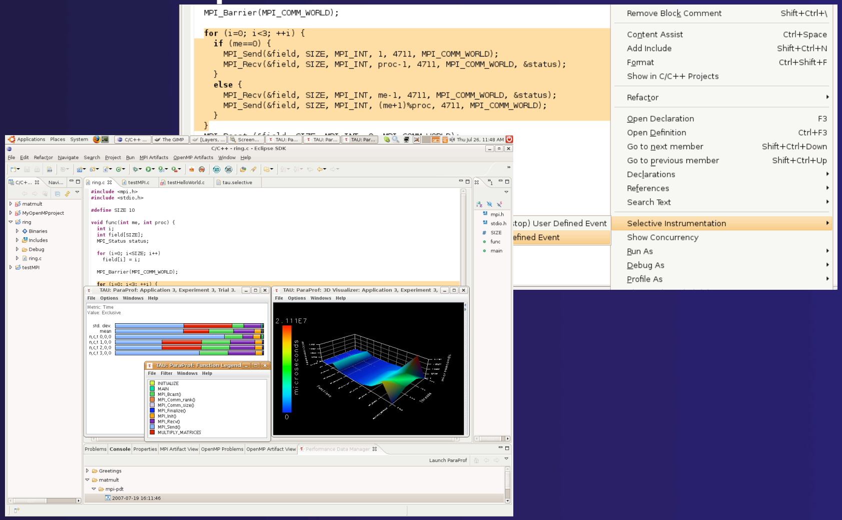
Select the PAPI counters to use with TAU

- [ ] PAPI_L1_DCM
- [ ] PAPI_L1_ICM
- [✓] PAPI_L2_DCM
- [ ] PAPI_L2_ICM
- [ ] PAPI_L1_TCM
- [ ] PAPI_L2_TCM
- [ ] PAPI_CA_SHR
- [ ] PAPI_CA_CLN
- [ ] PAPI_CA_ITV
- [ ] PAPI_TLB_IM
- [ ] PAPI_L1_LDM

Name: ring (1)

Analysis Options:
- [✓] MPI
- [ ] Callpath Profiling
- [✓] Phase Based Profiling
- [ ] Memory Profiling
- [ ] OPARI
- [ ] OpenMP
- [ ] Epilog
- [ ] VampirTrace
- [✓] PAPI — (● Preset Counters ○ Native Counters)
- [ ] Perflib
- [ ] Trace

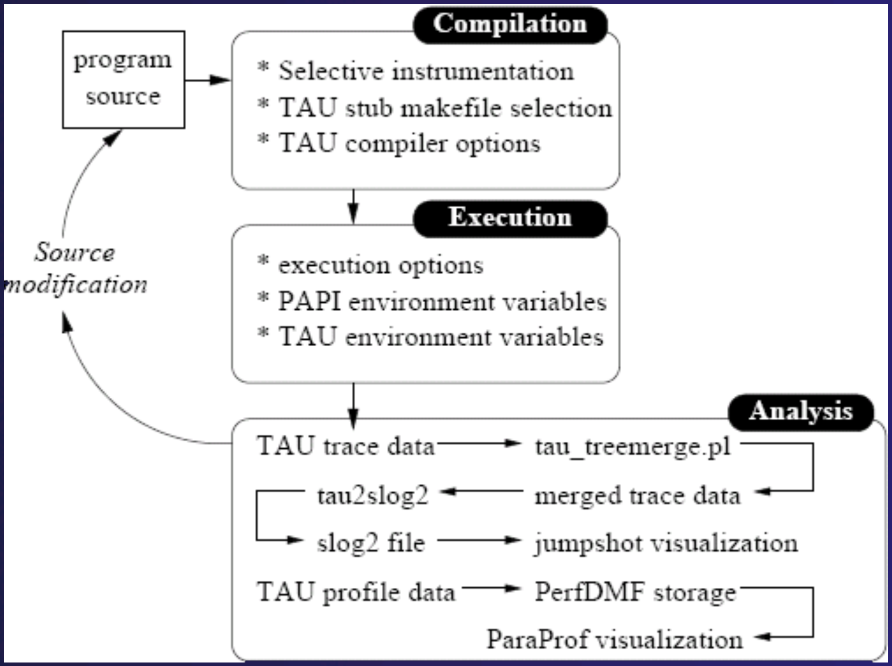
Select Makefile: Makefile.tau-phase-multiplecounters-mpi-papi-pdt


```
MPI_Barrier(MPI_COMM_WORLD);

for (i=0; i<3; ++i) {
  if (me==0) {
    MPI_Send(&field, SIZE, MPI_INT, 1, 4711, MPI_COMM_WORLD);
    MPI_Recv(&field, SIZE, MPI_INT, proc-1, 4711, MPI_COMM_WORLD, &status);
  }
  else {
    MPI_Recv(&field, SIZE, MPI_INT, me-1, 4711, MPI_COMM_WORLD, &status);
    MPI_Send(&field, SIZE, MPI_INT, (me+1)%proc, 4711, MPI_COMM_WORLD);
  }
}
```

Remove Block Comment Shift+Ctrl+\
Content Assist Ctrl+Space
Add Include Shift+Ctrl+N
Format Ctrl+Shift+F
Show in C/C++ Projects
Refactor
Open Declaration F3
Open Definition Ctrl+F3
Go to next member Shift+Ctrl+Down
Go to previous member Shift+Ctrl+Up
Declarations
References
Search Text
Selective Instrumentation
Show Concurrency
Run As
Debug As
Profile As
top) User Defined Event
efined Event
C/C++ - ring.c - Eclipse SDK
testHelloWorld.c
tau.selective
TAU: ParaProf: 3D Visualizer: Application 3, Experiment 3,
TAU: ParaProf: Function Legend
2.111E7

Initial Goal: TAU Integration

- TAU: Tuning and Analysis Utilities
  - Performance data collection and analysis for HPC codes
  - Numerous features
  - Command line interface
- The TAU Workflow:
  - Instrumentation
  - Execution
  - Analysis



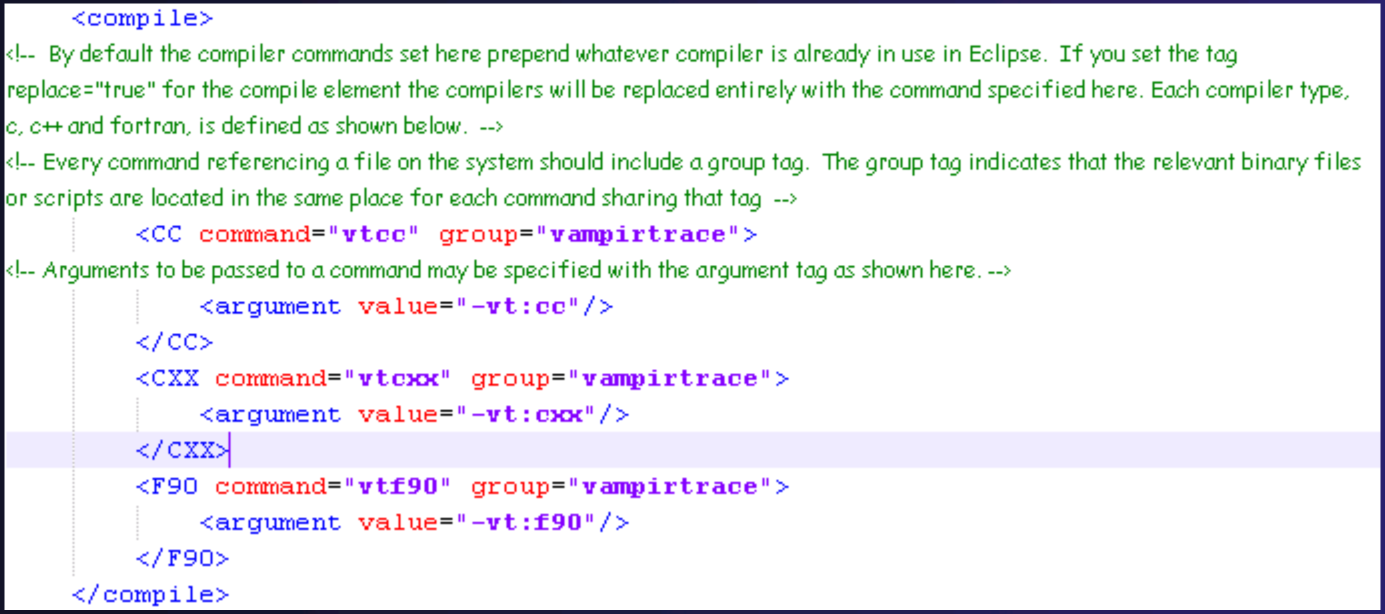
ETFw Motivation

- There are numerous command-line oriented development tools employed in HPC
- These can be complicated or time consuming to use
- IDE integration for individual development tools is slow and inconsistent
- We want all our development tools in one place with one interface
- We want our development tools to work together

ETFw: Development Tool Workflows

- Variations on ‘Compile, Execute, Analyze-Results’ are common to most software development
- These steps may be tedious and time consuming, especially over multiple iterations
- By defining both tool interfaces and behavior in an XML document these steps can be simplified and automated

