



parallel tools platform

<http://eclipse.org/ptp>

Eclipse and the Parallel Tools Platform

Greg Watson, IBM
g.watson@computer.org

Jay Alameda, NCSA
jalameda@ncsa.illinois.edu

Wyatt Spear, U. Oregon
wspear@cs.uoregon.edu

Beth Tibbitts, NCSA
beth@tibweb.com

Jeff Overbey, Auburn U.
jeffreyoverbey@acm.org

Galen Arnold, NCSA
arnoldg@ncsa.uiuc.edu

Steven R. Brandt, LSU
sbrandt@cct.lsu.edu

Chris Navarro, NCSA
cmnavarr@illinois.edu

July 14, 2014

Portions of this material are supported by or based upon work supported by

- The Defense Advanced Research Projects Agency (DARPA) under its Agreement No. HR0011-07-9-0002
- The Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (award number OCI 07-25070)
- The United States Department of Energy under Contract No. DE-FG02-06ER25752
- The SI2-SSI Productive and Accessible Development Workbench for HPC Applications, which is supported by the National Science Foundation under award number OCI 1047956



XSEDE[14]

Tutorial Outline

Time (Tentative!)	Module	Topics	Presenter
8:00-9:00	Eclipse Installation Intro/Overview } (1:00)	<ul style="list-style-type: none"> ✦ Installation of Eclipse and PTP ✦ Eclipse architecture & organization overview 	Beth Greg
9:00-9:30	Eclipse basics (1:45)	<ul style="list-style-type: none"> ✦ Eclipse basics; synchronized projects; Git support ✦ Editor features; MPI Features 	Beth
9:30-10:00	BREAK	Atrium Foyer	
10:00-11:15	(continue Basics)		
11:15-12:00	Build/run (1:00)	<ul style="list-style-type: none"> ✦ Building w/Makefile ✦ Target configurations and launching a parallel app ✦ Modules (Build Environment Mgmt) 	Steve
12:00 – 1:00	Lunch	Atrium Ballroom B & C	
1:00-1:15	(continue Build/Run)		
1:15-2:15	Debugging (1:00)	<ul style="list-style-type: none"> ✦ Debugging an MPI program 	Greg
2:15-3:30	Fortran Advanced Features } (1:15)	<ul style="list-style-type: none"> ✦ Fortran ✦ Search & refactoring ✦ GUI terminal ✦ XSEDE feature 	Jeff Steve Chris
3:30-4:00	BREAK	Atrium Foyer	
4:00-4:50	Performance Tuning & Analysis Tools (:50)	<ul style="list-style-type: none"> ✦ TAU, External Tools FrameWork (:40) ✦ Perfsuite 	Wyatt Chris
4:50-5:00	Wrap-up (:10)	<ul style="list-style-type: none"> ✦ Online resources; getting involved; feedback 	Beth

Final Slides, Installation Instructions

- ★ Please go to <http://wiki.eclipse.org/PTP/tutorials/XSEDE14> for slides and installation instructions

Installation

- ✦ Objective
 - ✦ To learn how to install Eclipse and PTP
- ✦ Contents
 - ✦ System Prerequisites
 - ✦ Eclipse Download and Installation of “Eclipse for Parallel Application Developers”
 - ✦ Installation Confirmation
 - ✦ Updating the PTP within your Eclipse to the latest release

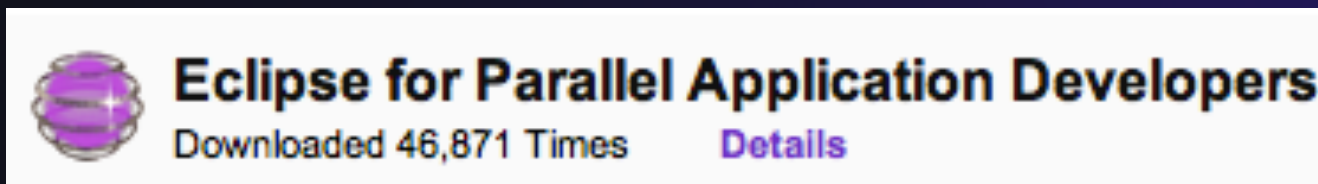
System Prerequisites

- ★ Local system (running Eclipse)
 - ★ Linux (just about any version)
 - ★ MacOSX (10.5 Leopard or higher)
 - ★ Windows (XP on)
- ★ Java: Eclipse requires Sun or IBM Java
 - ★ Only need Java runtime environment (JRE)
 - ★ Java 1.7 or higher
 - ★ Java 1.7 is the same as JRE Version 7
 - ★ The GNU Java Compiler (GCJ), which comes standard on Linux, will not work!
 - ★ OpenJDK, distributed with some Linux distributions, comes closer to working, but should not be used.
 - ★ See <http://wiki.eclipse.org/PTP/installjava>

Eclipse Packages

- ★ The current version of Eclipse (4.4) is also known as "Luna"
- ★ Eclipse is available in a number of different packages for different kinds of development
 - ★ <http://eclipse.org/downloads>
- ★ For PTP, we recommend the all-in-one download:
 - ★ Eclipse for Parallel Application Developers

New! See next slide for update



We often call this the "Parallel Package"

New! Parallel Package updated

- ★ The public Parallel Package on eclipse.org/downloads is only updated three times yearly
- ★ We are now building updated all-in-one packages with new releases of PTP already installed.
 - ★ You can use this, or just update the original one
 - ★ See next slides for updating...

To use already-updated package:

- ★ Go to <http://eclipse.org/ptp/downloads.php>
- ★ Under **File Downloads:**
- ★ Click on the link, and on the file downloads page, see **Parallel Application Developers Package** and download the appropriate file for your platform
 - ★ Mac OS X
 - ★ Linux X86 and X86_64
 - ★ Windows x86 and x86_64
- ★ Unzip or untar it



Exercise

1. Download the “Eclipse for Parallel Application Developers” package to your laptop
 - ★ Your tutorial instructions will provide the location of the package
 - ★ Make sure you match the architecture with that of your laptop
2. If your machine is Linux or Mac OS X, untar the file
 - ★ On Mac OS X you can just double-click in the Finder
3. If your machine is Windows, unzip the file
4. This creates an **eclipse** folder containing the executable as well as other support files and folders

Starting Eclipse

★ **Linux**

- ★ From a terminal window, enter
“<eclipse_installation_path>/eclipse/eclipse &”

★ **Mac OS X**

- ★ From finder, open the **eclipse** folder where you installed
- ★ Double-click on the **Eclipse** application
- ★ Or from a terminal window

