The Eclipse Parallel Tools Platform

and Scientific Application Development

Craig Rasmussen, LANL crasmussen@lanl.gov
Beth Tibbitts, IBM tibbitts@us.ibm.com
Greg Watson, IBM g.watson@computer.org

OSCON July 2007

"This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under its Agreement No. HR0011-07-9-0002"

Tutorial Outline / Schedule

Time	Module	Topics	Presenter
1:30-1:45	1. Overview of Eclipse and PTP	An understanding of the overall Eclipse and PTP architecture	Beth/Craig
1:45-2:15	2. Introduction to the Eclipse IDE	Basic features of the Eclipse IDE, incl. building, running and debugging a sample application	Craig Rasmussen
2:15-3:00	3. Advanced Development	Version Control, Bookmarks, Task Tags, Refactoring, Search	Craig Rasmussen
3:00-3:30	Break	Optional: Install eclipse on student laptops	
3:30-4:00	4. PTP and Parallel Language Development Tools	Introduction to PTP and MPI & OpenMP tools	Beth Tibbitts
4:00-4:30	5. Parallel Debugging	Eclipse parallel debugger	Beth Tibbitts
4:30 - 5:00	6. Eclipse and the Enterprise; related info	Further information about Eclipse, PTP and related tools	Beth & Craig

A word on versions…

- Note: PTP core currently supports CDT version 3.1.x, which requires Eclipse 3.2.x. (2006)
- Eclipse 3.3 and CDT 4.0 were released in June 2007, and CDT 4.0 provides many enhancements over CDT 3.1
- The slides in this tutorial
  - describe CDT 4.0 for PLDT and CDT-only features
    - So that you see the latest features!
  - PTP core (runtime, debugger) won't support CDT 4.0 until late '07 or early '08. Its features described here won't change significantly, however.

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

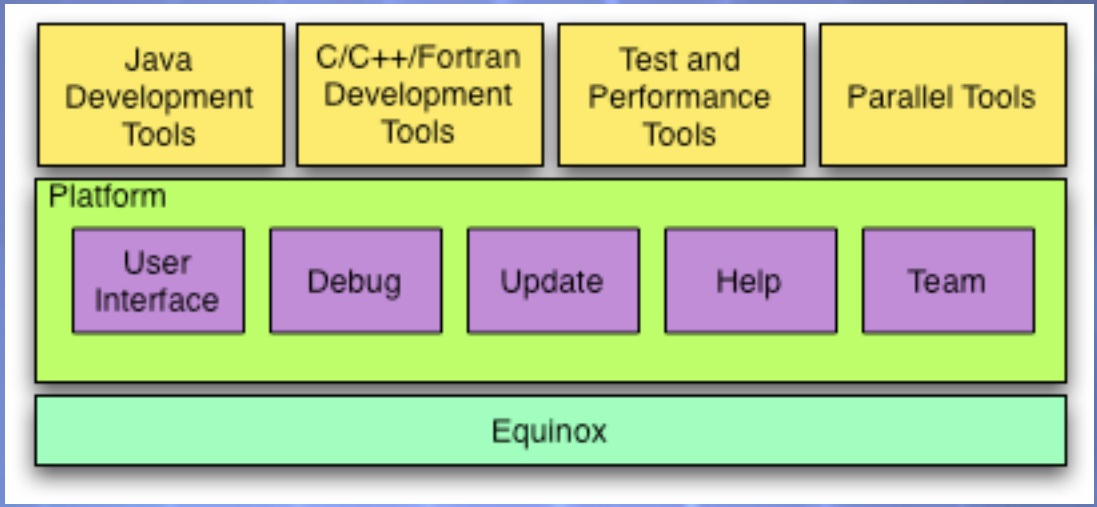
Members of Eclipse

June 2007

- 162 members in June ’07 (130 in March 2006)
  - 21 strategic members (16 in June 2006)
- 794 committers, representing 48 organizations

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 (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)
- 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

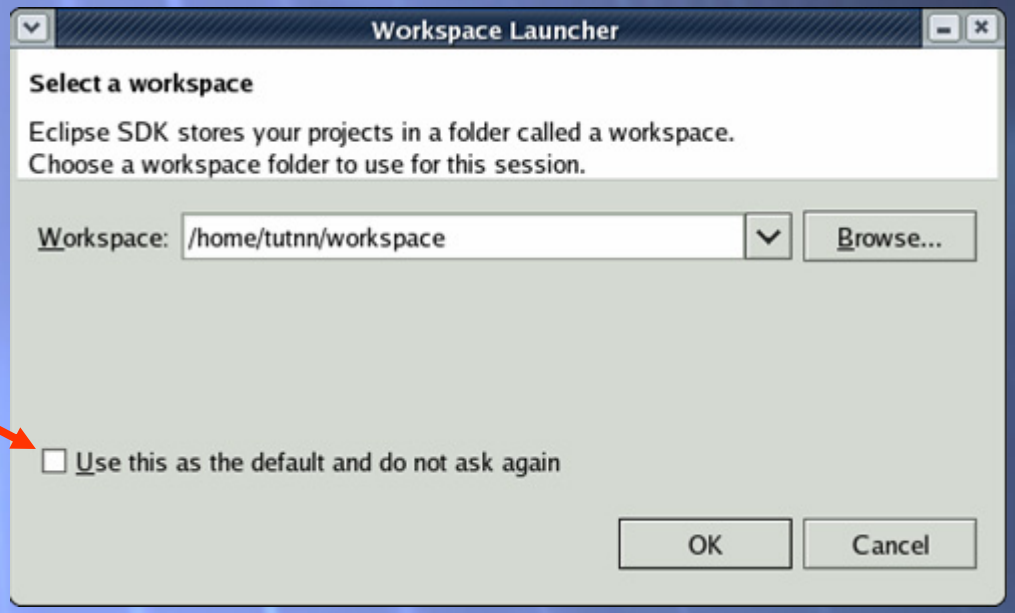
Module 2: Introduction to the Eclipse IDE

- 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

Specifying A Workspace

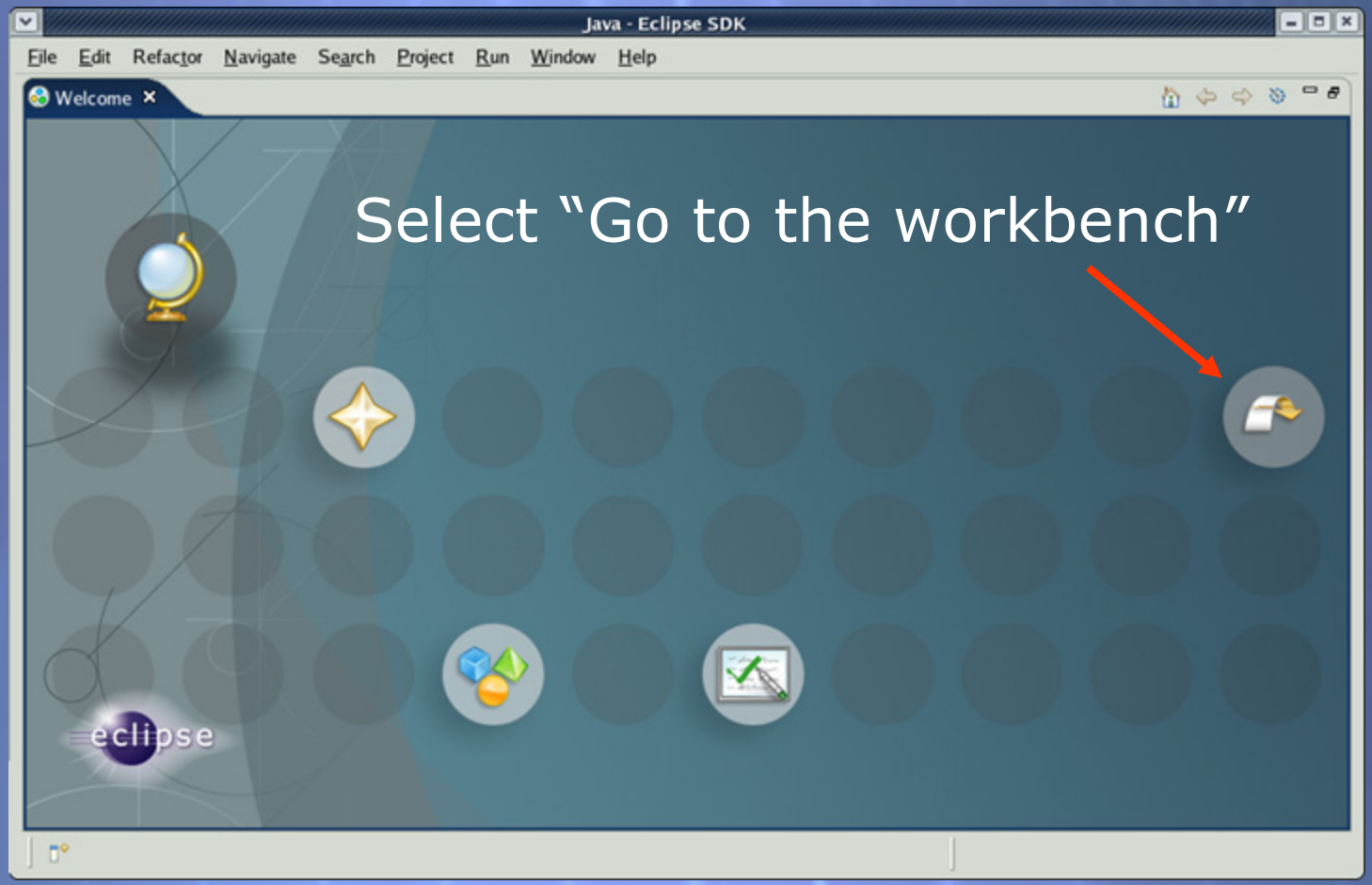
- Eclipse prompts for a workspace location at startup time
- The workspace contains all user-defined data
  - Projects and resources such as folders and files