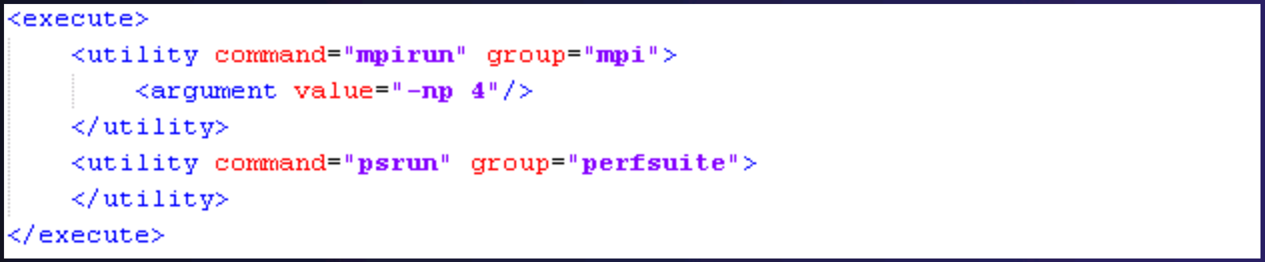
ETFw: The Build Phase



```
    <compile>
<!--  By default the compiler commands set here prepend whatever compiler is already in use in Eclipse.  If you set the tag
replace="true" for the compile element the compilers will be replaced entirely with the command specified here. Each compiler type,
c, c++ and fortran, is defined as shown below.  -->
<!-- Every command referencing a file on the system should include a group tag.  The group tag indicates that the relevant binary files
or scripts are located in the same place for each command sharing that tag  -->
        <CC command="vtcc" group="vampirtrace">
<!-- Arguments to be passed to a command may be specified with the argument tag as shown here. -->
            <argument value="-vt:cc"/>
        </CC>
        <CXX command="vtcxx" group="vampirtrace">
            <argument value="-vt:cxx"/>
        </CXX>
        <F90 command="vtf90" group="vampirtrace">
            <argument value="-vt:f90"/>
        </F90>
    </compile>
```

- Set compilers and arguments for each language
- Define UI for compiler/compiler-wrapper configuration

ETFw: The Execution Phase

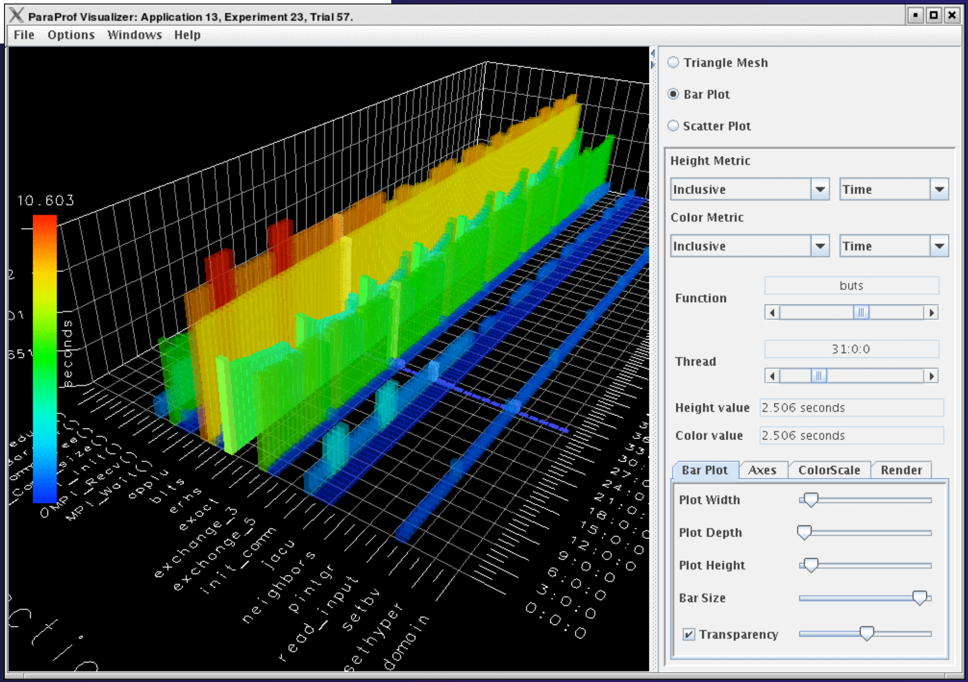


```xml
<execute>
    <utility command="mpirun" group="mpi">
        <argument value="-np 4"/>
    </utility>
    <utility command="psrun" group="perfsuite">
    </utility>
</execute>
```

- Specify composed execution tools such as Perfsuite or Valgrind
- Set launch environment variables
- Define variables and tool options in XML or provide a UI in the IDE
- Integrates with PTP parallel launch environment

ETFw: The Analysis/Post-Processing Phase

```
<analyze>
    <utility command="expert" group="kojak">
        <argument value="a.elg"/>
    </utility>
    <utility command="paraprof" group="tau">
        <argument value="a.cube"/>
    </utility>
</analyze>
```

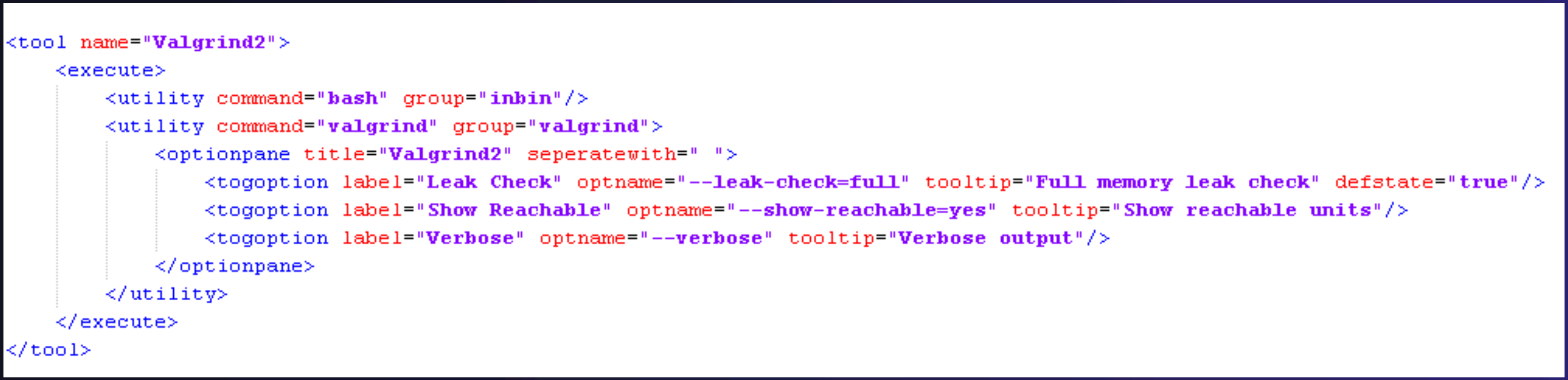


( ) Triangle Mesh
(●) Bar Plot
( ) Scatter Plot

Height Metric: Inclusive, Time
Color Metric: Inclusive, Time
Function: buts
Thread: 31:0:0
Height value: 2.506 seconds
Color value: 2.506 seconds
✓ Transparency

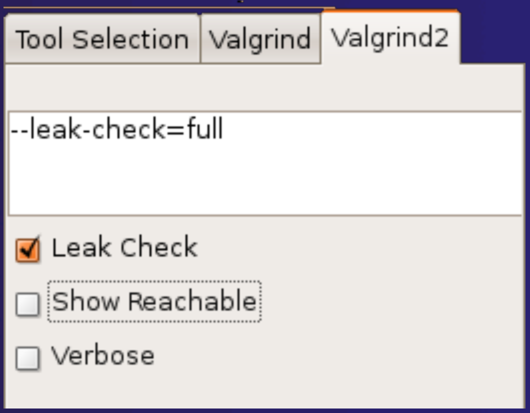
- Sequentially run tools on program output
- Launch external visualization tools

ETFw: XML-Defined UI Components



```
<tool name="Valgrind2">
    <execute>
        <utility command="bash" group="inbin"/>
        <utility command="valgrind" group="valgrind">
            <optionpane title="Valgrind2" seperatewith=" ">
                <togoption label="Leak Check" optname="--leak-check=full" tooltip="Full memory leak check" defstate="true"/>
                <togoption label="Show Reachable" optname="--show-reachable=yes" tooltip="Show reachable units"/>
                <togoption label="Verbose" optname="--verbose" tooltip="Verbose output"/>
            </optionpane>
        </utility>
    </execute>
</tool>
```

- Each pane constructs a set of options sent to a tool or a set of environment variables
- Numerous options for converting a command line interface into an intelligent GUI without Eclipse coding