★ **Windows**

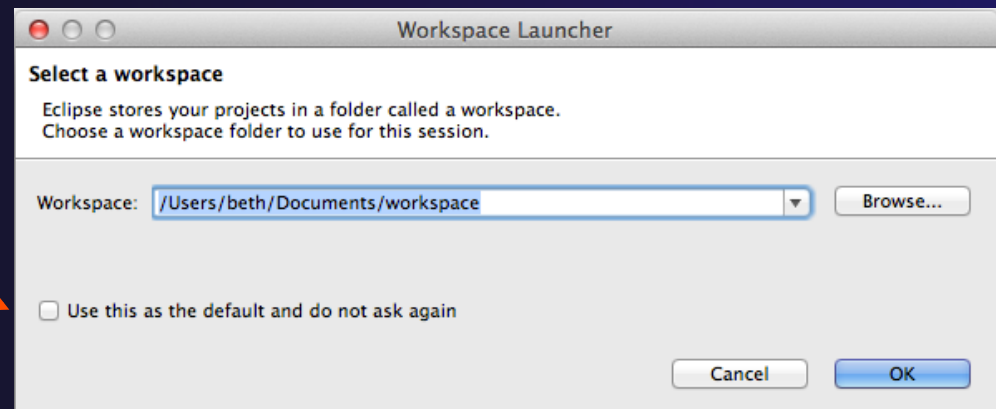
- ★ Open the **eclipse** folder
- ★ Double-click on the **eclipse** executable



Specifying A Workspace

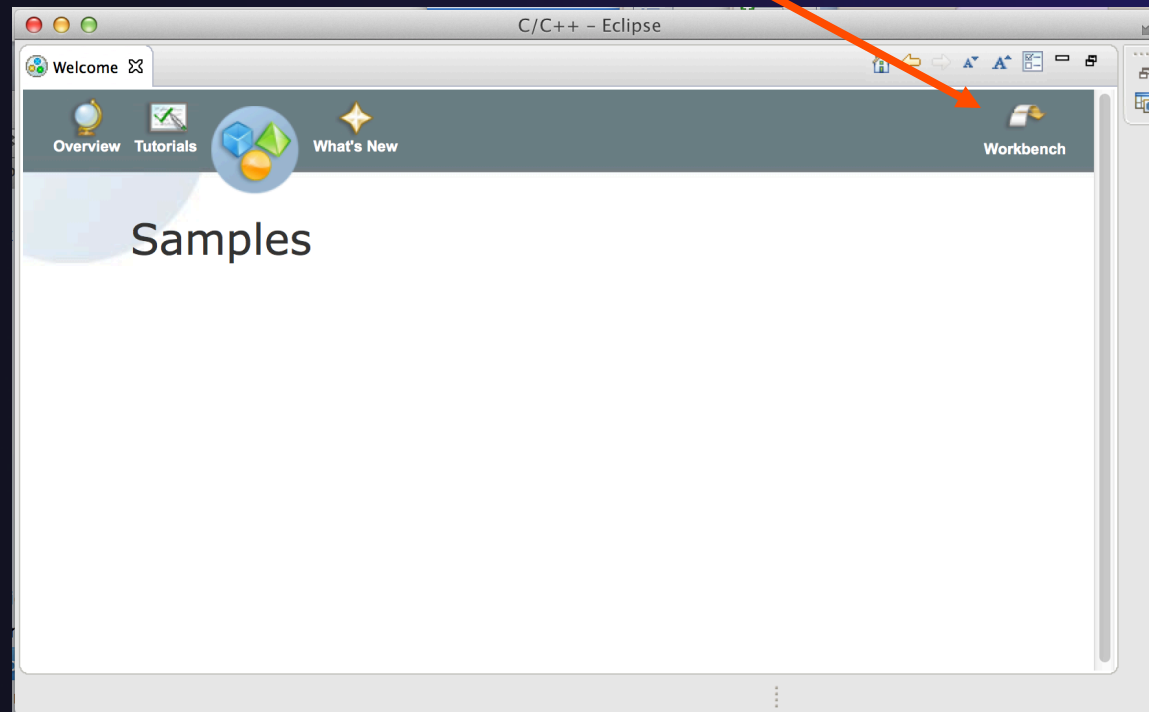
- ✦ Eclipse prompts for a workspace location at startup time
- ✦ The workspace contains all user-defined data
 - ✦ Projects and resources such as folders and files
 - ✦ The default workspace location is fine for this tutorial

The prompt can be turned off



Eclipse Welcome Page

- ★ Displayed when Eclipse is run for the first time
- Select "Workbench"

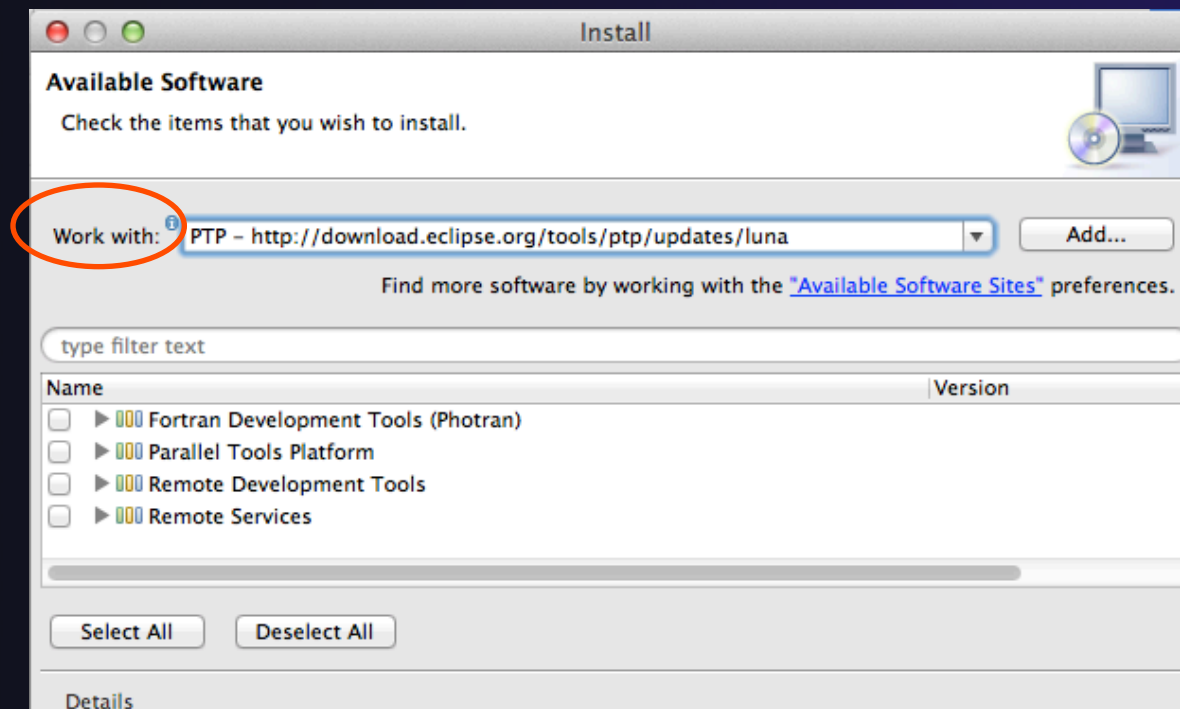


Checking for PTP Updates

- ✦ From time-to-time there may be newer PTP releases than the Luna release
 - ✦ Luna and "Parallel package" updates are released only in September and February
- ✦ PTP maintains its own update site with the most recent release
 - ✦ Bug fix releases can be more frequent than base Eclipse (e.g. Luna), and what is within the parallel package
- ✦ **You must enable (and install from) the PTP-specific update site before the updates will be found**