The prompt can be turned off



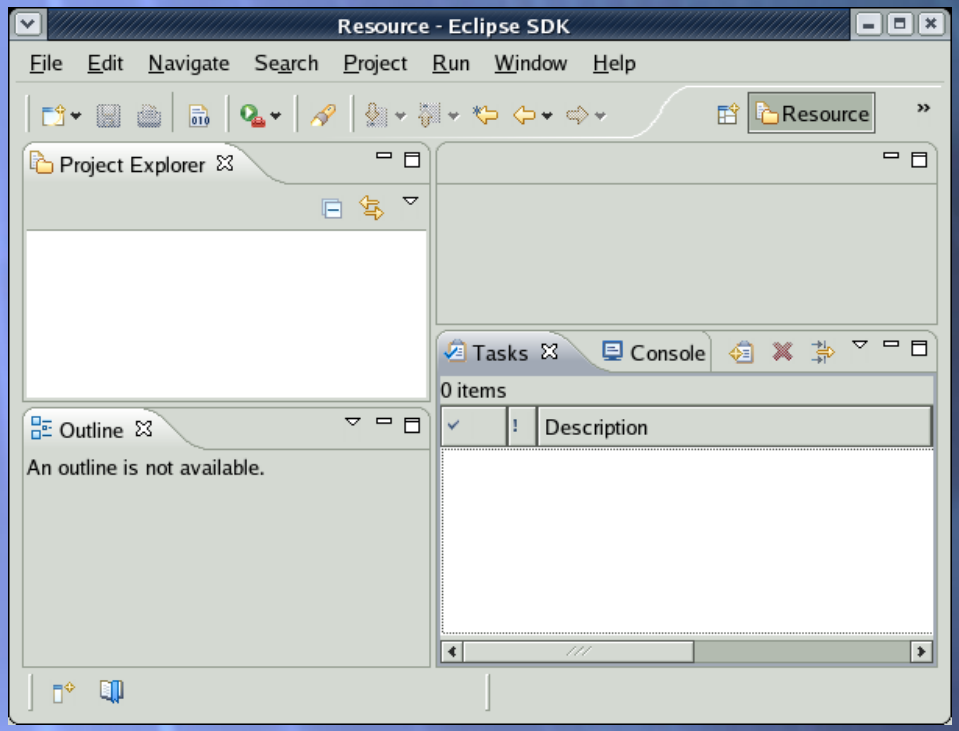
Eclipse Welcome Page

- 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



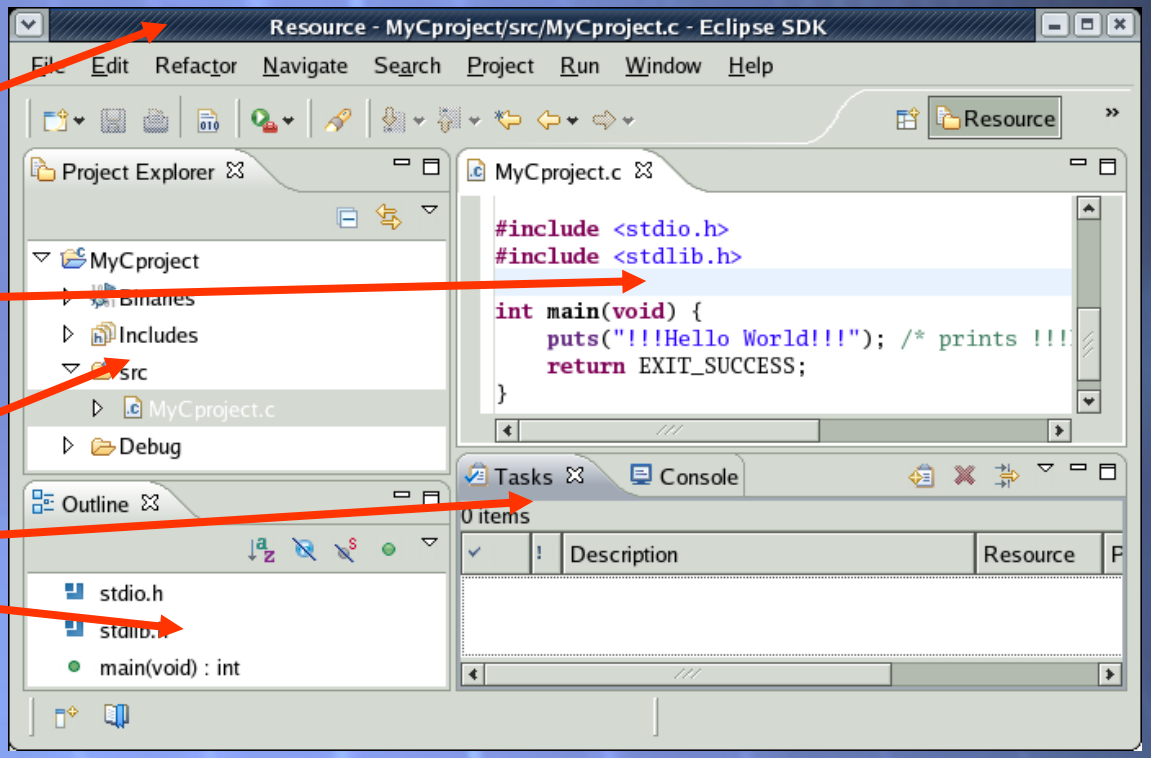
Workbench Components

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

perspective

editor

views

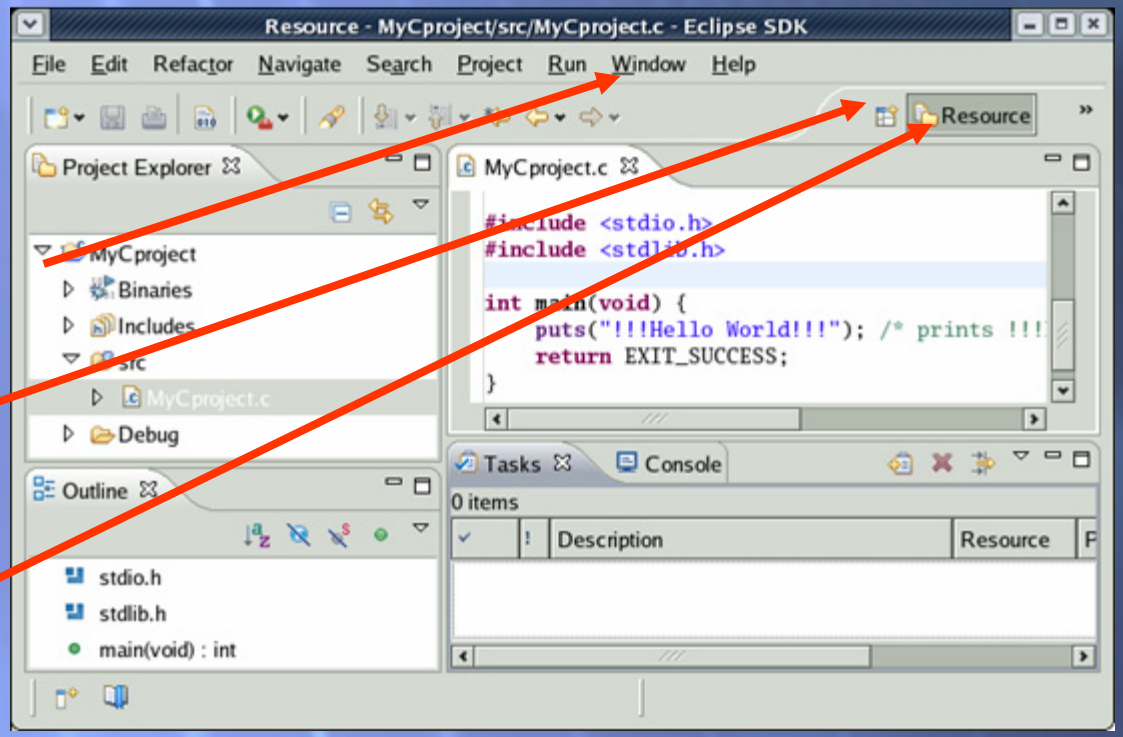


Perspectives

- 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

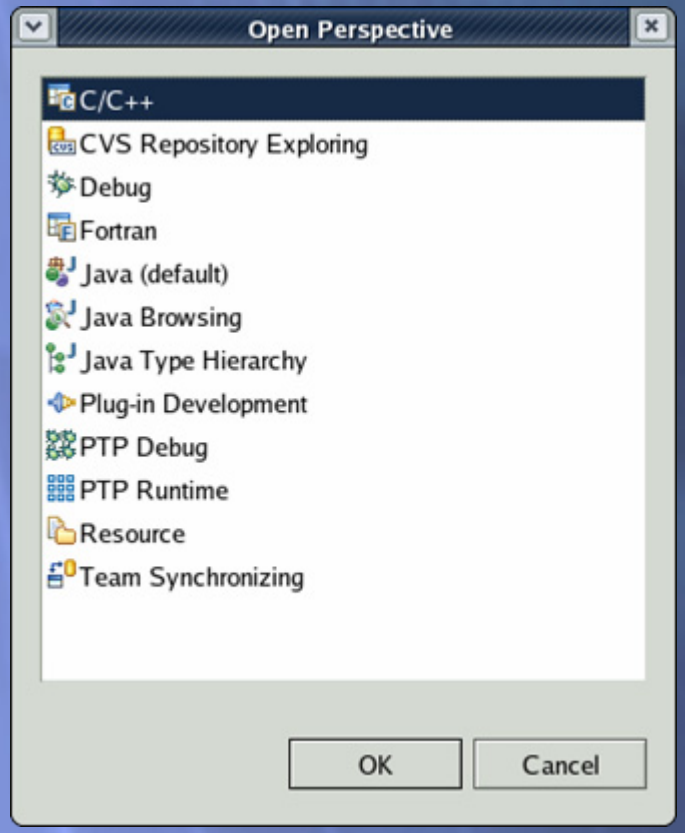
Switching Perspectives

- 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

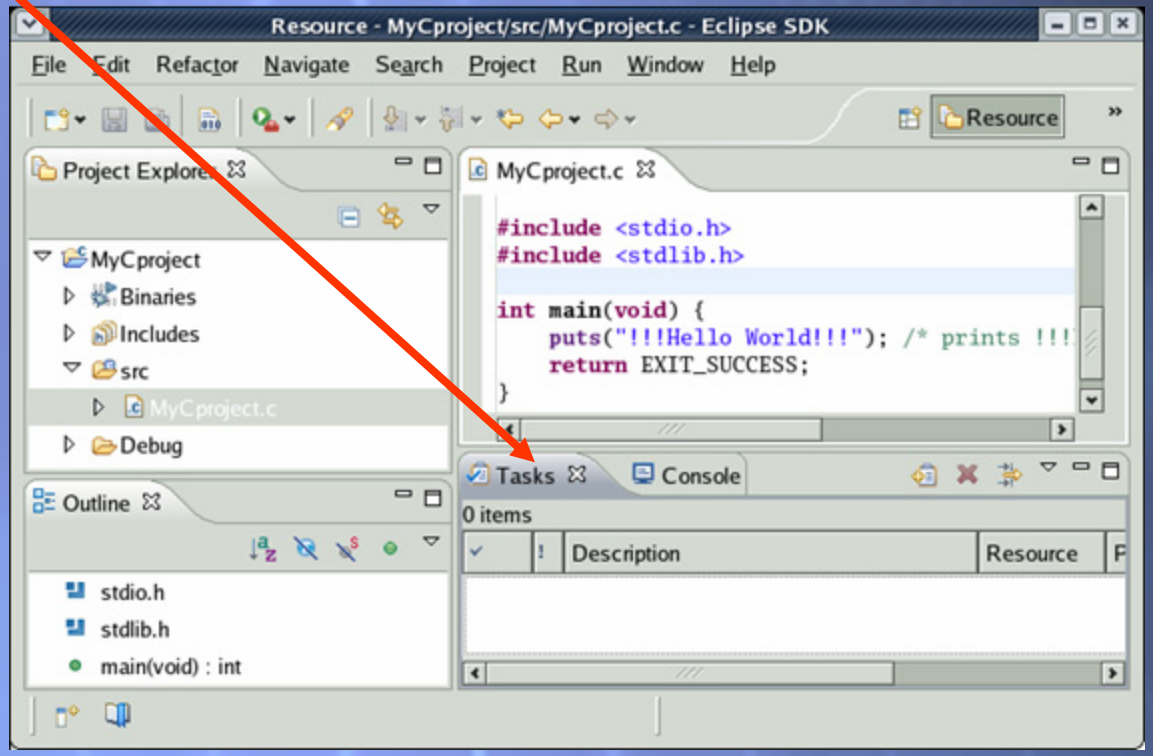
- Items such as shortcuts, menu items and views may be customized
  - **Window ▸ Customize Perspective...**
- Rearrange views by dragging
  - Try moving the outline view
- Save changes
  - **Window ▸ Save Perspective As...**
- Close Perspective
  - Right-click on perspective title and select **Close**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

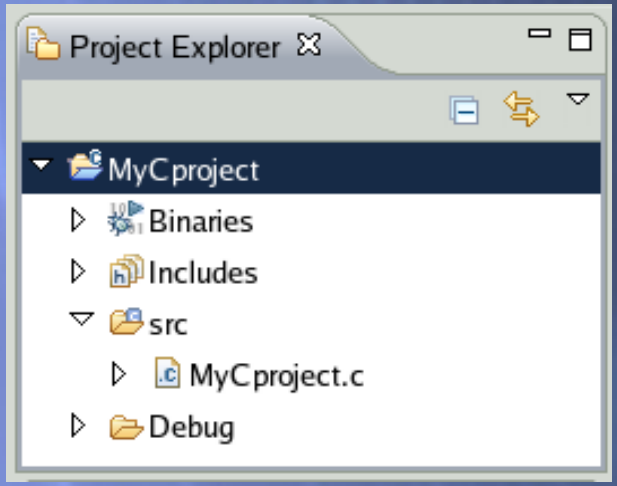
Stacked Views

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



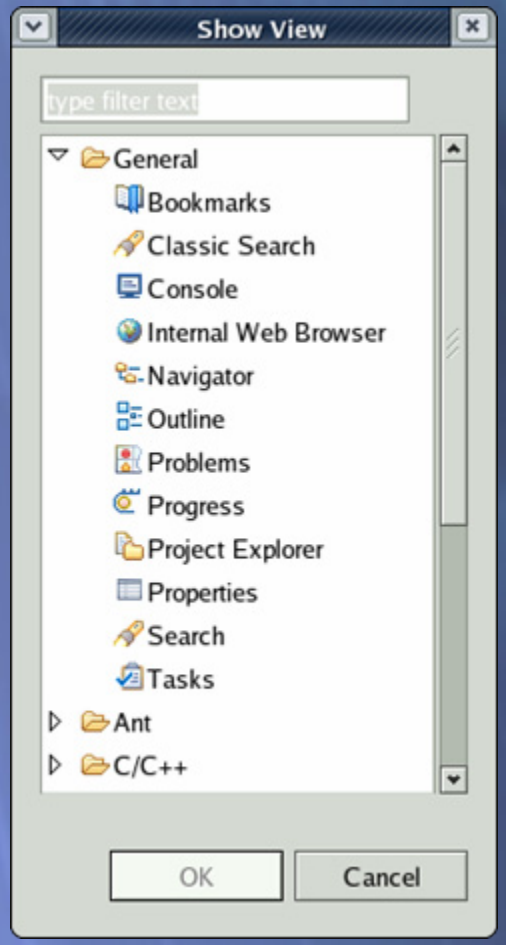
Project Explorer View

- 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



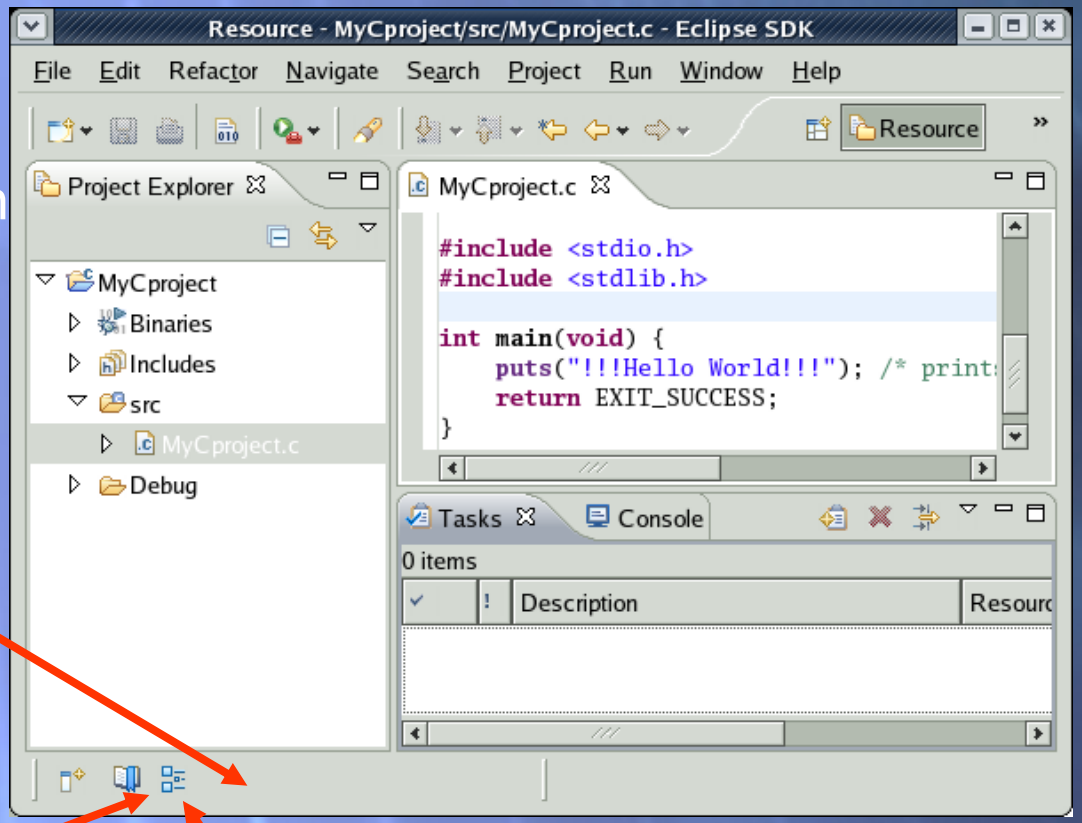
Opening a New View

- 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)

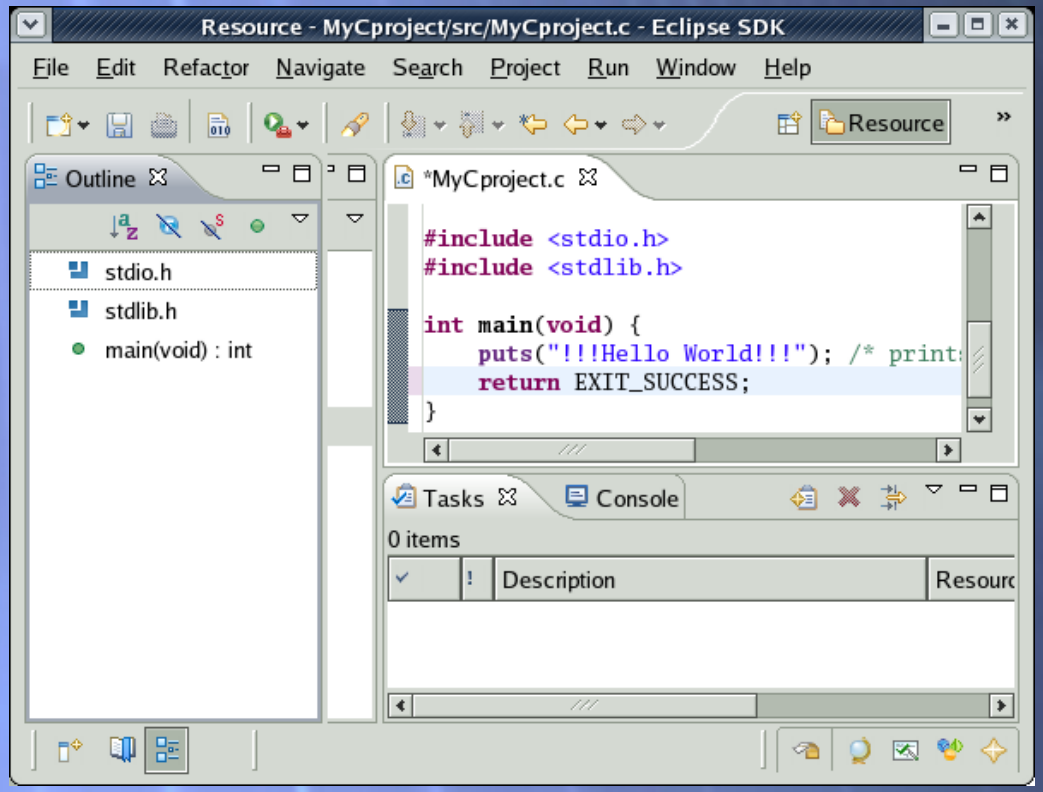
- Hidden views that can be quickly opened and closed
  - They take up space in the Workbench
- Fast views can be created by:
  - Dragging an open view to the shortcut bar
  - Selecting **Fast View** from the view's menu
- A Fast View is activated by clicking on its **Fast View** button



Outline view has been hidden in the shortcut bar

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

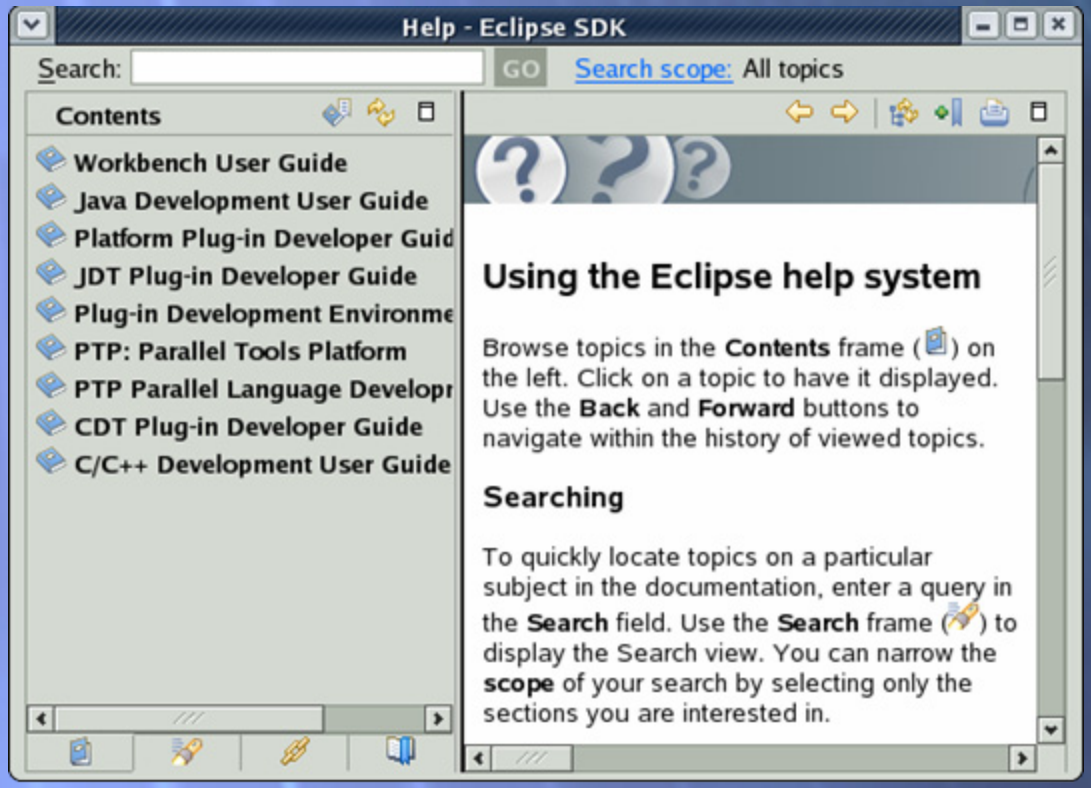
- An editor for a resource opens when you double-click on a resource
  - Editor type depends on the type of the resource, for example .c files are opened with the C/C++ editor
  - When an editor opens on a resource, it stays open across different perspectives
  - An active editor contains menus and toolbars specific to that editor
  - When you change a resource, an asterisk on the editor's title bar indicates unsaved changes

