Scientific Application Development Using Eclipse and the Parallel Tools Platform

Craig Rasmussen, LANL
Beth Tibbitts, IBM
Greg Watson, LANL

parallel tools platform
http://eclipse.org/ptp

Tutorial Outline (morning)

Time	Module	Outcomes	Presenter
8:30 - 9:00	Tutorial Introduction	Introduction to tutorial process and setup	Greg Watson
9:00 - 9:30	1. Overview of Eclipse and PTP	An understanding of the overall Eclipse and PTP architecture	Greg Watson
9:30 - 10:00	2. Introduction to the Eclipse IDE	Introduction to the basic features of the Eclipse IDE, including building, running and debugging a sample application	Craig Rasmussen
10:00 - 10:30	Break		
10:30 - 11:30	2. Introduction to the Eclipse IDE	Continued…	Craig Rasmussen
11:30 - 12:00	3. Installing Eclipse	Eclipse installed on your laptop	Greg Watson
12:00 - 1:30	Lunch Break		

parallel tools platform
http://eclipse.org/ptp

Tutorial Outline (afternoon)

Time	Module	Outcomes	Presenter
1:30 - 2:30	4. Advanced Development	Exploring advanced features of the Eclipse IDE	Craig Rasmussen
2:30 - 3:00	5. PTP and Parallel Language Development Tools	Introduction to PTP and MPI application development tools	Beth Tibbitts
3:00 - 3:30	Break		
3:30 - 3:45	5. PTP and Parallel Language Development Tools	Continued…	Beth Tibbitts
3:45 - 4:45	6. Parallel Debugging	Introduction to the Eclipse parallel debugger, locating and correcting a bug in a parallel code	Beth Tibbitts
4:45 - 5:00	Tutorial wrap up	Further information about Eclipse, PTP and related tools Obtain feedback from participants	

parallel tools platform
http://eclipse.org/ptp

Tutorial Introduction

- Tutorial is divided into modules
- Each module has clear objectives
- Modules usually comprise short theory component, followed by hands-on
- Raise your hand if you get stuck
- Ask questions at any time
-  Indicates hands-on

Hands-on Setup

- Hands-on activities require use of X-Windows software
  - Avoids installing OpenMPI on your laptop (takes a long time)
  - Runs Eclipse remotely on the 'parallel' machine
  - Displays an Eclipse session on your laptop
- A full version of Eclipse will also be installed on your laptop in Module 3
  - This will allow you explore more Eclipse features later
- Linux:
  - Should not require any additional setup
- MacOS X:
  - May require X11 installation
- Windows:
  - Requires installation of X11

X11 Installation (MacOS X)

- Check for existing installation:
  - **/Applications/Utilities/X11**
- If installation required:
  - Open **macosx** folder on **TutorialCD**
  - Double-click on
    - **X11User10.3.pkg** for MacOS X 10.3
    - **X11User10.4.pkg** for MacOS X 10.4
  - Follow installer prompts
    - Accept default options

X11 Installation (Windows)

- Options:
  - Install Cygwin/X
    - Cygwin installation required for optional Module 3
  - Install StarNet X-Win32
    - Installs new application
  - Boot from XLiveCD
    - No installation, slower
  - Boot from Knoppix CD
    - No installation, requires manual setup

Cygwin/X Installation

- Open the **TutorialCD** in **My Computer**
- Open the **cygwin** folder
- Double-click on **setup**
- Select **Next >**
- Select **Install from Local Directory**
- If not already, enter **C:\cygwin** in **Root Directory**
- Select **Next >**
- On the **Local Package Directory** page
  - Select the **cygwin** folder on the **TutorialCD** via the **Browse...** button, then **OK**
- Select **Next >**
- Select **Next >** to accept default packages
  - This will take about 5 minutes
- Select **Create icon on Desktop**, **Finish** then **OK**

parallel tools platform
http://eclipse.org/ptp

Module 1: Overview of Eclipse and PTP

- Objective
  - To introduce participants to the Eclipse platform and PTP
- Contents
  - History
  - What is Eclipse?
  - Who is using Eclipse?
  - What is PTP?

History

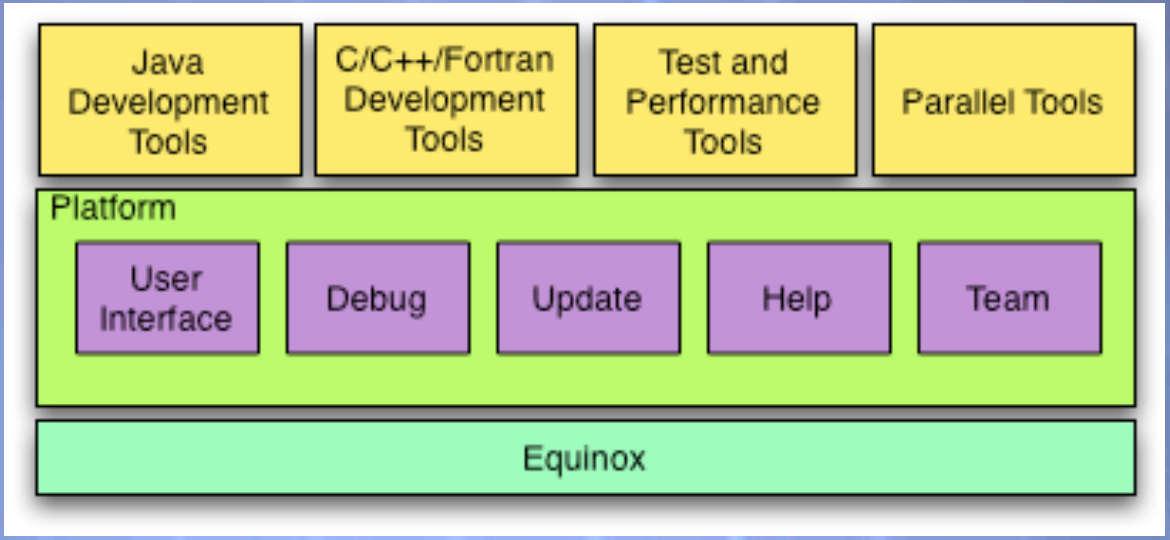
- Originally developed by Object Technology International (OTI) and purchased by IBM for use by internal developers
- Released to open-source community in 2001, managed by consortium
  - Eclipse Public License (EPL)
  - Based on IBM Common Public License (CPL)
- Consortium reorganized into independent not-for-profit corporation, the Eclipse Foundation, in early 2004
  - Participants from over 100 companies

Eclipse Foundation

- Board of Directors drawn from four classes of membership:
  - Strategic Developers, Strategic Consumer, Add-in Providers, and Open Source project leaders
- Full-time Eclipse management organization
- Councils guide the development done by Eclipse Open Source projects
  - Requirements
  - Architecture
  - Planning
- Currently 9 projects and over 50 subprojects

What is Eclipse?

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



Equinox

- OSGi framework implementation model
  - Formerly known as the Open Services Gateway initiative
  - Standard for application lifecycle management
- Provides the most fundamental Eclipse infrastructure
  - Plug-ins (known as a bundle)
  - Bundle install, update and uninstall
  - Bootstrap and launching
  - Extension registry
- Introduced in Eclipse 3.0

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
  - Dynamic update/install service

Plug-ins

- Java Development Tools (JDT)
- Plug-in Development Environment (PDE)
- C/C++ Development Tools (CDT)
- Parallel Tools Platform (PTP)
- 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…

Who is using Eclipse?

- Commercial tool developers
  - Accelerated Technology, Catalyst Systems, Codign Software, Compuware Corp, Exadel, HP, ILOG, IBM, Intel, Lattix, Mentor Graphics, Monta Vista, MySQL, Novell, Palm, QNX, Wind River
- Commercial application developers
  - Actuate, Applied Biosystems, Bay Breeze Software, BSI, Crypto Intelligence, DeltaLearn, eClarus Software, EzMgt, Future Management, IBM, Incremental, Infonoia, iMEDIC, Innovation Gate, ITscope, Market Contours, nulogy, Recursa Software, Redbird Software, RPC Software, ForeFlight, SkyWalker Software, SnapXT, Sphere Networks, Third Brigade
- Commercial application users
  - Adobe, Agence France Press, AlterPoint, Bank SinoPac, City of Stuttgart, Compass Group, DailmerChrysler, NASA JPL, Plum Canary, Refractions Research, RSS Solutions, SAS

What is 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

parallel tools platform
http://eclipse.org/ptp

Module 2: Introduction to the Eclipse IDE

- Objective
  - Gain an understanding of how to use Eclipse to develop applications
- Contents
  - Brief introduction to the Eclipse IDE
  - Create a simple application
  - Run and debug simple application

parallel tools platform
http://eclipse.org/ptp

Launching Eclipse

- For the tutorial, everyone will run Eclipse from an X-Windows terminal
- Once the X-Windows terminal is started, connect to the parallel machine
- Log on to the parallel machine using the supplied username
- Launch Eclipse on the parallel machine
  - Eclipse will be displayed locally on your laptop

parallel tools platform
http://eclipse.org/ptp

Starting X-Windows

- Start the X-Windows Application
  - Linux
    - Open a terminal window
  - MacOS X
    - Double-click **/Applications/Utilities/X11**
  - Windows (cygwin/X)
    - Double-click **cygwin** icon on Desktop
    - At the prompt, enter **startxwin.sh**
  - Windows (other options)
    - Tutorial instructor will guide you

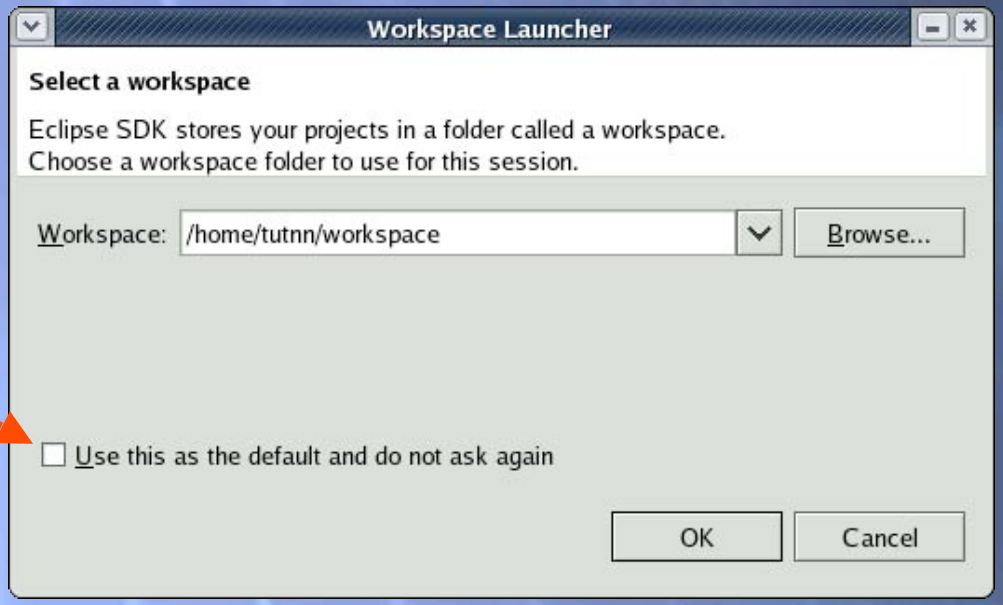
Connecting

- Once you have a terminal prompt, enter:
  - `ssh -Y` *username@hostname*
  - Enter password when prompted
  - Your username, hostname, and password can be found on the cover of your tutorial CD
- After you have logged on to the parallel machine, enter:
  - `eclipse &`

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

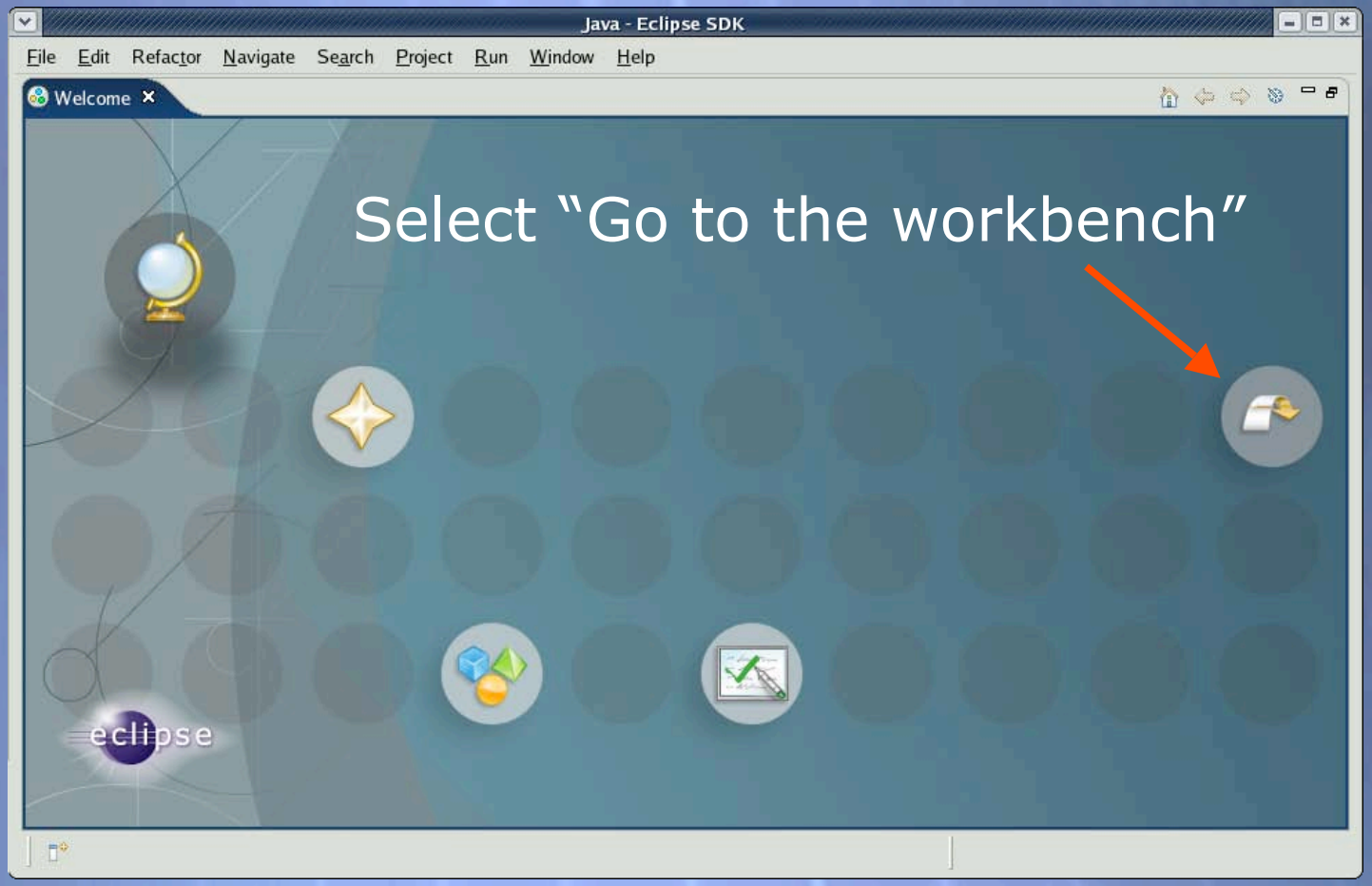
The prompt can be turned off



parallel tools platform
http://eclipse.org/ptp

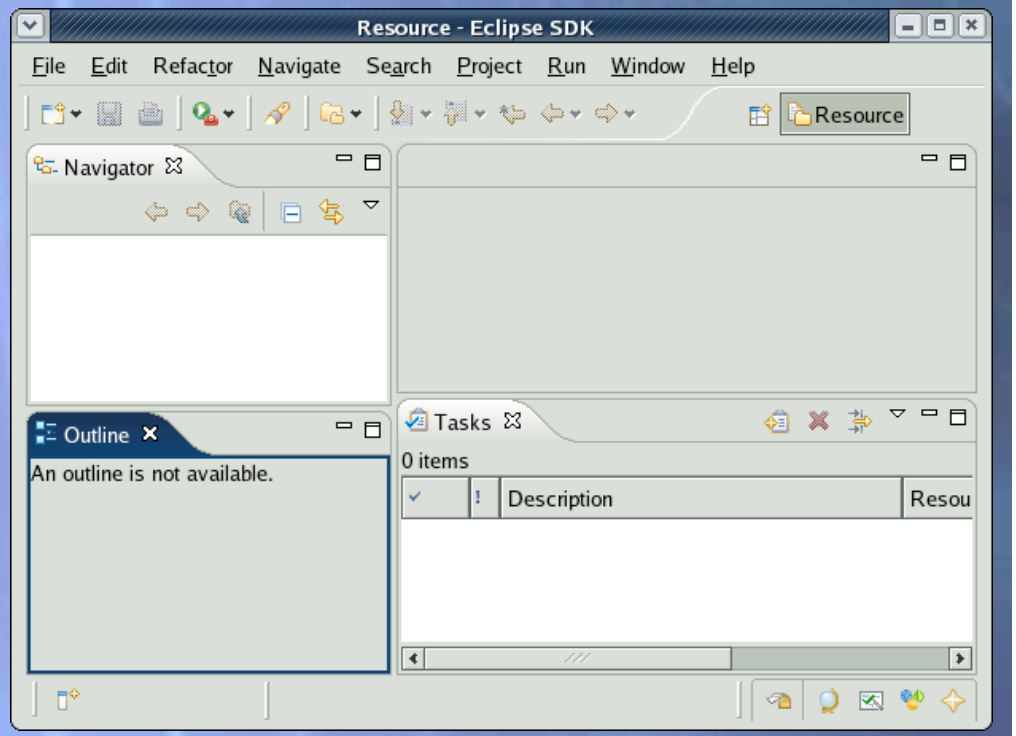
Eclipse Welcome Page

- Displayed when Eclipse is run for the first time



Workbench

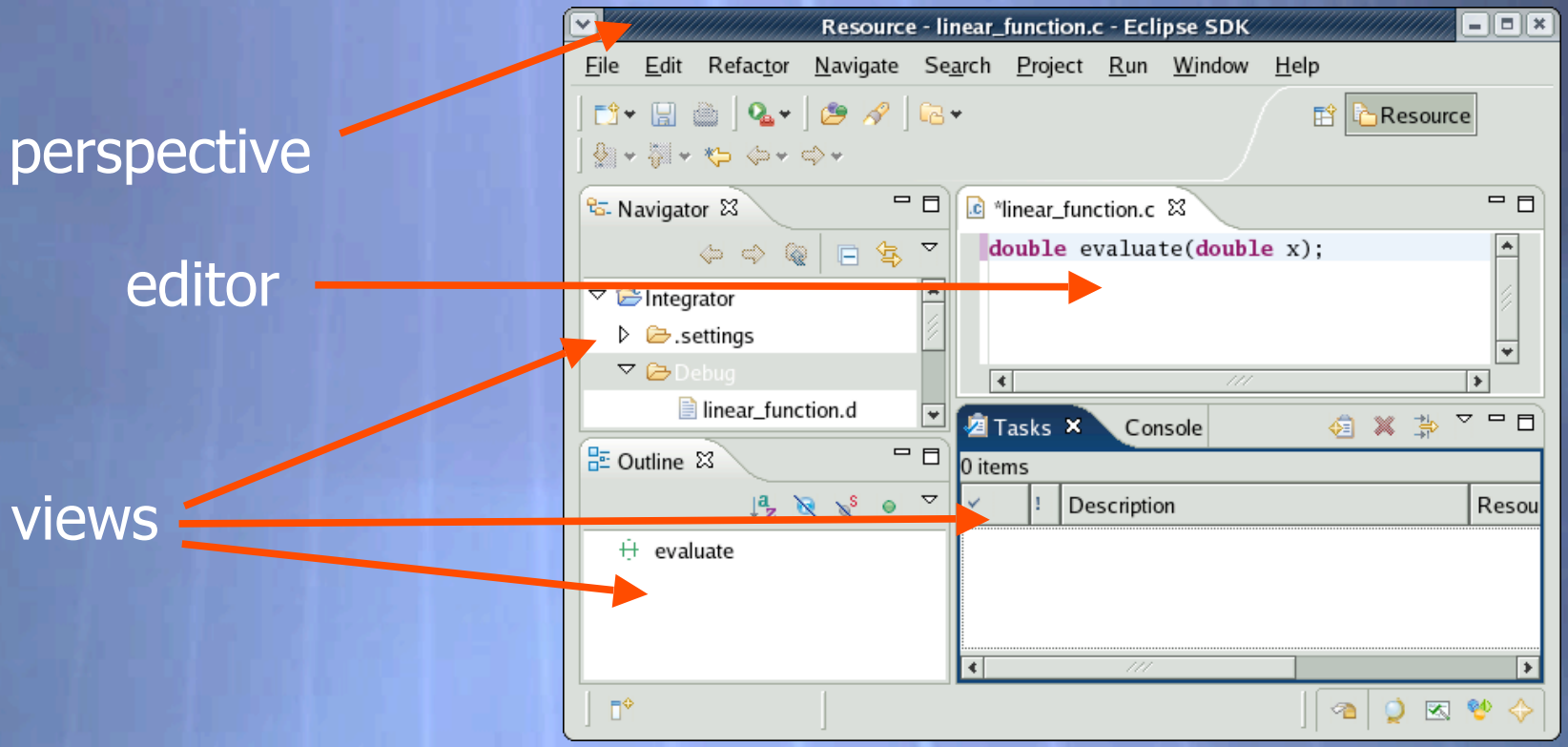
- The Workbench represents the desktop development environment
  - It contains a set of tools for resource management
  - It provides a common way of navigating through the resources
- Multiple workbenches can be opened at the same time