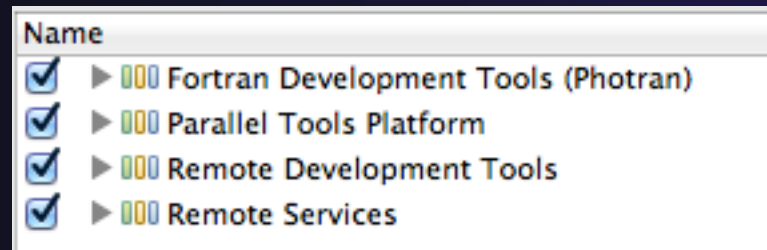
Updating PTP

- ★ Now select **Help>Install New Software...**
 - ★ In the **Work With:** dropdown box, select this update site, or enter it:
<http://download.eclipse.org/tools/ptp/updates/luna>



Updating PTP (2)

- ★ Easiest option is to “Select All” - which updates existing PTP features and adds a few more

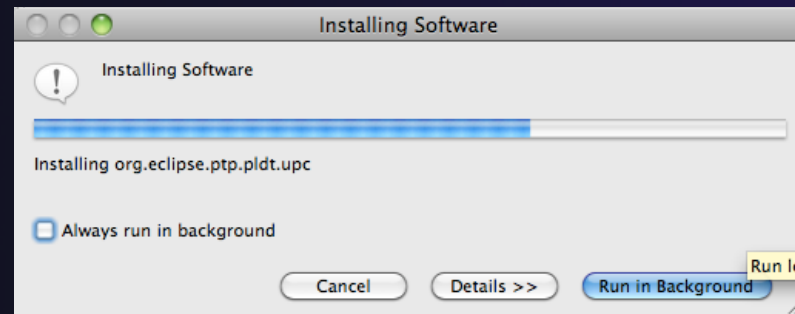


Note: for this tutorial, this installs extra features we'll refer to later anyway (TAU, PerfSuite)

- ★ Select **Next** to continue updating PTP
- ★ Select **Next** to confirm features to install

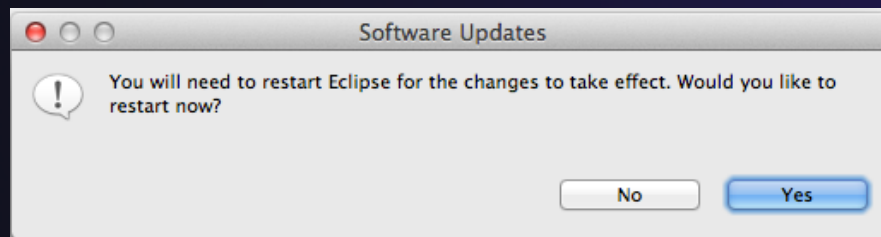
Updating PTP (3)

- ★ Accept the License agreement and select **Finish**



Updating PTP - restart

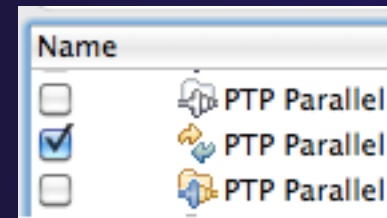
- ★ Select **Yes** when prompted to restart Eclipse



Updating Individual Features

- ★ It's also possible (but a bit tedious) to update all the PTP features without adding any new features
 - ★ Open each feature and check the ones you want to update

- ★ Icons indicate: Grey plug: already installed
Double arrow: can be updated
Color plug: Not installed yet

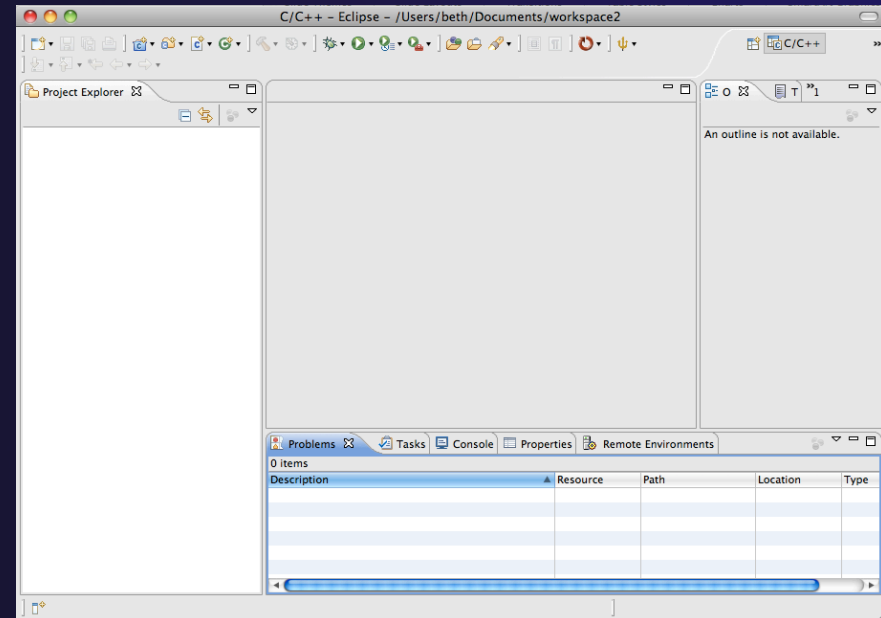


- ★ Note: if network is slow, consider unchecking:

Contact all update sites during install to find required software

Restart after Install

- ★ If any new top-level features are installed, they will be shown on the welcome screen
- ★ We only updated PTP, so we land back at C/C++ Perspective



- ★ **Help>About** or **Eclipse > About Eclipse ...** will indicate the release of PTP installed
- ★ Further **Help>Check for Updates** will find future updates on the PTP Update site