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…



A Simple Application

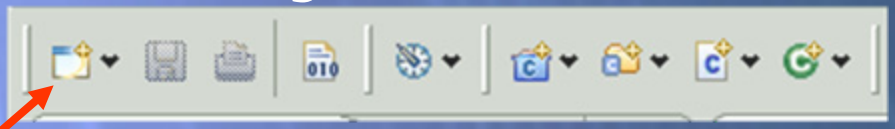
- Create a C Project
- Add files
  - Source files (ending in .c)
  - A makefile is automatically created
- Build application
  - Done automatically
- Debug application
  - Create a Debug Configuration

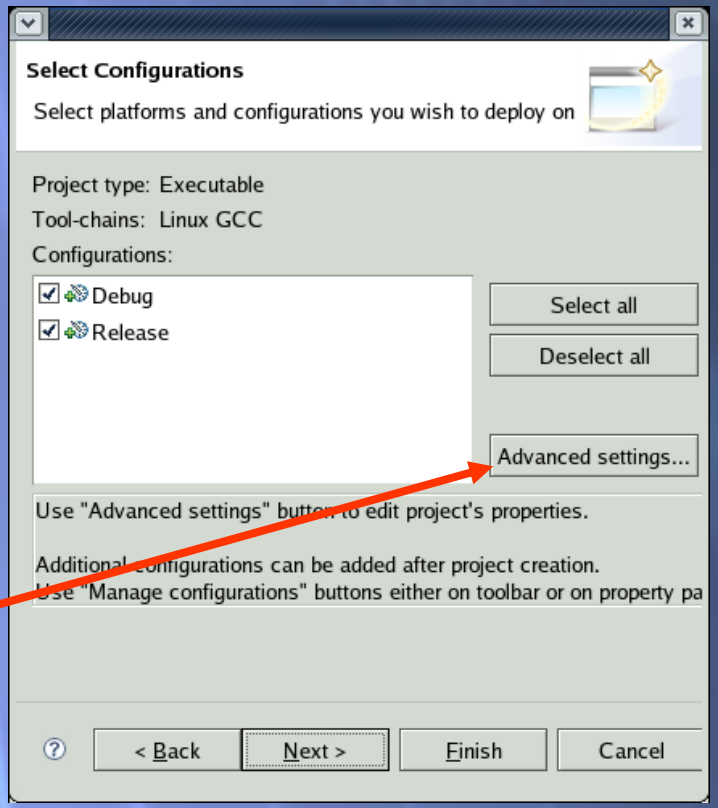
CDT Projects

- A Project contains the resources of an application
- 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 C Project

- Create a project (in C/C++ Perspective)
  - **File ▸ New ▸ C Project**
  - Or select **New Project** button
- Give it a name: e.g. Integrator
- Select Project Type (or default is OK)
  - **Next>**
- On **Select Configurations** page, click "Advanced Settings" to see Project properties
- Select **Finish**





Creating a C Project CDT 3.1

- Terminology is a bit different from CDT 4.0…
- CDT (C/C++ Development Tools) 3.1 project types:
  - Managed Make
  - Standard Make
- To 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 **samples** folder from PTP tutorial CD; 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)
  - If project doesn't build automatically, select the build icon on the toolbar. 
- 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
    Under that, select the **Settings** 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 ▸ Open Debug Dialog...** to specify details of running the application
  - Or for quick launch: click arrow next to debug button, then **Debug As**
    - Select **Local C/C++ Local Application**
  - Switch to Debug Perspective if prompted

Debugging (1)

- 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

Other things to try, if there's time:

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

Module 3: Advanced Development

- Objective
  - Create and build a Standard Make Project from source files in CVS
- Contents
  - Version control
  - Standard Make Projects
  - C/C++/Fortran
  - Bookmarks, Task Tags
  - Refactoring
  - Searching

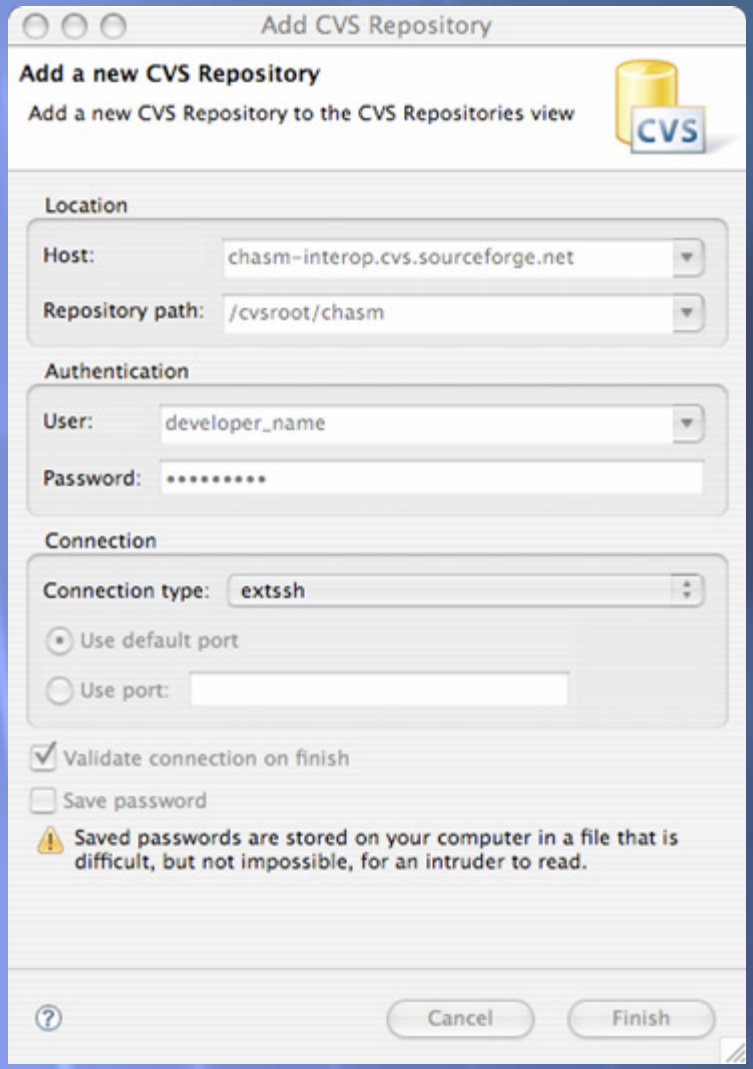
Version Control (CVS)

- Version control provided through the **Project Explorer View**, in the **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 **CVS Repositories View**, then select **New ▸ Repository Location...**
- Fill out options
  - Host, repository path
  - Username and password
  - Connection type
- Select **Finish**

Correction: Repository path: /cvsroot/chasm-interop



Checkout Chasm Code

- Open the repository, then open HEAD
  - Right-click on **chasm ▸ Check out As…**
  - Select **Finish**
  - Select **C ▸ C project**
  - Select **Next>**
- Enter **Project name** and **location**
  - Workspaces tend to be temporary so do not use default location
- Select **Finish**
- Switch to the **C/C++ Perspective**
- In Chasm Project **Properties**
  - Deselect Generate Makefiles automatically

Standard Make Project

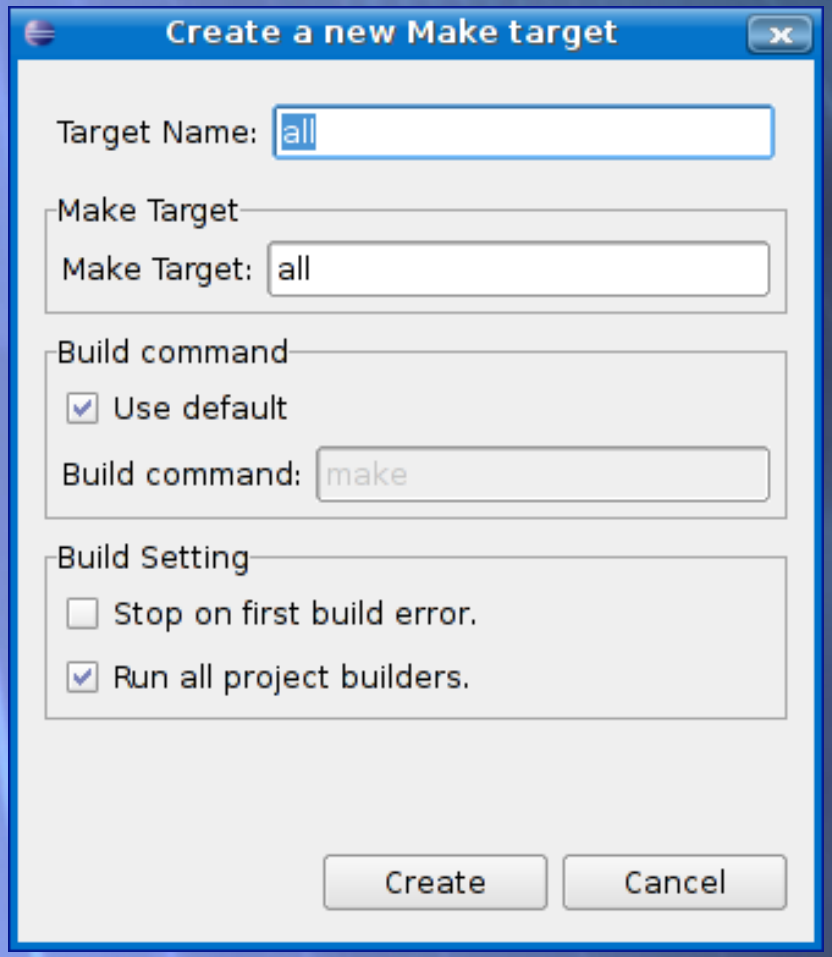
- *Standard Make* projects are different from *Managed Make* projects
  - Project Makefiles must be created
- Can create project Makefiles with the Makefile Editor
  - Syntax highlighting and Outline view
- autoconf often used to create Makefiles for open source projects
  - Must refresh after running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

Building Chasm 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**

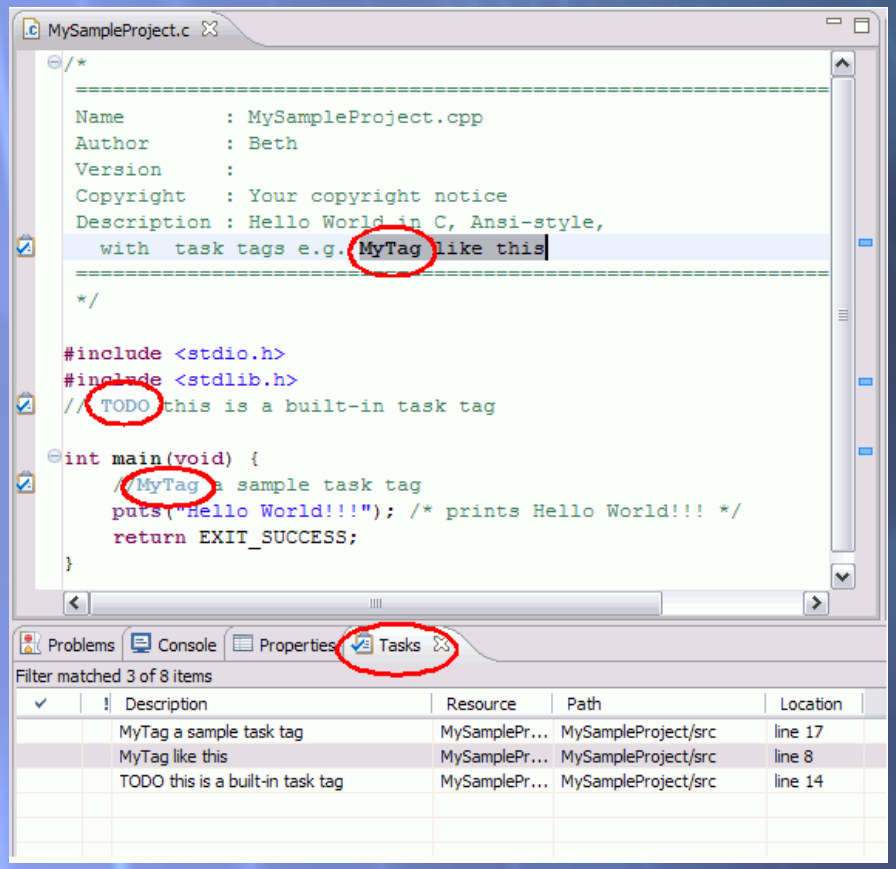


Create a Bookmark

- 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**

Create a Task Tag

- Task tags are identifiers in C/C++ comments
- TODO is a built-in task tag
- Configure your own task tag in Window > Preferences
  - Under C/C++, select Task Tags
- Add a Task tag by typing it in a source file comment
  - i=i+1 // TODO this is a task tag
- The build locates task tags during compilation
- View task tags in the Tasks View
  - If it's not shown, Window > Show View > Other… Open **General** and select **Tasks**



Commit Changes

- Select the **Projects Explorer** view
- Notice the '>' before the file name(s)
  - Indicates a file has been modified
- Right-click on the **chasm** Project
  - Select **Team ▸ Synchronize With Repository**
  - Confirm switch to perspective if asked
- Expand the **chasm** 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

- Refactoring
  - Modifying source code without changing its external behavior
- Searching
  - Based on languages elements, not just textual

Refactoring

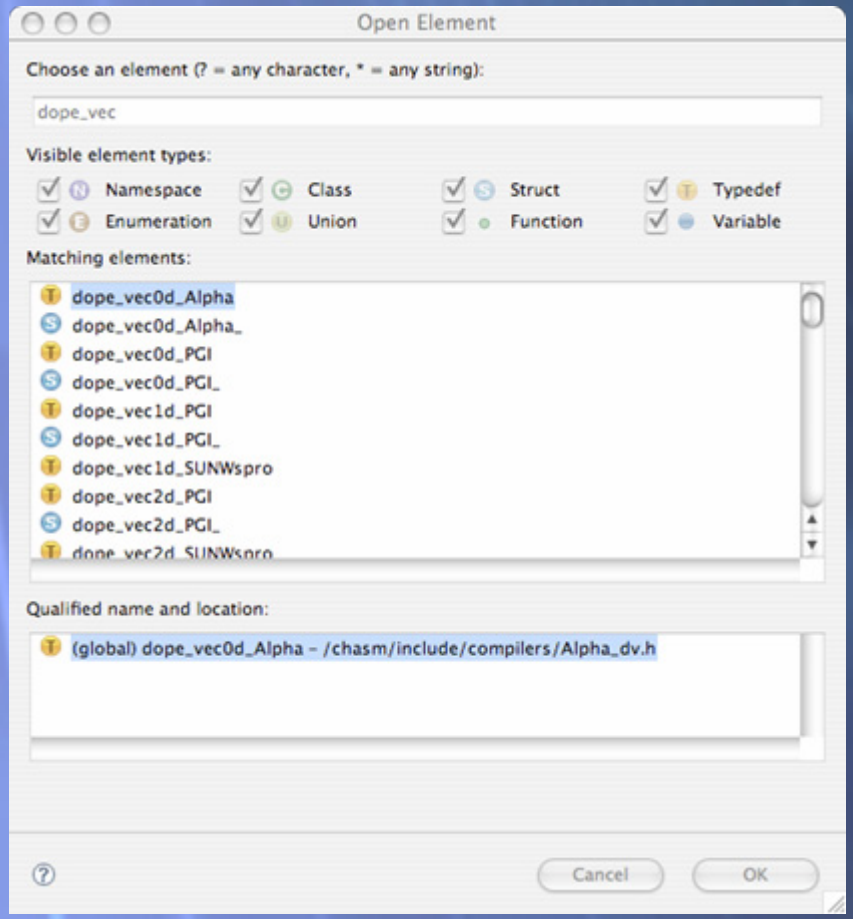
- Rename
  - Select **C/C++ Perspective**
  - Open **src/compilers/GNU.c**
  - Use **Outline View** to scroll to **setArrayDesc_GNU**
  - Click in editor view on declaration of **rank**
  - Select menu item **Refactor ▸ Rename**
  - Change **rank** to **rank_renamed**
  - Notice that change is semantic not textual
- Introduce Implicit None
- Constant promotion