parallel tools platform
http://eclipse.org/ptp

Workbench Components

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



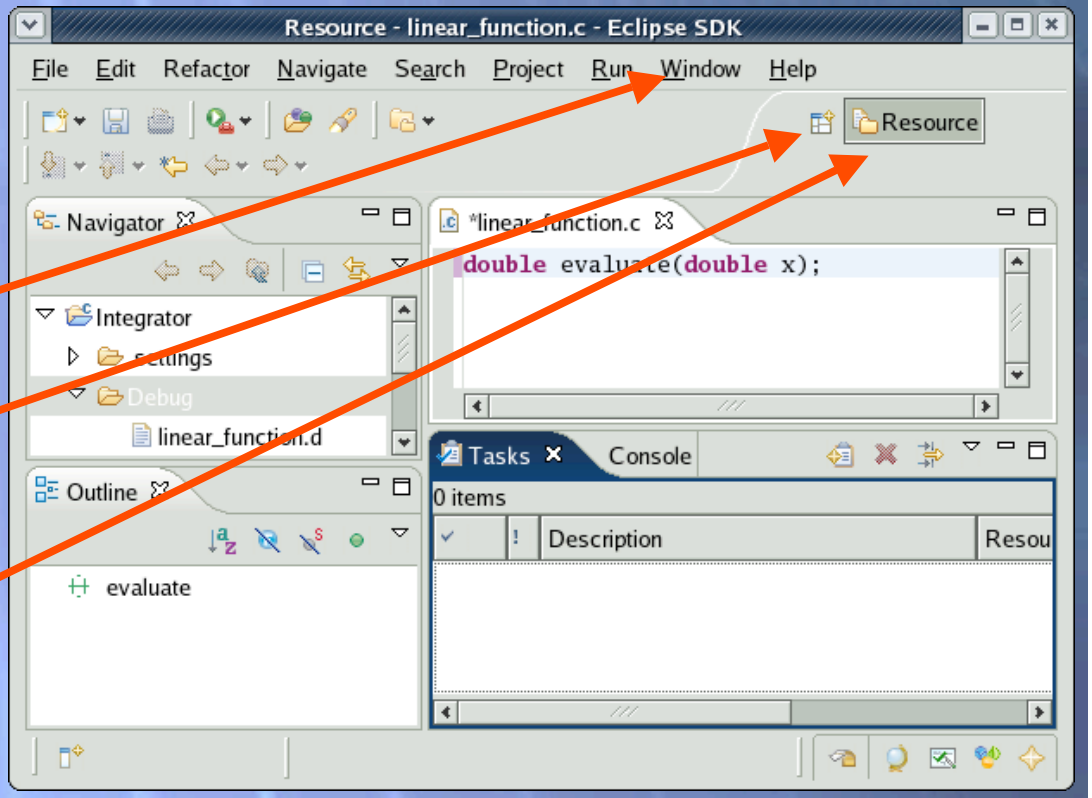
Perspectives

- 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
  - Make Perspective for manipulating compiled code (C/C++, Fortran)
  - Debug Perspective for debugging applications
- You can easily switch between perspectives

parallel tools platform
http://eclipse.org/ptp

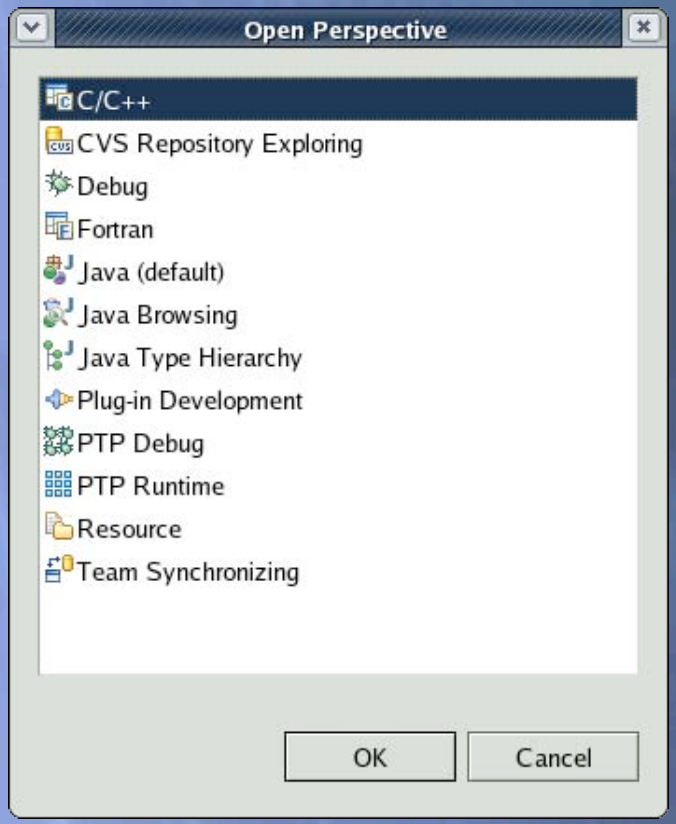
Switching Perspectives

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



Available Perspectives

- By default, certain perspectives are available in the Workbench
- We've also installed C/C++ and Fortran perspectives



Customizing Perspectives

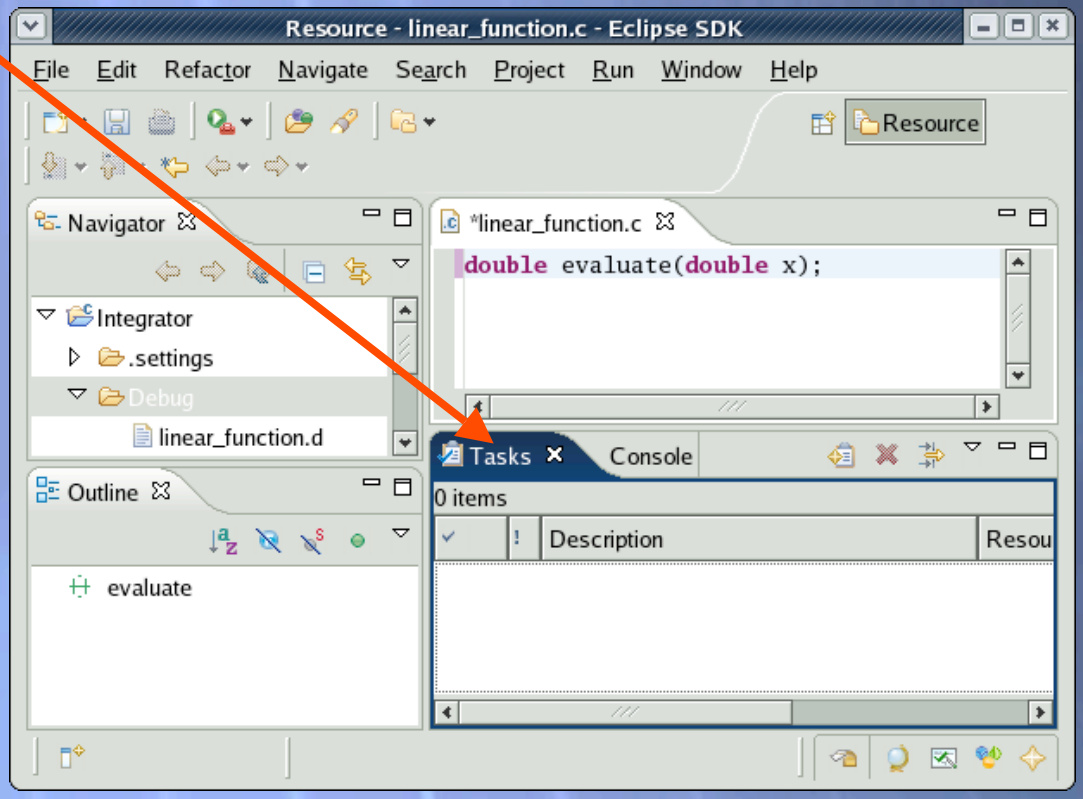
- Items such as shortcuts, menu items and views may be customized
  - **Window ▸ Customize Perspective...**
- Rearrange views by dragging
  - Try moving the outline view
- Save changes
  - **Window ▸ Save Perspective As...**
- Close Perspective
  - Right-click on perspective title and select **Close**
- Reset Perspective
  - **Window ▸ Reset Perspective** resets the current perspective to its default layout

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

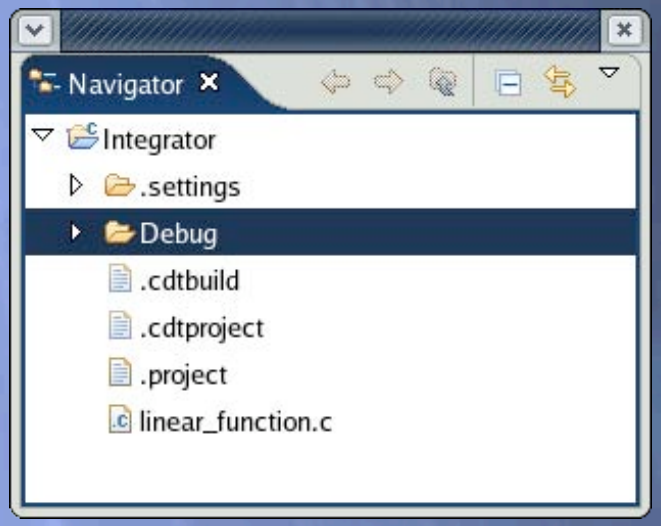
Stacked Views

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



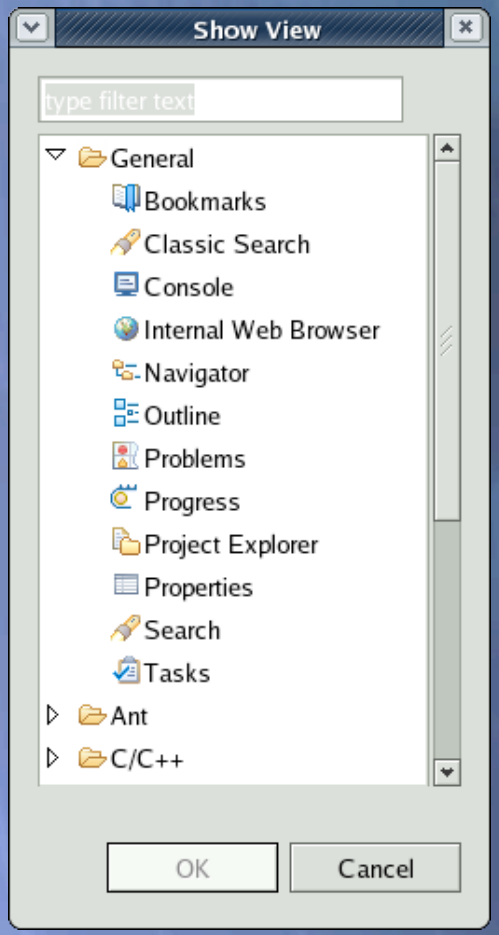
Projects View

- 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 Navigator View



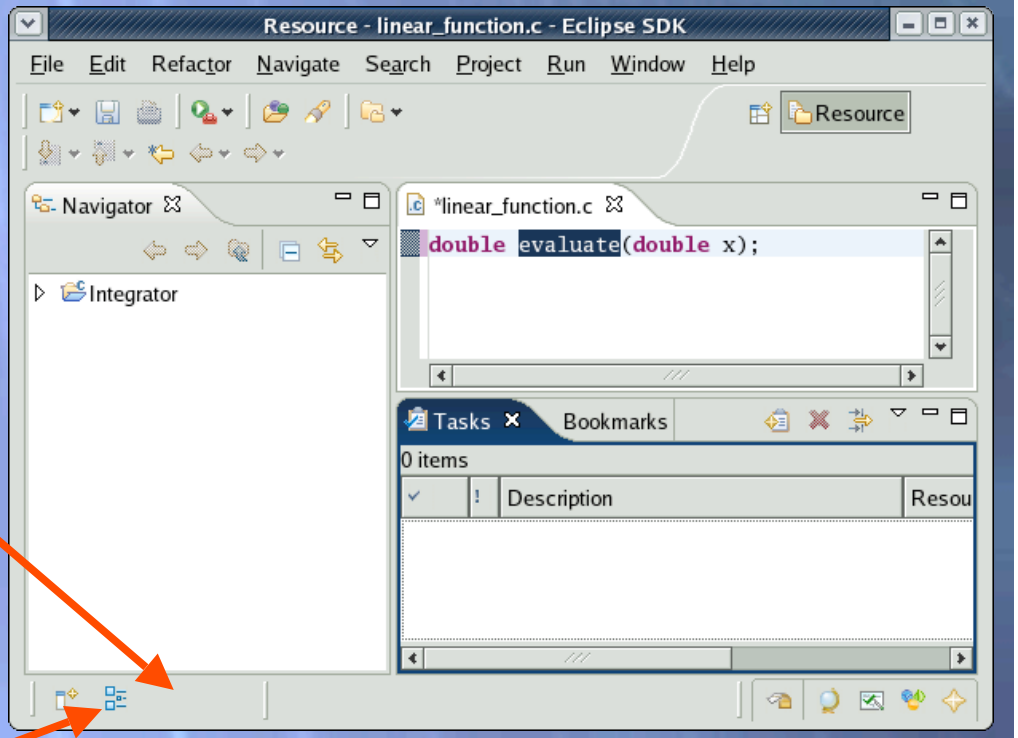
Opening a New View

- To open a view:
  - Choose **Window ▸ Show View ▸ Other...**
  - The **Show View** dialog comes up
  - Select the view to be shown
  - Select **OK**



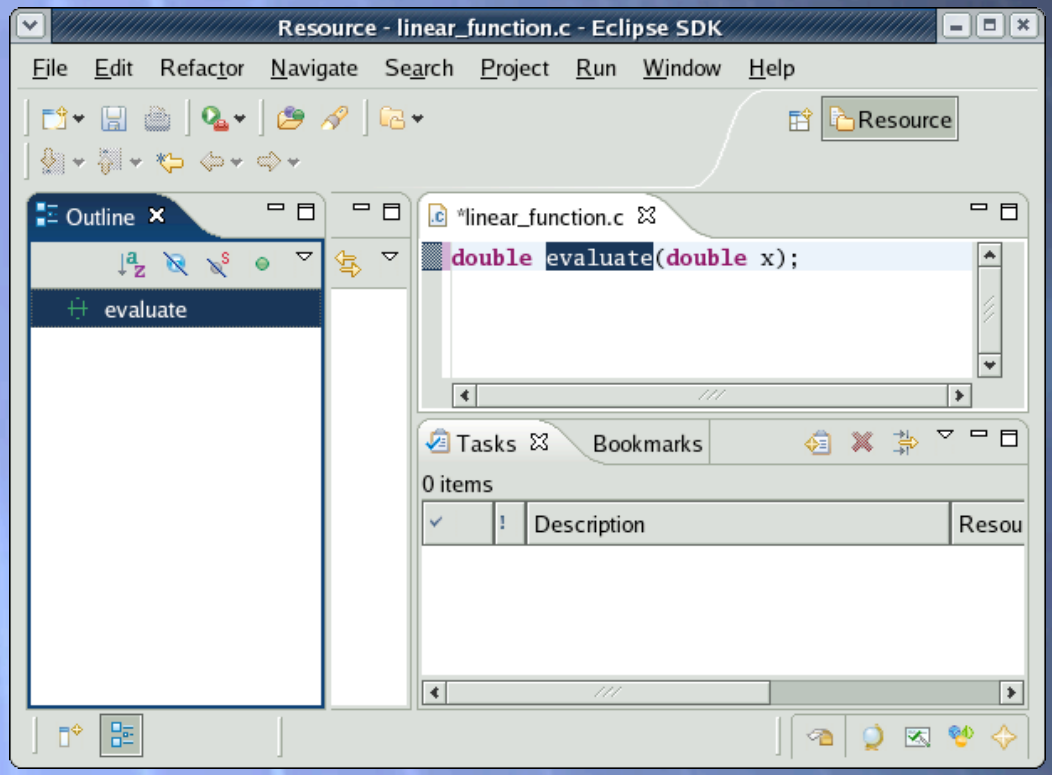
Fast Views (1)

- Hidden views that can be quickly opened and closed
  - They take up space in the Workbench
- Fast views can be created by:
  - Dragging an open view to the shortcut bar
  - Selecting **Fast View** from the view's menu
- A Fast View is activated by clicking on its **Fast View** button



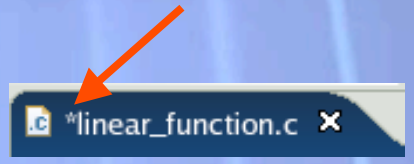
Fast Views (2)

- Clicking on the Fast View opens the view in the current perspective
- Clicking outside of the view makes it hidden again
- Turn off the Fast View by selecting **Fast View** from the view's menu again



Editors

- An editor for a resource opens when you double-click on a resource
  - Editor type depends on the type of the resource, for example .c files are opened with the C/C++ editor
  - 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



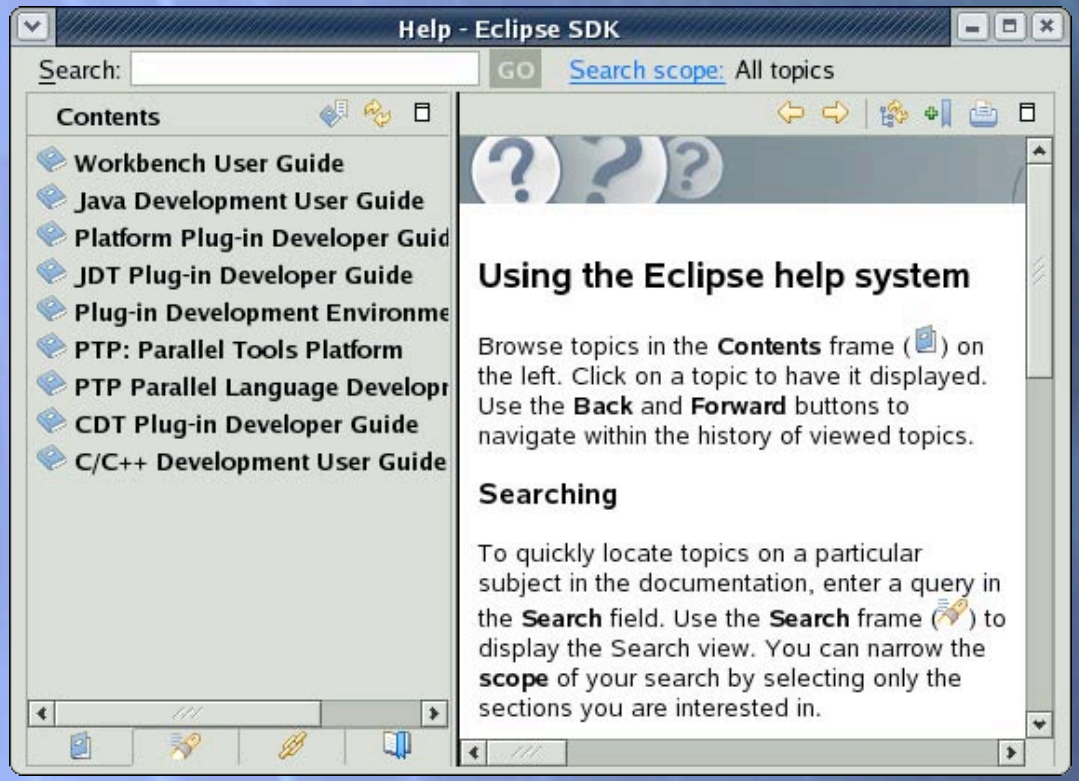
parallel tools platform
http://eclipse.org/ptp

Preferences

- Preferences provide a way for you to customize your Workbench
  - By selecting **Window ▸ Preferences...**
- For example:
  - Use Emacs bindings
  - Modify editor folding defaults
    - E.g., fold all macro definitions
  - Associate file types with file extensions
    - E.g., *.f03 with the Fortran editor
  - Toggle automatic builds
  - Change key sequence shortcuts
    - E.g., Ctrl+/ for Comment