Exercise

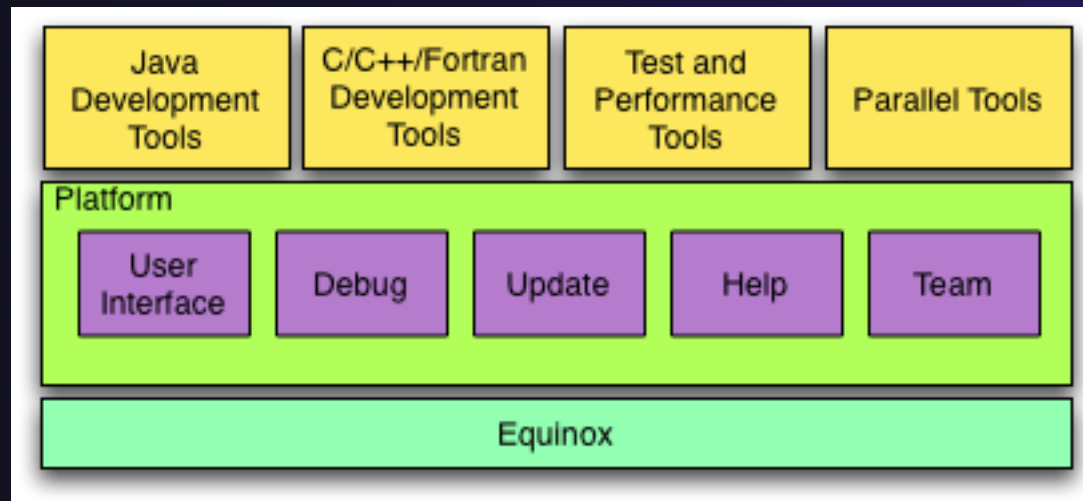
1. Launch Eclipse and select the default workspace
2. Configure Eclipse to check for PTP updates
3. Update all PTP features to the latest level
4. Install the optional features of PTP, including TAU and PerfSuite
 - Selecting *all* features accomplishes 3. and 4.
5. Restart Eclipse once the installation is completed

Introduction

- ✦ Objective
 - ✦ To introduce the Eclipse platform and PTP
- ✦ Contents
 - ✦ New and Improved Features
 - ✦ What is Eclipse?
 - ✦ What is PTP?

What is Eclipse?

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

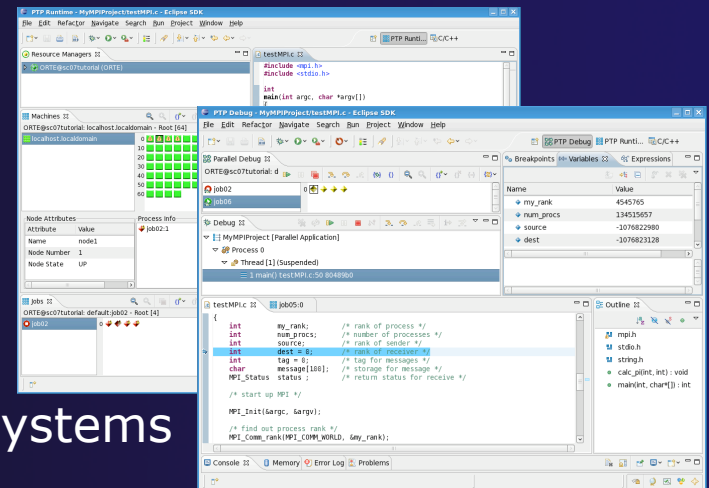


Eclipse Features

- ✦ Full development lifecycle support
- ✦ Revision control integration (CVS, SVN, Git)
- ✦ Project dependency management
- ✦ Incremental building
- ✦ Content assistance
- ✦ Context sensitive help
- ✦ Language sensitive searching
- ✦ Multi-language support
- ✦ Debugging

Parallel Tools Platform (PTP)

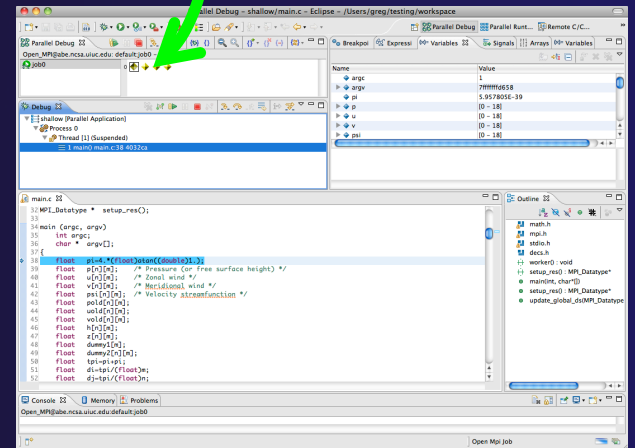
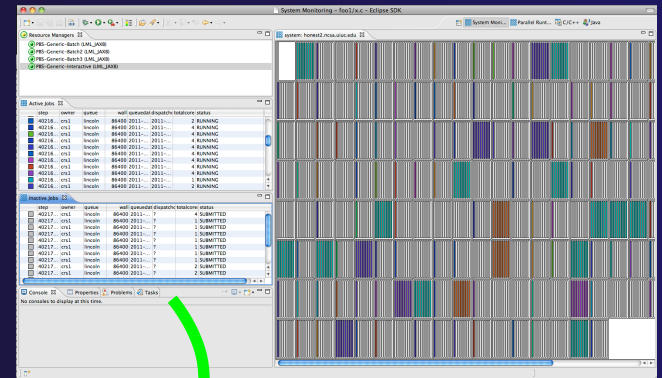
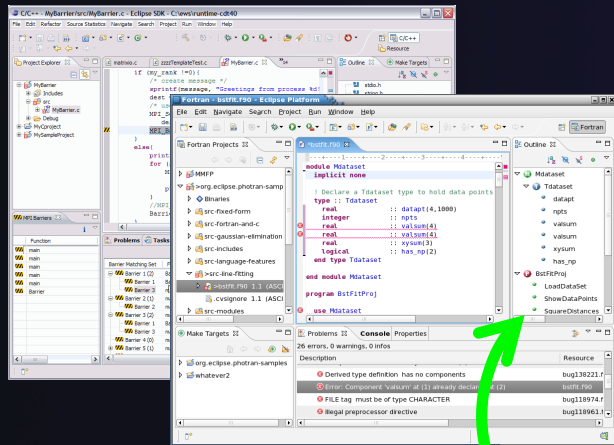
- ★ The Parallel Tools Platform aims to provide a highly integrated environment specifically designed for parallel application development
- ★ Features include:
 - ★ An integrated development environment (IDE) that supports a wide range of parallel architectures and runtime systems
 - ★ A scalable parallel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