Searching

- 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

Type Navigation

- Choose **C/C++ Perspective**
- Select **Navigate ▸ Open Element…**
- Enter 'dope_vec' in text box
- All matching types are displayed



Module 4: 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 (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 Language Development Tools (1)

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

Parallel Language Development Tools (2)

- More PLDT features:
  - 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.
- Included in PTP 2.0
  - MPI Barrier analysis

A word on versions…

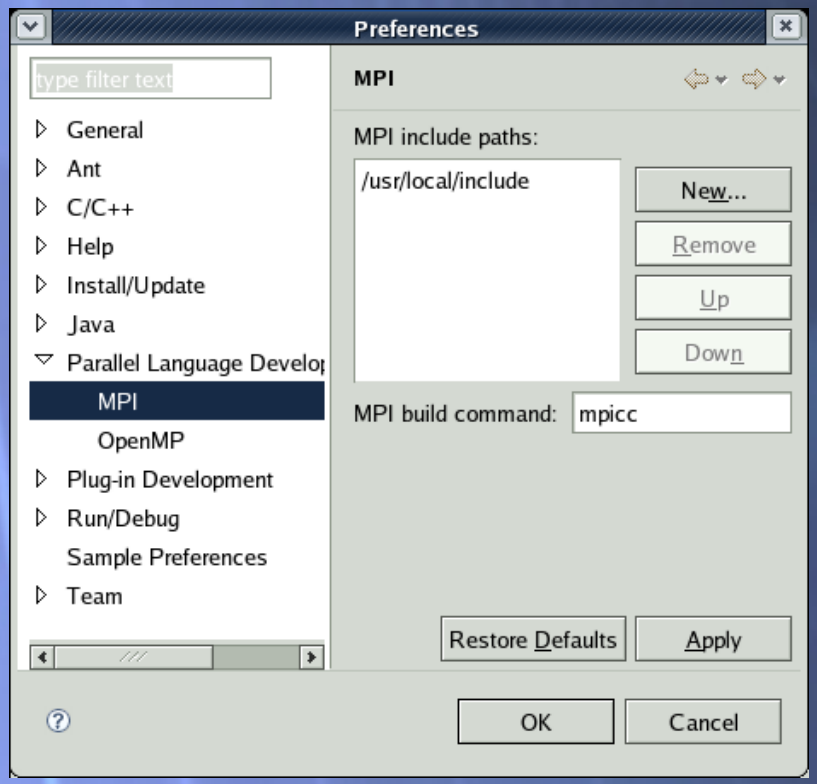
- Note: PTP core currently supports CDT version 3.1.x, which requires Eclipse 3.2.x. (2006)
- Eclipse 3.3 and CDT 4.0 were released in June 2007, and CDT 4.0 provides many enhancements over CDT 3.1
- The slides in this tutorial
  - describe CDT 4.0 for PLDT and CDT-only features
    - So that you see the latest features!
  - PTP core (runtime, debugger) won't support CDT 4.0 until late '07 or early '08. Its features described here won't change significantly, however.

Terminology

- In CDT 3.1 there were two distinct types of C/C++ projects
  - Managed Make project – CDT handles the makefile and build process
  - Standard Make project – "bring your own" makefile
- In CDT 4.0 there is a single project type
  - A "C Project" or "C++ project"
  - In the project creation you can select "Makefile project" if you want to "bring your own."
    - Otherwise we consider it a "Managed Make" project
    - We will still use the old terminology at times.

PLDT Preferences

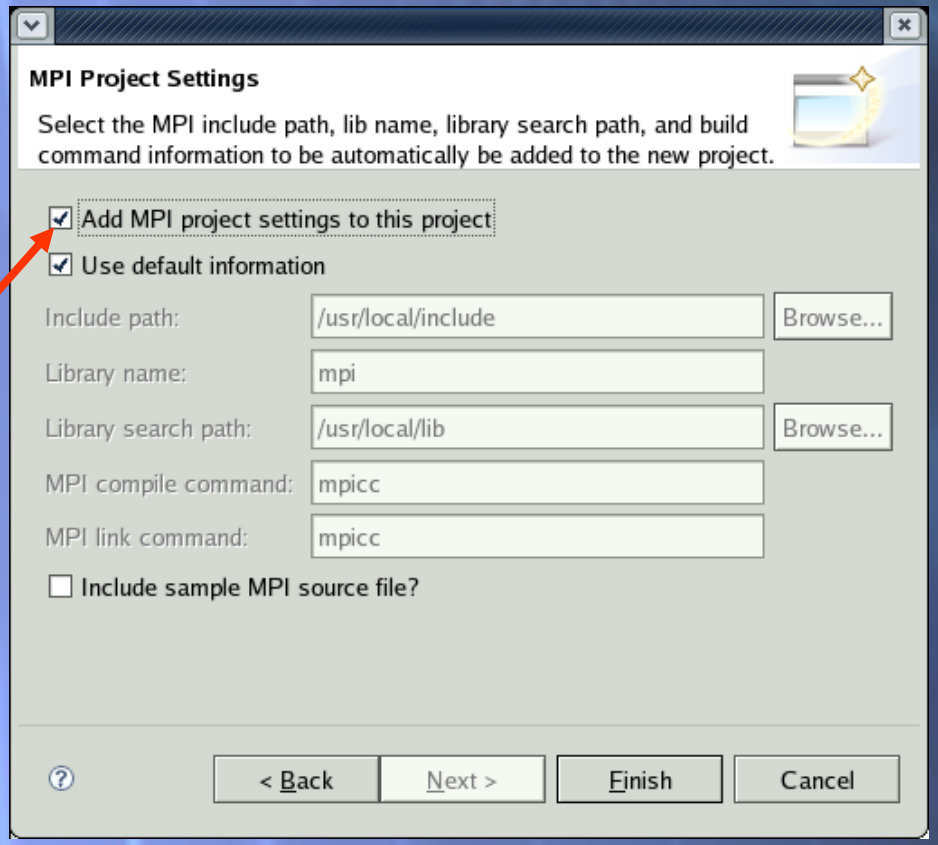
- To use the PTP Parallel Language Development Tools feature for MPI development, you need to
  - Specify the MPI include path
  - Specify the MPI build command
- Open **Window ▸ Preferences...**
  - Open the **PTP** item [if PTP is installed]
  - Open the **Parallel Language Development Tools** item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



Managed Make Project Setup

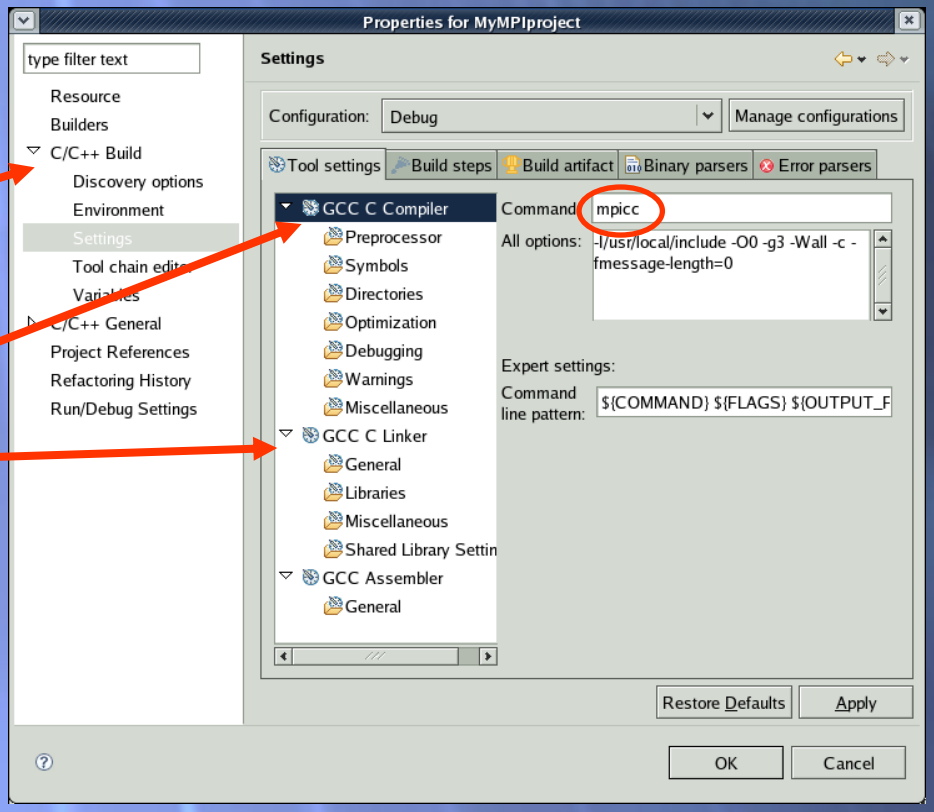
- Create a new **C Project**
  File > New > C Project
- Name the project e.g. 'helloMPI' and click 'Next'
- After the **Select Configurations** page, you should see the **MPI Project Settings** wizard page
- Check **Add MPI project settings to this project** to update the project information.
- Change default values if necessary

- Currently only works on Managed Make C projects
- (Note: we plan to add a **New MPI Project** project type in a later version)



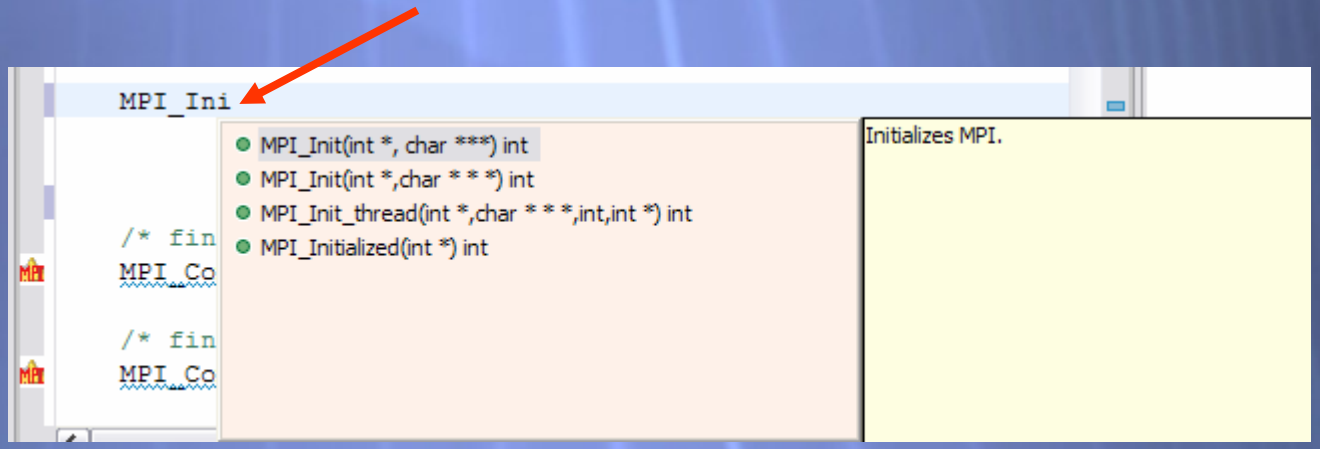
Changing the Project Properties

- *If* you wish to change the way your MPI program is built:
  - Open the project properties
  - Select **C/C++ Build**
  - Select **Settings**
  - Select **GCC C Compiler**
    - Change the command
  - Select **GCC C Linker**
    - Change the command
  - It's also possible to change compiler/linker arguments
- The MPI Project wizard set these for you, so it isn't necessary to change them.

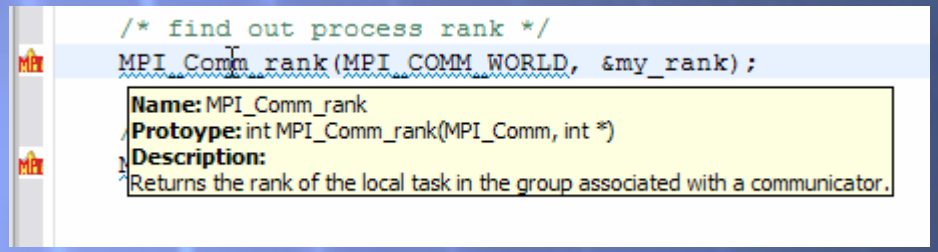


Content Assist

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

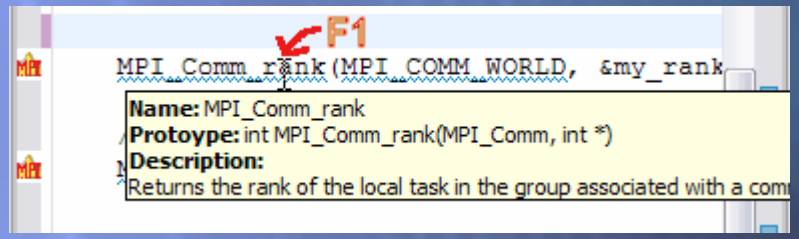


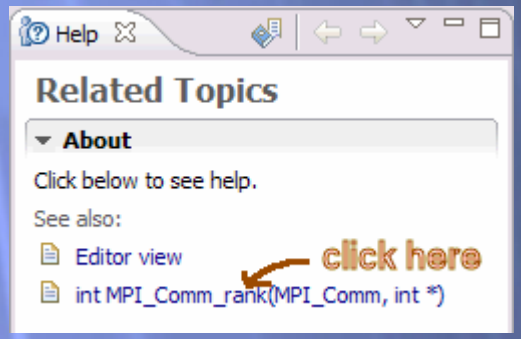
- Hover over the MPI Artifact identified in the source file to see additional information about that function call, for example

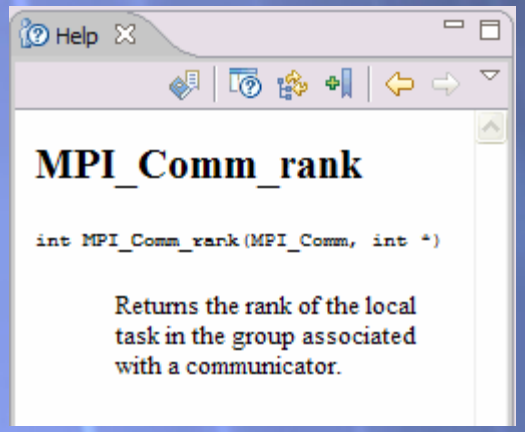


Context Sensitive Help

- 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
- Or, 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 (Ctrl-space)
- Try context sensitive help
  - Win: F1 Linux: Ctrl-F1 Mac: Help key

Project Created and Built

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

C/C++ - mpitest.c - Eclipse SDK - /home/tibbitts/workspace

File Edit Refactor Navigate Search Project Run MPI Artifacts OpenMP Artifacts Window Help

C/C++ Proj... Navigator

- helloMPI
  - Binaries
  - Includes
  - Debug
  - mpitest.c
- shallow

mpitest.c main.c

```
#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;
}
```

Outl... Mak...