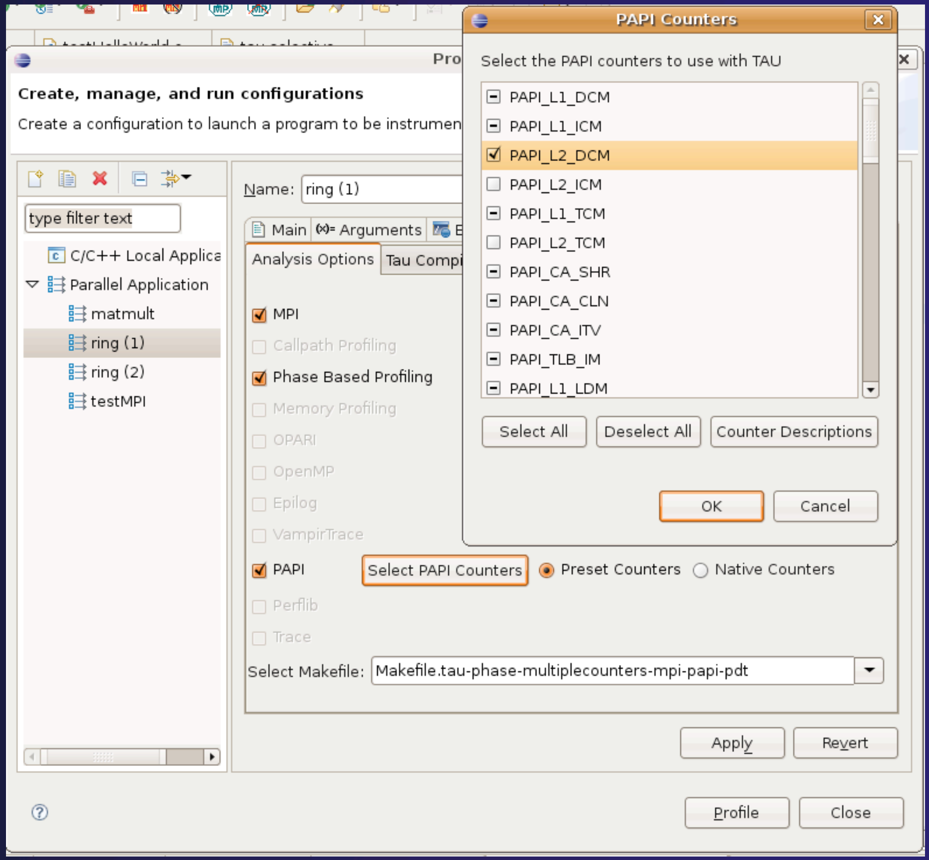


```
--leak-check=full
```

[✓] Leak Check
[ ] Show Reachable
[ ] Verbose

ETFw: Advanced Components

- Extension points allow integration with UIs and workflow behavior too complex to define in XML
- Logical and iterative workflows for successive executions and parametric studies


Create, manage, and run configurations

Create a configuration to launch a program to be instrumen

type filter text

C/C++ Local Applica
Parallel Application
matmult
ring (1)
ring (2)
testMPI

Name: ring (1)

Main | Arguments | Analysis Options | Tau Compi

- [x] MPI
- [ ] Callpath Profiling
- [x] Phase Based Profiling
- [ ] Memory Profiling
- [ ] OPARI
- [ ] OpenMP
- [ ] Epilog
- [ ] VampirTrace
- [x] PAPI — Select PAPI Counters — [✓ Preset Counters] Native Counters
- [ ] Perflib
- [ ] Trace

Select Makefile: Makefile.tau-phase-multiplecounters-mpi-papi-pdt

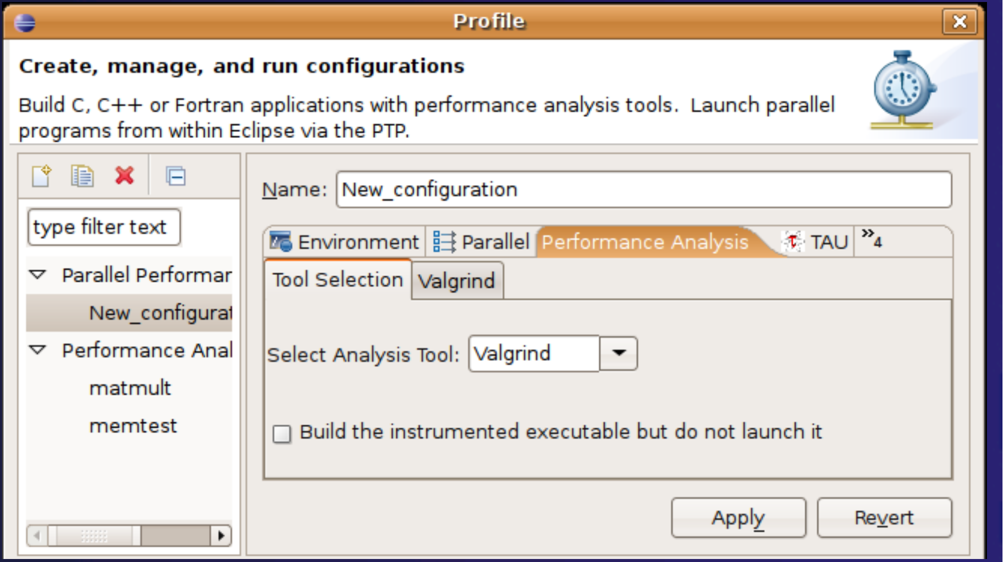
PAPI Counters

Select the PAPI counters to use with TAU

- [ ] PAPI_L1_DCM
- [ ] PAPI_L1_ICM
- [x] PAPI_L2_DCM
- [ ] PAPI_L2_ICM
- [ ] PAPI_L1_TCM
- [ ] PAPI_L2_TCM
- [ ] PAPI_CA_SHR
- [ ] PAPI_CA_CLN
- [ ] PAPI_CA_ITV
- [ ] PAPI_TLB_IM
- [ ] PAPI_L1_LDM

ETFw: Using Workflows

- New workflows are added to the ETFw launch configuration system
- Multiple workflow configurations can be defined and saved for different use cases
- XML Workflow definitions can be saved and reused in different environments



Create, manage, and run configurations

Build C, C++ or Fortran applications with performance analysis tools. Launch parallel programs from within Eclipse via the PTP.

type filter text
- Parallel Performar
  - New_configurat
- Performance Anal
  - matmult
  - memtest

Name: New_configuration

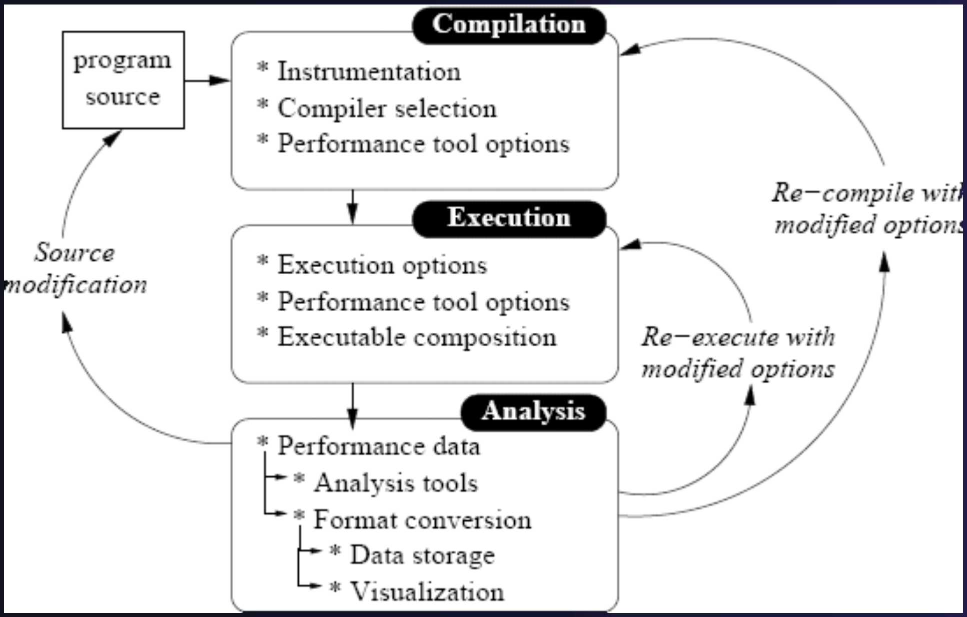
Environment | Parallel | Performance Analysis | TAU | »4

Tool Selection | Valgrind

Select Analysis Tool: Valgrind

☐ Build the instrumented executable but do not launch it

ETFw: General Purpose Workflow



- ✦ Automated
- ✦ Generalized
- ✦ Quick performance analysis and other development tool integration
- ✦ Exposes tool capabilities to the user

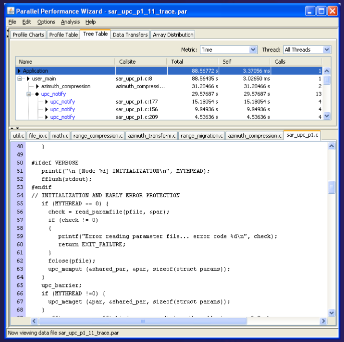
ETFw: Continuing Development

- Integration with PTP Remote Development Tools
- Additional options for GUI definition
- Generalization of TAU specific features such as hardware counter selection and performance data storage

- Contact: Wyatt Spear

Parallel Performance Wizard (PPW)

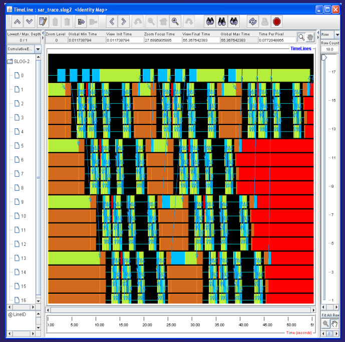
- Full-featured performance tool for PGAS programming models
  - Currently supports UPC, SHMEM, and MPI
  - Extensible to support other models
  - PGAS support by way of Global Address Space Performance (GASP) interface (http://gasp.hcs.ufl.edu)
- PPW v2.1 features:
  - Easy-to-use scripts for backend data collection
  - User-friendly GUI with familiar visualizations
  - Advanced automatic analysis support
- More information and free download: http://ppw.hcs.ufl.edu