Eclipse PTP Family of Tools

Coding & Analysis
(C, C++, Fortran)

Launching & Monitoring

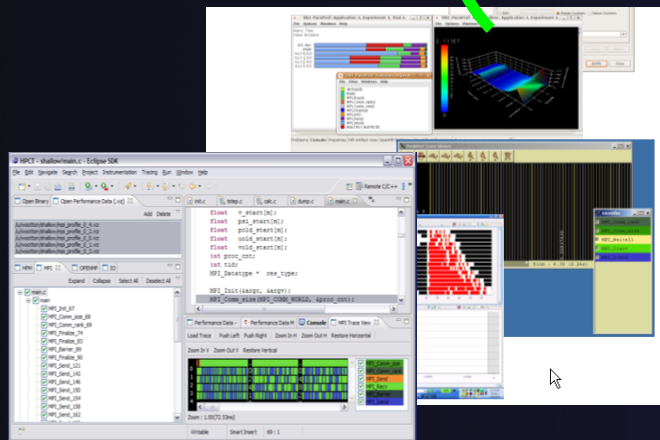


Performance Tuning
(TAU, PerfSuite, ...)

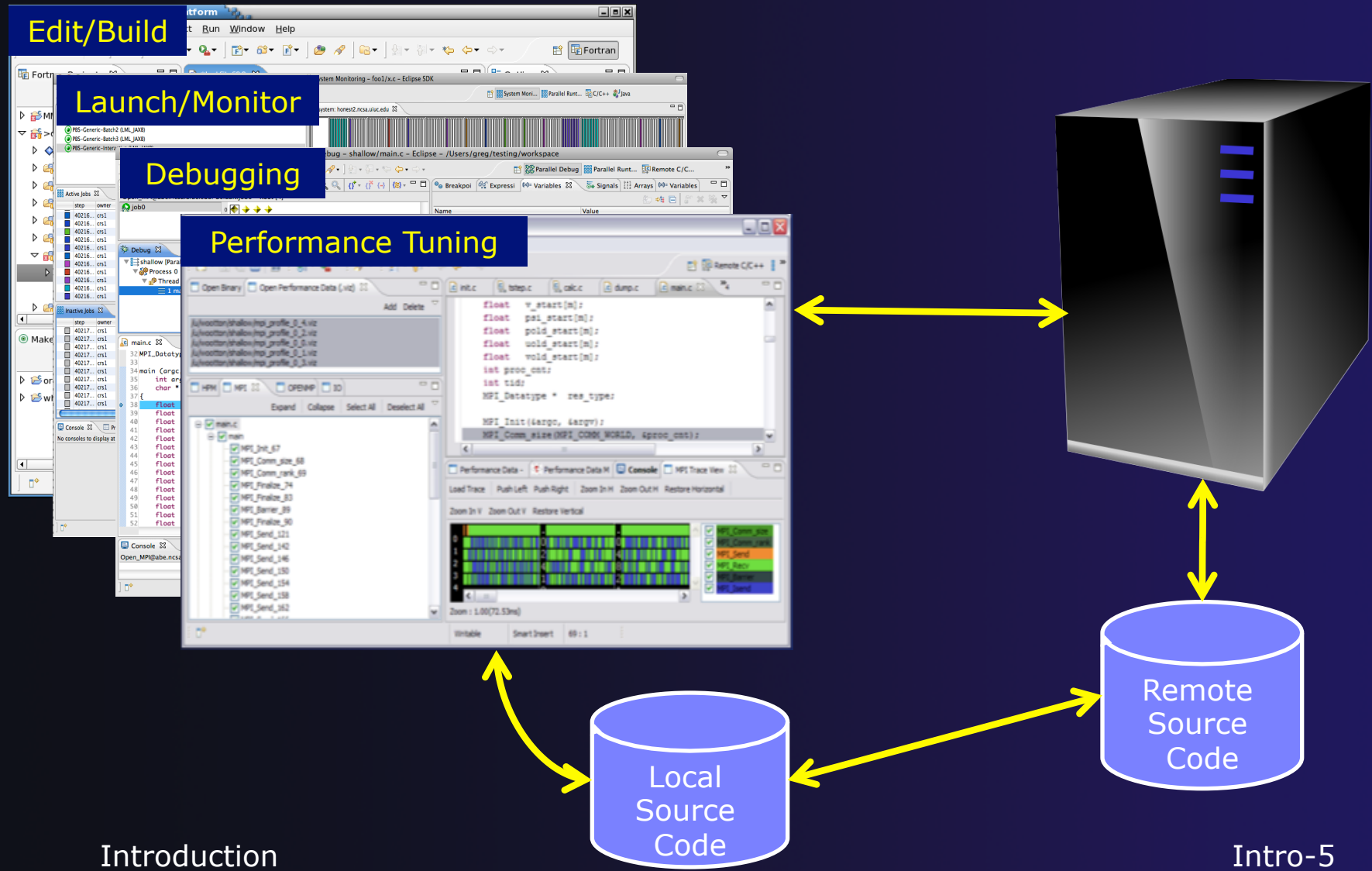
Parallel Debugging

Introduction

Intro-4



How Eclipse is Used



Introduction

Intro-5

Eclipse Basics

★ Objective

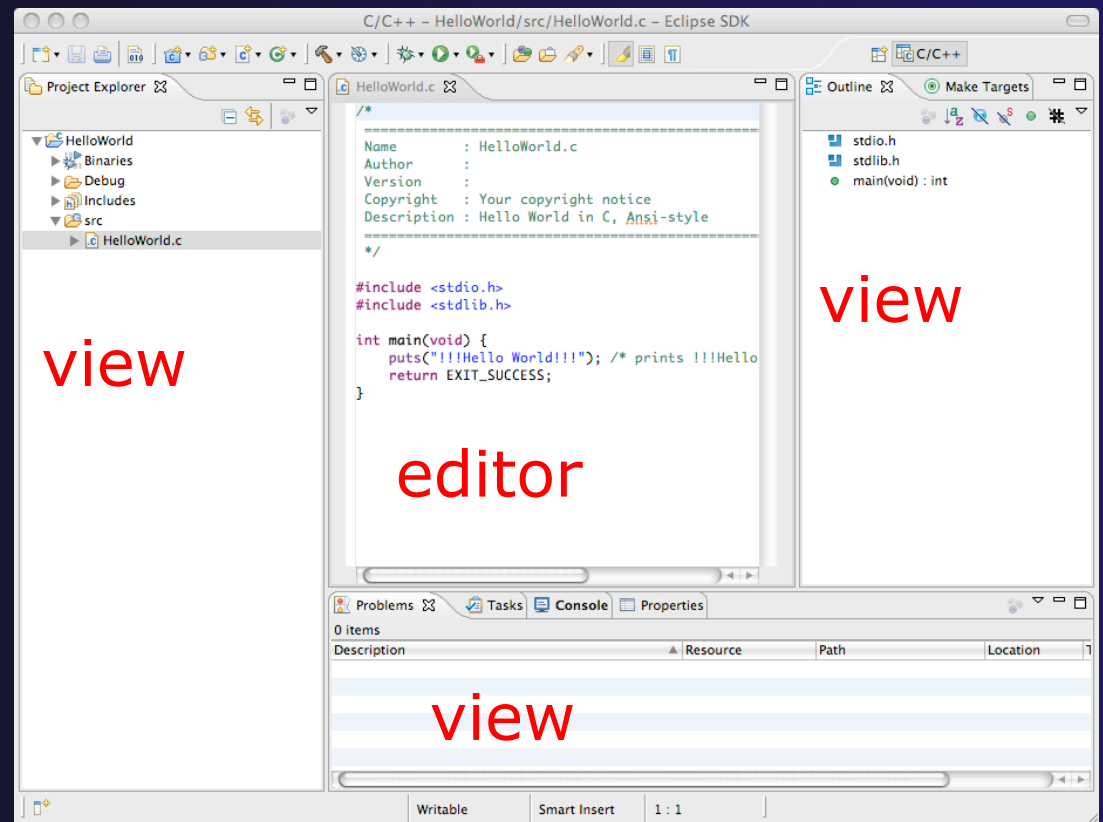
- ★ Learn about basic Eclipse workbench concepts: projects,
- ★ Learn about projects: local, synchronized, remote

★ Contents

- ★ Workbench components: Perspectives, Views, Editors
- ★ Local, remote, and synchronized projects
- ★ Learn how to create and manage a C project
- ★ Learn about Eclipse editing features

Eclipse Basics

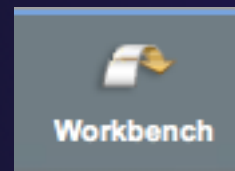
- ★ A *workbench* contains the menus, toolbars, editors and views that make up the main Eclipse window
- ★ The workbench represents the desktop development environment
 - ★ Contains a set of tools for resource mgmt
 - ★ Provides a common way of navigating through the resources