- stdio.h
- mpi.h
- main

Help

MPI_Comm_rank

int MPI_Comm_rank(MPI_Comm, int

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

Go To: All Topics Search Related Topics Bookmarks

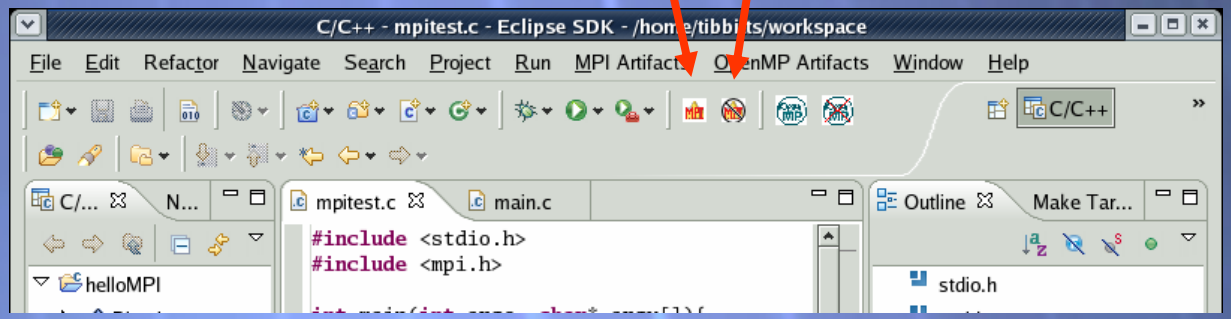
Problems Console Properties MPI Artifact View

C-Build [helloMPI]

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

make -k all
Building file: ../mpitest.c
Invoking: GCC C Compiler
mpicc -I/usr/local/include -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -
MF"mpitest.d" -MT"mpitest.d" -o"mpitest.o" "../mpitest.c"
Finished building: ../mpitest.c

Building target: helloMPI
Invoking: GCC C Linker
mpicc -L/usr/local/lib -o"helloMPI"  ./mpitest.o   -lmpi
Finished building target: helloMPI

Build complete for project helloMPI
```

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



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

C/C++ - mpitest.c - Eclipse SDK - /home/tibbitts/workspace

File Edit Refactor Navigate Search Project Run MPI Artifacts OpenMP Artifacts Window Help

helloMPI
- Binaries
- Includes
- Debug
- mpitest.c
- shallow

mpitest.c | main.c

```
#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;
}
```

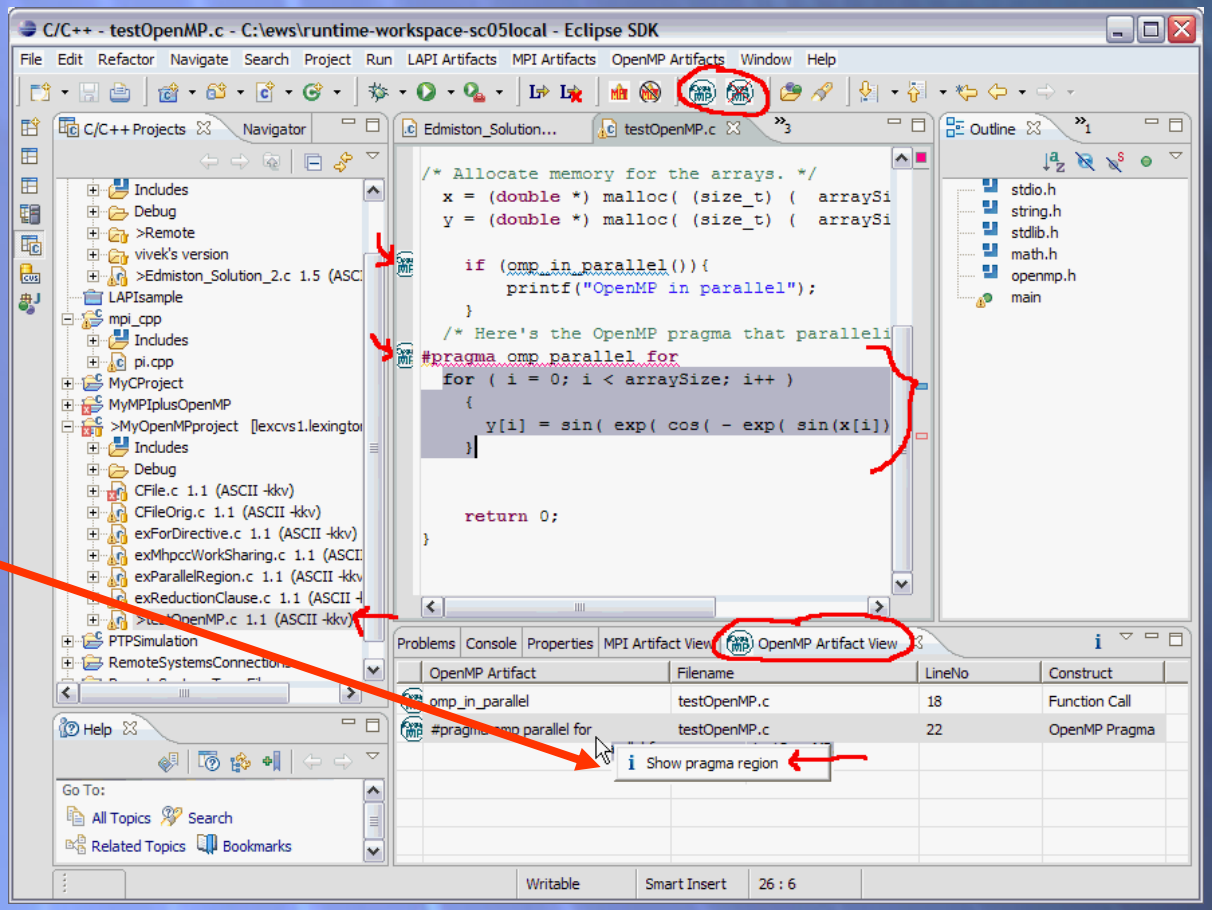
Outline: stdio.h, mpi.h, main

Problems | Console | Properties | MPI Artifact View

MPI Artifact	Filename	LineNo	Construct
MPI_Init	mpitest.c	6	Function C
MPI_Comm_rank	mpitest.c	7	Function C
MPI_Finalize	mpitest.c	9	Function C

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)



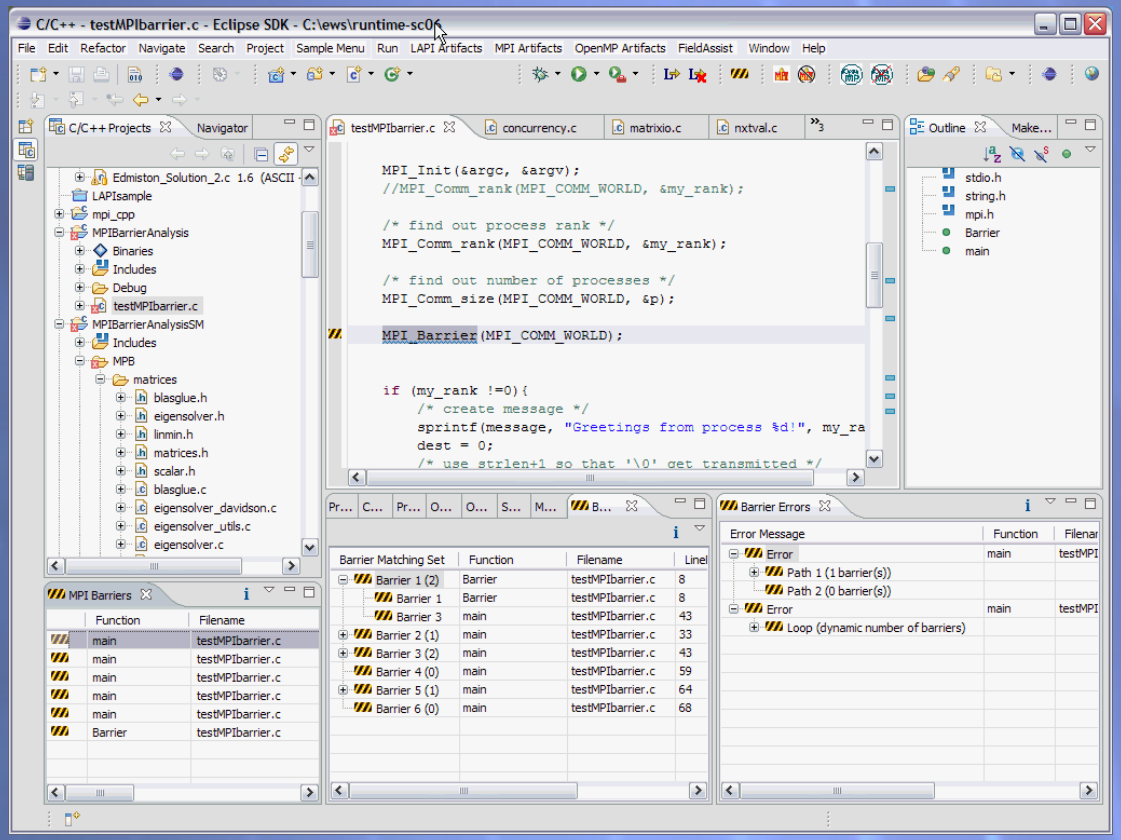
Show Concurrency

- 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

testregion.c | cfg.c | IncludeExample.c | MacroExample.c

```
#include <stdio.h>

int findme(int a)
{
    int f, c,d;