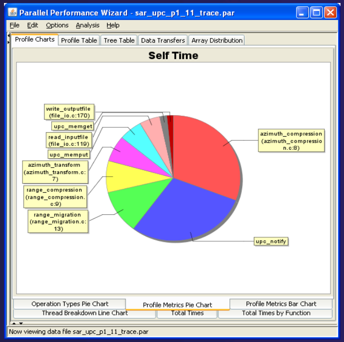


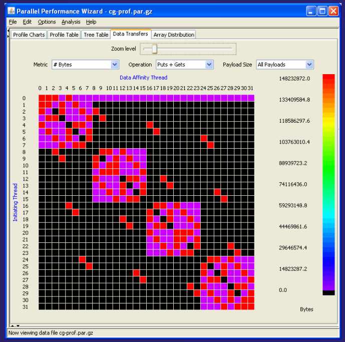
Name	Callsite	Total	Self	Calls
Application		88.56772 s	3.37056 ms	1
user_main	sar_upc_p1.c:8	88.56435 s	3.02650 ms	1
azimuth_compression	azimuth_compressi...	31.20466 s	31.20466 s	2
upc_notify		29.57687 s	29.57687 s	13
upc_notify	sar_upc_p1.c:177	15.18054 s	15.18054 s	4
upc_notify	sar_upc_p1.c:156	9.84936 s	9.84936 s	4
upc_notify	sar_upc_p1.c:209	4.53636 s	4.53636 s	4

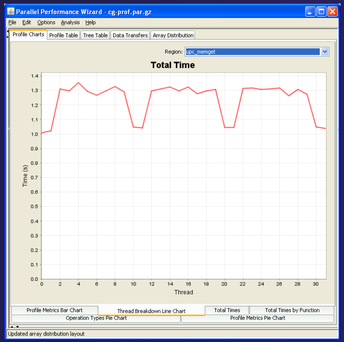
```
    }

#ifdef VERBOSE
    printf("\n [Node %d] INITIALIZATION\n", MYTHREAD);
    fflush(stdout);
#endif
// INITIALIZATION AND EARLY ERROR PROTECTION
    if (MYTHREAD == 0) {
      check = read_paramfile(pfile, &par);
      if (check != 0)
      {
         printf("Error reading parameter file... error code %d\n", check);
         return EXIT_FAILURE;
      }
      fclose(pfile);
      upc_memput (&shared_par, &par, sizeof(struct params));
    }
    upc_barrier;
    if (MYTHREAD !=0) {
      upc_memget (&par, &shared_par, sizeof(struct params));
    }
```

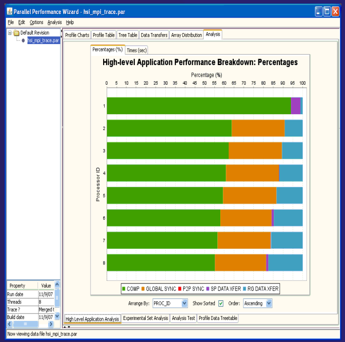
Now viewing data file sar_upc_p1_11_trace.par




Parallel Performance Wizard - sar_upc_p1_11_trace.par
Self Time
write_outputfile (file_io.c:170)
upc_memget
read_inputfile (file_io.c:119)
upc_memput
azimuth_transform (azimuth_transform.c:7)
range_compression (range_compression.c:9)
range_migration (range_migration.c:13)
azimuth_compression (azimuth_compression.c:8)
upc_notify
Operation Types Pie Chart
Profile Metrics Pie Chart
Profile Metrics Bar Chart
Thread Breakdown Line Chart
Total Times
Total Times by Function
Now viewing data file sar_upc_p1_11_trace.par

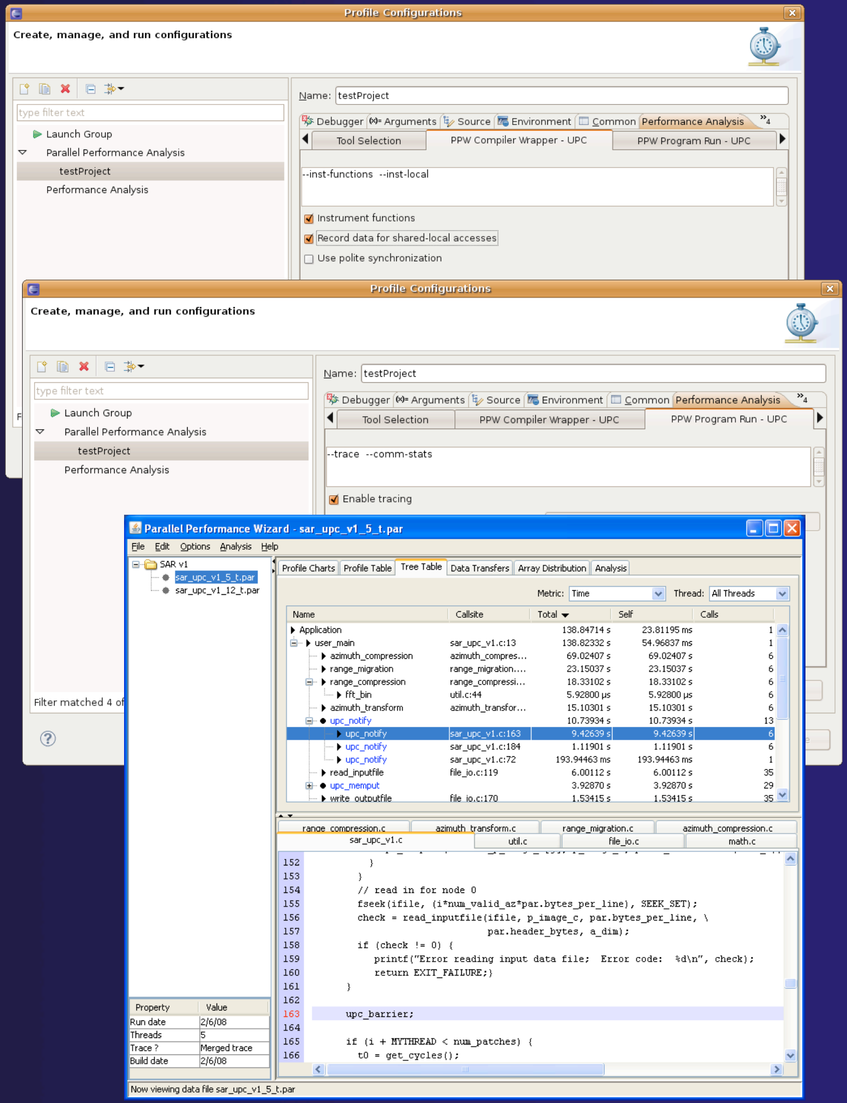

Parallel Performance Wizard - cg.prof.par.gz
Data Transfers
Zoom level
Metric: # Bytes
Operation: Puts + Gets
Payload Size: All Payloads
Data Affinity Thread
Initiating Thread
Bytes
Now viewing data file cg.prof.par.gz


Parallel Performance Wizard - cg.prof.par.gz
Region: upc_memget
Total Time
Time (s)
Thread
Updated array distribution layout


Parallel Performance Wizard - fst_mpi_trace.par
High-level Application Performance Breakdown: Percentages
Percentage (%)
Processor ID
COMP
GLOBAL SYNC
P2P SYNC
SP DATA XFER
RQ DATA XFER

PPW Integration via ETFw

- We implement the ETFw to make PPW's capabilities available within Eclipse
  - Compile with instrumentation, parallel launch with PPW
  - Generates performance data file in workspace, PPW GUI launched
- PPW is often used for UPC application analysis
  - ETFw extended to support UPC
  - Many UPC features in PTP
- For more information:
  - http://ppw.hcs.ufl.edu
  - ppw@hcs.ufl.edu



Profile Configurations

Create, manage, and run configurations

Name: testProject

```
--inst-functions  --inst-local
```

[✓] Instrument functions
[✓] Record data for shared-local accesses
[ ] Use polite synchronization

Profile Configurations

Create, manage, and run configurations

Name: testProject

```
--trace  --comm-stats
```

[✓] Enable tracing

Filter matched 4 of

Parallel Performance Wizard - sar_upc_v1_5_t.par

Metric: Time Thread: All Threads

Name	Callsite	Total	Self	Calls
Application		138.84714 s	23.81195 ms	1
user_main	sar_upc_v1.c:13	138.82332 s	54.96837 ms	1
azimuth_compression	azimuth_compres…	69.02407 s	69.02407 s	6
range_migration	range_migration.…	23.15037 s	23.15037 s	6
range_compression	range_compressi…	18.33102 s	18.33102 s	6
fft_bin	util.c:44	5.92800 µs	5.92800 µs	6
azimuth_transform	azimuth_transfor…	15.10301 s	15.10301 s	6
upc_notify		10.73934 s	10.73934 s	13
upc_notify	sar_upc_v1.c:163	9.42639 s	9.42639 s	6
upc_notify	sar_upc_v1.c:184	1.11901 s	1.11901 s	6
upc_notify	sar_upc_v1.c:72	193.94463 ms	193.94463 ms	1
read_inputfile	file_io.c:119	6.00112 s	6.00112 s	35
upc_memput		3.92870 s	3.92870 s	29
write_outputfile	file_io.c:170	1.53415 s	1.53415 s	35

```
152          }
153        }
154        // read in for node 0
155        fseek(ifile, (i*num_valid_az*par.bytes_per_line), SEEK_SET);
156        check = read_inputfile(ifile, p_image_c, par.bytes_per_line, \
157                               par.header_bytes, a_dim);
158        if (check != 0) {
159          printf("Error reading input data file;  Error code:  %d\n", check);
160          return EXIT_FAILURE;}
161      }
162
163      upc_barrier;
164
165      if (i + MYTHREAD < num_patches) {
166        t0 = get_cycles();
```