Help

- Access help
  - **Help ▶ Help Contents**
  - **Search**
  - **Dynamic Help…**
- What's there…
- Context sensitive help…



parallel tools platform
http://eclipse.org/ptp

A Simple Application

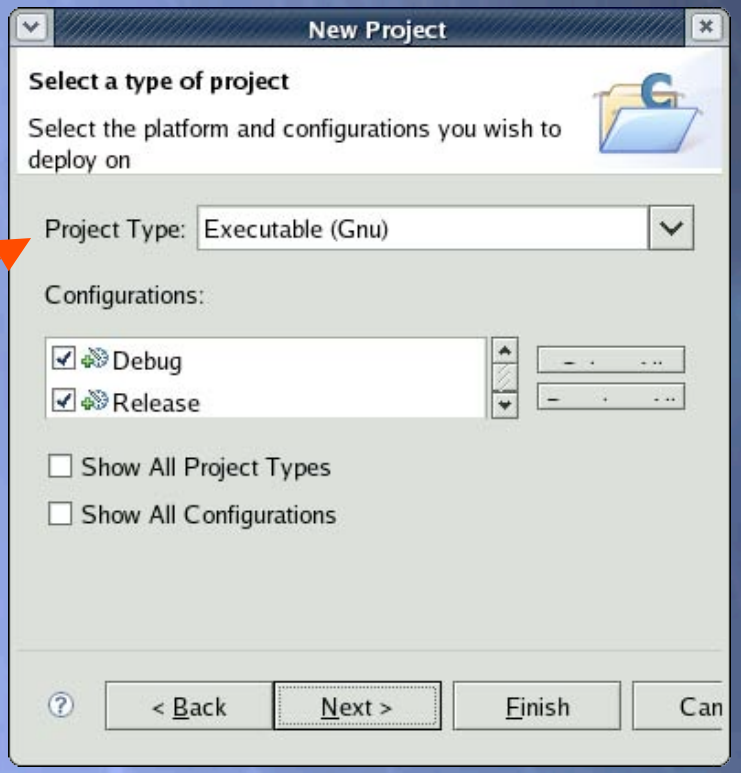
- Create a Project
  - Managed Make C Project
- Add files
  - Source files (ending in .c)
  - A makefile is automatically created
- Build application
  - Done automatically
- Debug application
  - Create a Run Configuration

CDT Projects

- A Project contains the resources of an application
- CDT (C/C++ Development Tools) project types:
  - Managed Make
  - Standard Make
- Resources are visible in Navigator or C/C++ Projects View
- Project Type is very **important**
  - Selects project builder (linker)
  - C++, Fortran, or C

Creating a Managed Make Project

- Create a project (in C/C++ Perspective)
  - **File▸New▸Managed Make C Project**
  - Or select **New Project** button
- Give it a name: Integrator
  - **Next>**
- Select Project Type
  - **Next>**
- On Indexer tab, select **Full C/C++ Indexer**
- Select **Finish**



Add Resources

- Import existing files from file system
  - Right-click on project, select **Import...**
  - Under **General**, select **File System** then **Next**
  - Input **From directory:** using **Browse...**
  - Select **code** folder and then **OK**
  - Check **linear_function.c** and **integrator.c**
  - Select **Finish**
- Can also create new source files
  - **File ▸ New ▸ Source File**

Fix Error in File

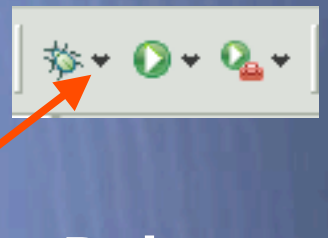
- Project fails to build
  - Note red icon on filename
- Click on **Problems** View tab
- Fix error in **linear_function.c**
  - Double-click on the file in the **C/C++ Projects** view to open an editor
- Save file; project will automatically rebuild when file is saved
  - **File▸Save** (or Ctrl-S)
- Look at console view to see build progress
  - There is still another error

Project Properties

- To fix the next error, add the GNU Scientific Library to the build process:
  - Right-click on Project and select the **Properties** menu item
  - Select the **C/C++ Build** item
  - Select **GCC C Linker ▸ Libraries** from the **Tool Settings** tab
    - Click on the **'+'** icon next to **Libraries (-l)** to add the library
    - Enter 'gsl' in the dialog box and select **OK**
  - Select **OK** to close the **Project Properties**

Launch Configuration

- A Launch Configuration is needed to run or debug an application
- To create a launch configuration:
  - Select **Run▸Debug...**
  - Or click arrow next to debug button, then **Debug...**
  - Select **C/C++ Local Application** and select the **New** button
  - You may have to select the **Search Project** button for the **C/C++ Application** binary
  - Select **gdb/mi** from the **Debugger tab**
  - You may add command line arguments using the **Arguments tab**
  - Select **Debug**



Debugging (1)

- If asked, select **Yes** to confirm switching to the Debug Perspective after creating the launch configuration
- Set a breakpoint by double-clicking on the left vertical bar in the editor (at `sum = 0.0;` line)
- To continue running, click on **Resume** button
- Click on **Step Over** button until line with getRandomNumber()
- Click on **Step Into** button to enter getRandomNumber()

Debugging (2)

- Examine variables in Variables View
  - Clicking on a variable will display its value
- Look at the result value in `getRandomNumber()`
- Click on the **Step Return** button
- Finish by clicking on the **Resume** or **Terminate** button

Play (if time)

- Add `printf()` to program
- Change variable name

Module 3: Installing Eclipse

- Objective
  - To learn how to install Eclipse
  - To install Eclipse on your laptop
  - This is an optional module
- Contents
  - Software prerequisites
  - Installing Eclipse
  - Installing CDT and PTP

Software Prerequisites

- Java (1.5 or later)
- Cygwin (for Windows)
- make, gcc, and gdb (or other vendor compilers)
- gfortran (only required for Fortran support)
- OpenMPI or MPICH2 (only required for PTP Runtime)

Pre-installation Overview

	Eclipse	C/C++/Fortran		Fortran	PTP
	Java	Cygwin	make/gcc /gdb	gfortran	OpenMPI
Windows	install	install		install	
Linux	install		install	install	build & install
MacOS X	update			install	build & install

Java Installation

- Download Sun or IBM versions
  - Only need Java runtime environment (JRE)
  - Java 1.5 is the same as JRE 5.0
- Latest Sun JRE is in the java folder on tutorial CD:
    - jre-1_5_0_08-windows-i586-p.exe
    - jre-1_5_0_08-windows-amd64.exe
    - jre-1_5_0_08-linux-i586.bin
    - jre-1_5_0_08-linux-amd64.bin
    - J2SE50Release3.dmg

Java Installation (Linux)

- Open a terminal window
- Mount your CDROM if necessary

```
mount /media/cdrom
```

- Enter the commands below:
  - Replace ***cdrom*** with the location of your CDROM (usually **/media/cdrom**) and ***arch*** with your computer architecture (usually **i586**)

```
cd
cdrom/java/jre-1_5_0_08-linux-arch.bin
```

  - hit space until you are asked to agree to license, then enter 'yes')

```
PATH=~/jre1.5.0_08/bin:$PATH
```

- Add to your PATH in your login file if required

Java Installation (MacOS X)

- Check Java version
  - Open **/Applications/Utilities/Terminal**
  - Enter the command:

```
java -version
```

- If java version is not "1.5.0_NN" or similar
  - From the Finder, open **TutorialCD**
  - Open the java folder
  - Double-click on the **J2SE50Release3.dmg** disk image
  - Open the mounted disk and double-click on the Installer icon and follow instructions
  - Open the Java Preferences Utility in **/Applications/Utilities/Java/J2SE 5.0/**
  - Set the **Java Applet Runtime Settings** to use version J2SE 5.0

Java Installation (Windows)

- Open the **TutorialCD** in **My Computer**
- Open the **java** folder
- Double-click on **jre-1_5_0_08-windows-*arch***
  - Replace ***arch*** with your computer architecture (most likely **i586-p**)
- Follow installer wizard prompts
  - Accept default options

Eclipse Installation Overview

	Eclipse SDK	CDT Feature	PTP Feature	PTP Runtime
Windows	install	update	update	N/A
Linux	install	update	update	build & install
MacOS X	install	update	update	install

Eclipse Installation

- The base component of Eclipse is known as the Eclipse SDK
- The Eclipse SDK is downloaded as a single zip or gzipped tar file
- Unzipping or untaring this file creates a directory containing the main executable
- Copies of the Eclipse SDK for each operating system type are located in the **eclipse** folder on the tutorial CD

Eclipse SDK Installation (Linux)

- Open a terminal window
- Mount CDROM if not already
- Enter the commands below:
  - Replace ***cdrom*** with the location of your CDROM (usually **/media/cdrom**)
  - If your machine is *not* x86 based, use either the **-ppc** or **-x86_64** versions (not on CDROM)

```
cd
tar -zxvf cdrom/eclipse/eclipse-SDK-3.2-linux-gtk.tar.gz
```

Eclipse SDK Installation (MacOS X)

- From the Finder, open **TutorialCD**
- Open the **eclipse** folder
- Double-click on **eclipse-SDK-3.2-macosx-carbon.tar.gz**
- Will create new eclipse folder in your **downloads** location
  - Specified in Safari
- Drag new **eclipse** folder to **Applications** (or wherever you want to install it)

Eclipse SDK Installation (Windows)

- Open the **TutorialCD** in **My Computer**
- Open the **eclipse** folder
- Unzip the following file:

  eclipse-SDK-3.2-win32.zip

- Choose a location on your hard drive where you want to install Eclipse (e.g. C:\)
  - An **eclipse** folder will be created at this location

Starting Eclipse

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

```
cd
eclipse/eclipse &
```

- **MacOS X**
  - From finder, open the **Applications**▸**eclipse** folder
  - Double-click on the **Eclipse** application
- **Windows**
  - Open the **eclipse** folder
  - Double-click on the **eclipse** executable

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

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 manually by copying files to the features and plugins directories in the main eclipse directory
- Eclipse comes preconfigured with the Callisto Discovery Site that contains a large number of features

Configuring an Update Site

- Two types of sites: remote and local
  - Archive Site is a local site packaged as a zip or jar file
- Remote Site requires Internet access
- We will use a Local Site for the tutorial
- To configure a Local Site:
  - Choose **Help ▸ Software Updates ▸ Find and Install...**
  - Select **Search for new features to install**
  - Click **Next >**
  - Click **New Local Site...**
  - Navigate to your CDROM, select the **updatesite** folder and click **Choose** (**OK** on Linux)
  - Click **OK**

C/C++ Developer Tools (CDT) Feature Installation

- The CDT feature can be installed using the Eclipse Callisto Discovery Site
  - Requires Internet access
- CDT is also available on the tutorial CDROM
  - We will install CDT via the Local Site

parallel tools platform
http://eclipse.org/ptp

Parallel Tools Platform (PTP) Feature Installation

- Will install Fortran support and the Parallel Language Development Tools (PLDT)
- Can install the PTP feature using the PTP Remote Update Site
  - Requires Internet access
  - Specify the following Name and URL:
    - PTP
    - http://download.eclipse.org/technology/ptp/update-site
- PTP is also available on the tutorial CDROM
  - We will install PTP via the Local Site

Installing the CDT and PTP Features

- Use the previously created local update site to install the CDT feature:
  - Select **Tutorial Site**
  - Click **Finish**
  - From **Search Results**, select **Tutorial Site**
  - Click **Next >**
  - Accept the license terms
  - Click **Next >**
  - Click **Finish**
  - For **Feature Verification**, click **Install All**
  - Restart workbench when asked

Installing the PTP Runtime

- Normally installed on a parallel machine
  - e.g. a cluster
- Can install on a single system
- Not currently available for Windows
- Requires OpenMPI to be built and installed
  - This process depends on the type of machine
- In most cases, installing PTP on Linux can be done using pre-compiled binaries, but sometimes must be done from source
- Installing PTP on MacOS X can be done using pre-compiled binaries
- Beyond the scope of this tutorial

parallel tools platform
http://eclipse.org/ptp

Module 4: Advanced Development

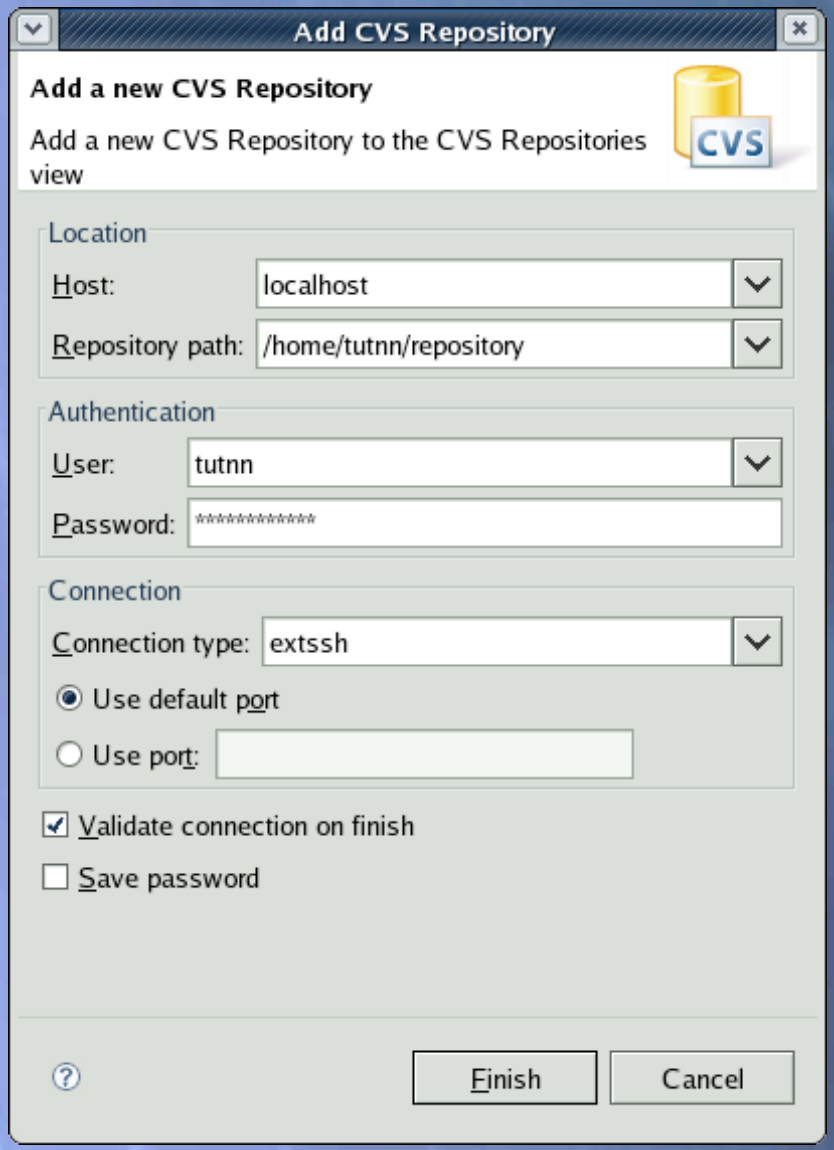
- Objective
  - Create and build a Standard Make Project from source files in CVS
- Contents
  - Version control
  - Standard Make Projects
  - Fortran
  - Refactoring
  - Searching

Version Control (CVS)

- Version control provided through the **Projects View Team** context menu
- Provides familiar actions:
  - Commit…
  - Update…
- Also less used tasks:
  - Create/Apply Patch…
  - Tag as Version
  - Branch…
  - Merge…
  - Add to .cvsignore…

Add Repository Location

- Select **Window ▸ Open Perspective ▸ Other...**
- Select **CVS Repository Exploring** then **OK**
- Right-click in **Repository View**, then select **New ▸ Repository Location...**
- Fill out options as shown
  - Replace *tutnn* with your username
  - Enter your password in the **Password** field
- Select **Finish**

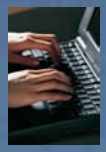


Checkout POP Code

- Open the repository, then open HEAD
  - Right-click on **POP ▸ Check out As...**
  - Select **Finish**
  - Select **Fortran ▸ Standard Make Fortran Project**
  - Select **Next>**
- Enter **Project name** and **location**
  - Workspaces tend to be temporary so do not use default location
- Select **Finish**
- Switch to the **Fortran Perspective**

Standard Make Project

- *Standard Make* projects are different from *Managed Make* projects
  - Project Makefiles must be created
- Can create project Makefiles with the Makefile Editor
  - Syntax highlighting and Outline view
- autoconf often used to create Makefiles for open source projects
  - Must refresh after running configure script
- Refresh whenever file system is modified outside of Eclipse

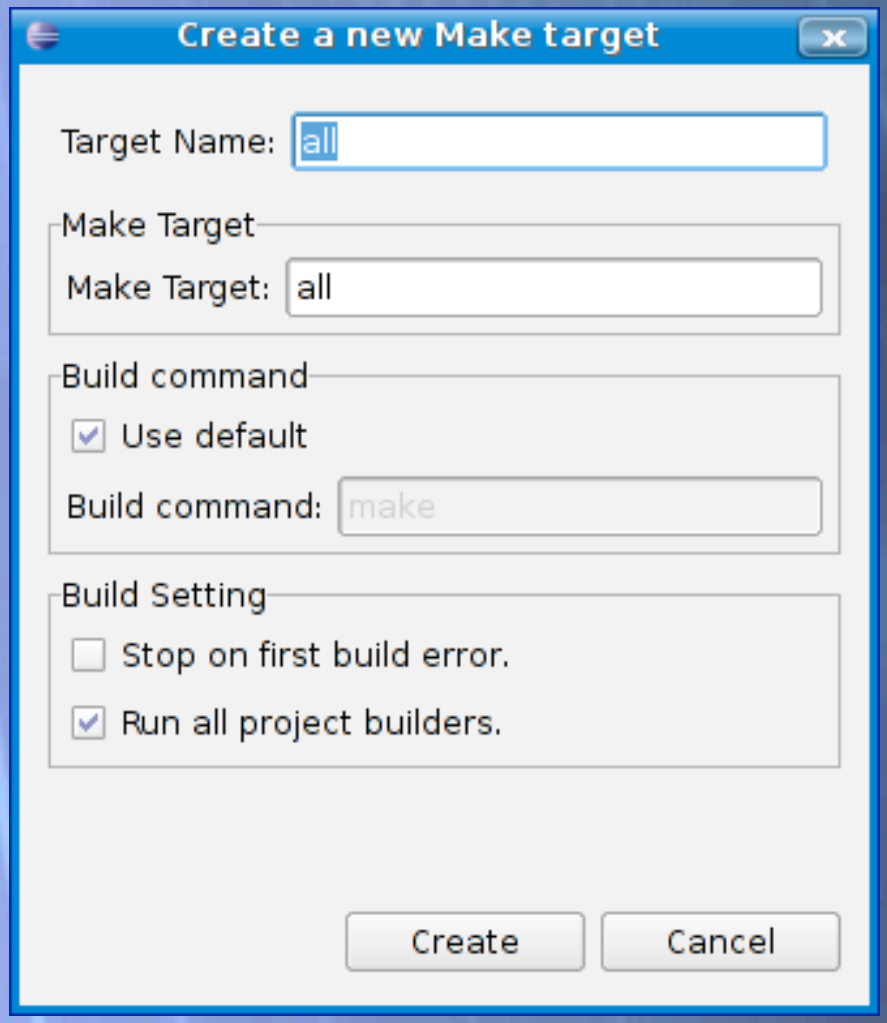


Building POP code

- Most projects will now have to be configured
  - This is project dependent
  - Do whatever is needed from a terminal window, often ./configure
  - This should create/configure all project Makefiles
  - (We have already done this for you)
- Refresh the project to sync with file system
  - Right-click on project and select **Refresh**

Building

- Create a Make Target named ‘all’
  - Right-click on the project in **Make Targets View**
  - Select **Add Make Target**
  - Select **Create**



Fix Error

- Build the POP application
  - Expand **POP** folder in **Make Targets** view
  - Double-click on the **all** make target
- Notice the red error icon on the project folder 
- Open source file with the error marker
  - Scroll to error by selecting the red icon in the gray border on right side of editor
  - Fix the error by deleting an '=' sign
  - Save the file
- Rebuild
- We won't actually run the code
  - Running of parallel codes is covered later

Create a Bookmark

- A bookmark reminds you of useful information
- Add a bookmark by right-clicking in the gray border on left side of editor and select **Add Bookmark...**
  - Provide a bookmark name, then select **OK**
- View bookmarks by selecting **Window▸Show View▸Other...**
  - Open **General** and select **Bookmarks**

Commit Changes

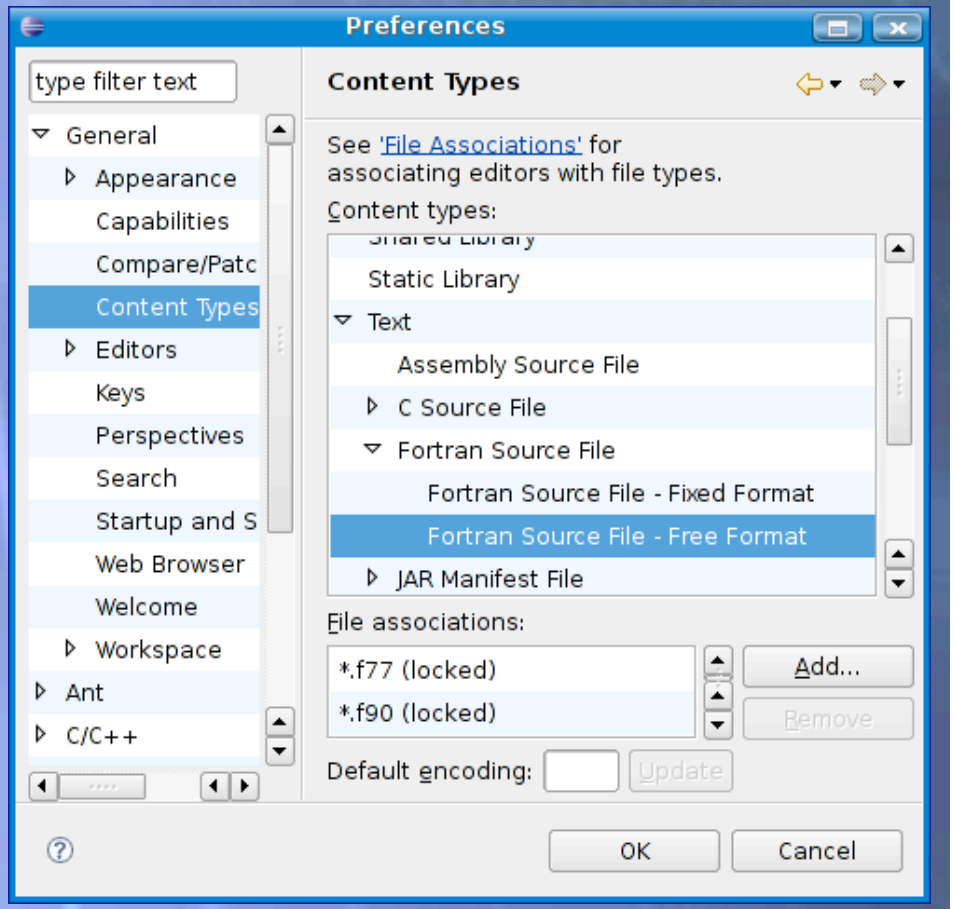
- Select the **Fortran Projects** view
- Notice the '>' before the file name(s)
  - Indicates a file has been modified
- Right-click on the **POP** Project
  - Select **Team ▸ Synchronize With Repository**
  - Confirm switch to perspective if asked
- Expand the **POP** folder
  - Double-click on a file name to view differences
- Commit changes
  - Right-click on the file name, select **Commit...** and enter a comment
  - Select **Finish**