    #pragma omp parallel
    {
        a++;
        for(int i=0; i<a; i++) {
            d++;
            #pragma omp barrier
            a=a+d;
            if (a==f)
              {if (a==c) a=f;}
            else {
              f=a;
              #pragma omp barrier
            }
        }
    }
}
```

PTP PLDT: MPI Barrier Analysis



Verify barrier synchronization in C/MPI programs

- Interprocedural static analysis.
- Output:
  1) For verified programs, lists barrier statements that synchronize together (match)

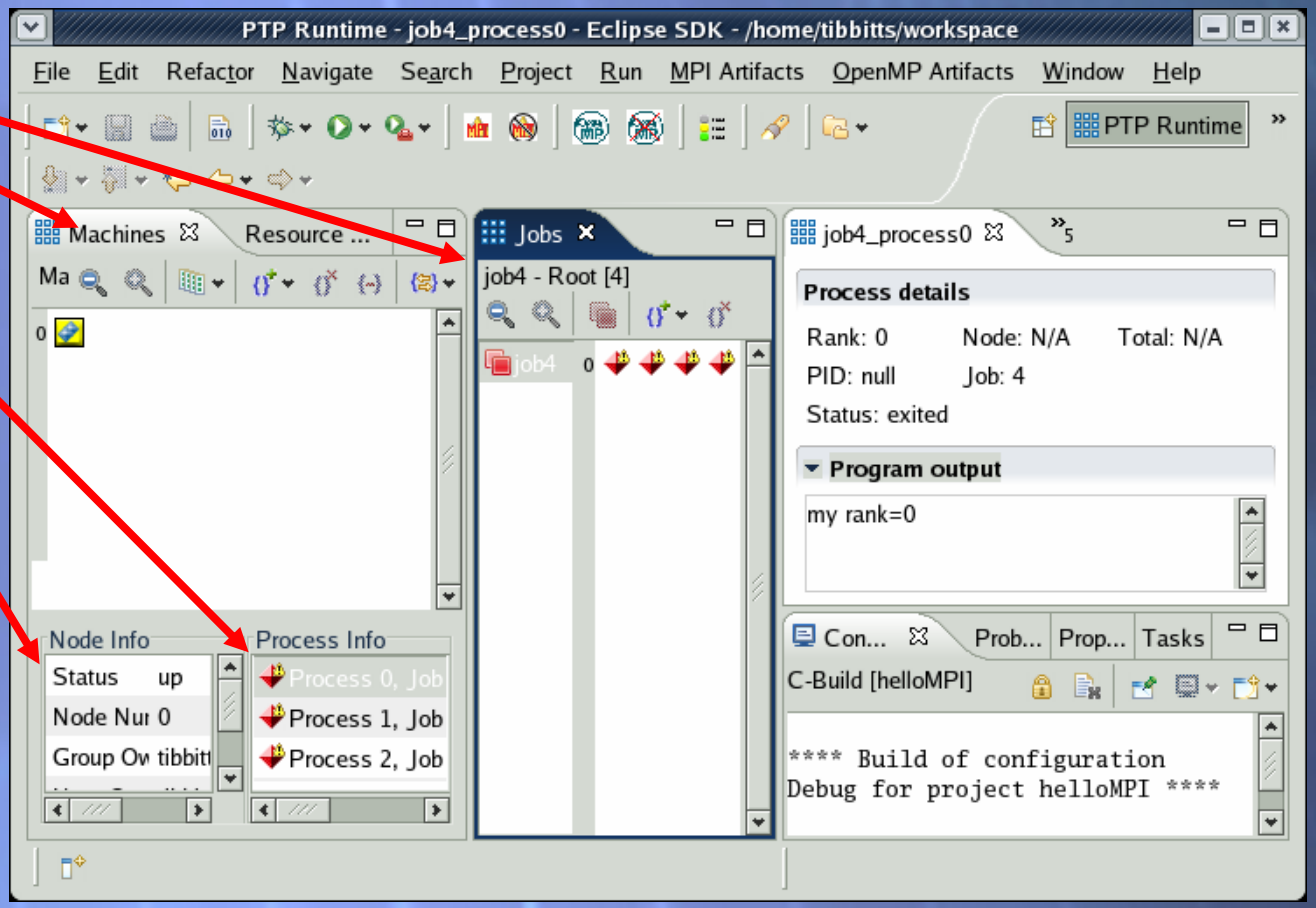
  2) For synchronization errors, reports counter example that illustrates and explains the error.

See PPOPP paper: Yuan Zhang and Evelyn Duesterwald. "Barrier matching for programs with textually unaligned barriers." In Proceedings of the Symposium on **Principles and Practice of Parallel Programming**, March 2007.

PTP Runtime Perspective (1)

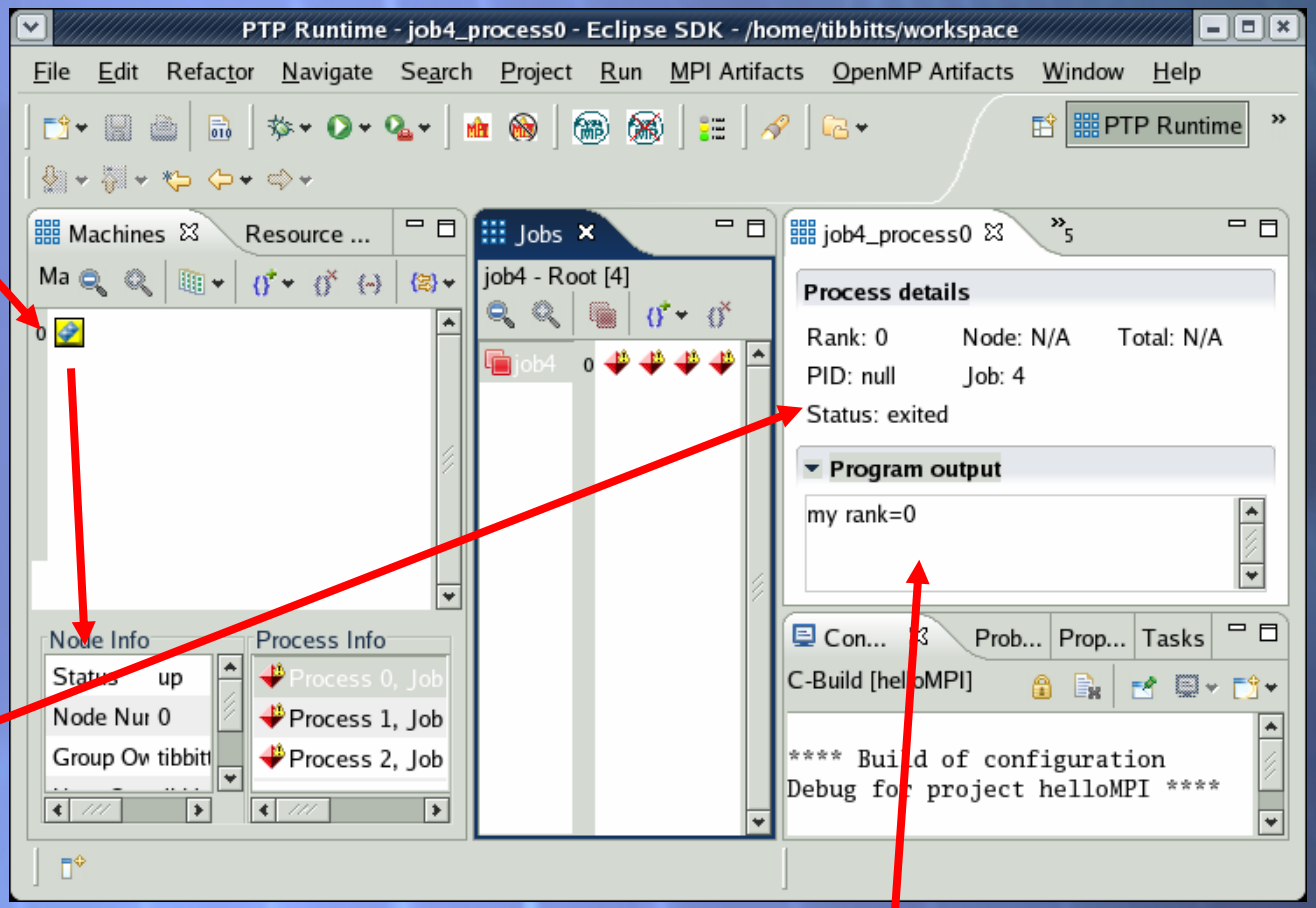
- Jobs view
- Machines view
- Processes on node
- Node details

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



PTP Runtime Perspective (2)

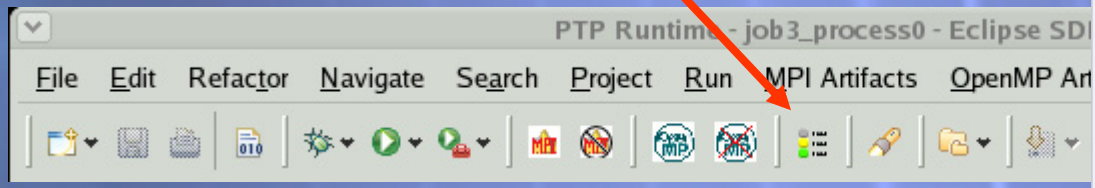
- Double-click a node in machines view to see more info
- Hover over node or process for tooltip popup
- Double-click a process to see process detail

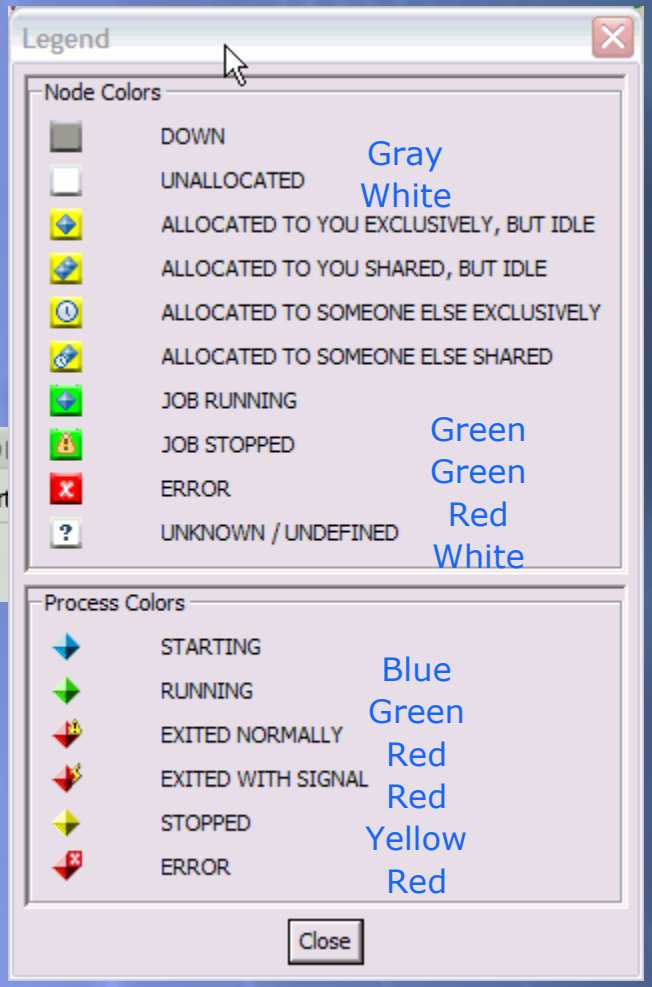


- Process output

Process and Job Icons

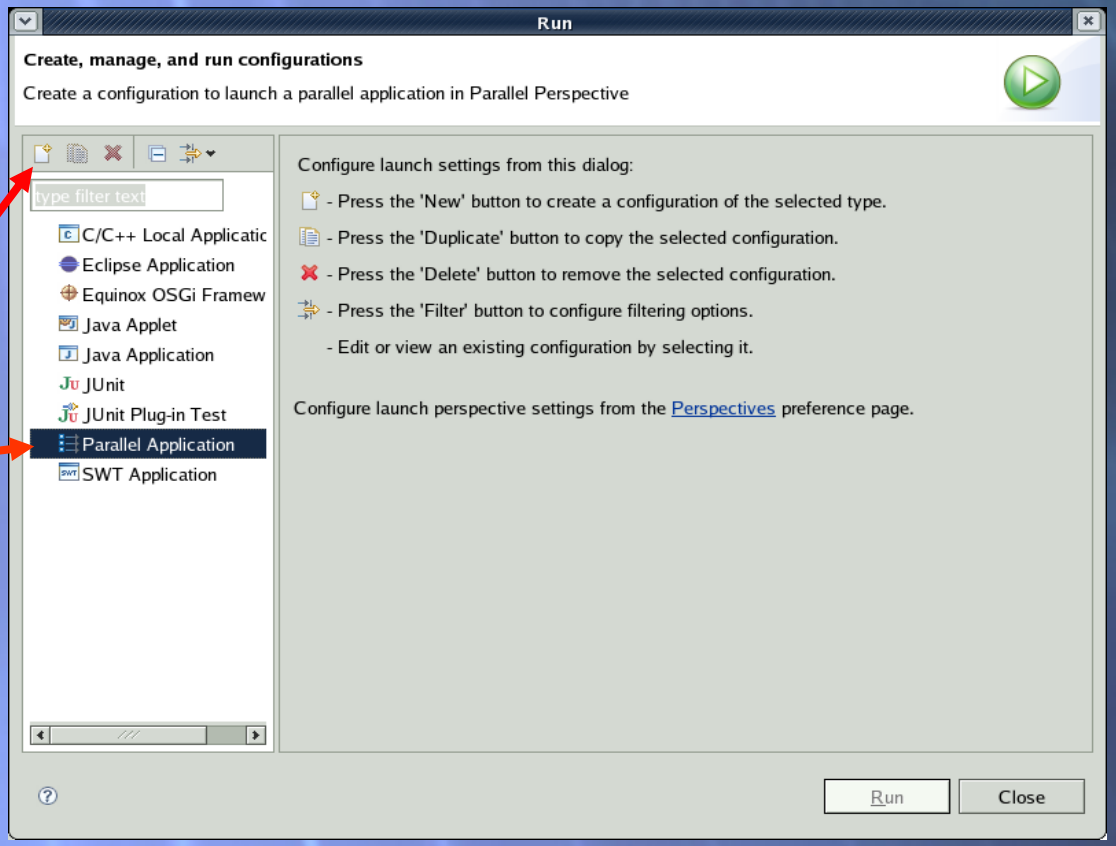
- Use legend icon in toolbar





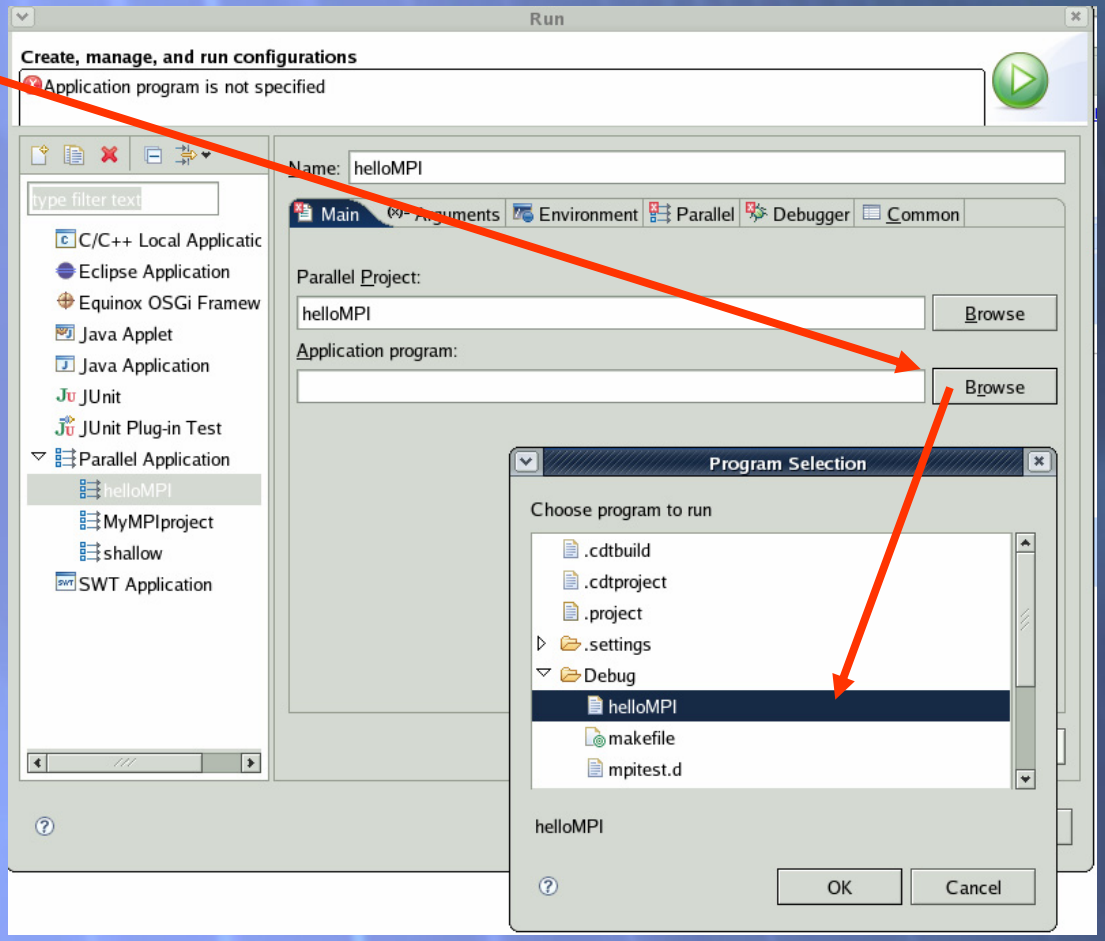
Running a Parallel Program (1)

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



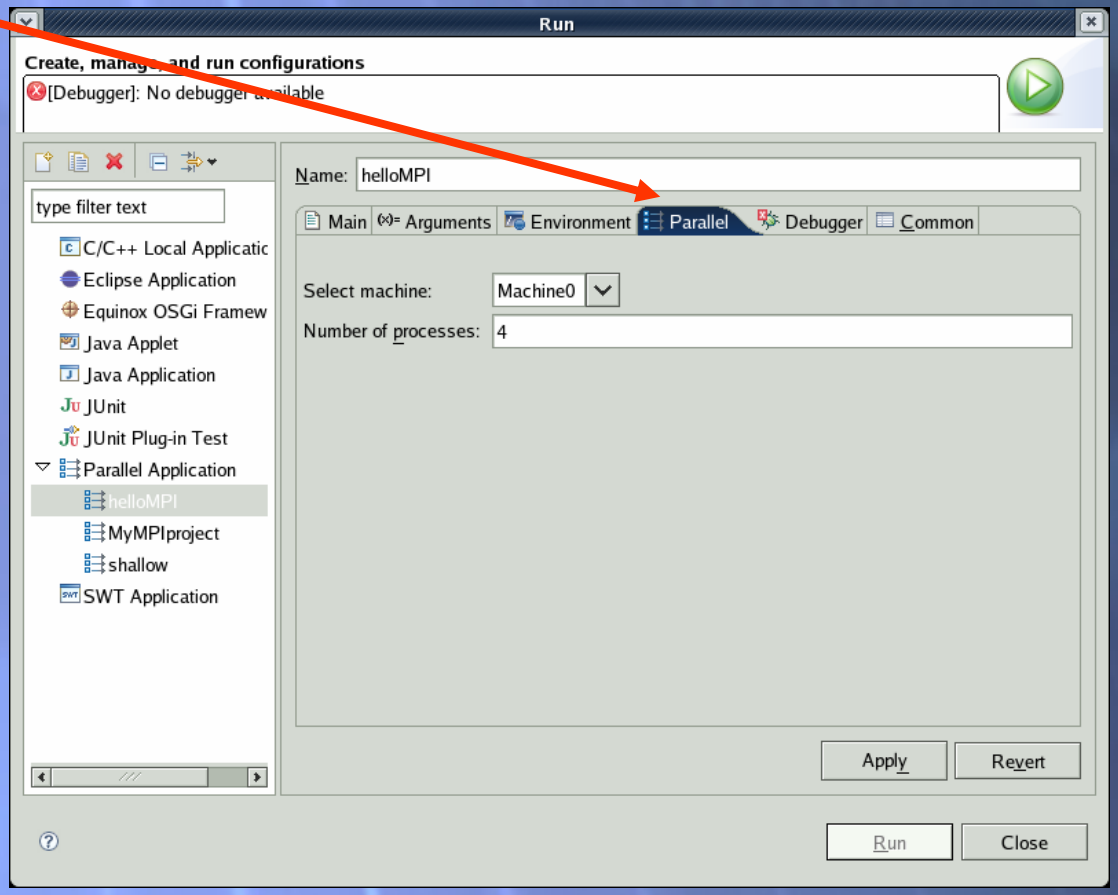
Running a Parallel Program (2)

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



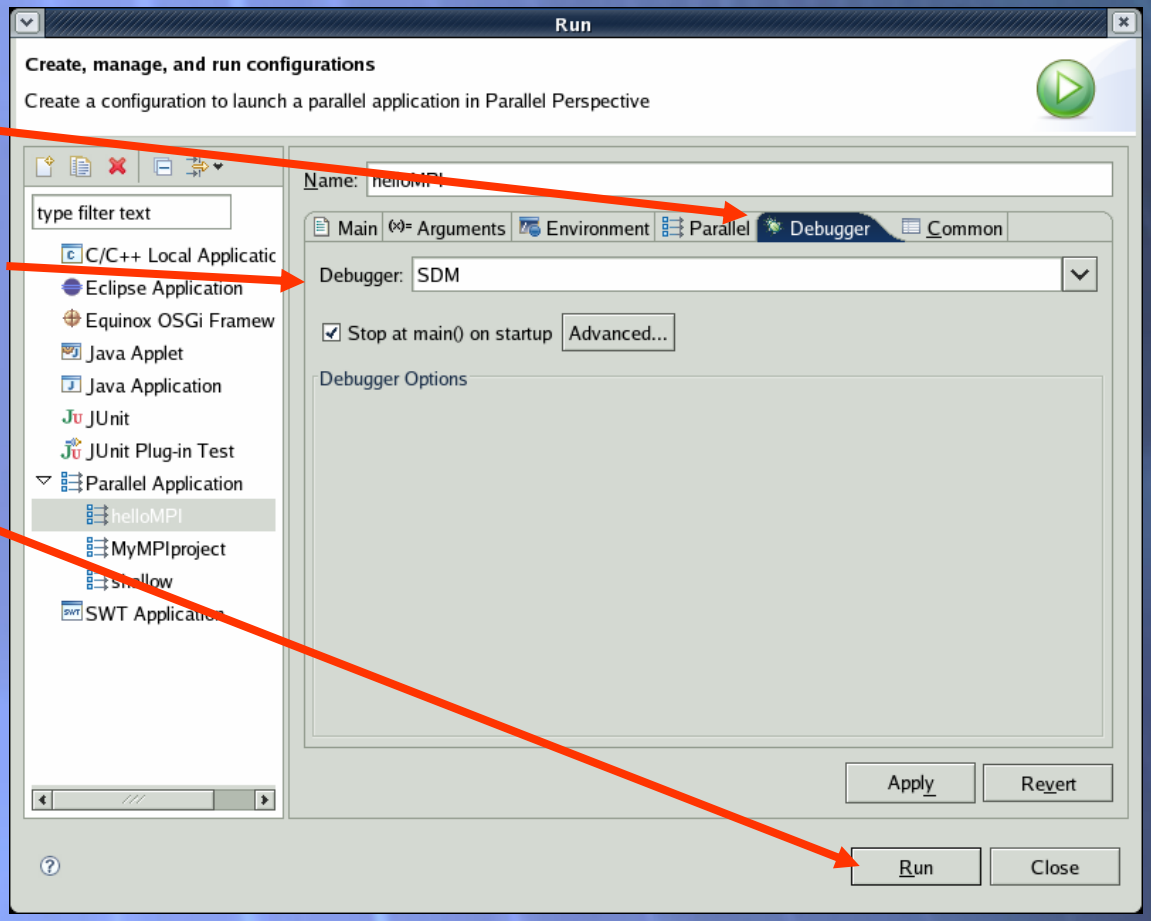
Running a Parallel Program (3)

- 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

PTP Runtime - job4_process0 - Eclipse SDK - /home/tibbitts/workspace

File Edit Refactor Navigate Search Project Run MPI Artifacts OpenMP Artifacts Window Help

Machines | Resource ... | Jobs | job4_process0

job4 - Root [4]

Process details

Rank: 0 Node: N/A Total: N/A

PID: null Job: 4

Status: exited

Program output

my rank=0

Node Info: Status up; Node Nu 0; Group Ow tibbitt

Process Info: Process 0, Job; Process 1, Job; Process 2, Job

Con... | Prob... | Prop... | Tasks

C-Build [helloMPI]

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

Module 5: 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

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

PTP Debug - main.c - Eclipse SDK - /home/tibbitts/ews/ptp-tutorial

File Edit Refactor Navigate Search Project Run MPI Artifacts OpenMP Artifacts Window Help

Parallel Debug
job4 - Root [3]
job2
job4
0

Debug
MyShallowProject [Parallel Application]
Process 0 (Suspended)
Thread [0] (Suspended)
0 main() at main.c:49

Breakpoints | Variables | Array View | Expressions

Name	Value
argc	1
argv	0x00000000
p	
u	
v	
psi	

main.c

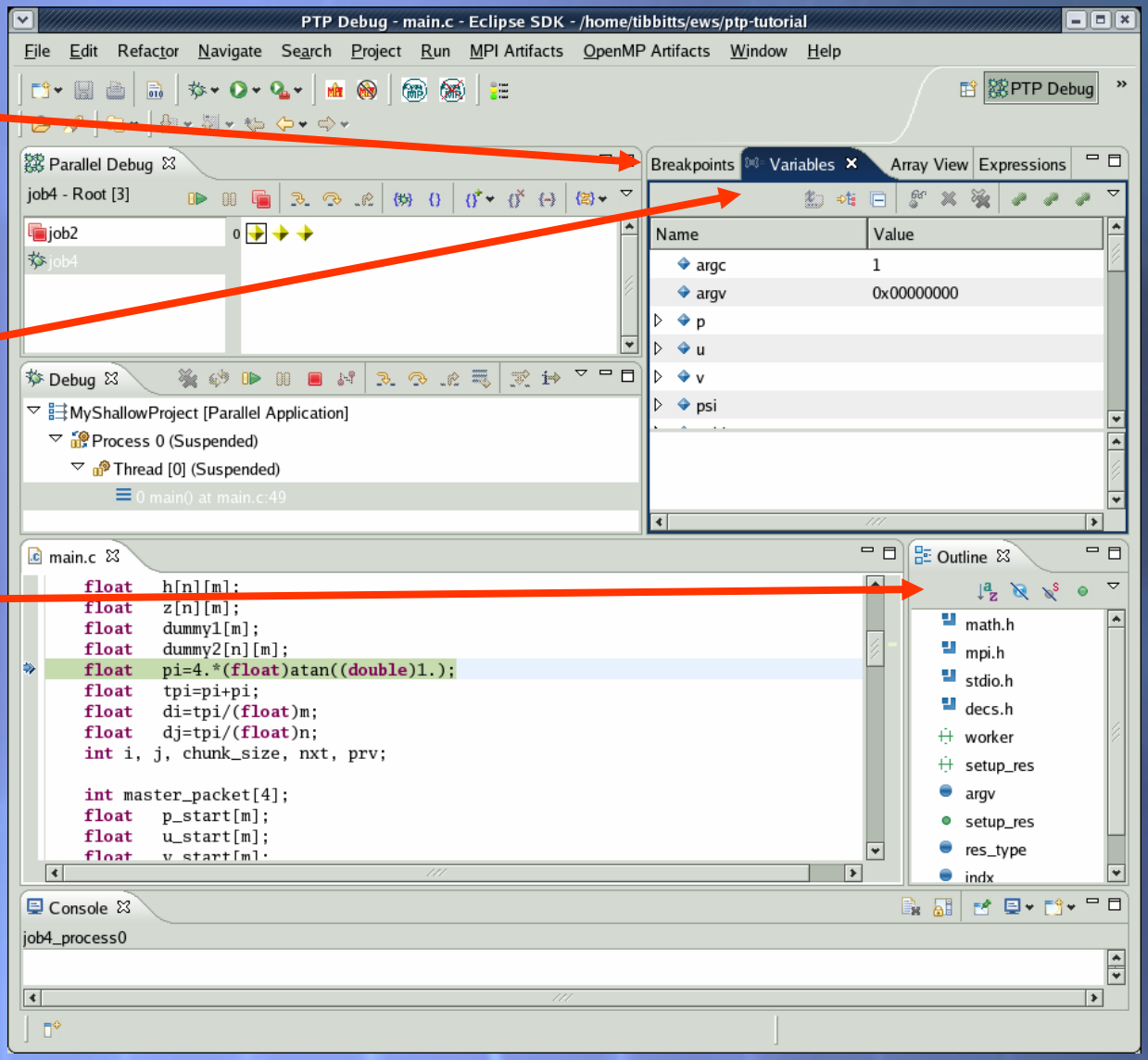
```
float   h[n][m];
float   z[n][m];
float   dummy1[m];
float   dummy2[n][m];
float   pi=4.*(float)atan((double)1.);
float   tpi=pi+pi;
float   di=tpi/(float)m;
float   dj=tpi/(float)n;
int i, j, chunk_size, nxt, prv;