Property	Value
Run date	2/6/08
Threads	5
Trace ?	Merged trace
Build date	2/6/08

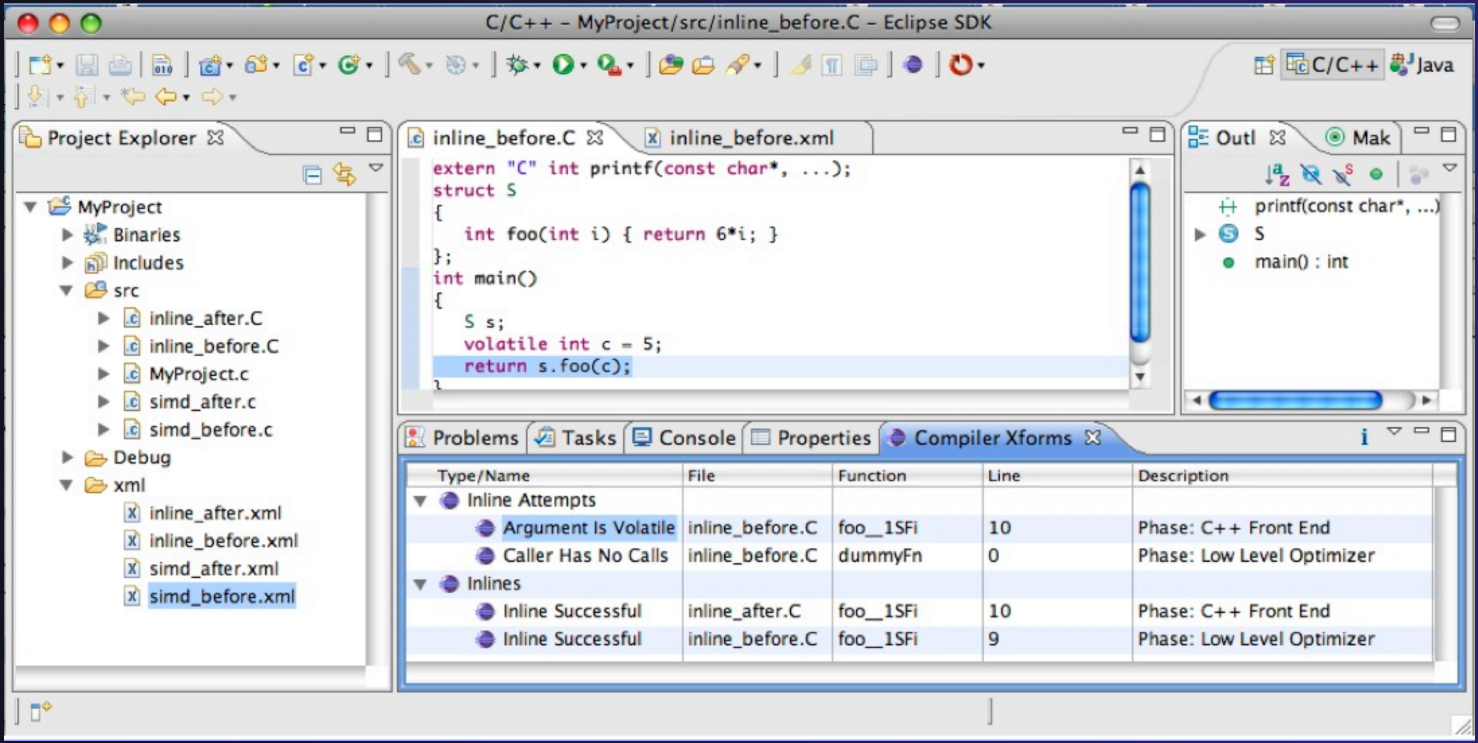
Now viewing data file sar_upc_v1_5_t.par

ETFw Feedback view

- New view to show externally acquired info e.g. from compilers and performance tools via XML, and map to source code lines.
- New extension point for customization
- New to PTP 3.0
- Examples:
  - Compile optimization report: optimizations that were made, and could not be made
  - Performance tool data includes recommendations mapped to source code lines

ETFw Feedback view

- New view makes messages more readable
- Easy navigation to source code lines



```
extern "C" int printf(const char*, ...);
struct S
{
   int foo(int i) { return 6*i; }
};
int main()
{
   S s;
   volatile int c = 5;
   return s.foo(c);
```

Type/Name	File	Function	Line	Description
Inline Attempts				
Argument Is Volatile	inline_before.C	foo__1SFi	10	Phase: C++ Front End
Caller Has No Calls	inline_before.C	dummyFn	0	Phase: Low Level Optimizer
Inlines				
Inline Successful	inline_after.C	foo__1SFi	10	Phase: C++ Front End
Inline Successful	inline_before.C	foo__1SFi	9	Phase: Low Level Optimizer

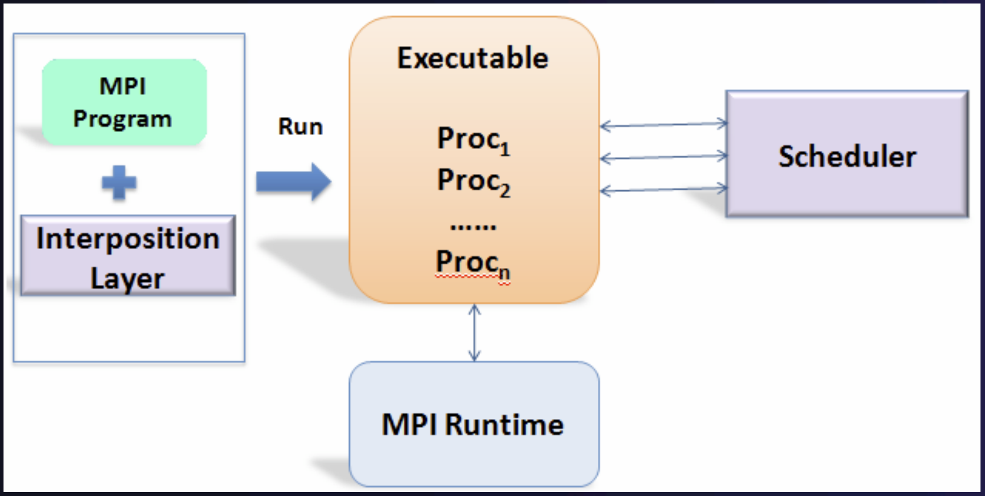
ETFw Feedback view

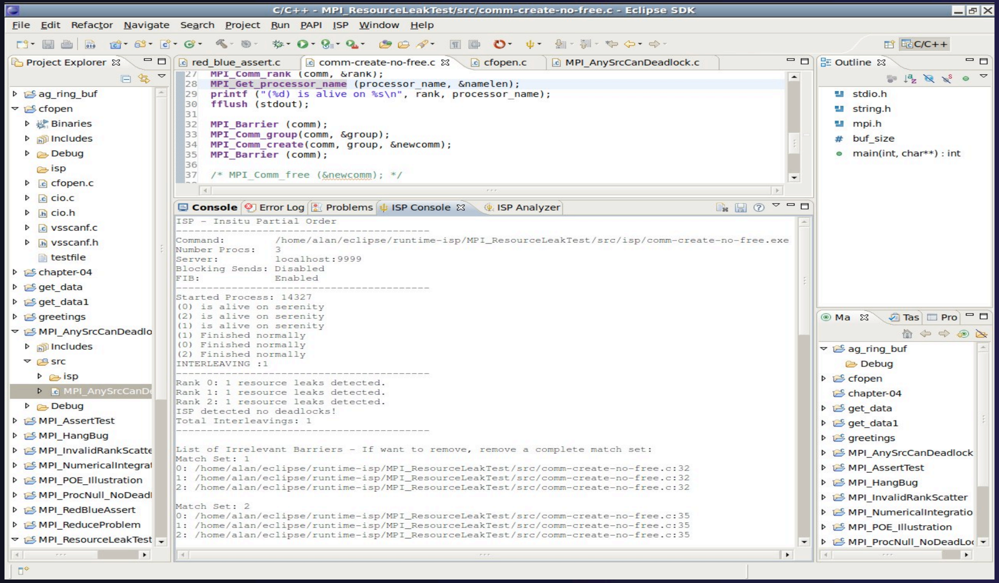
- Many existing tools provide information that can be mapped to source code lines
  - Compiler errors, warnings, suggestions
  - Performance tool findings
- ETFw feedback view provided to aid construction of these views
  - Currently geared toward data provided by tools in XML files
- Existing ETFw facilities aid the CALL of external tools from PTP
  - Feedback view aids the exposition of results to the user

ISP – In-situ Partial Order

- Dynamic verifier for MPI applications, to detect
  - Deadlocks
  - Assertion violations
  - MPI object leaks
- Contributed to PTP by the University of Utah
  - Available in PTP 3.0 (late Nov.)
- Offers rigorous coverage guarantees
  - Rigorous coverage of communication/synchronization behaviors
  - Determines relevant interleavings, replaying them as necessary
- Tested on several different MPI implementations
  - MPICH2, OpenMPI, Microsoft MPI, MVAPICH, and IBM MPI