Advanced Features

- Associating file types with editors
  - *.f should be opened in Fortran editor
- Refactoring
  - Modifying source code without changing its external behavior
- Searching
  - Based on languages elements, not just textual

Editors and File Types

- It is possible to associate an editor with a file type
- Choose **Window ▸ Preferences**
- Expand **General**
- Select **Content Types**
- Expand **Text**
- Expand **Fortran Source File**
- Select **Free Format**
- Select **Add...**
- Enter ***.f** and select **OK**



Refactoring

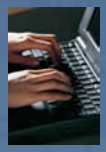
- Rename
- Introduce Implicit None
- Constant promotion
  - Select **Fortran** Perspective
  - Open **forcing_ap.F90**
  - Right-click in editor view and select **FDT Refactor ▸ Constant Promotion...**
  - Make changes by repeatedly selecting **Replace/Next**
  - Select **Close**

Searching (1)

- Language-based searching
- Search for Language Elements
  - e.g., C++ Class, Function, Method, Variable, Field, Namespace
- Limit search to Declarations, Definitions, References
- Type navigation
- Fortran
  - text based only for now

Searching (2)

- C/C++ Searching
  - Select **Search▶C/C++...**
  - Consider the possibilities!
- Search for Fortran TYPE declarations
  - Select **File Search** tab
  - Deselect **Case sensitive**
  - Enter 'type' in **Containing text** field
  - Enter '*.F90' in **File name patterns** field
  - Select **Search**

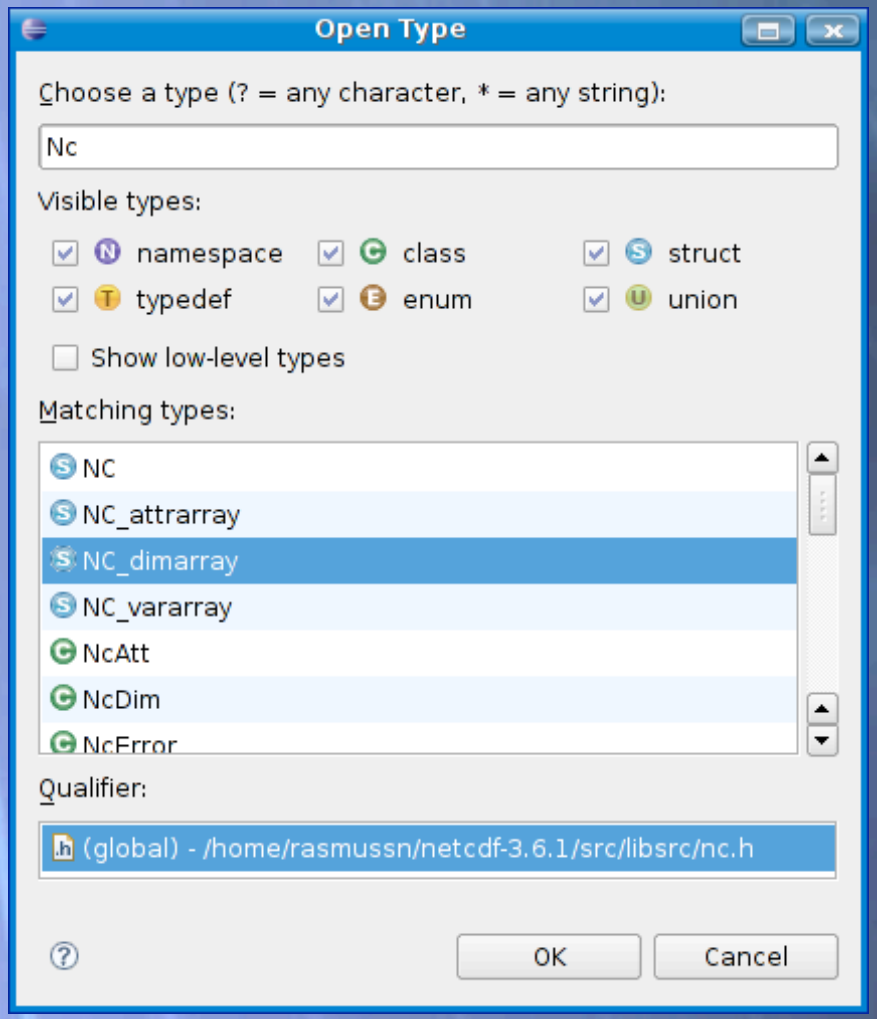


Searching (3)

- The **boundary.F90** file should be displayed in the **Search** view
- Double-click on the file in the **Search** view to open the editor
- Select some of the light-blue icons in right-gray border of editor
- Can you find a TYPE() declaration?
  - This highlights the advantage of language-aware searching

Type Navigation

- Doesn't work yet for Fortran
- Choose **Fortran or C/C++ Perspective**
- Select **Navigate ▸ Open Type...**
- Enter 'Nc' in text box
- All matching types are displayed



parallel tools platform
http://eclipse.org/ptp

Module 5: PTP and Parallel Language Development Tools

- Objective
  - Learn to develop and run a parallel program
- Contents
  - Learn to use PTP's Parallel Language Development Tools
  - Learn to launch a parallel job and view it via the PTP Runtime Perspective

parallel tools platform
http://eclipse.org/ptp

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

parallel tools platform
http://eclipse.org/ptp

Parallel Language Development Tools (1)

- Features
  - Analysis of C and C++ code to determine the location of MPI and OpenMP Artifacts
  - "Artifact View" indicates locations of Artifacts found in source code
  - Navigation to source code location of artifacts
  - Content assist via **ctrl+space**
  - Hover help
  - Reference information about the MPI and OpenMP calls via Help
    - **F1** on Windows
    - **ctrl-F1** on Linux
    - **Help** on Mac

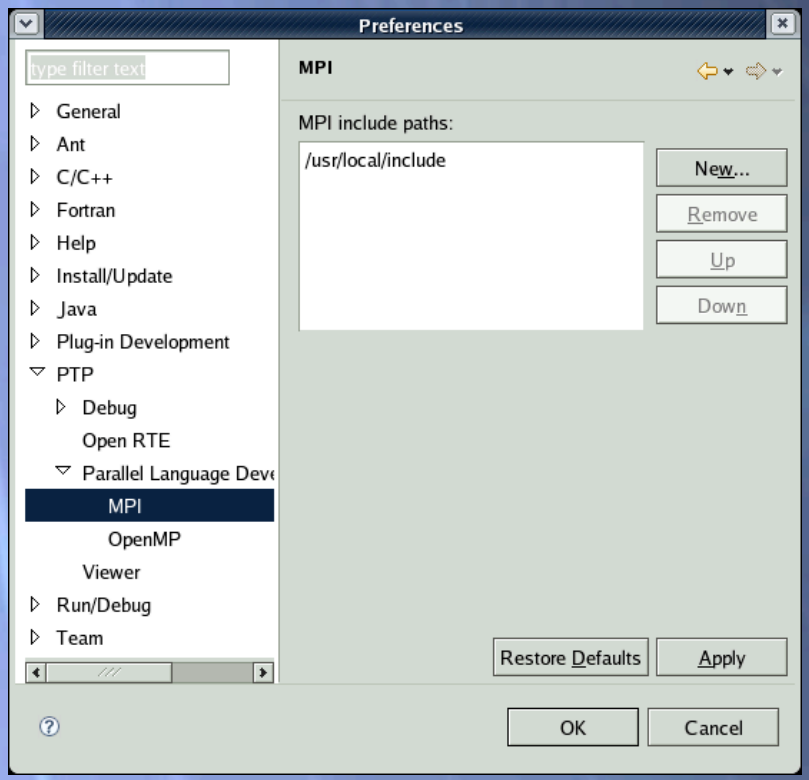
parallel tools platform
http://eclipse.org/ptp

Parallel Language Development Tools (2)

- These features for MPI were available in the PTP v1.0 MPI Development Tools
- New features for PLDT v1.0:
  - OpenMP features similar to MPI
  - OpenMP problems view of common errors
  - OpenMP "show #pragma region" action
  - OpenMP "show concurrency" action
  - MPI New project wizard automatically configures Managed Make MPI projects.

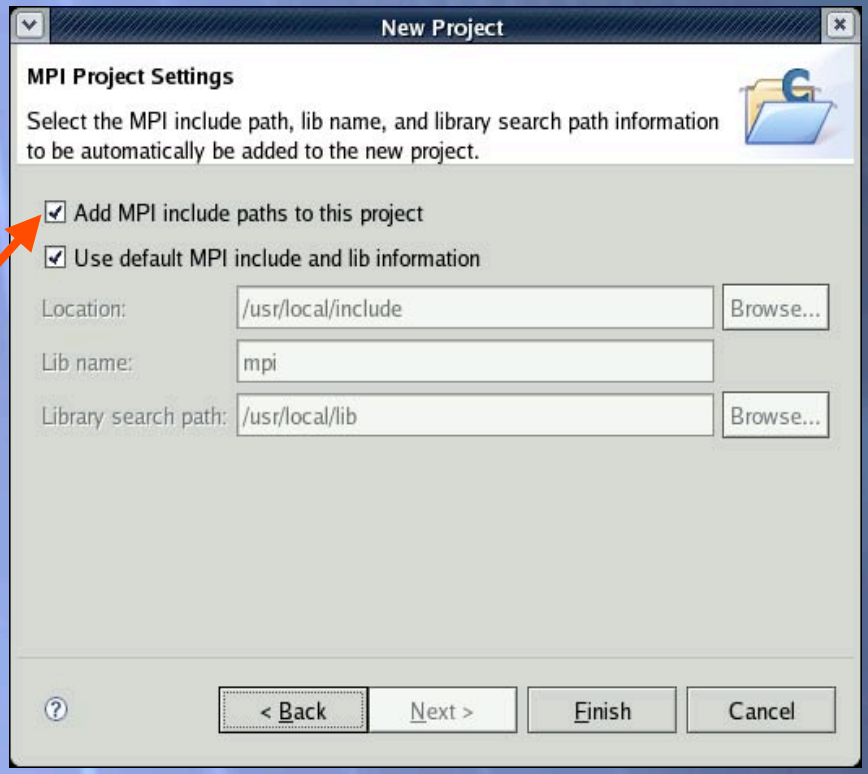
PLDT Preferences

- To use the PTP Parallel Language Development Tools feature for MPI development, you need to
  - Specify the MPI include path in preferences
- Open **Window ▸ Preferences...**
  - Open the **PTP** item
  - Open the **Parallel Language Development Tools** item
  - Select **MPI**
  - Select **New...** to add MPI include path
- If running OpenMP, add its include file location here too



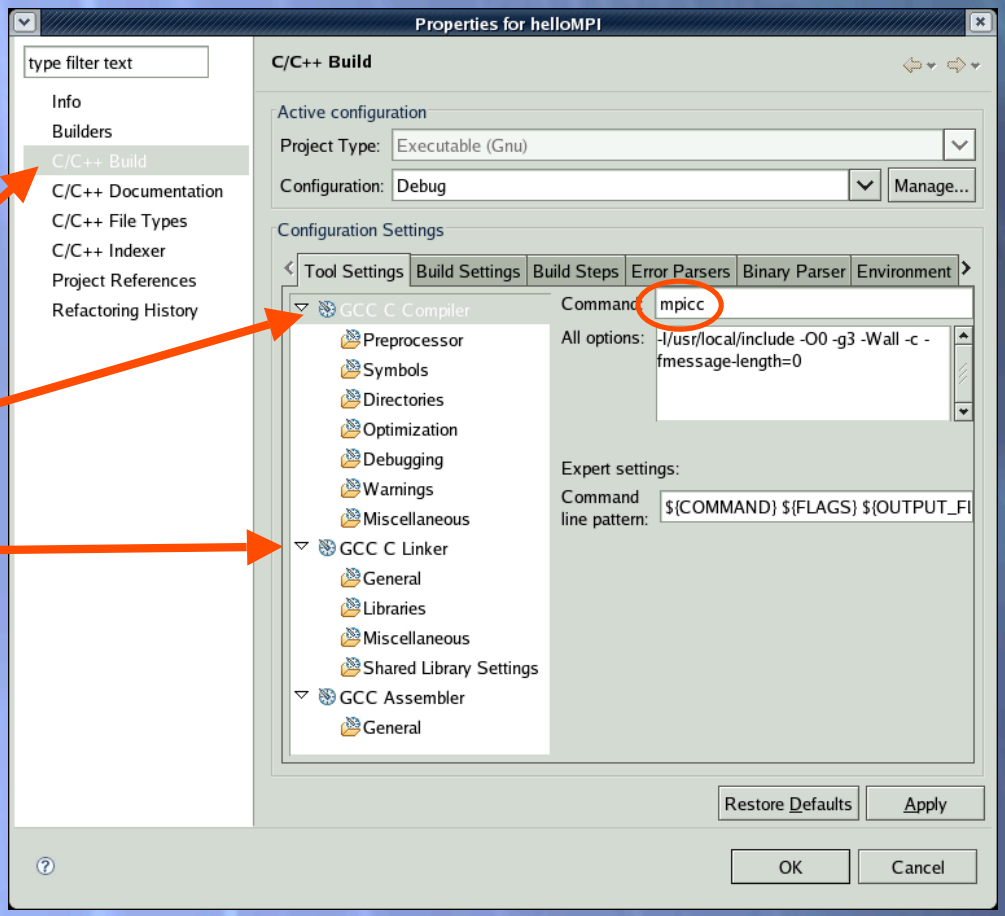
Managed Make Project Setup

- Create a new **Managed Make C Project**
- Name the project e.g. 'helloMPI'
- After the **Additional Project Settings** wizard page, you should see the **MPI Project Settings** wizard page
- Check **Add MPI include paths to this project** to add include and lib paths to MPI projects
- Currently only works on Managed Make C projects
- (Note: we plan to add a **New MPI Project** project type in a later version)



Changing the MPI build Command

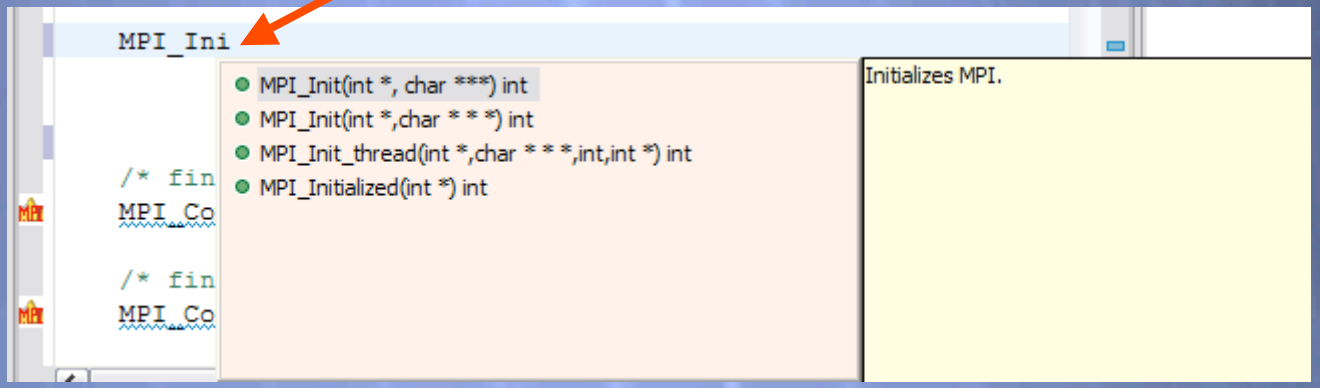
- If you wish to change the way your MPI program is built:
  - Open the project properties
  - Select **C/C++ Build**
  - Select **GCC C Compiler**
    - Change the command
  - Select **GCC C Linker**
    - Change the command
  - It's also possible to change compiler/linker arguments



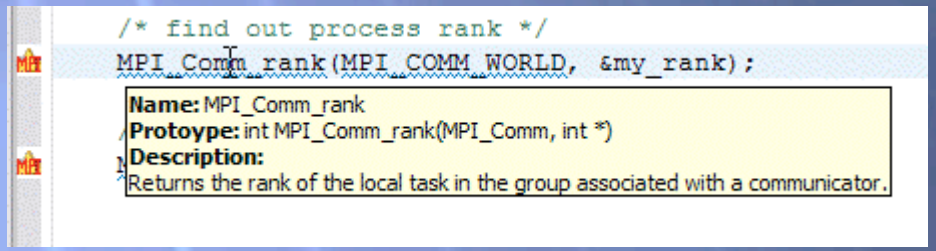
parallel tools platform
http://eclipse.org/ptp

Content Assist

- Type an incomplete MPI function name e.g. "MPI_Ini" into the editor, and hit **ctrl-space**



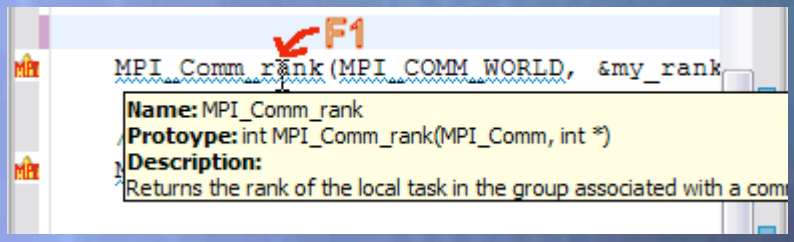
- Hover over the MPI Artifact identified in the source file to see additional information about that function call, for example

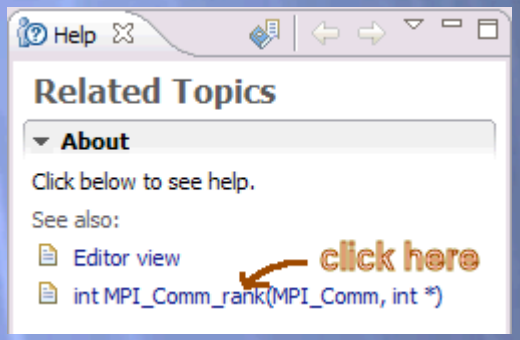


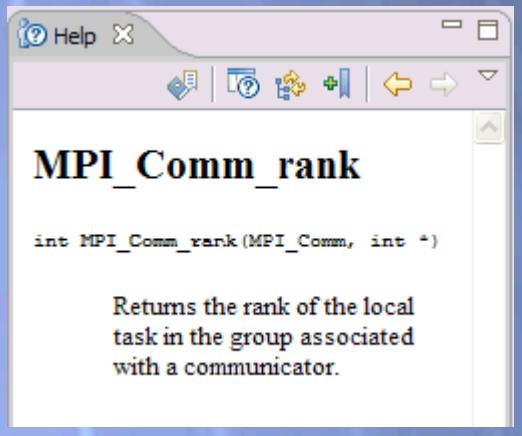
parallel tools platform
http://eclipse.org/ptp

Context Sensitive Help

- Press help key when the cursor is within a function name
  - Windows: **F1**
  - Linux: **ctrl-F1**
  - MacOS X: **Help**
- A help view appears (**Related Topics**) which shows additional information
- Click on the function name to see more information







Create Source File

- Create new source file called ‘mpitest.c’
  - Right click on project
  - Select **New▸Source File**
  - An editor view will automatically open on the empty file
- Double-click on any source file in project view to open an editor on that file