- ★ Multiple workbenches can be opened at the same time
- ★ Only one workbench can be open on a *workspace* at a time



perspective

Perspectives

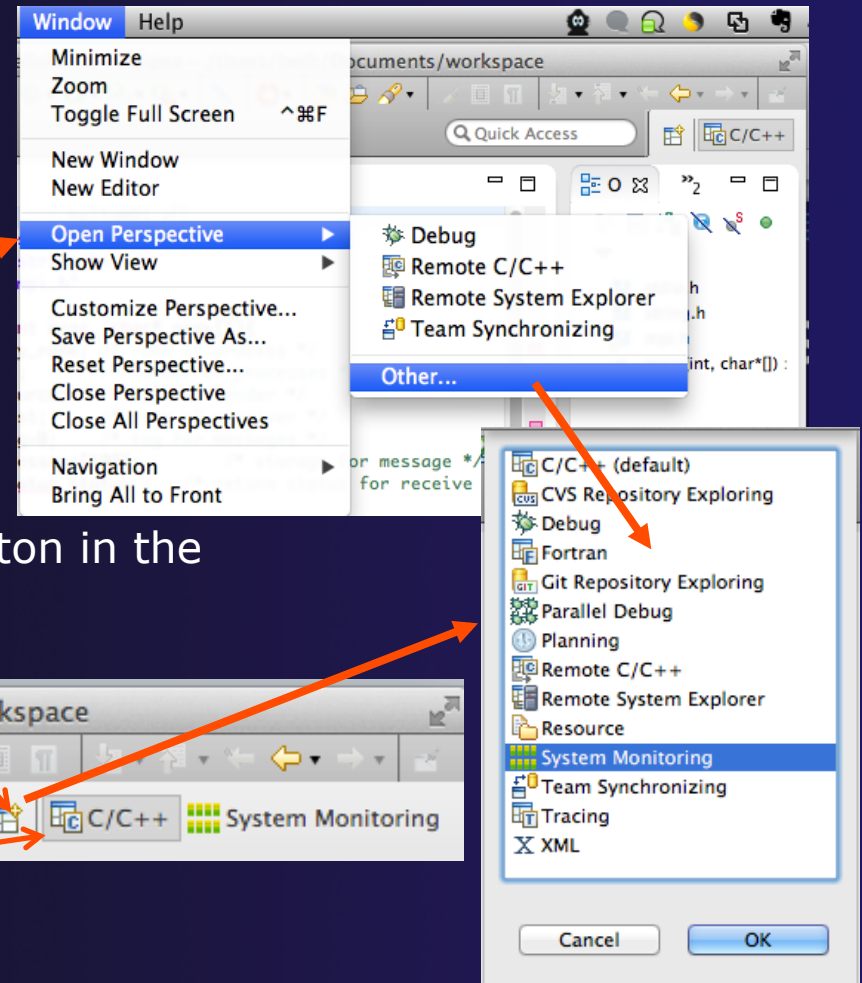
- ✦ Perspectives define the layout of views and editors in the workbench
- ✦ They are *task oriented*, i.e. they contain specific views for doing certain tasks:
 - ✦ **C/C++ Perspective** for manipulating compiled code
 - ✦ **Debug Perspective** for debugging applications
 - ✦ **System Monitoring Perspective** for monitoring jobs
- ✦ You can easily switch between perspectives
- ✦ If you are on the Welcome screen now, select “Go to Workbench” now



Switching Perspectives

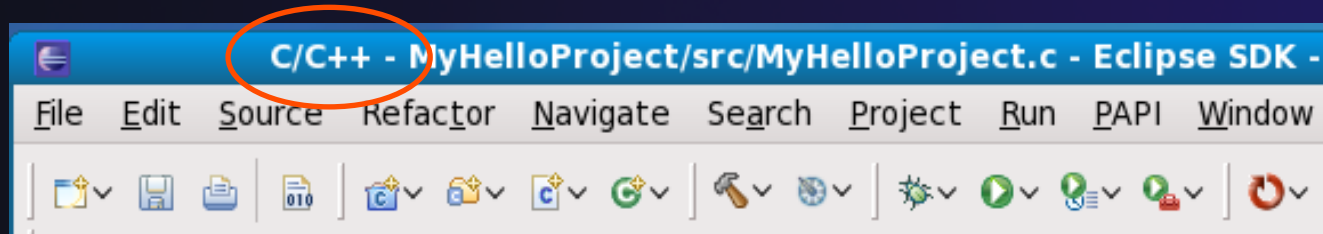
★ Three ways of changing perspectives

1. Choose the **Window>Open Perspective** menu option
Then choose **Other...**
2. Click on the **Open Perspective** button in the upper right corner of screen (hover over it to see names)
3. Click on a perspective shortcut button