ISP - Dynamic Formal Verification for MPI





```
27  MPI_Comm_rank (comm, &rank);
28  MPI_Get_processor_name (processor_name, &namelen);
29  printf ("(%d) is alive on %s\n", rank, processor_name);
30  fflush (stdout);
31
32  MPI_Barrier (comm);
33  MPI_Comm_group(comm, &group);
34  MPI_Comm_create(comm, group, &newcomm);
35  MPI_Barrier (comm);
36
37  /* MPI_Comm_free (&newcomm); */
```

```
ISP - Insitu Partial Order
-----------------------------------------
Command:        /home/alan/eclipse/runtime-isp/MPI_ResourceLeakTest/src/isp/comm-create-no-free.exe
Number Procs:   3
Server:         localhost:9999
Blocking Sends: Disabled
FIB:            Enabled
-----------------------------------------
Started Process: 14327
(0) is alive on serenity
(2) is alive on serenity
(1) is alive on serenity
(1) Finished normally
(0) Finished normally
(2) Finished normally
INTERLEAVING :1
-----------------------------------------
Rank 0: 1 resource leaks detected.
Rank 1: 1 resource leaks detected.
Rank 2: 1 resource leaks detected.
ISP detected no deadlocks!
Total Interleavings: 1
-----------------------------------------

List of Irrelevant Barriers - If want to remove, remove a complete match set:
Match Set: 1
0: /home/alan/eclipse/runtime-isp/MPI_ResourceLeakTest/src/comm-create-no-free.c:32
1: /home/alan/eclipse/runtime-isp/MPI_ResourceLeakTest/src/comm-create-no-free.c:32
2: /home/alan/eclipse/runtime-isp/MPI_ResourceLeakTest/src/comm-create-no-free.c:32

Match Set: 2
0: /home/alan/eclipse/runtime-isp/MPI_ResourceLeakTest/src/comm-create-no-free.c:35
1: /home/alan/eclipse/runtime-isp/MPI_ResourceLeakTest/src/comm-create-no-free.c:35
2: /home/alan/eclipse/runtime-isp/MPI_ResourceLeakTest/src/comm-create-no-free.c:35
```

- Recommended best use of ISP is during application development
- Automatically instruments and runs user code, displaying results
- A variety of GUIs facilitate debugging and code understanding

(Image courtesy of Steve Parker, U of Utah)

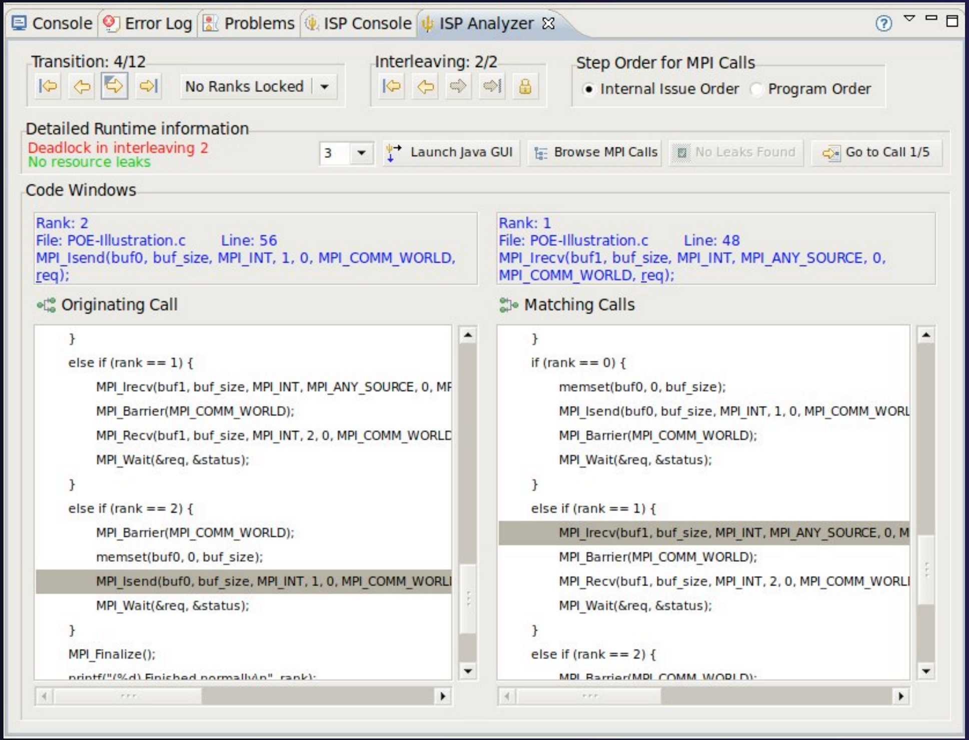
ISP Success Stories

- Umpire Tests
  - http:// www.cs.utah.edu / fv / ISP-Tests
  - Documents bugs missed by tests, caught by ISP
- MADRE (EuroPVM/MPI 2007)
  - Previously documented deadlock detected
- N-Body Simulation Code
  - Previously unknown resource leak caught during EuroPVM/MPI 2009 tutorial !
- Large Case Studies
  - ParMETIS, MPI-BLAST, IRS (Sequoia Benchmark), and a few SPEC-MPI benchmarks could be handled
- Full Tutorial including LiveDVD ISO available
  - Visit http:// www.cs.utah.edu / fv / ISP-Release

Eclipse CDT/PTP based ISP GUI

ISP Plug-in uses Eclipse CDT and PTP Highlights Bugs, and facilitates Post-Verification Review / Debugging

The MPI Happens-Before Graph shows required orderings **and** communication matches



Transition: 4/12 No Ranks Locked

Interleaving: 2/2

Step Order for MPI Calls: [✓ Internal Issue Order] Program Order

Detailed Runtime information

Deadlock in interleaving 2
No resource leaks

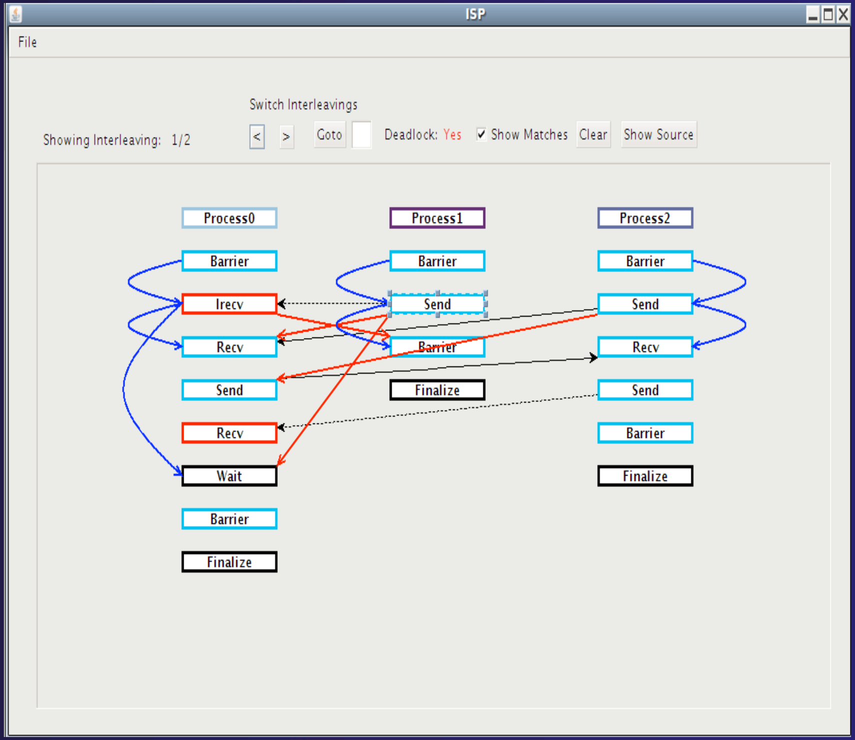
3

Rank: 2
File: POE-Illustration.c Line: 56
MPI_Isend(buf0, buf_size, MPI_INT, 1, 0, MPI_COMM_WORLD, req);

```
    }
    else if (rank == 1) {
        MPI_Irecv(buf1, buf_size, MPI_INT, MPI_ANY_SOURCE, 0, MP
        MPI_Barrier(MPI_COMM_WORLD);
        MPI_Recv(buf1, buf_size, MPI_INT, 2, 0, MPI_COMM_WORLD
        MPI_Wait(&req, &status);
    }
    else if (rank == 2) {
        MPI_Barrier(MPI_COMM_WORLD);
        memset(buf0, 0, buf_size);
        MPI_Isend(buf0, buf_size, MPI_INT, 1, 0, MPI_COMM_WORLI
        MPI_Wait(&req, &status);
    }
    MPI_Finalize();
    printf("(%d) Finished normally\n", rank);
```

Rank: 1
File: POE-Illustration.c Line: 48
MPI_Irecv(buf1, buf_size, MPI_INT, MPI_ANY_SOURCE, 0, MPI_COMM_WORLD, req);

```
    }
    if (rank == 0) {
        memset(buf0, 0, buf_size);
        MPI_Isend(buf0, buf_size, MPI_INT, 1, 0, MPI_COMM_WORL
        MPI_Barrier(MPI_COMM_WORLD);
        MPI_Wait(&req, &status);
    }
    else if (rank == 1) {
        MPI_Irecv(buf1, buf_size, MPI_INT, MPI_ANY_SOURCE, 0, M
        MPI_Barrier(MPI_COMM_WORLD);
        MPI_Recv(buf1, buf_size, MPI_INT, 2, 0, MPI_COMM_WORLI
        MPI_Wait(&req, &status);
    }
    else if (rank == 2) {
        MPI_Barrier(MPI_COMM_WORLD);
```



Showing Interleaving: 1/2

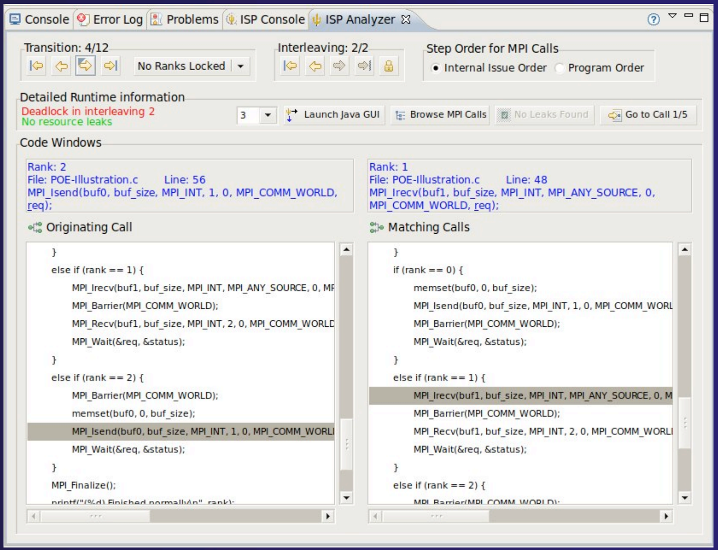
Deadlock: Yes

[✓ Show Matches]