Enter Program

- Type in hello world program

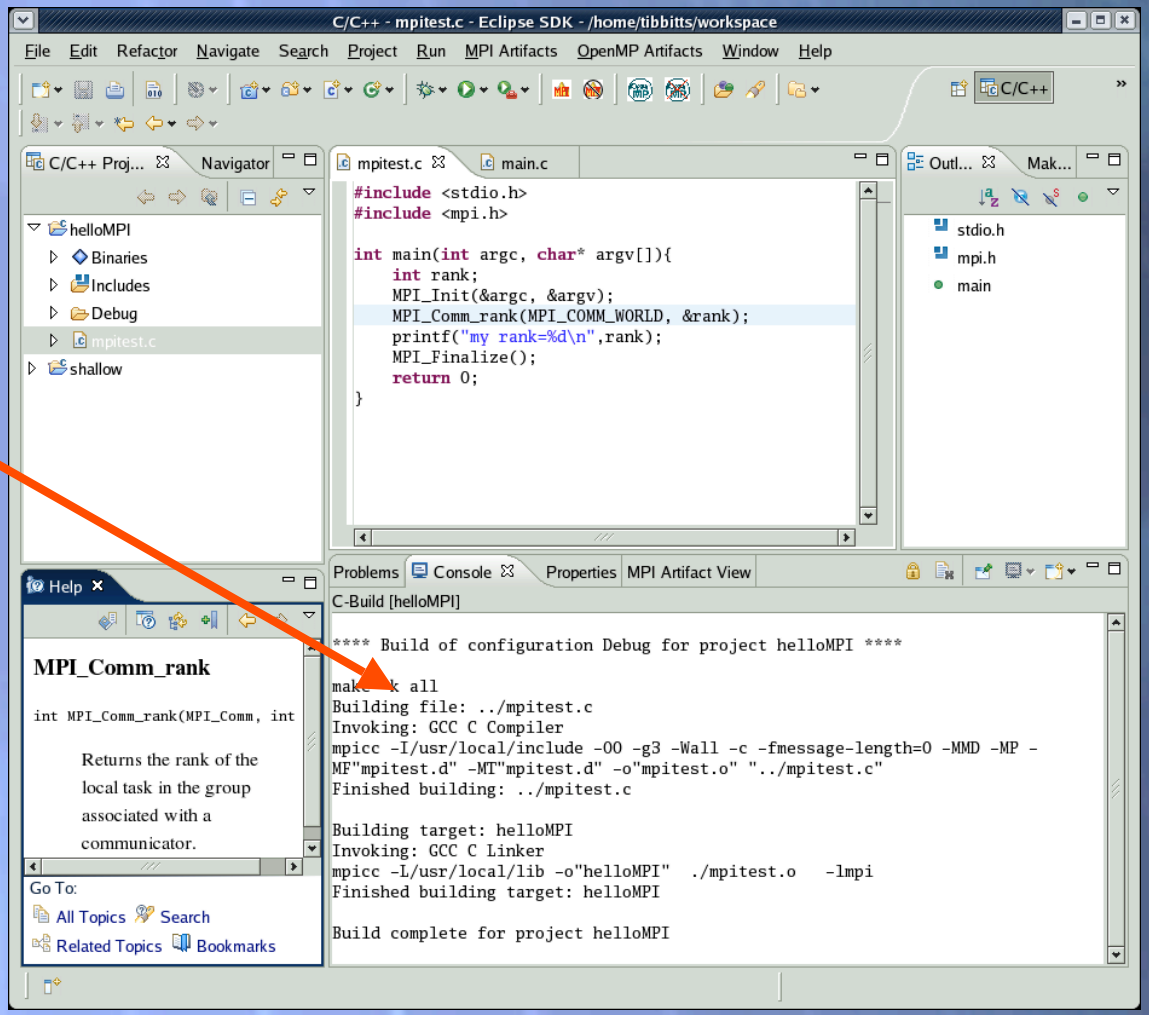
```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char *argv[]) {
    int rank;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    printf("my rank=%d\n", rank);
    MPI_Finalize();
    return 0;
}
```

- Try content assist
- Try context sensitive help

parallel tools platform
http://eclipse.org/ptp

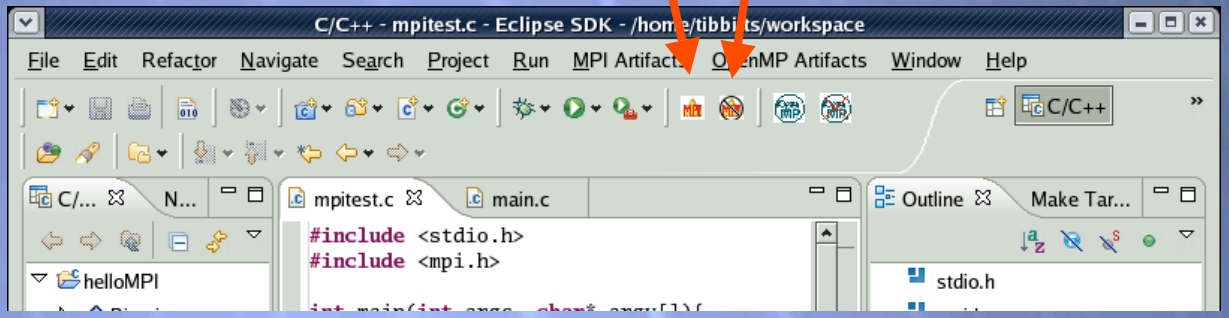
Project Created and Built

- Save the source file
  - Should build automatically
  - Console shows results of build



Show MPI Artifacts (1)

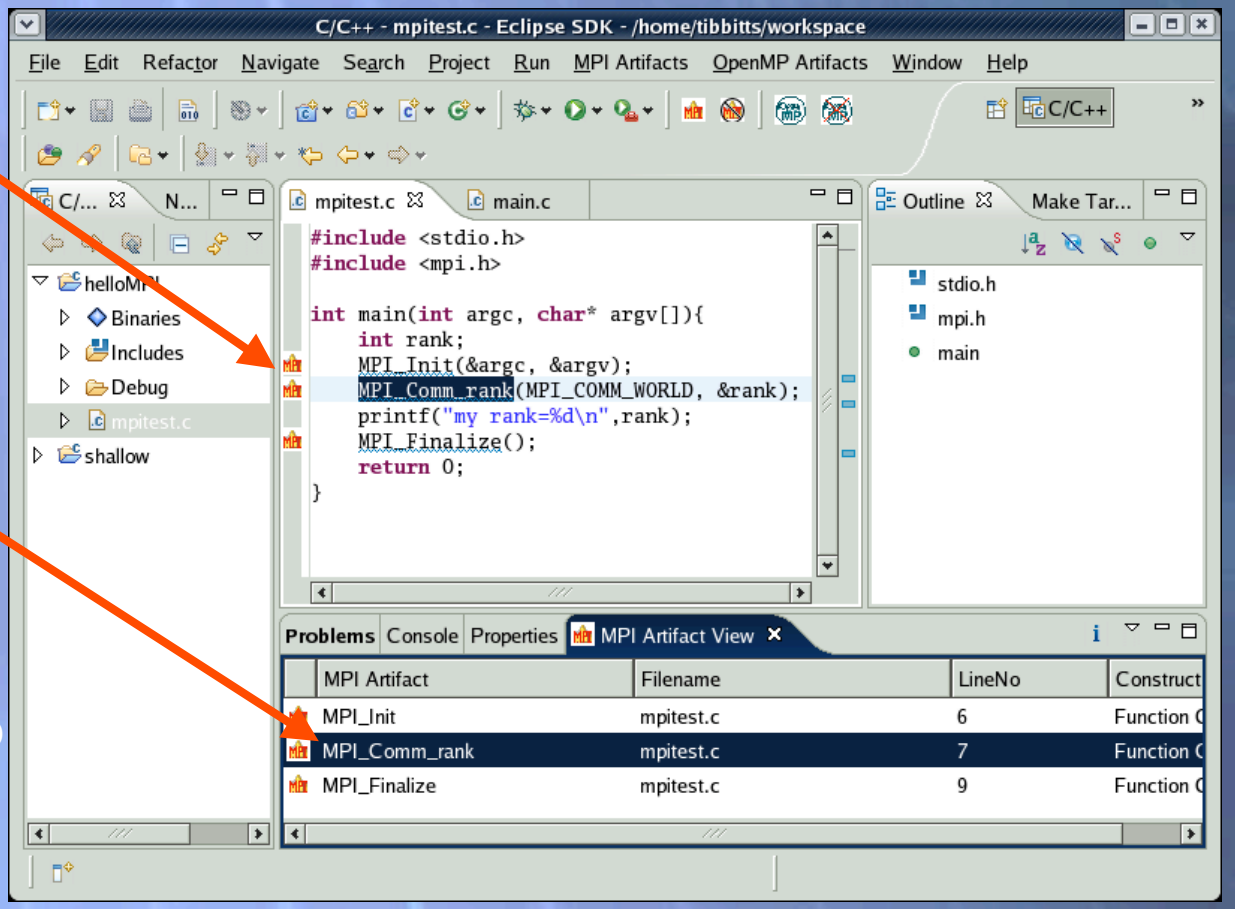
- Select the source folder/file to analyze, to find the MPI artifacts
- Click the **Add MPI Artifacts** button in the tool bar to annotate source with markers
- Click the **Clear MPI Artifacts** to remove all the MPI artifacts



parallel tools platform
http://eclipse.org/ptp

Show MPI Artifacts (2)

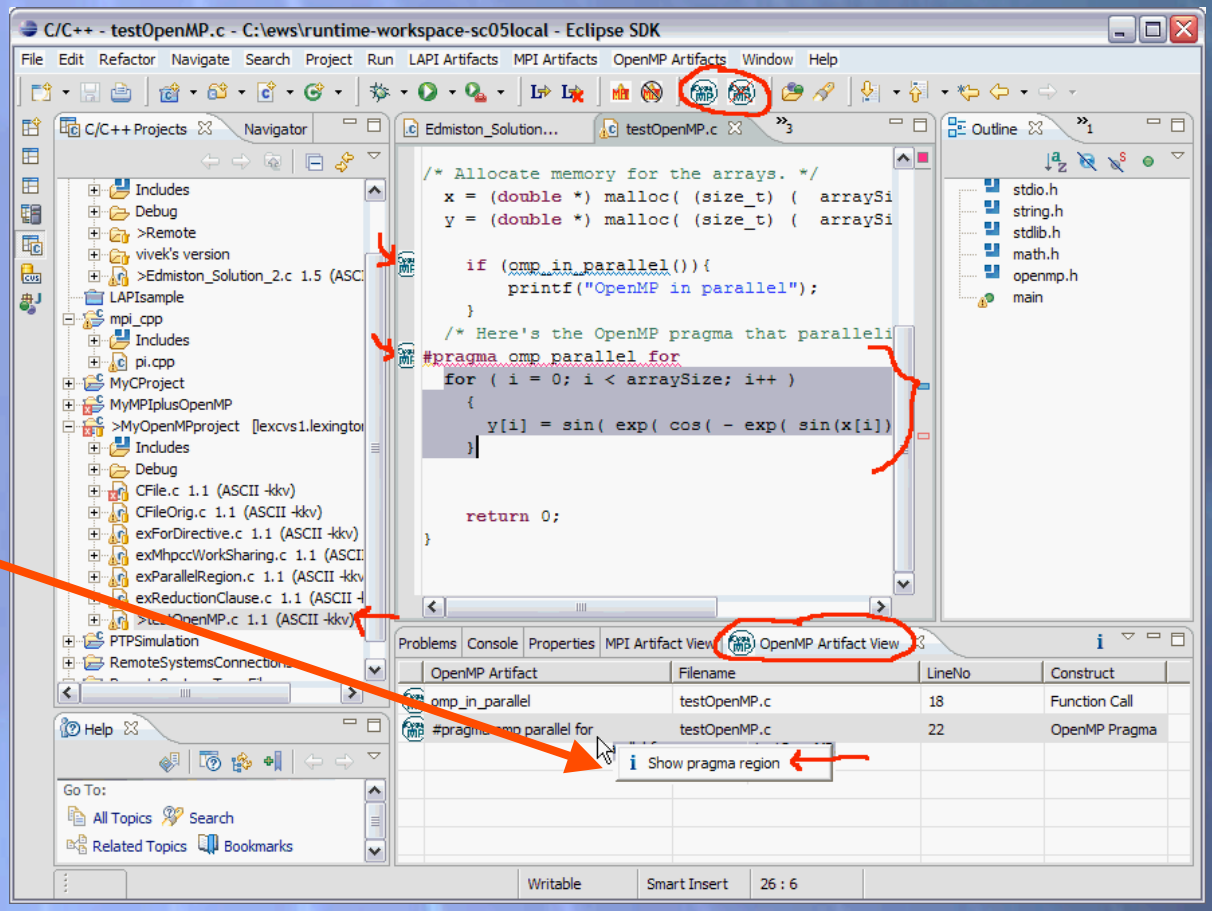
- Markers indicate the location of the artifacts in the editor
- In **MPI Artifact View** sort by any column (click on column heading)
- Navigate to source code line by double-clicking on the artifact
- Run the analysis on another file and its markers will be added to the view



parallel tools platform
http://eclipse.org/ptp

OpenMP tools

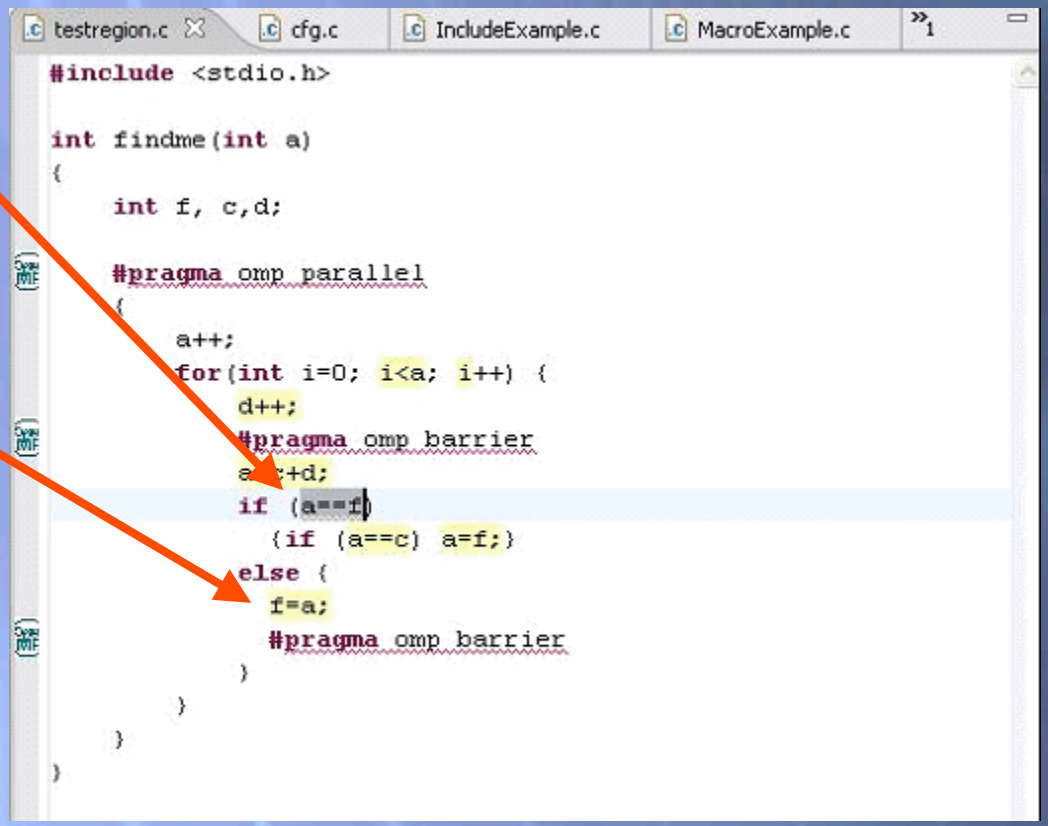
- Similar functions to the MPI tools
- Find artifacts, sort, navigate to source code
- Help, content assist, etc.
- Show #pragma regions
- Show Concurrency (next slide)



parallel tools platform
http://eclipse.org/ptp

Show Concurrency

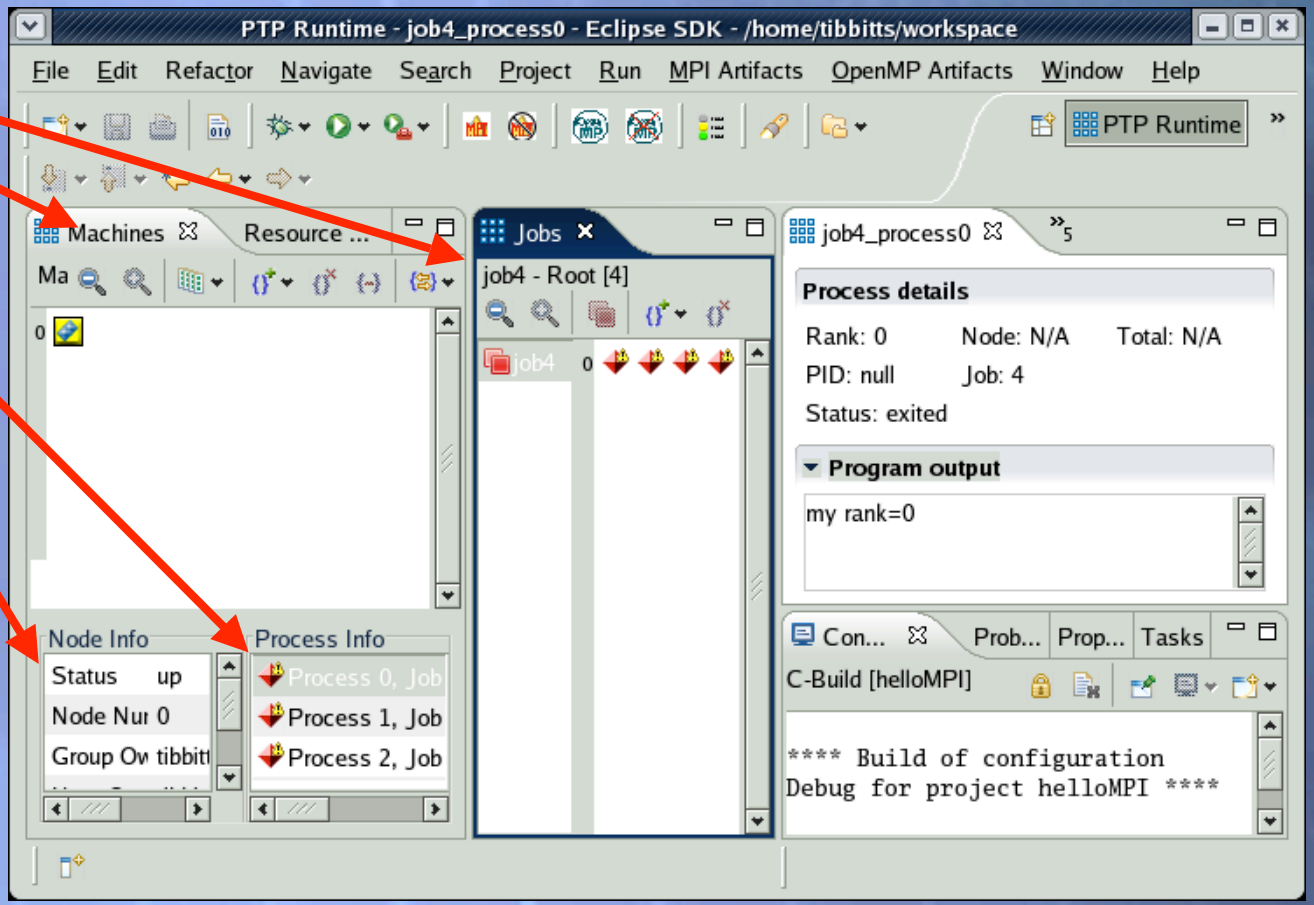
- Select a statement
- Select the context menu on the highlighted statement, and click **Show concurrency**
- Other statements will be highlighted in yellow
- The yellow highlighted statements can execute concurrently to the selected statement



parallel tools platform
http://eclipse.org/ptp

PTP Runtime Perspective (1)