int master_packet[4];
float   p_start[m];
float   u_start[m];
float   v_start[m];
```

Outline
math.h
mpi.h
stdio.h
decs.h
worker
setup_res
argv
setup_res
res_type
indx

Console
job4_process0

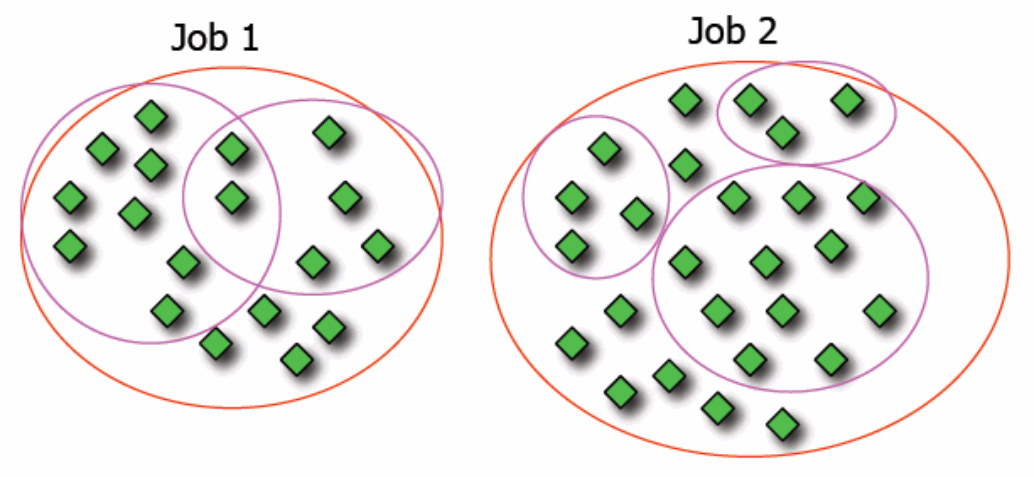
The PTP Debug Perspective (2)

- **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



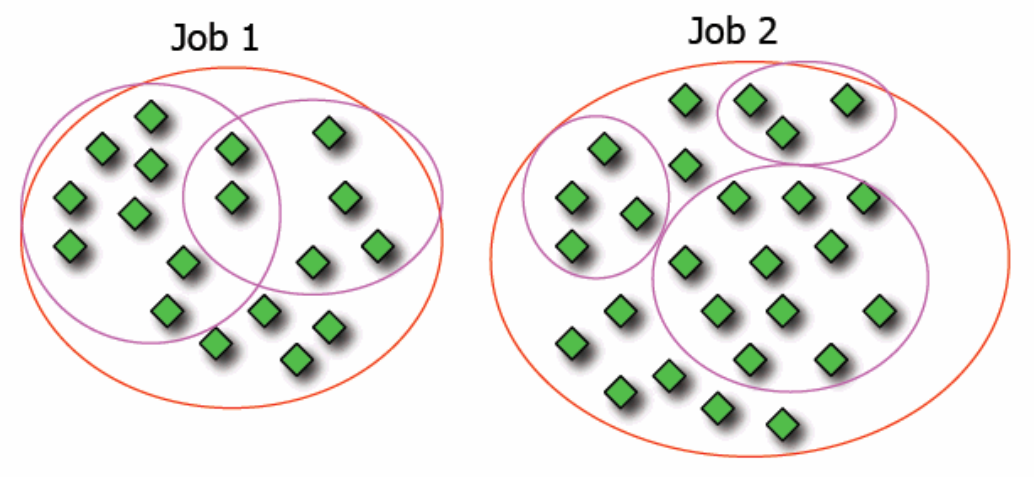
Process Sets (1)

- 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



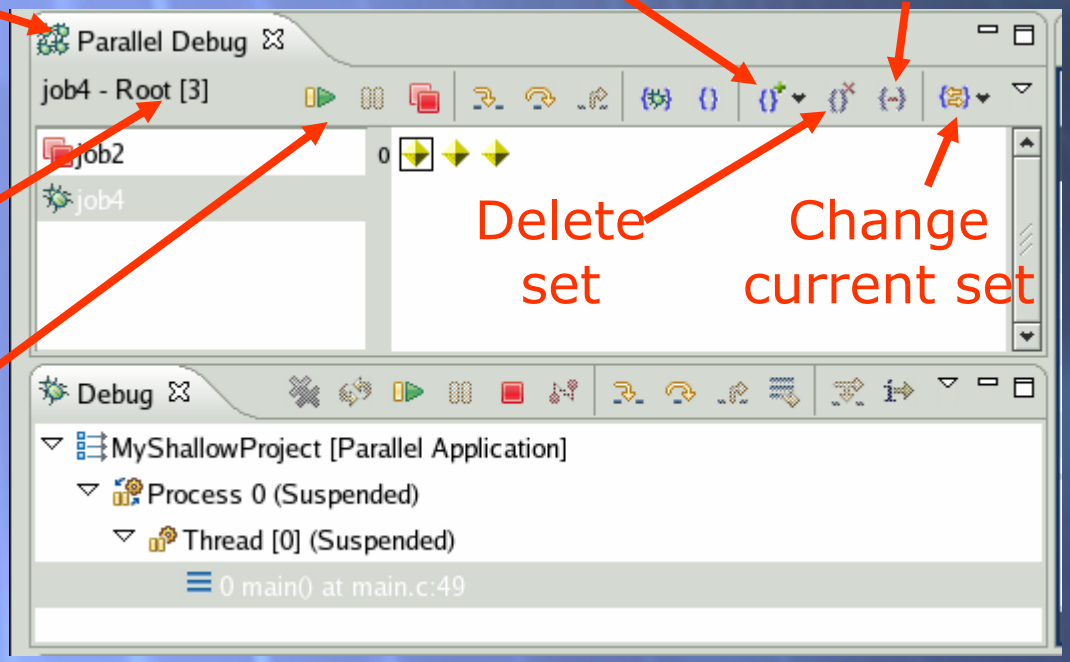
Process Sets (2)

- 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



Operations on Process Sets

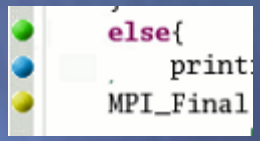
- Use the icons in the toolbar of the **Parallel Debug** view to create, modify, and delete process sets, and to change the current process set
- Current process set is listed next to job name along with number of processes in the set
- Debug operations on the **Parallel Debug** View toolbar always apply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



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

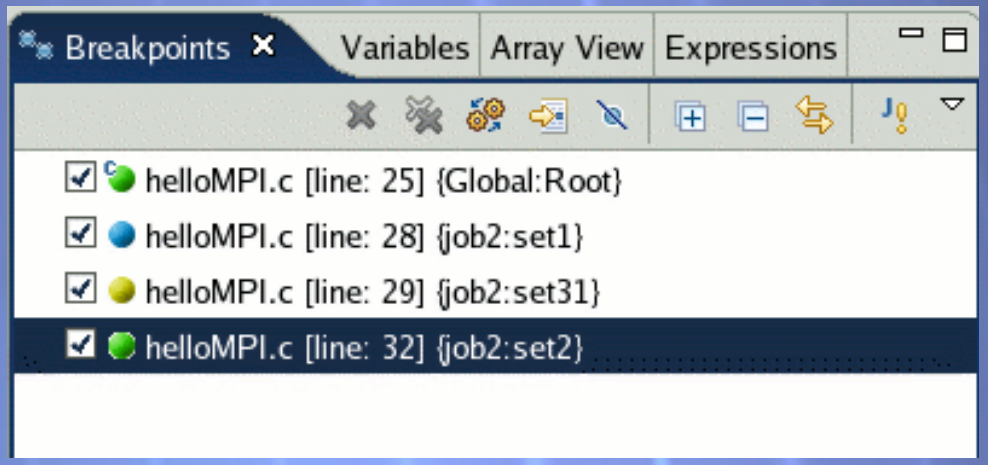
Setting Breakpoints

- 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

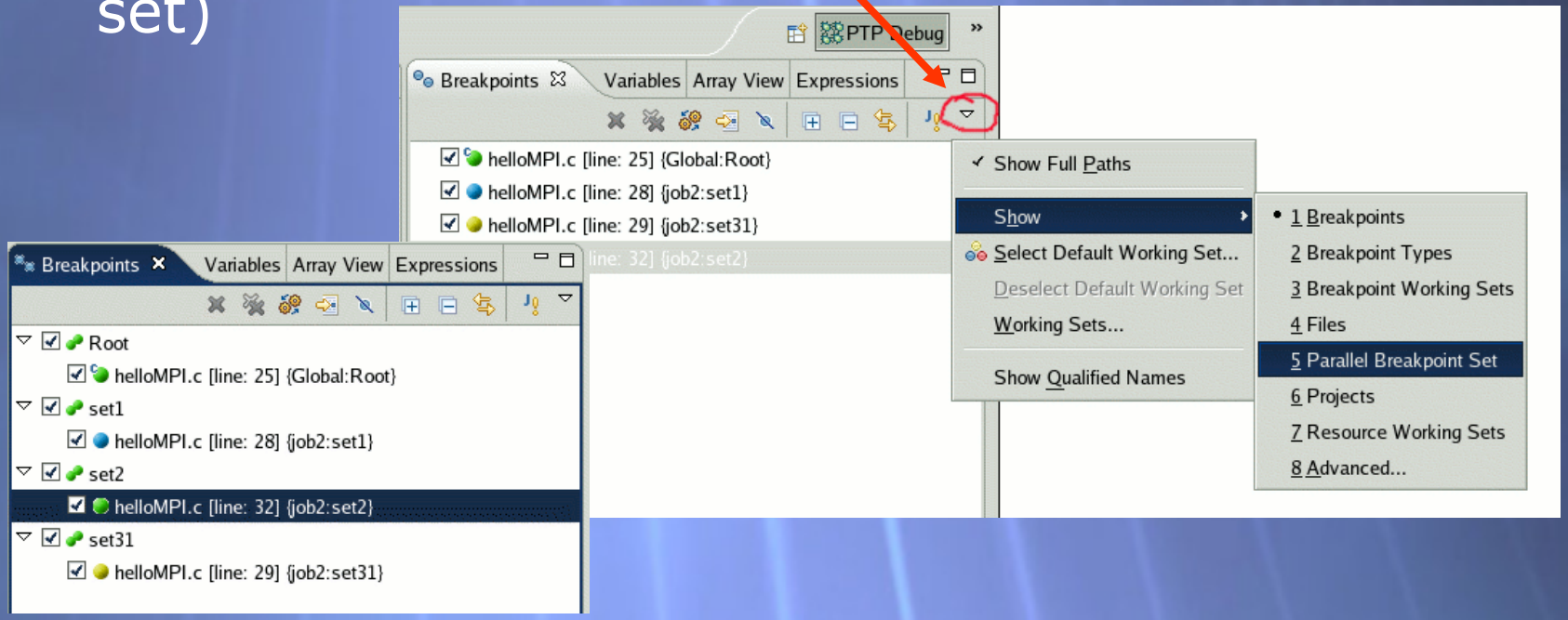
Breakpoint Information (1)

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



Breakpoint Information (2)

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

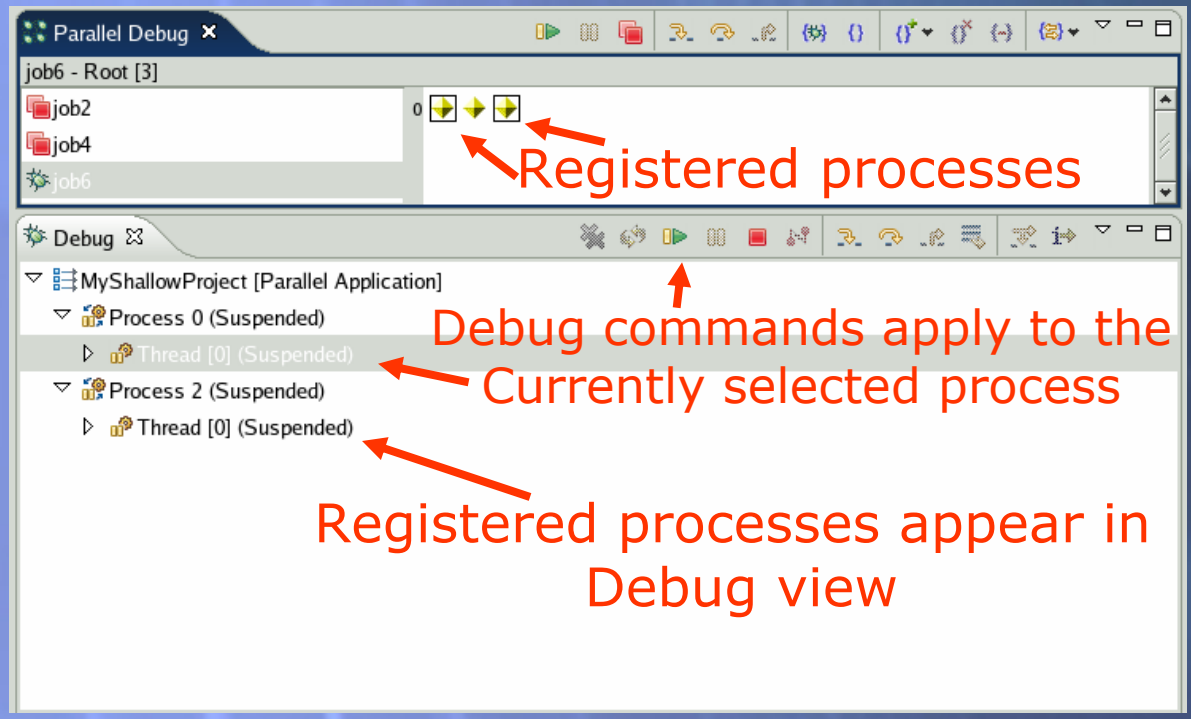


Process Registration (1)

- 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

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 Debug View– debug commands control *groups* of processes

Debug View – debug commands controls *single* processes (*registered* processes)

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

Current Instruction Pointer (2)

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

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

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

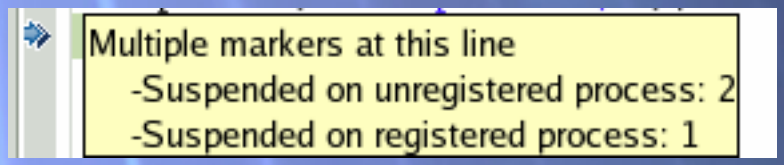

Multiple processes marker


Registered process marker


Un-registered process marker

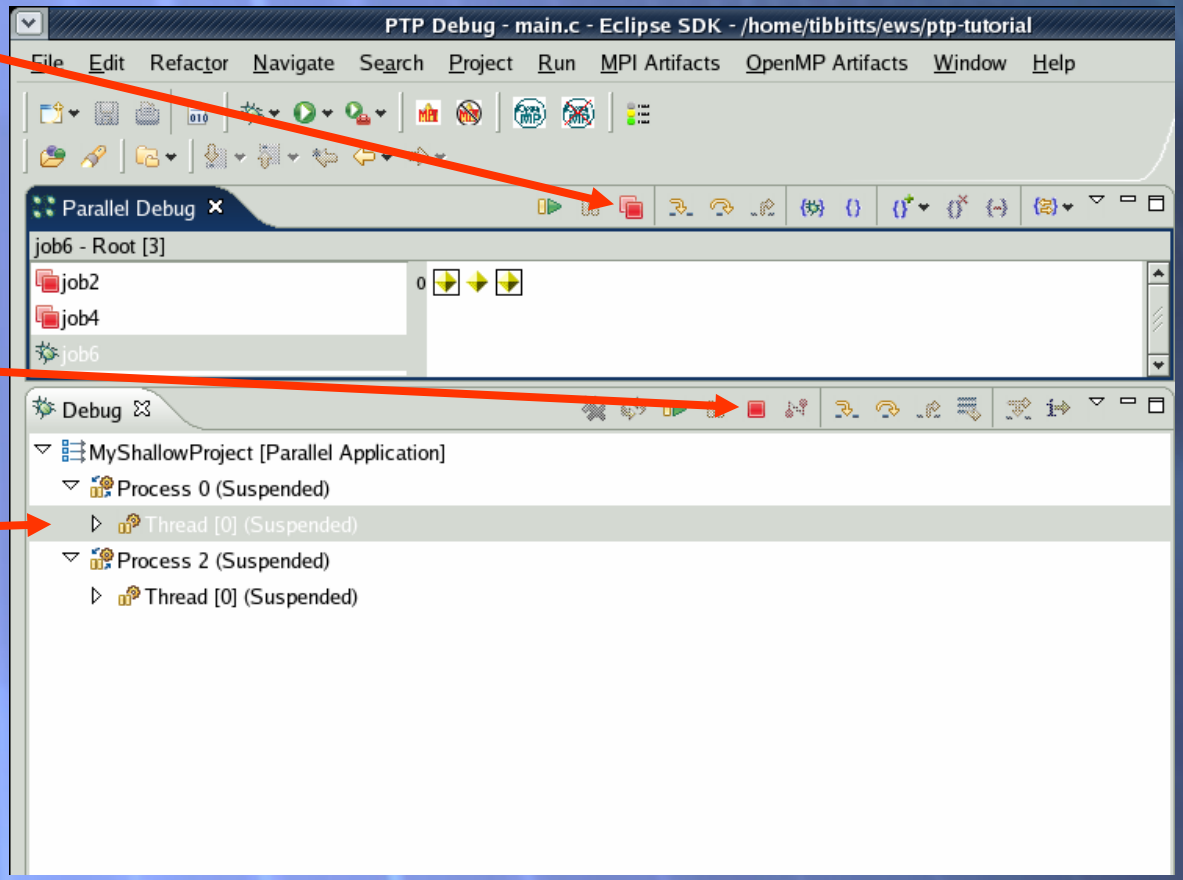
- The marker depends on the type of process stopped at that location

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



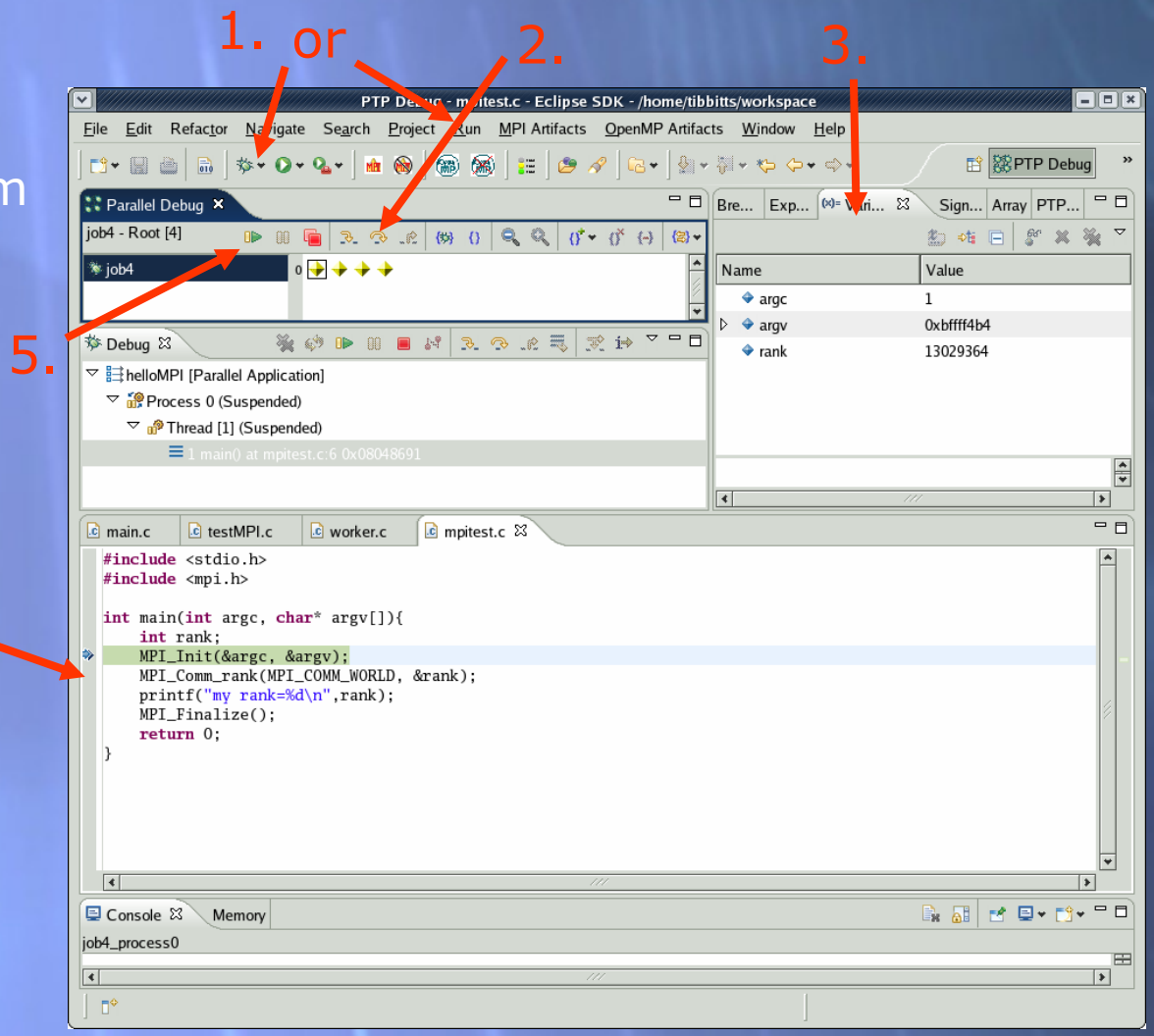
Terminating a Debug Session

- 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



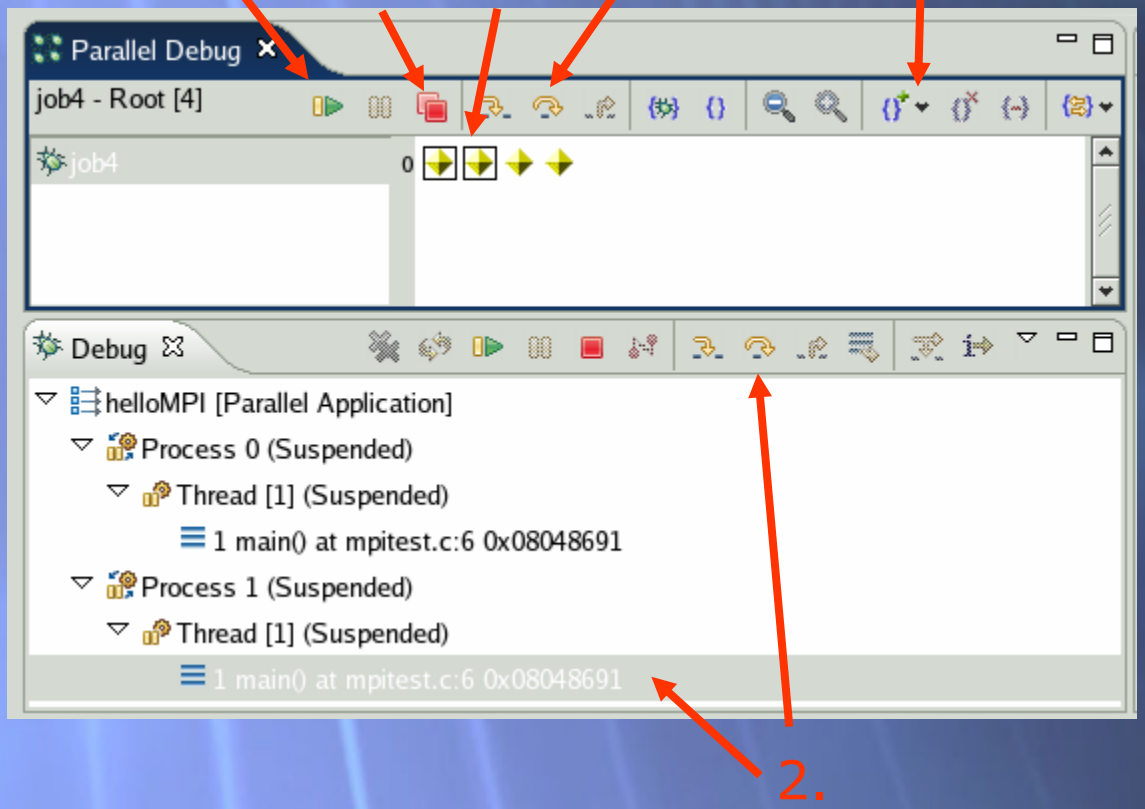
Basic Debug Commands

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



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



Module 6: Eclipse and the Enterprise

- Objective
  - How Eclipse can benefit enterprise development
  - Learn about other tools related to PTP
  - PTP upcoming features
- Contents
  - Links to other tools, including performance tools
  - Planned features for new versions of PTP
  - Additional documentation

Existing Enterprise Features

- Distributed development
  - CVS, Subversion
  - Commercial (e.g. ClearCase)
- Complete development lifecycle coverage
  - Design (UML), edit, debug, test, etc.
- Code refactoring
- Rich client platform (RCP)
  - Application framework

Trends in Enterprise Computing

- Limited exploitation of parallelism to date
  - Instruction level parallelism (on chip)
  - Symmetric multiprocessing (off chip)
  - Multi-core (2-4 cores)
- Significant parallelism now in reach of the Enterprise
  - Clusters (e.g. server farms) + cheap interconnect (e.g. Infiniband)
  - Manycore (8+ cores)
  - Compute off-load (e.g. GPGPUs)
  - Hybrid architectures (e.g. Cell)

How will PTP help?

- Until new programming models/languages become available
  - Better tools to support explicit parallelism (e.g. MPI, OpenMP, threads)
  - Simplify interaction with parallel systems
  - Provide parallel debugging support
- Once new programming models/languages are available
  - Integrate parallel languages with IDE
  - Provide new tools to aid the developer

PTP-Related Tools

- TAU – Tuning and Analysis Utilities
  - http://www.cs.uoregon.edu/research/tau
  - Eclipse plug-in integrates external performance tool instrumentation for PTP
- Performance Visualization: TuningFork
  - Performance visualization Eclipse plug-ins from IBM Research
  - Available on alphaWorks now
  - http://www.alphaworks.ibm.com/tech/tuningfork
  - Enhancements for parallel computing underway
- Upcoming: Performance framework for PTP

TAU
(Tuning and Analysis Utilities)

Demo presented by Wyatt Spear, wspear@cs.uoregon.edu
http://www.cs.uoregon.edu/research/**tau**/

- TAU Features
  - Highly scalable and portable: works on numerous operating systems and architectures
  - Supports many data collection and analysis options, including hardware counters, callpath profiling and memory profiling
  - Allows output and conversion of performance data to several trace and profile formats
- TAU Eclipse Plug-ins
  - Simple configuration of TAU instrumentation and data collection options
  - Automatic 'one-click' instrumentation, compilation, execution and data-collection
  - Profile database and analysis tools integrated with Eclipse, including source callback

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
- … and many more!

PTP Upcoming Features (1)

- PTP 2.0 (late 2007 / early 2008)
  - Resource management
    - Support for viewing of jobs in queues
    - View and query the status of queues
    - Submit and control jobs
    - Resource managers supported
      - Possibly SLURM, LoadLeveler , LSF, MOAB ?
  - Additional runtime support for MPICH2 improved
  - PLDT enhancements – PLDT 2.0
    - MPI barrier analysis – to detect possible deadlocks
    - PLDT 2.0 early access builds are available
    - Requires CDT 4.0 and Eclipse 3.3 (Europa)

PTP Upcoming Features (2)

- PTP 2.0 – (late 2007 / early 2008)
  - Remote services support
    - Allow projects to reside on remote systems
    - Ability to build projects remotely
    - Ability to launch and debug projects remotely
  - Debugger improvements

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

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
- "An Integrated Tools Platform for Multi-Core Enablement," Beth Tibbitts & Evelyn Duesterwald, STMCS: Second Workshop on Software Tools for Multi-Core Systems, March 2007
  - http://www.isi.edu/~mhall/stmcs07/program.html

PTP Tutorial Feedback

- Please complete feedback form
- Your feedback is valuable!

Thanks for attending
We hope you found it useful

Appendix: 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
  - (Installing all of PTP is beyond the scope of this tutorial)

Software Prerequisites

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

- Note: for Windows, an easier solution is to install the "Wascana" package of Eclipse CDT for windows, available on the tutorial CD or at:
  - http://wascana.sourceforge.net/
  - That's all you need; ignore the rest of this section!

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 untarring 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-3.3-linux-gtk.tar.gz
```