Download / documentation: http:// www.cs.utah.edu / fv / ISP-Eclipse

ISP Analyzer View

- Reports program errors, and runtime statistics
- Debug-style source code stepping of interleavings
  - Point-to-point / Collective Operation matches
  - Internal Issue Order / Program Order views
  - Rank Lock feature
- One click to visit the Eclipse editor, to examine:
  - Calls involved in deadlock
    - helps root-cause deadlock
  - MPI Object Leaks sites
    - helps root-cause leaks
  - Assertion Violations
    - takes view to failing assertion



Step Order for MPI Calls: [✓ Internal Issue Order] Program Order

Detailed Runtime information
Deadlock in interleaving 2
No resource leaks
3

Rank: 2
File: POE-Illustration.c Line: 56
MPI_Isend(buf0, buf_size, MPI_INT, 1, 0, MPI_COMM_WORLD, req);

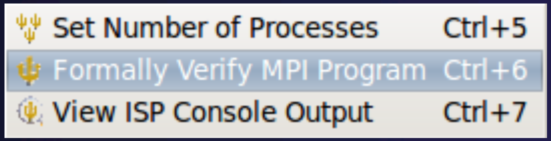
```
    }
    else if (rank == 1) {
        MPI_Irecv(buf1, buf_size, MPI_INT, MPI_ANY_SOURCE, 0, MP
        MPI_Barrier(MPI_COMM_WORLD);
        MPI_Recv(buf1, buf_size, MPI_INT, 2, 0, MPI_COMM_WORLD
        MPI_Wait(&req, &status);
    }
    else if (rank == 2) {
        MPI_Barrier(MPI_COMM_WORLD);
        memset(buf0, 0, buf_size);
        MPI_Isend(buf0, buf_size, MPI_INT, 1, 0, MPI_COMM_WORL
        MPI_Wait(&req, &status);
    }
    MPI_Finalize();
    printf("(%d) Finished normally\n", rank);
```

Rank: 1
File: POE-Illustration.c Line: 48
MPI_Irecv(buf1, buf_size, MPI_INT, MPI_ANY_SOURCE, 0, MPI_COMM_WORLD, req);

```
    }
    if (rank == 0) {
        memset(buf0, 0, buf_size);
        MPI_Isend(buf0, buf_size, MPI_INT, 1, 0, MPI_COMM_WORL
        MPI_Barrier(MPI_COMM_WORLD);
        MPI_Wait(&req, &status);
    }
    else if (rank == 1) {
        MPI_Irecv(buf1, buf_size, MPI_INT, MPI_ANY_SOURCE, 0, M
        MPI_Barrier(MPI_COMM_WORLD);
        MPI_Recv(buf1, buf_size, MPI_INT, 2, 0, MPI_COMM_WORLI
        MPI_Wait(&req, &status);
    }
    else if (rank == 2) {
        MPI_Barrier(MPI_COMM_WORLD);
```

Running ISP

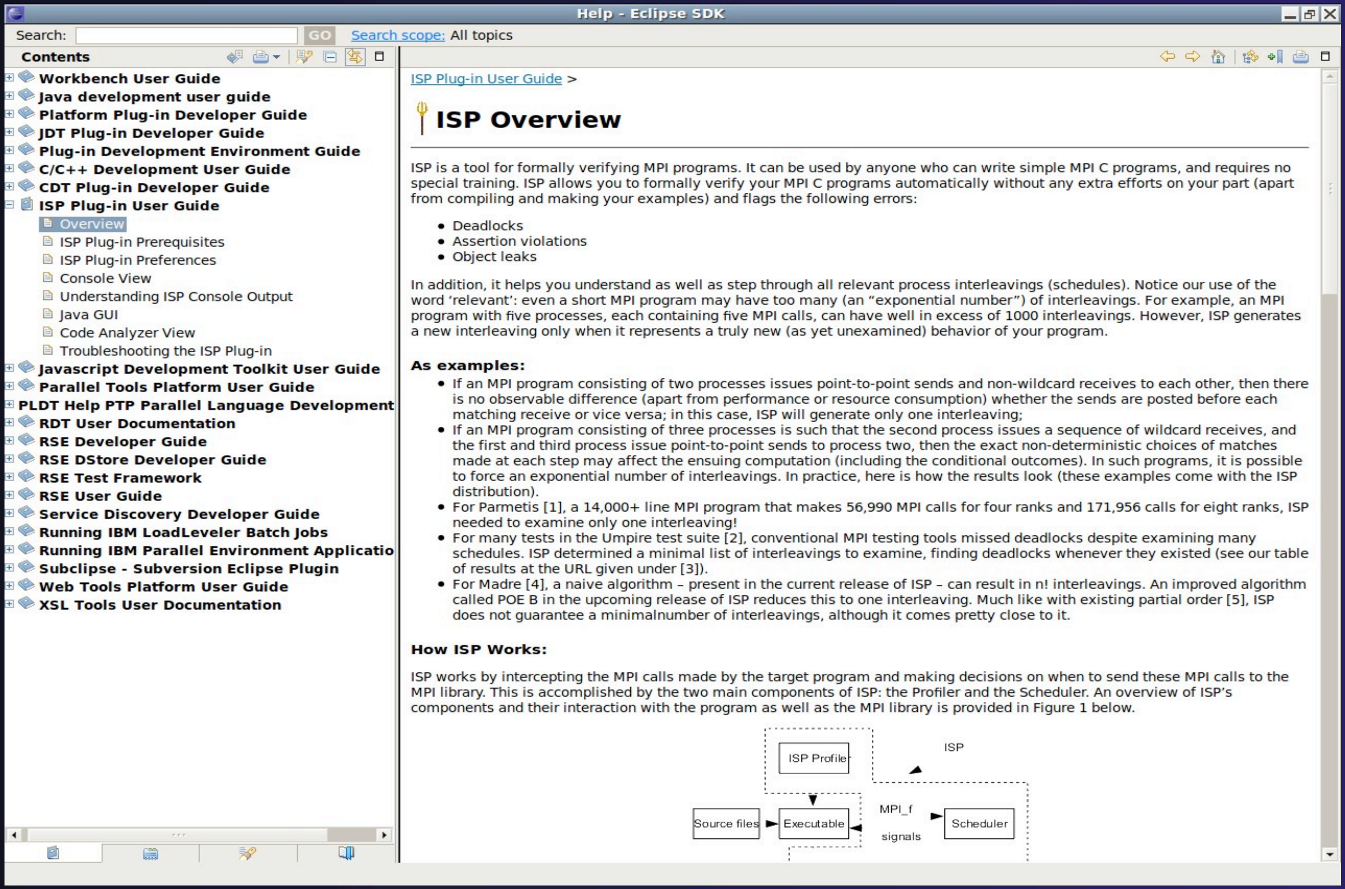
- Create an MPI C Project within C/C++ Perspective
  - Make sure your project builds and runs correctly
- Set preferences and via ISP Preference Page
- From the trident icon or the ISP menu, user can:


Set Number of Processes Ctrl+5
Formally Verify MPI Program Ctrl+6
View ISP Console Output Ctrl+7

  - Context menus may also be used from Project Explorer
- Formally Verifying MPI Program
  - Launches ISP
  - Generates log file for Post-Verification Analysis Views
- Dedicated ISP Console accompanies Analyzer View

ISP Integrated Help

Extensive graphical aids & trouble shooting section



Help - Eclipse SDK

Search: GO Search scope: All topics

Contents

- Workbench User Guide
- Java development user guide
- Platform Plug-in Developer Guide
- JDT Plug-in Developer Guide
- Plug-in Development Environment Guide
- C/C++ Development User Guide
- CDT Plug-in Developer Guide
- ISP Plug-in User Guide
  - Overview
  - ISP Plug-in Prerequisites
  - ISP Plug-in Preferences
  - Console View
  - Understanding ISP Console Output
  - Java GUI
  - Code Analyzer View
  - Troubleshooting the ISP Plug-in
- Javascript Development Toolkit User Guide
- Parallel Tools Platform User Guide
- PLDT Help PTP Parallel Language Development
- RDT User Documentation
- RSE Developer Guide
- RSE DStore Developer Guide
- RSE Test Framework
- RSE User Guide
- Service Discovery Developer Guide
- Running IBM LoadLeveler Batch Jobs
- Running IBM Parallel Environment Applicatio
- Subclipse - Subversion Eclipse Plugin
- Web Tools Platform User Guide
- XSL Tools User Documentation

ISP Plug-in User Guide >

ISP Overview

ISP is a tool for formally verifying MPI programs. It can be used by anyone who can write simple MPI C programs, and requires no special training. ISP allows you to formally verify your MPI C programs automatically without any extra efforts on your part (apart from compiling and making your examples) and flags the following errors:

- Deadlocks
- Assertion violations
- Object leaks

In addition, it helps you understand as well as step through all relevant process interleavings (schedules). Notice our use of the word 'relevant': even a short MPI program may have too many (an "exponential number") of interleavings. For example, an MPI program with five processes, each containing five MPI calls, can have well in excess of 1000 interleavings. However, ISP generates a new interleaving only when it represents a truly new (as yet unexamined) behavior of your program.