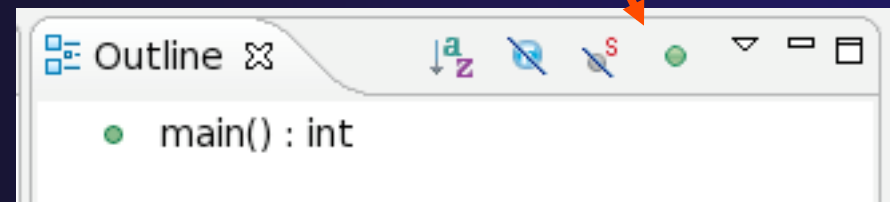
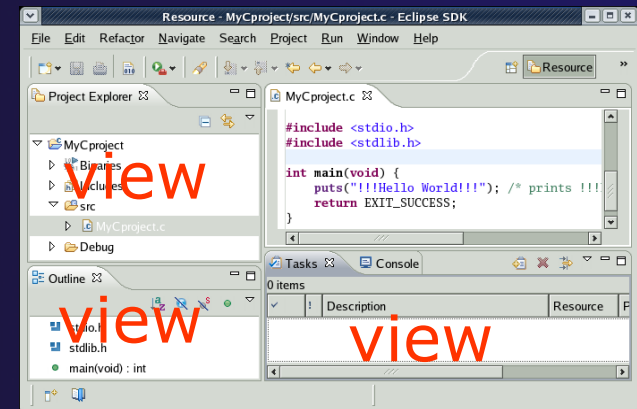
Which Perspective?