Eclipse SDK Installation (MacOS X)

- From the Finder, open **TutorialCD**
- Open the **eclipse** folder
- Double-click on **eclipse-SDK-3.3-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-3.3-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 3.3 comes preconfigured with a link to the Europa Discovery Site that contains a large number of features
  - Europa projects are guaranteed to work with Eclipse 3.3

Installing Eclipse Features with 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
- To install CDT via a Remote Site:
  - Choose **Help ▸ Software Updates ▸ Find and Install…**
  - Select **Search for new features to install**
  - Click **Next >**
  - Check **Europa Discovery Site**
  - Click **Finish**
  - Select a Mirror site if asked
  - In the resulting Updates dialog, expand "Europa Discovery Site" and check "C and C++ Development"
  - Click Next, accept license agreement terms, Next, Finish.
  - When done downloading, Select "Install All"
  - When prompted to restart eclipse, answer "Yes"

Installing Eclipse
from the Web

- Download Eclipse
  - http://eclipse.org/downloads
- Download the Eclipse SDK for your platform
  - If you don't need Java or plug-in development, you might install just the Eclipse "Platform."
    - Eclipse *Platform* is "bare bones"
    - Eclipse *SDK* includes Java and plug-in development tools
  - Unzip or untar on your machine
  - This is just the base Eclipse.
    CDT/PTP install covered separately.
- Launch eclipse
  - From <install-dir>/eclipse/eclipse *executable*
  - Or from command line: `eclipse &`

Installing

continued

- After launching eclipse,
  - Help > Software Updates > Find and install
  - Select other features to install
  - If any "red X's" indicated pre-reqs, click "Select required" button to select them.
  - For CDT and PTP install information, see http://www.eclipse.org/ptp/docs/install.html and follow directions at least through CDT install.
  - Restart eclipse after CDT installation, as prompted