- Jobs view
- Machines view
- Processes on node
- Node details

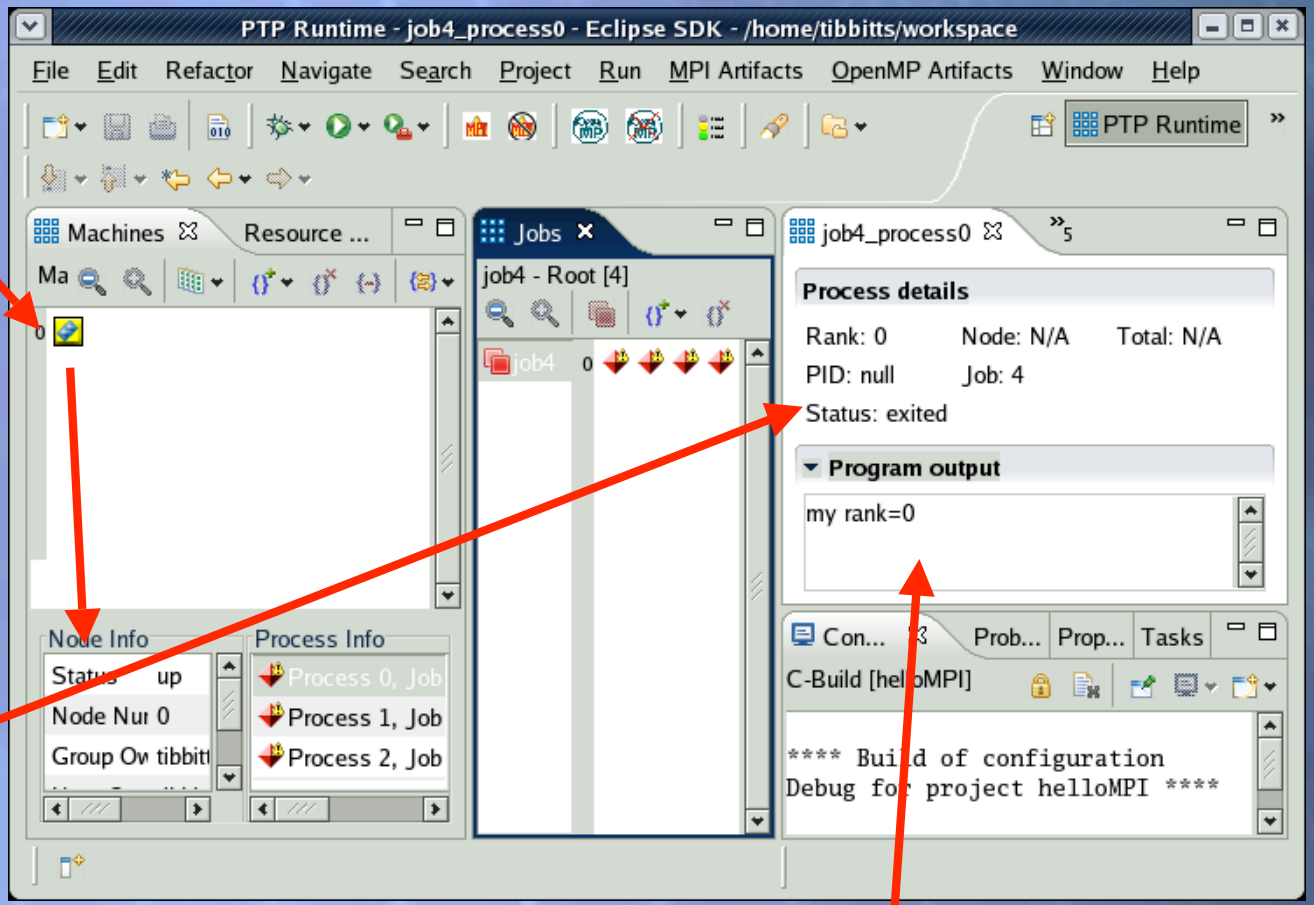


To see **Jobs** view *beside* (not behind) **Machines** view, drag its tab to the right until its outline occupies about half of the Machines view

parallel tools platform
http://eclipse.org/ptp

PTP Runtime Perspective (2)

- Double-click a node in machines view to see more info
- Hover over node or process for tooltip popup
- Double-click a process to see process detail

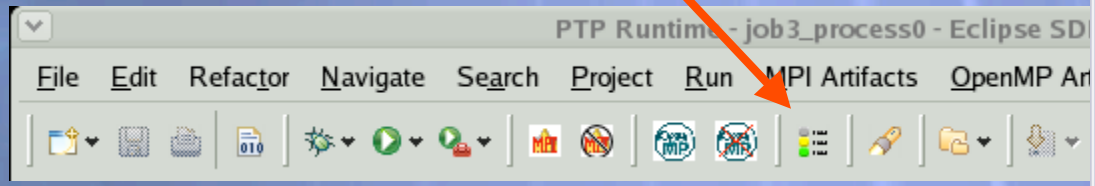


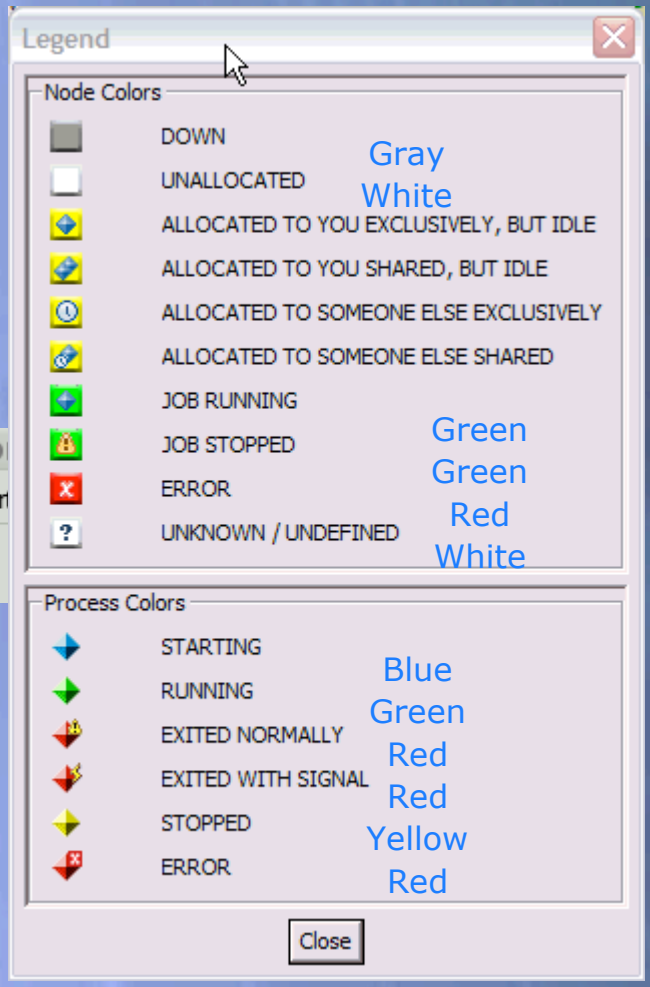
- Process output

parallel tools platform
http://eclipse.org/ptp

Process and Job Icons

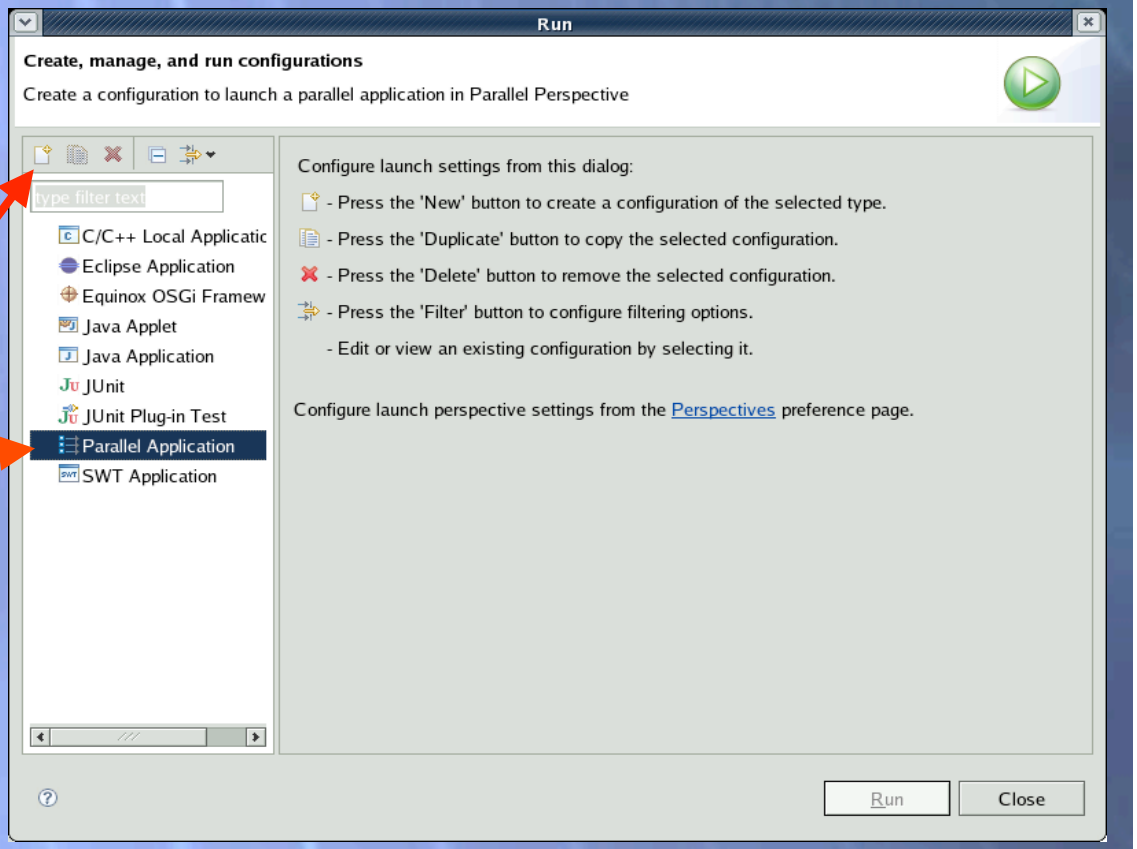
- Use legend icon in toolbar





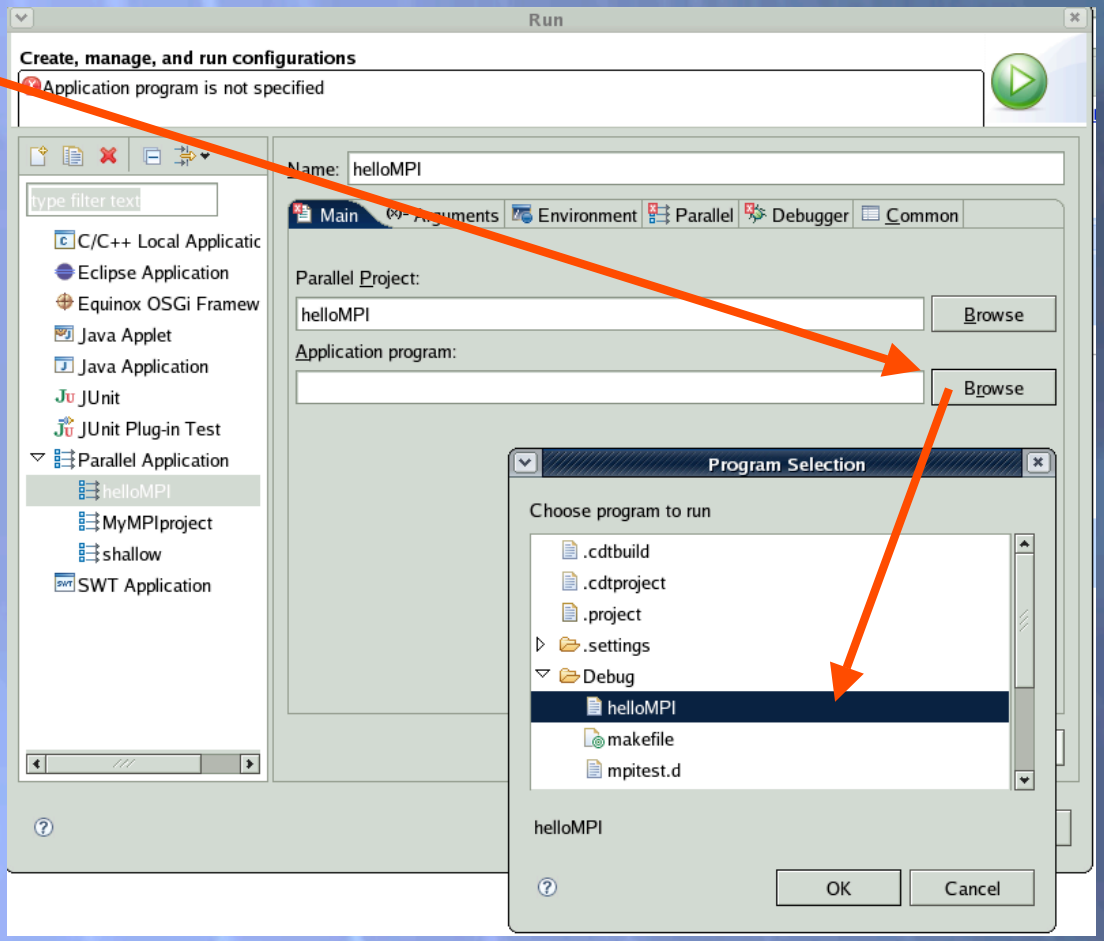
Running a Parallel Program (1)

- Create a parallel launch configuration:
  - Open the run configuration dialog **Run▸Run...**
  - Select **Parallel Application**
  - Select the **New** button



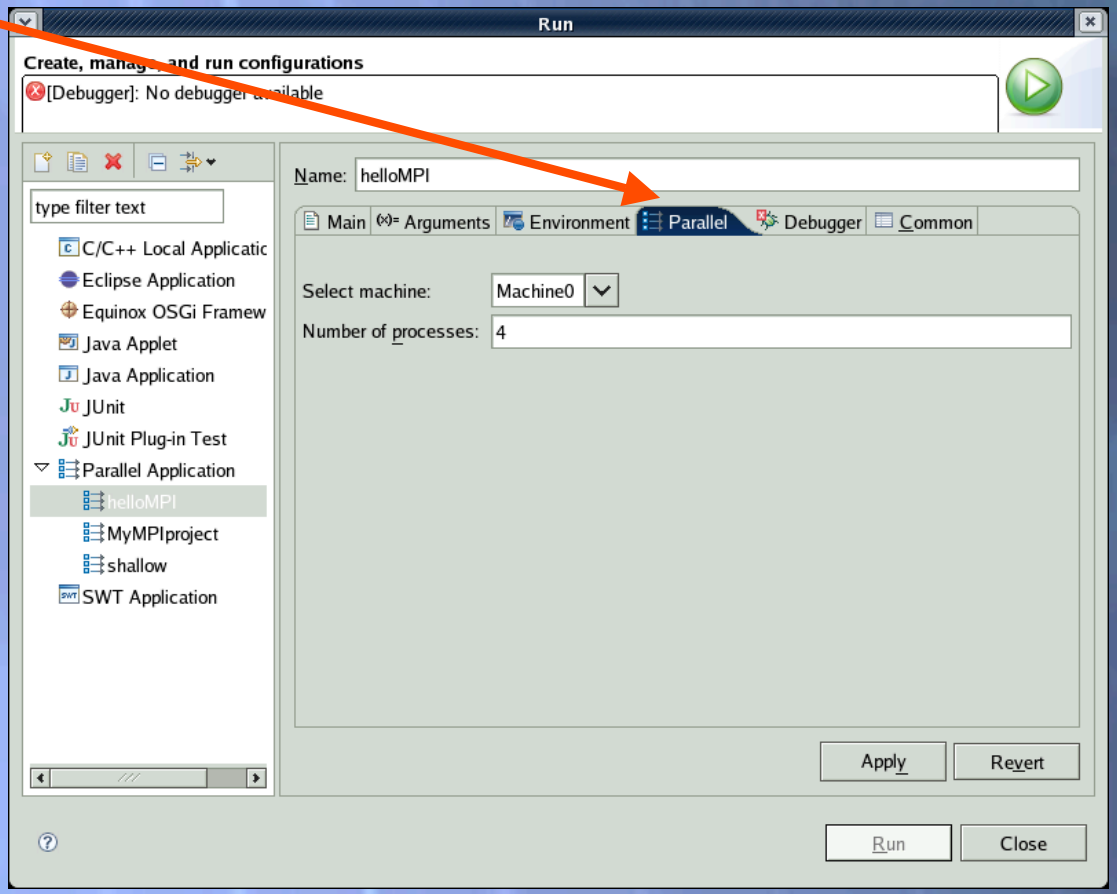
Running a Parallel Program (2)

- In **Main** tab, select **Browse** button to find the **Application program** (executable)
- Probably under **Debug** configuration



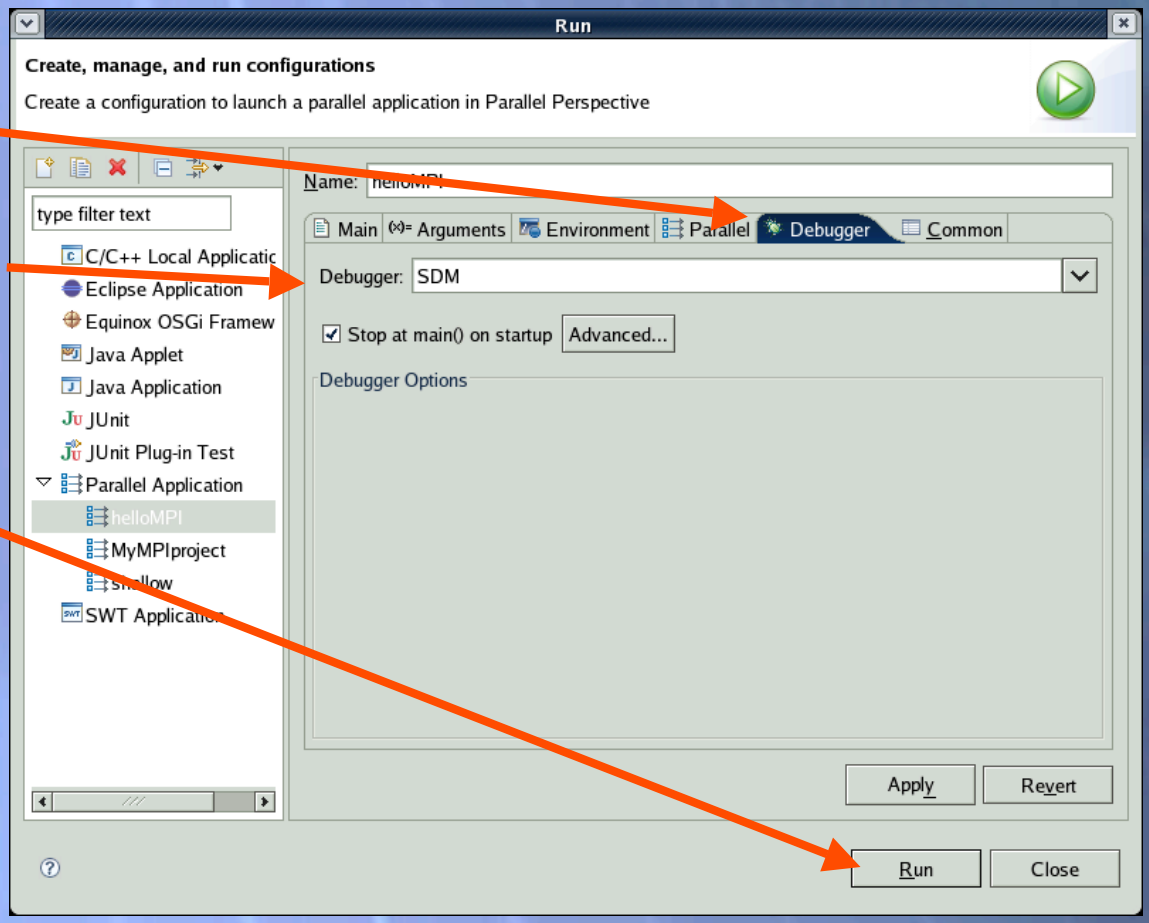
Running a Parallel Program (3)

- Select **Parallel** tab
- Enter the number of processes for this job
- 4 is a good number for this tutorial



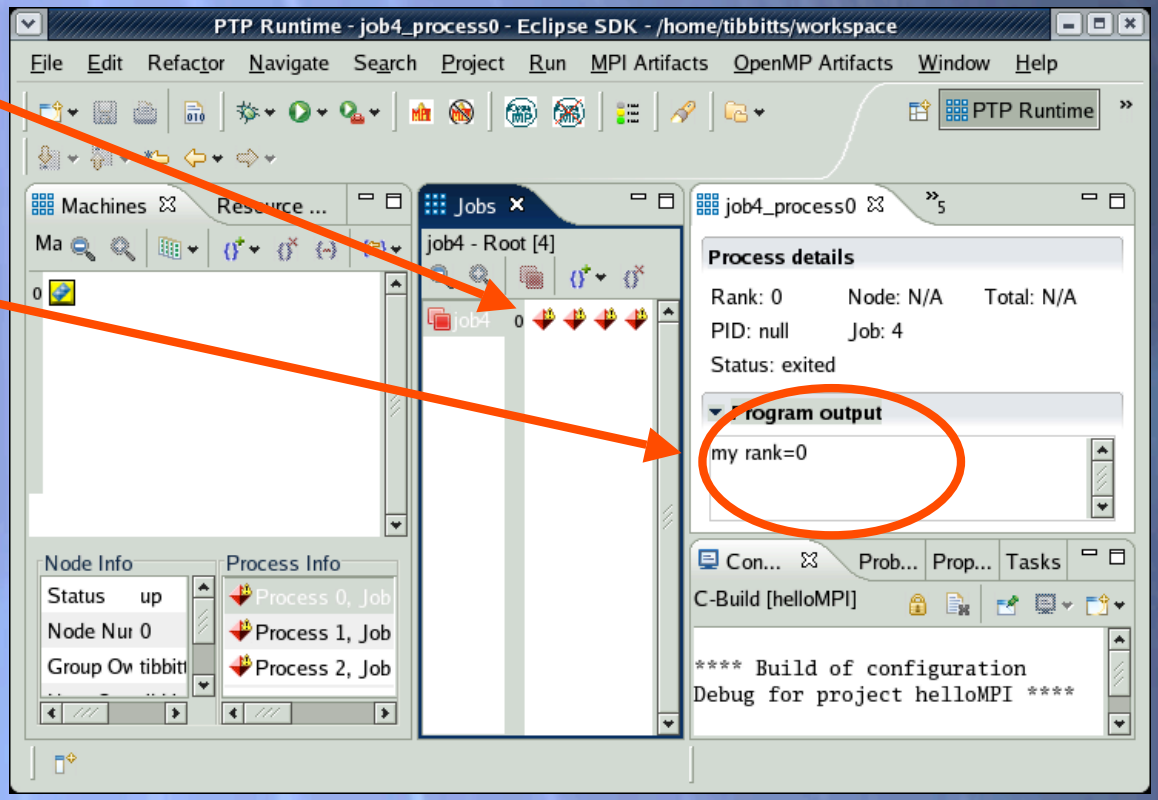
Running a Parallel Program (4)