As examples:

- If an MPI program consisting of two processes issues point-to-point sends and non-wildcard receives to each other, then there is no observable difference (apart from performance or resource consumption) whether the sends are posted before each matching receive or vice versa; in this case, ISP will generate only one interleaving;
- If an MPI program consisting of three processes is such that the second process issues a sequence of wildcard receives, and the first and third process issue point-to-point sends to process two, then the exact non-deterministic choices of matches made at each step may affect the ensuing computation (including the conditional outcomes). In such programs, it is possible to force an exponential number of interleavings. In practice, here is how the results look (these examples come with the ISP distribution).
- For Parmetis [1], a 14,000+ line MPI program that makes 56,990 MPI calls for four ranks and 171,956 calls for eight ranks, ISP needed to examine only one interleaving!
- For many tests in the Umpire test suite [2], conventional MPI testing tools missed deadlocks despite examining many schedules. ISP determined a minimal list of interleavings to examine, finding deadlocks whenever they existed (see our table of results at the URL given under [3]).
- For Madre [4], a naive algorithm – present in the current release of ISP – can result in n! interleavings. An improved algorithm called POE B in the upcoming release of ISP reduces this to one interleaving. Much like with existing partial order [5], ISP does not guarantee a minimalnumber of interleavings, although it comes pretty close to it.

How ISP Works:

ISP works by intercepting the MPI calls made by the target program and making decisions on when to send these MPI calls to the MPI library. This is accomplished by the two main components of ISP: the Profiler and the Scheduler. An overview of ISP's components and their interaction with the program as well as the MPI library is provided in Figure 1 below.

PTP Adv. Development: Summary

- A diversity of other tools aid parallel development
  - Parallel Language Development Tools: MPI, OpenMP, UPC
    - MPI Barrier deadlock detection, etc.
  - External Tools Framework (ETFw) eases integration of existing (command-line, etc.) tools
    - TAU Performance Tuning uses ETFw
    - PPW (Parallel Perf. Wizard) uses ETFw for UPC analysis
    - New Feedback view maps tool findings with src code
  - MPI Analysis: ISP
- A diversity of contributors too!
  - We welcome other contributions. Let us help!

Module 8: Other Tools and Wrap-up

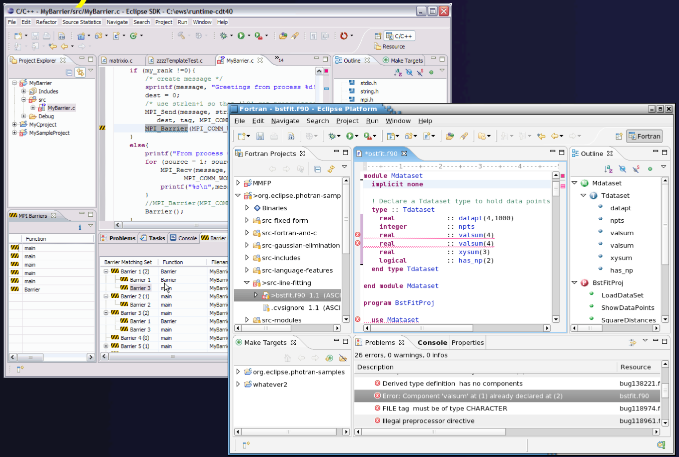
- Objective
  - How to find more information on PTP
  - Learn about other tools related to PTP
  - See PTP upcoming features
- Contents
  - Links to other tools, including performance tools
  - Planned features for new versions of PTP
  - Additional documentation
  - How to get involved

NCSA HPC Workbench

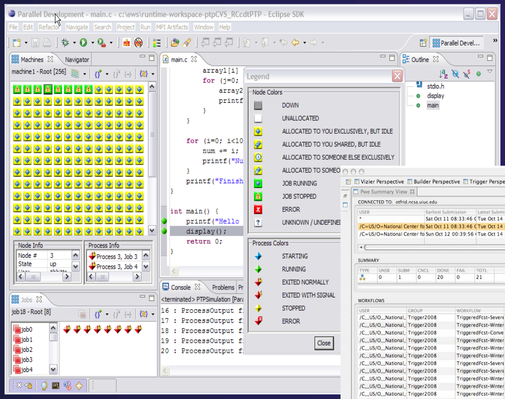
- Tools for NCSA Blue Waters
  - http://www.ncsa.illinois.edu/BlueWaters/
  - Sustained Petaflop system
- Based on Eclipse and PTP
- Includes some related tools
  - Performance tools
  - Scalable debugger
  - Workflow tools (https://wiki.ncsa.uiuc.edu/display/MRDPUB/MRD+Public+Space+Home+Page)
- Part of the enhanced computational environment described at:
  http://www.ncsa.illinois.edu/BlueWaters/ece.html

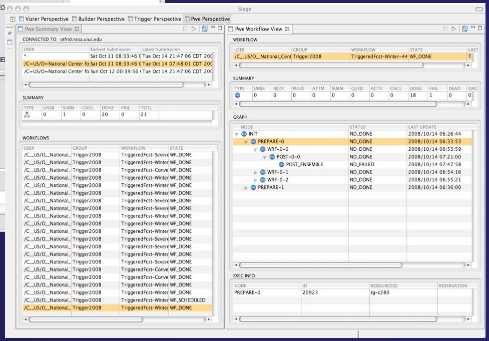
NCSA HPC Workbench

Coding & Analysis (CDT, PLDT, Photran)

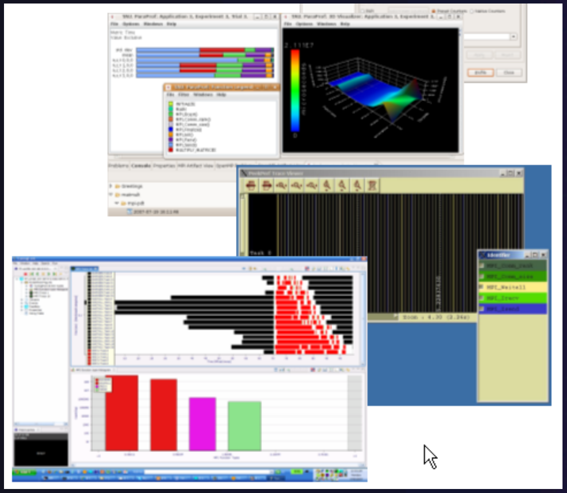


PTP Launching & Monitoring

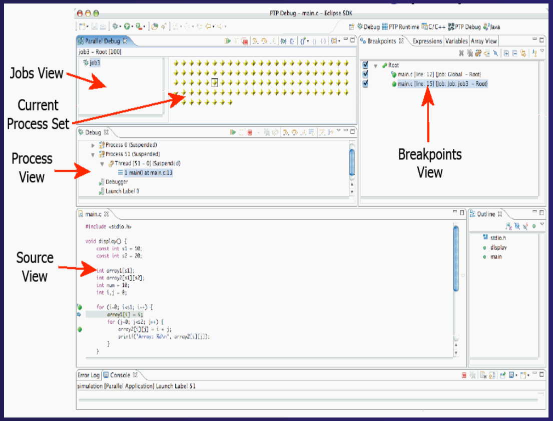




Workflow



Performance Tuning (HPC toolkit, HPCS toolkit, RENCI, …)



PTP Debugging

Useful Eclipse Tools

- Python
  - http://pydev.sourceforge.net
- Ruby
  - http://sourceforge.net/projects/rubyeclipse
- Subversion (now an Eclipse project)
  - http://eclipse.org/subversive
  - Or Subclipse: http://subclipse.tigris.org/
- Git (now an Eclipse project)
  - http://www.eclipse.org/egit
- … and many more!

Future PTP Features

- Support for multicore development
  - Building on Cell IDE and other multicore tools
- Resource managers to support for PBS, LSF, and Blue Gene
- Transitioning debugger to Scalable Tools Communication Infrastructure (STCI)
- Scalability improvements
  - UI to support 1M processes
  - Optimized communication protocol
  - Very large application support

Online Information

- Information about PTP
  - Main web site for downloads, documentation, etc.
    - http://eclipse.org/ptp
  - Developers' wiki for designs, planning, meetings, etc.
    - http://wiki.eclipse.org/PTP
  - Articles and other documents
    - http://wiki.eclipse.org/PTP/articles
- Information about Photran
  - Main web site for downloads, documentation, etc.
    - http://eclipse.org/photran
  - User's manuals
    - http://wiki.eclipse.org/PTP/photran/documentation

Mailing Lists

- PTP Mailing lists
  - Major announcements (new releases, etc.) - low volume
    - http://dev.eclipse.org/mailman/listinfo/ptp-announce
  - User discussion and queries - medium volume
    - http://dev.eclipse.org/mailman/listinfo/ptp-user
  - Developer discussions - high volume
    - http://dev.eclipse.org/mailman/listinfo/ptp-dev
- Photran Mailing lists
  - User discussion and queries
    - http://dev.eclipse.org/mailman/listinfo/photran
  - Developer discussions –
    - http://dev.eclipse.org/mailman/listinfo/photran-dev

Getting Involved

- See http://eclipse.org/ptp
- Read the developer documentation on the wiki
- Join the mailing lists
- Attend the monthly developer meetings
    - Teleconference each second Tuesday, 1:00 pm ET

- PTP will only succeed with your participation!

PTP Tutorial Feedback

- Please complete feedback form
- Your feedback is valuable!

Thanks for attending
We hope you found it useful