- In **Debugger** tab, select **SDM** from the **Debugger** drop-down menu
- Select **Run** button to launch the application



Viewing Program Output

- Double-click on process 0 (diamond icon) in the jobs view
- Standard output will be visible in process detail view
- Double-click on other processes to see their stdout



parallel tools platform
http://eclipse.org/ptp

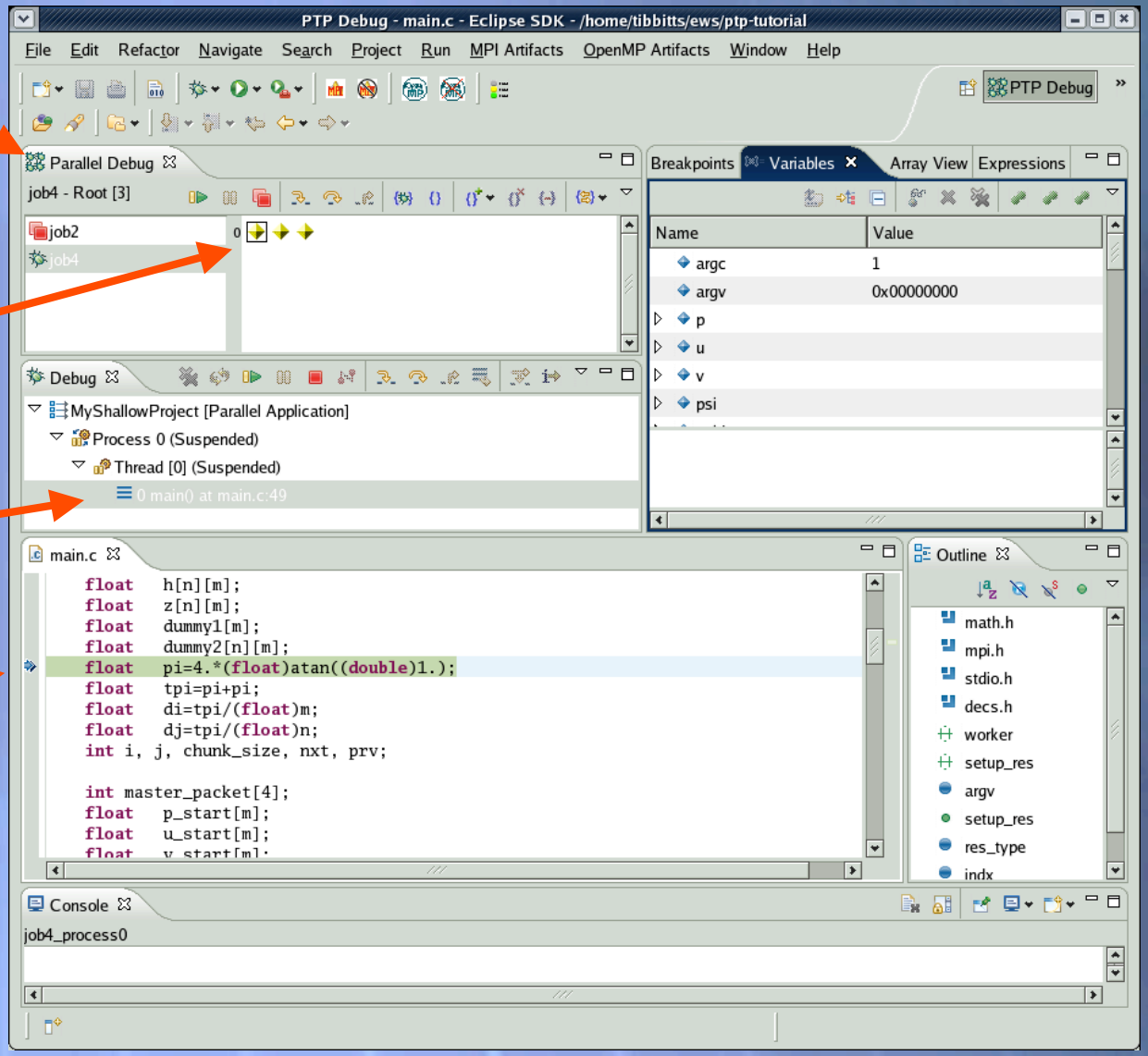
Module 6: Parallel Debugging

- Objective
  - Learn the basics of debugging parallel programs with PTP
- Contents
  - Launching a parallel debug session
  - The PTP Debug Perspective
  - Parallel Breakpoints
  - Current Instruction Pointer
  - Process sets: controlling sets of processes
  - Process registration: controlling individual processes

parallel tools platform
http://eclipse.org/ptp

The PTP Debug Perspective (1)

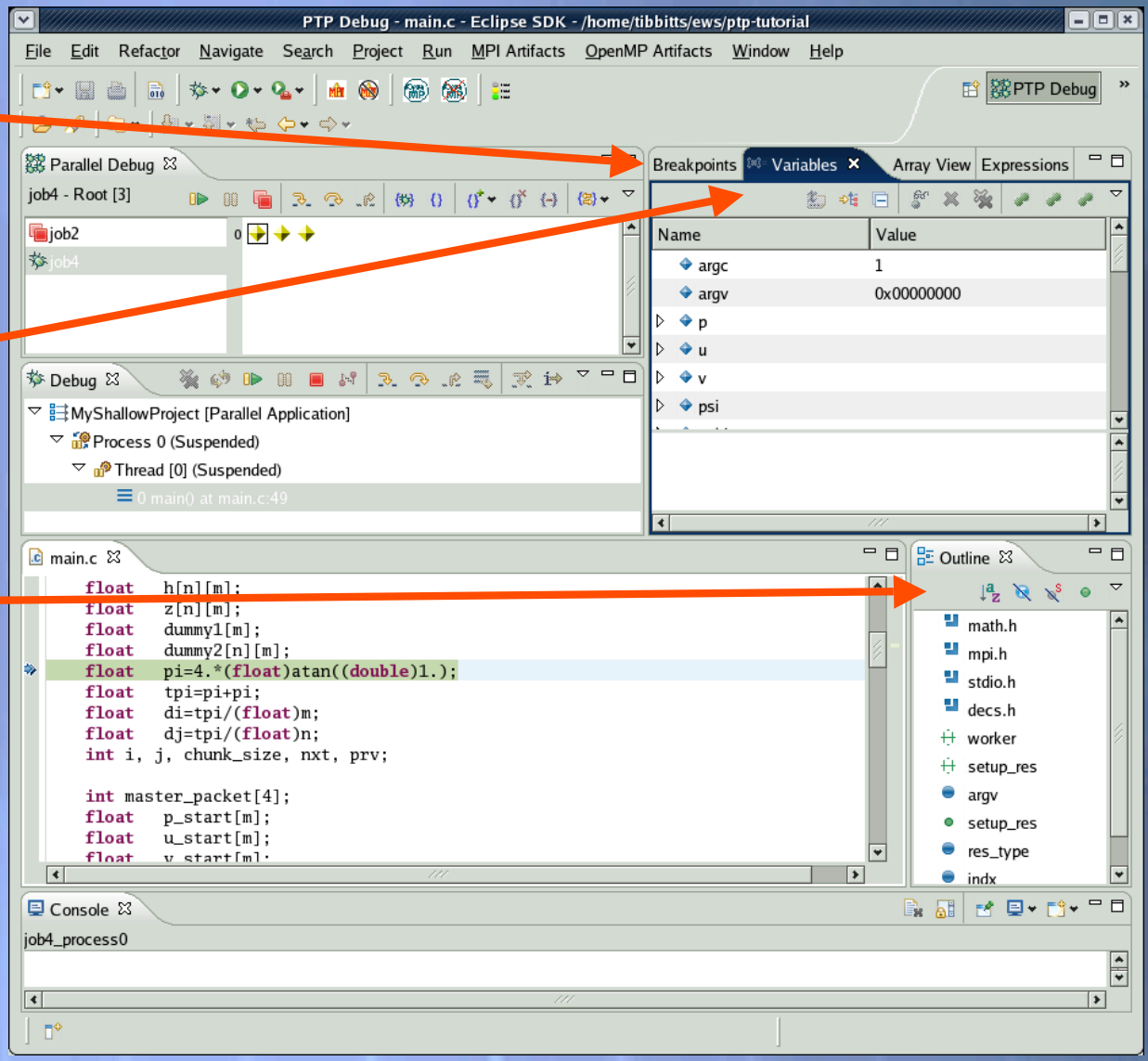
- **Parallel Debug** view shows currently running jobs, including the processes in the current process set
- **Debug** view shows threads and call stack for individual registered processes
- **Source** view shows the current instruction pointer for all processes



parallel tools platform
http://eclipse.org/ptp

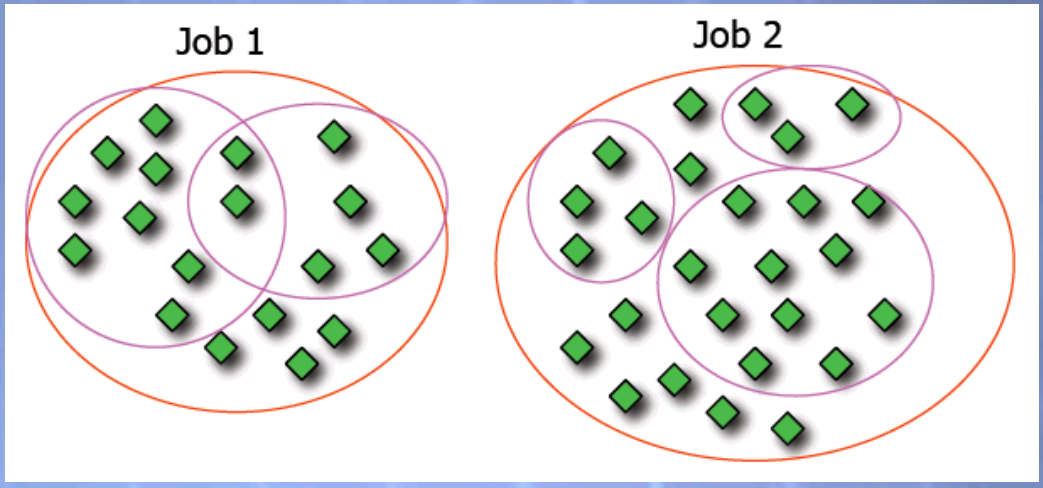
The PTP Debug Perspective (2)

- **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



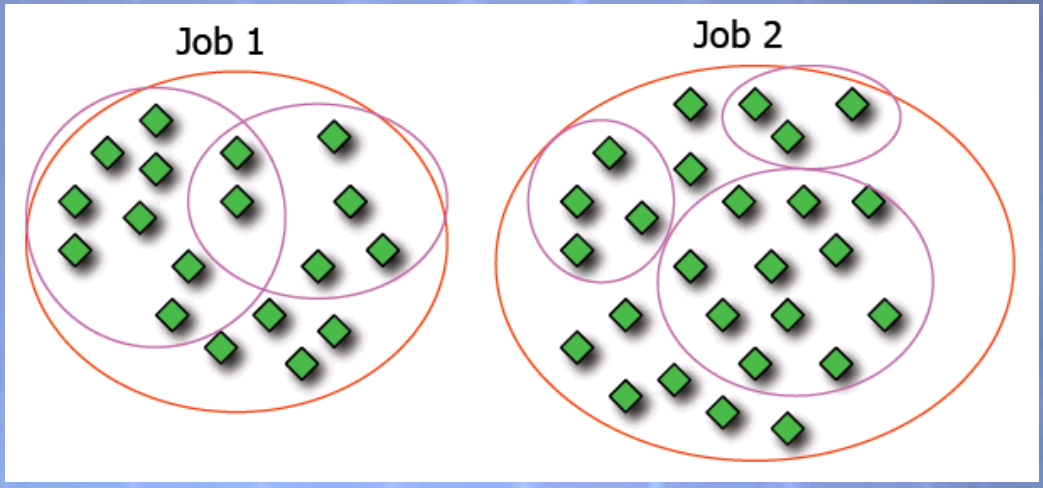
Process Sets (1)

- 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)

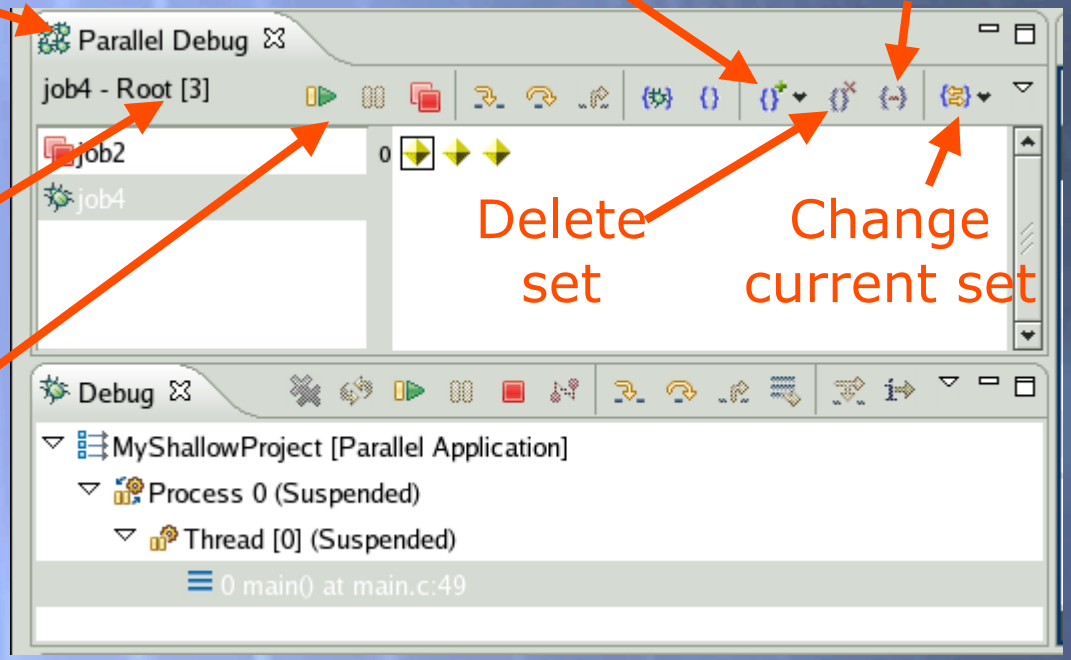
- When a parallel debug session is first started, all processes are placed in a set, called the **Root** set
- Sets are always associated with a single job
- A job can have any number of process sets
- A set can contain from 1 to the number of processes in a job



parallel tools platform
http://eclipse.org/ptp

Operations on Process Sets

- Use the icons in the toolbar of the **Parallel Debug** view to create, modify, and delete process sets, and to change the current process set
- Current process set is listed next to job name along with number of processes in the set
- Debug operations on the **Parallel Debug** View toolbar always apply to the current set:
  - Resume, suspend, stop, step into, step over, step return

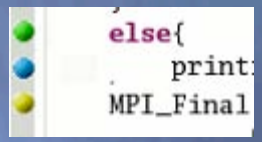


parallel tools platform
http://eclipse.org/ptp

Breakpoints

- Two types of parallel breakpoints
- Global breakpoints
  - Apply to all processes, all jobs

- Set Breakpoints
  - Apply only to processes in a particular set (which can include the root set) for a single job
  - When the job completes, the breakpoints are no longer available
- Set breakpoints are colored depending on which processes the breakpoint applies to:
  - Green indicates the breakpoint set is the same as the current set.
  - Blue indicates some processes in the breakpoint set are also in the current set (i.e. the process sets overlap)
  - Yellow indicates the breakpoint set is different from the current set



parallel tools platform
http://eclipse.org/ptp

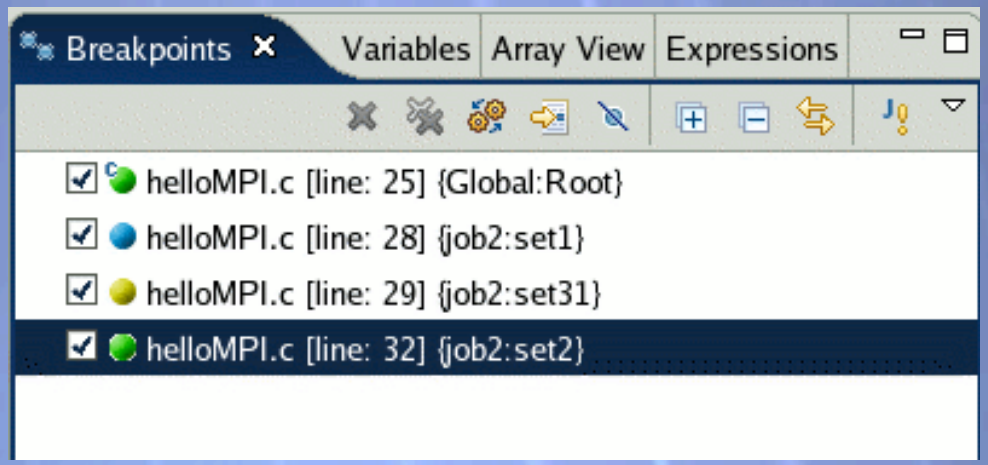
Setting Breakpoints

- To create a set breakpoint
  - Make sure the current job is selected
  - Select the root process set, or any other 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 context menu

- 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

Breakpoint Information (1)

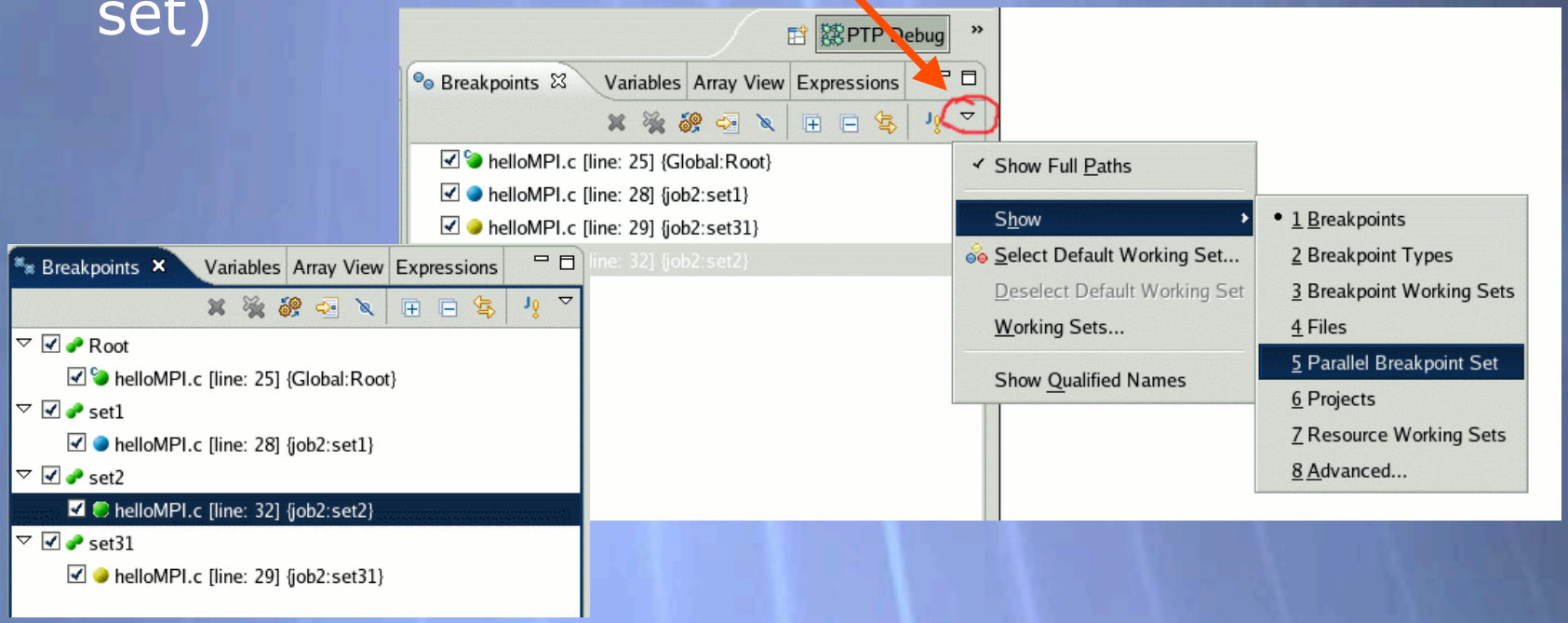
- Hover over breakpoint icon
  - Will show the sets this breakpoint applies to
- Select **Breakpoints** view
  - Will show all breakpoints in all projects



parallel tools platform
http://eclipse.org/ptp

Breakpoint Information (2)

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

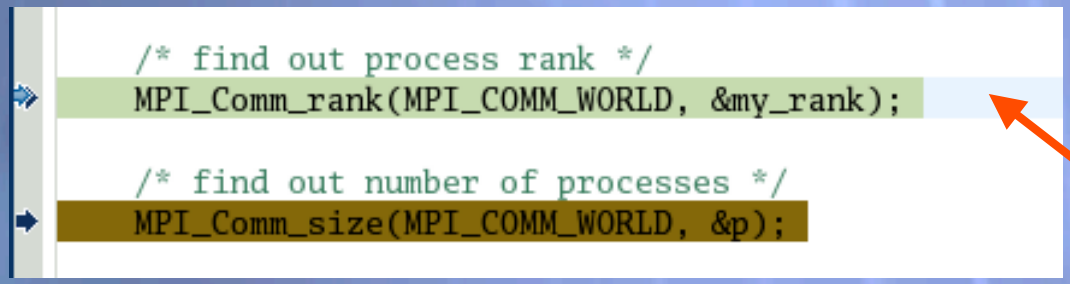


Current Instruction Pointer (1)

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

parallel tools platform
http://eclipse.org/ptp

Current Instruction Pointer (2)



- The highlight color depends on the stack frame:
  - **Green:** Registered Process
  - **Brown:** Unregistered Process
  - **Blue:** Tracks current stack frame

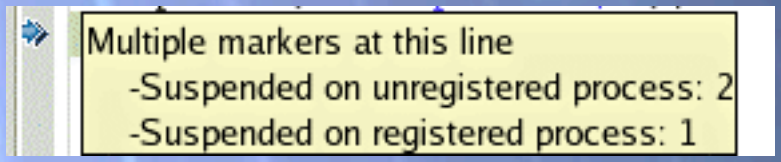
- The marker depends on the type of process stopped at that location


Multiple processes marker


Registered process marker


Un-registered process marker

- Hover for more details about the processes suspend at that location



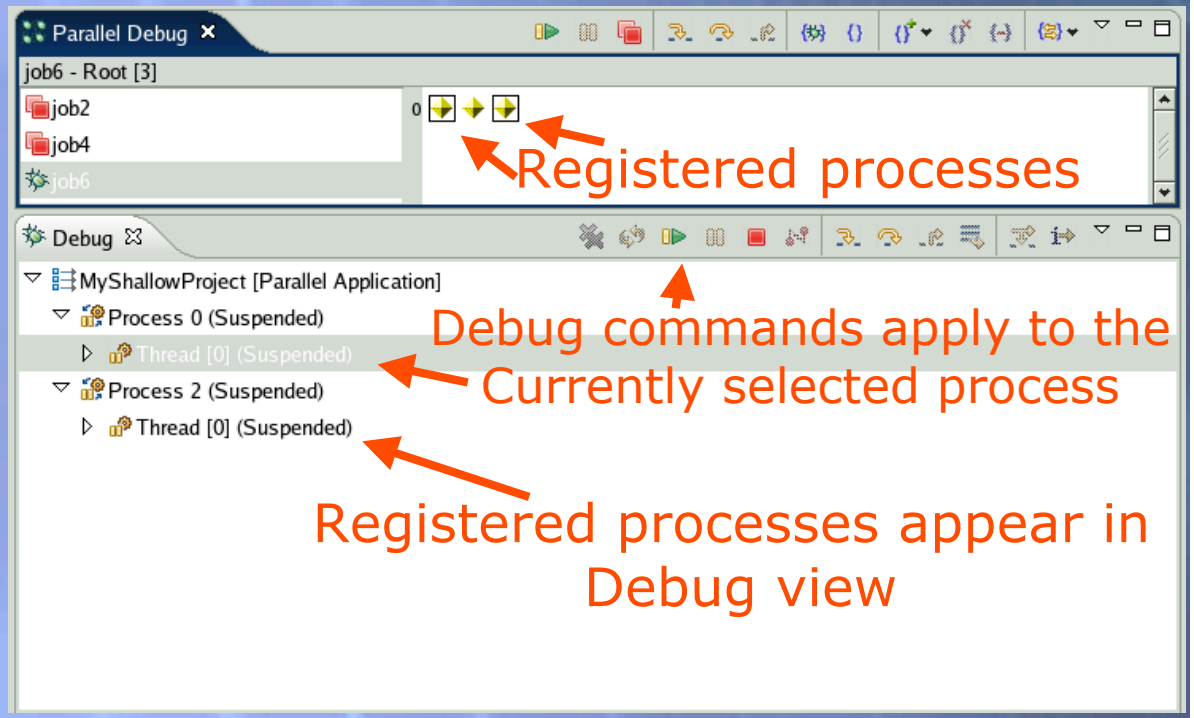
Process Registration (1)

- 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, appear in the **Debug** view
- Any number of processes can be registered, and processes can be registered or un-registered at any time

parallel tools platform
http://eclipse.org/ptp

Process Registration (2)

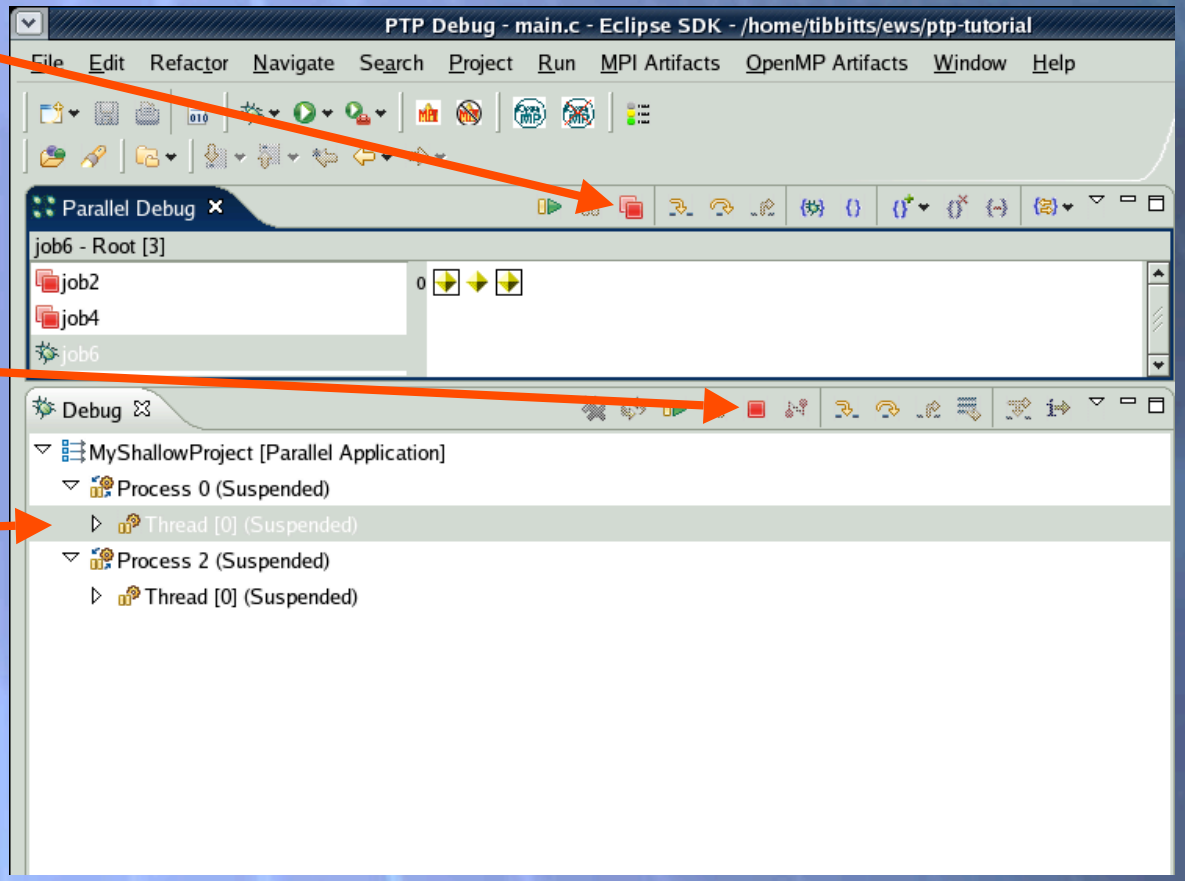
- To register a process, double-click its process icon in the **Parallel Debug** view
- Note that the process icon is surrounded by a box
  - The process appears in the debug view.
- To un-register a process, double-click on the same process icon
- Debug commands in the **Debug** view control the single process that is currently selected in that view



parallel tools platform
http://eclipse.org/ptp

Terminating a Debug Session

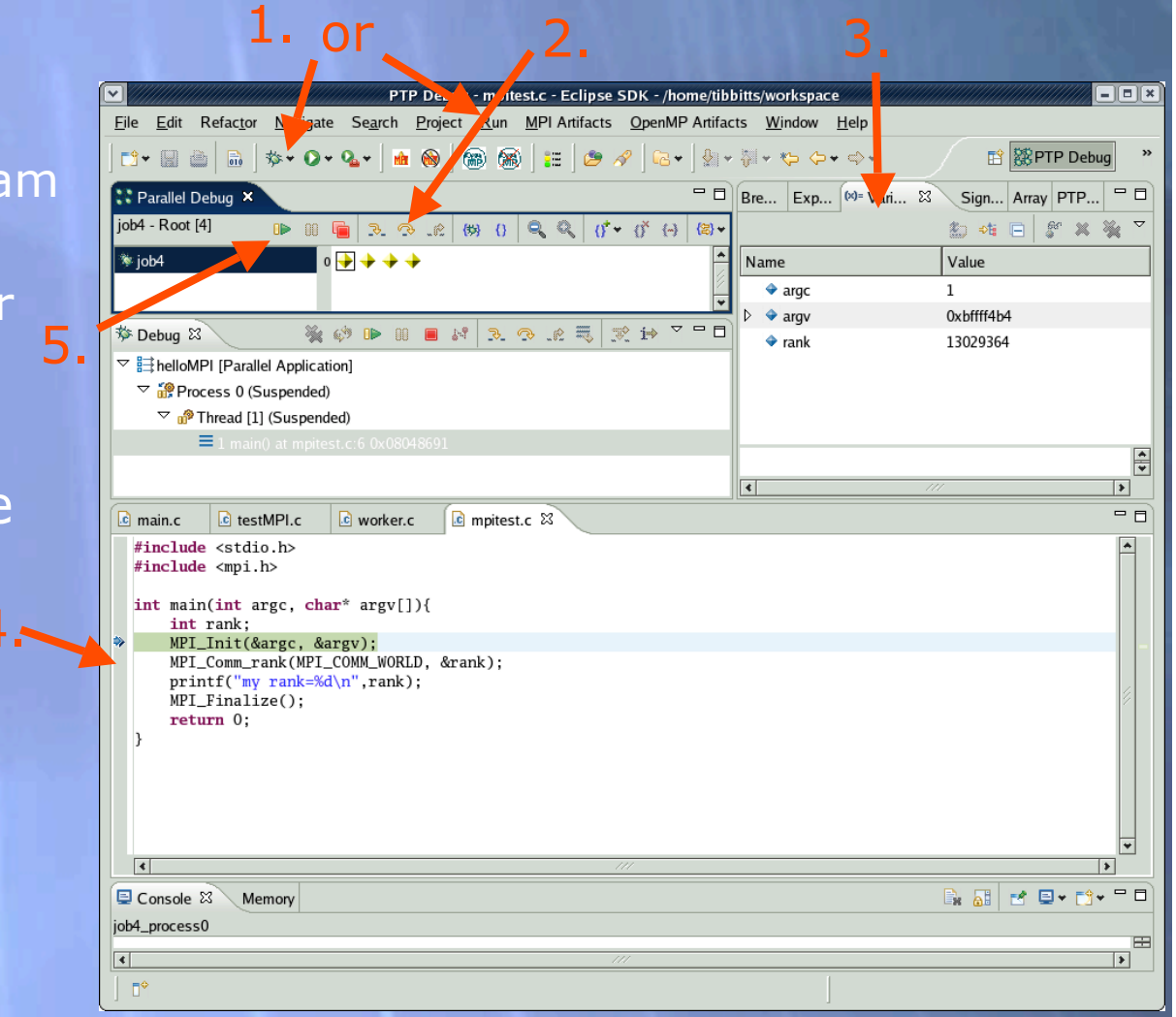
- Click on the terminate icon in the **Parallel Debug** view to terminate all processes
- Click on the terminate icon in the **Debug** view to terminate the currently selected process



parallel tools platform
http://eclipse.org/ptp

Basic Debug Commands

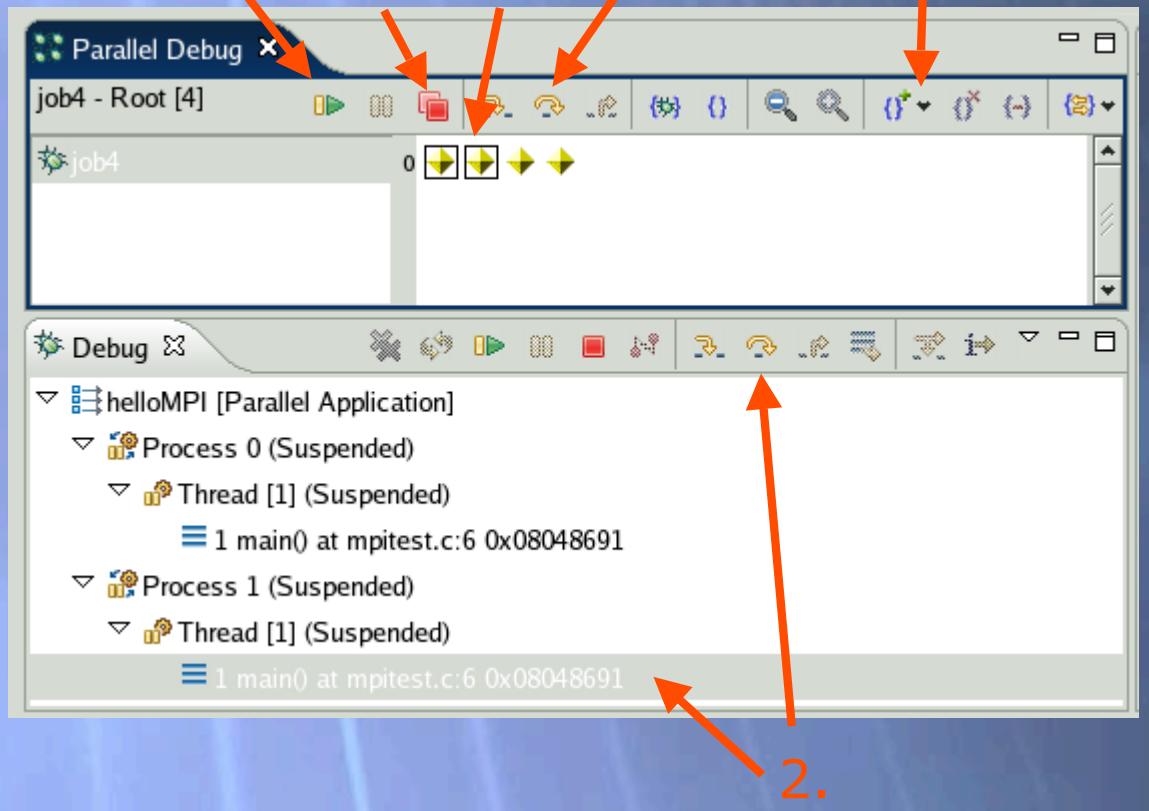
1. Launch debug session
   - Using helloMPI program
   - Use same launch configuration used for running
2. Step Over
3. Watch variable change
4. Set a breakpoint
5. Run to breakpoint



parallel tools platform
http://eclipse.org/ptp

Debug Actions on Processes and Process Sets

1. Register a (different) process
2. Step the registered process
3. Create a process set (select process icons first)
4. (a) step and (b) run the set
5. Terminate debug session



parallel tools platform
http://eclipse.org/ptp

Debugging a More Complex Application

- Shallow water weather model
- 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 modeling" by Washington and Parkinson
- Already in your workspace
  - Also on tutorial CD in **samples** folder

Code Structure

- Distributed memory model
- Master/worker
  - Master is task ID 0
  - Master initializes data and sends to workers
  - Workers perform computation on their data
  - Workers may exchange data during computation
  - Workers send results back to master
- Double-click on main.c
  - `main()` is in `main.c`
- Master code follows test for `tid != 0` (line 87)

Code Structure (cont…)

- Worker code starts at `worker()` in `worker.c`
- Worker receives data from master (line 107)
- Main time step loop starts on line 137
- Computation is performed by call to `calcuvzh()` (line 145)
  - `calcuvzh()` is located in calc.c
- Time tendencies are calculated by call to `timetend()` (line 154)
  - `timetend()` is located in `tend.c`
- Results are returned at line 180

Building the Application

- Switch to **C/C++ perspective**
- Select the **Make Targets** view
- Open the **shallow** project
- Double-click on the **all** target
- Bring **Console** view to front to see build progress

Running the Application

- Select **Run ▸ Run...** to open the run configuration dialog
- Select **Parallel Application**
- Select the **New** button
- Select the **shallow** project if not already visible
- Select **Browse** and choose the **shallow** executable for **Application program**
- Select the **Parallel** tab and enter 3 for **Number of processes**
  - must be an odd number
- Select the **Debugger** tab and choose **SDM** from the **Debugger** drop down menu
- Select **Apply**, then **Run**
- Eclipse will automatically switch to the **PTP Runtime** perspective

Examining Output

- Three process icons should be visible in the **Jobs** view
- Double-click on process 0 and the standard output should be visible in the **Process** view
- Examine the value for `Potential energy` in `Cycle 950`
  - The correct value is `128225.086`
  - There must be a bug!

Debugging the Application

- Select **Run ▸ Debug History ▸ shallow**
  - Or just click on the Debug icon on the toolbar
- The application should launch and switch to the **PTP Debug** perspective

Debugging Hints (1)

- There is an error somewhere in the worker calculation
- Place a breakpoint on the `worker()` call in `main.c`
- Continue execution by selecting the resume button (green arrow on **Parallel Debug** view)
  - You'll notice that process 0 remains running (green) while processes 1 and 2 are suspended (yellow)
- Step into the `worker()` call
  - `worker.c` should open and show the current line marker on the first statement
- Place a breakpoint in the main timestep loop
  - Line 144 is a good place
- Continue execution again
  - The processes should now suspend at the breakpoint

Debugging Hints (2)

- Double click on the process 1 and process 2 icons
  - This will register the processes so you can view variables
- Compare variable values that should be the same across processes
  - Make sure **Variable** view is visible
  - Select stack frame of each process in turn
  - The `ncycle`, `time` and `tdt` variables are likely candidates since they should be the same for each process
- Continue execution and check values again
- When you locate a difference
  - Inspect code where the variable is updated or modified
  - This is likely to be the source of the error
- When you find the error
  - Exit the debugger, recompile and re-run the code

parallel tools platform
http://eclipse.org/ptp

Module 7: Other Tools

- Objective
  - Learn about other tools related to PTP
  - PTP upcoming features
  - Where to go for further information
  - Tutorial feedback
- Contents
  - Links to other tools, including performance tools
  - Planned features for new versions of PTP
  - Additional documentation

parallel tools platform
http://eclipse.org/ptp

PTP-Related Tools

- TAU – Tuning and Analysis Utilities
  - http://www.cs.uoregon.edu/research/tau
  - Eclipse plug-in integrates instrumentation for PTP
- Performance Explorer
  - Performance visualization Eclipse plug-ins from IBM Research

Useful Eclipse Tools

- CDT – C/C++ Development Tools
  - http://eclipse.org/cdt
- TPTP – Testing and Performance Tools Platform
  - http://eclipse.org/tptp
- Python
  - http://pydev.sourceforge.net
- Subversion (CVS replacement)
  - http://subclipse.tigris.org

parallel tools platform
http://eclipse.org/ptp

PTP Upcoming Features (1)

- PTP 1.1 (4Q 2006)
  - Resource management
    - Support for viewing of jobs in queues
    - One resource manager supported, probably SLURM
  - Additional runtime support
    - MPICH2
  - PLDT enhancements
    - OpenMP support

parallel tools platform
http://eclipse.org/ptp

PTP Upcoming Features (2)

- PTP 2.0 – Summer 2007
  - Remote services support
    - Allow projects to reside on remote systems
    - Ability to build projects remotely
    - Ability to launch and debug projects remotely
  - Resource management
    - View and query the status of queues
    - Submit and control jobs
    - Second implementation of a resource manager
      - Possibly LoadLeveler or LSF

parallel tools platform
http://eclipse.org/ptp

PTP Upcoming Features (3)

- PTP Performance Analysis Framework
  - Goal: Integrate Instrumentation, Measurement, and Analysis for a variety of tools
  - http://wiki.eclipse.org/index.php/PTP/designs/perf

parallel tools platform
http://eclipse.org/ptp

Recent PTP Publications

- "Developing Scientific Applications Using Eclipse," Computing in Science & Engineering, vol. 8, no. 4, July/August 2006, pp. 50-61
  - Link on http://eclipse.org/ptp web page
- "A Model-Based Framework for the Integration of Parallel Tools", Proceedings of the IEEE International Conference on Cluster Computing, Barcelona, September 2006
  - Link on http://eclipse.org/ptp web page
- IBM developerWorks article:
  - http://www-128.ibm.com/developerworks/edu/os-dw-os-ecl-ptp.html

parallel tools platform
http://eclipse.org/ptp

PTP Tutorial Feedback

- Please complete feedback form
- Your feedback is valuable!

Thanks for attending
We hope you found it useful