- ★ The current perspective is displayed in the title bar



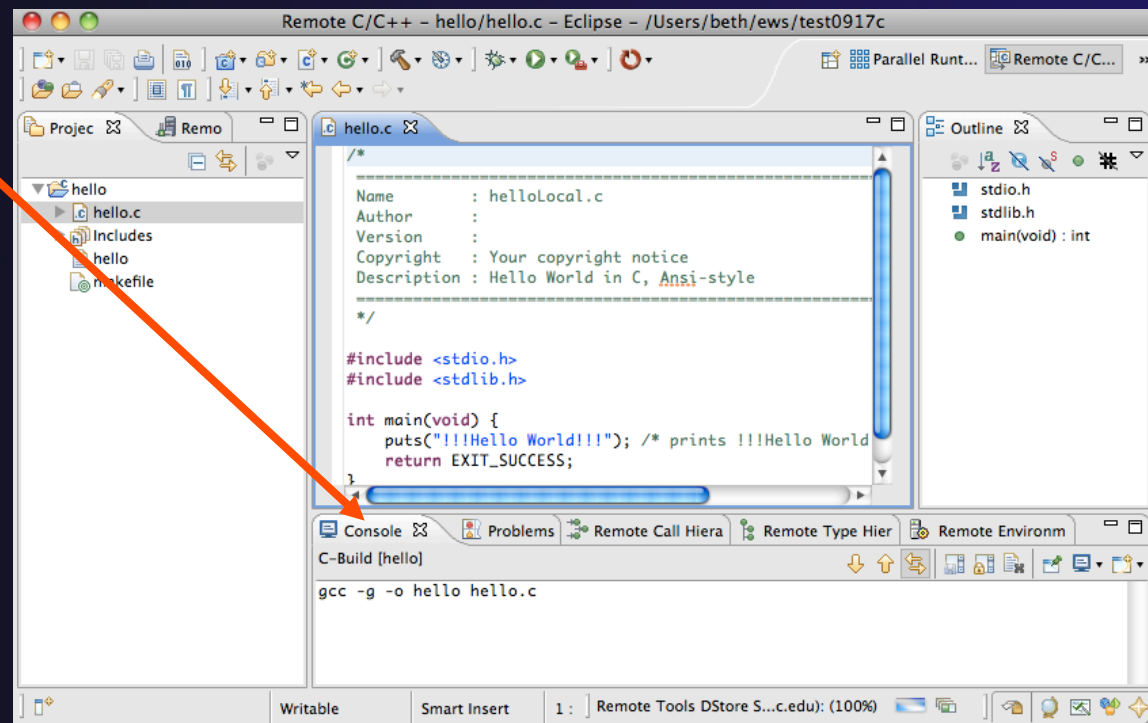
Views

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



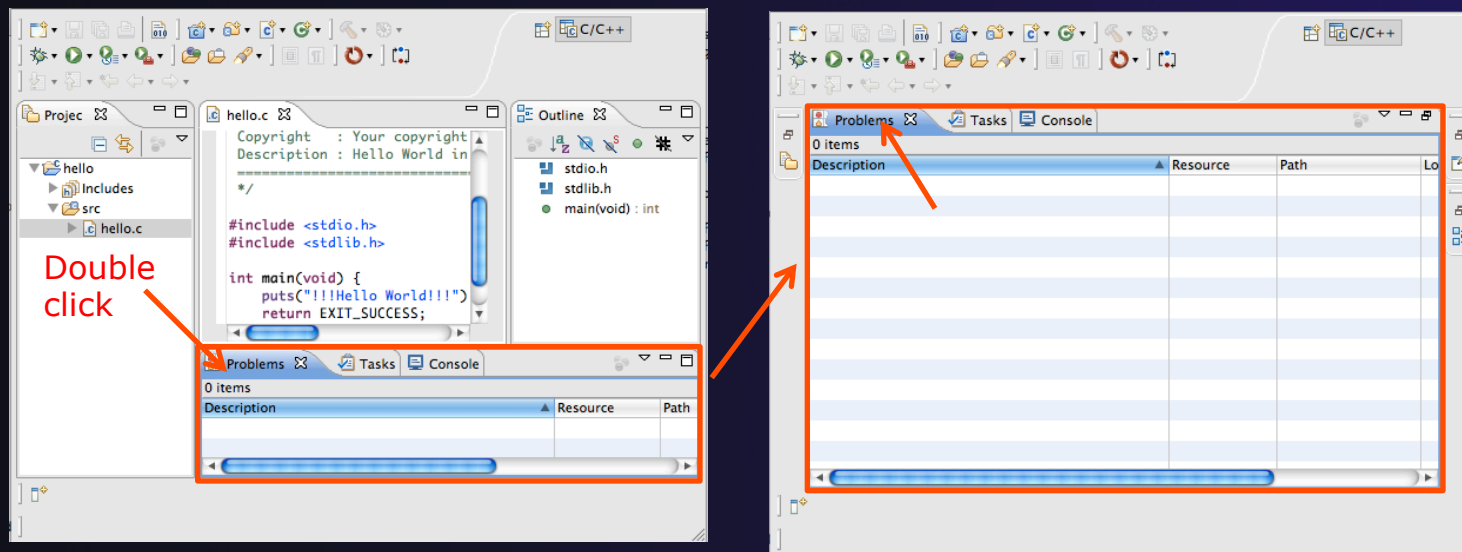
Stacked Views

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



Expand a View

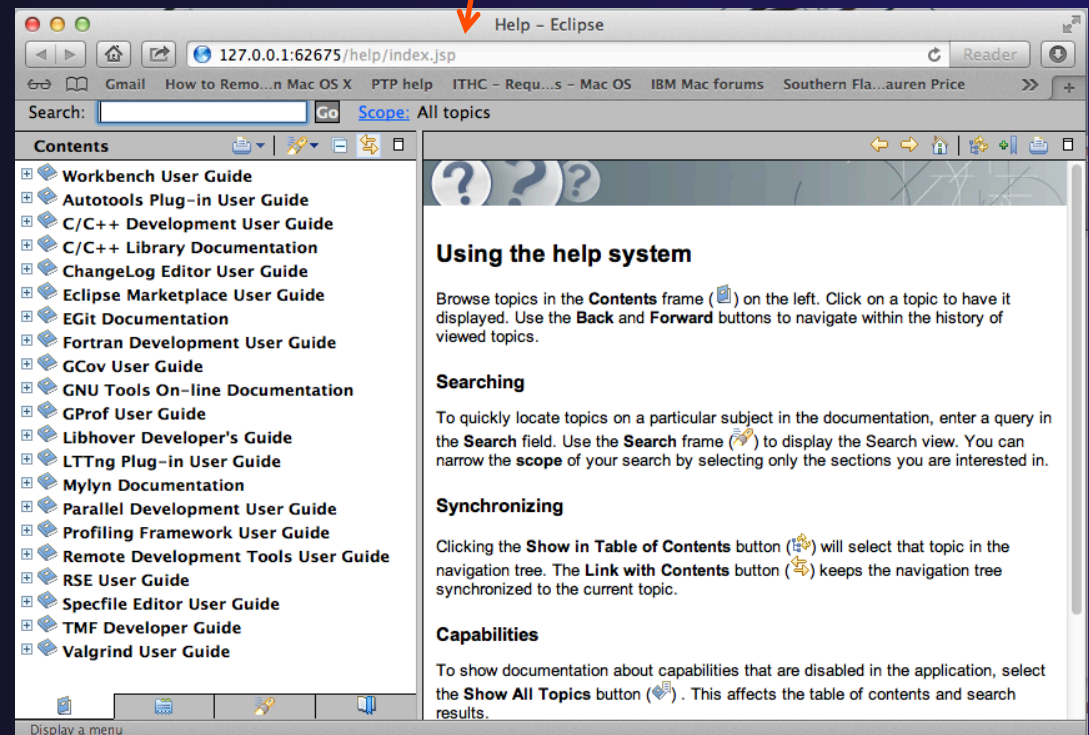
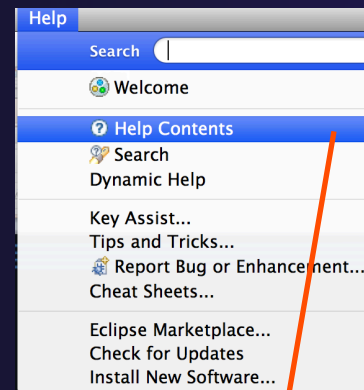
- ★ Double-click on a view/editor's tab to fill the workbench with its content;
- ★ Repeat to return to original size



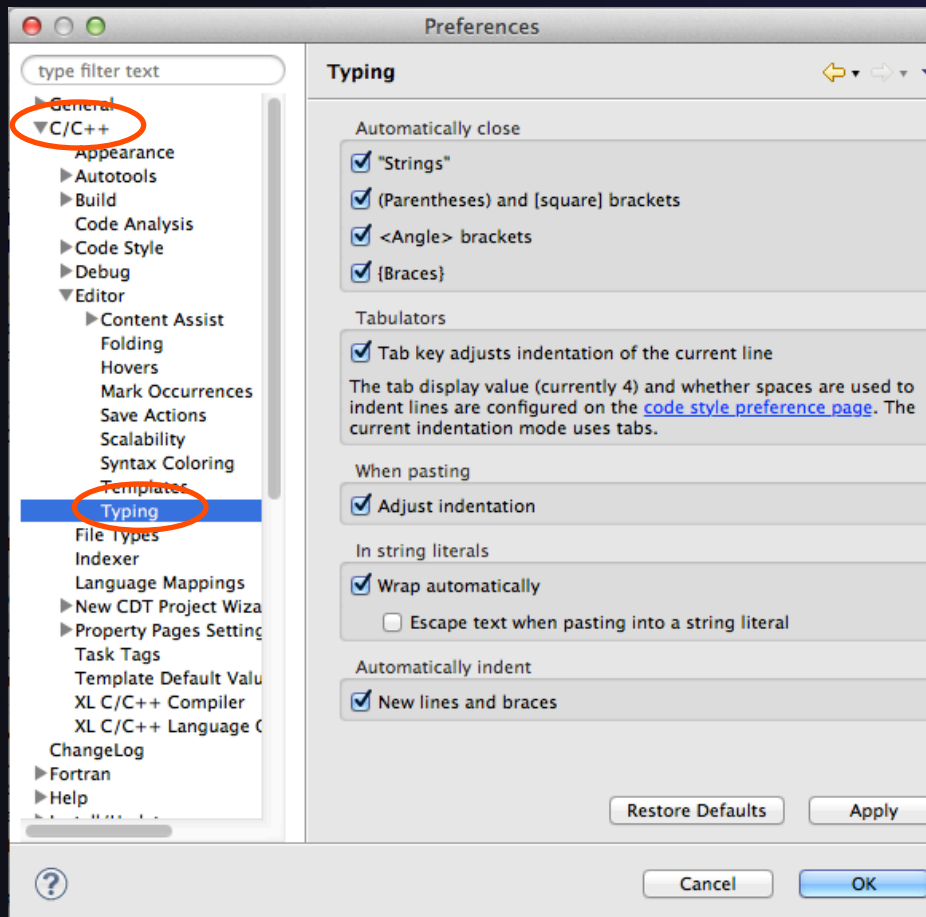
- ★ Window > Reset Perspective returns everything to original positions

Help

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

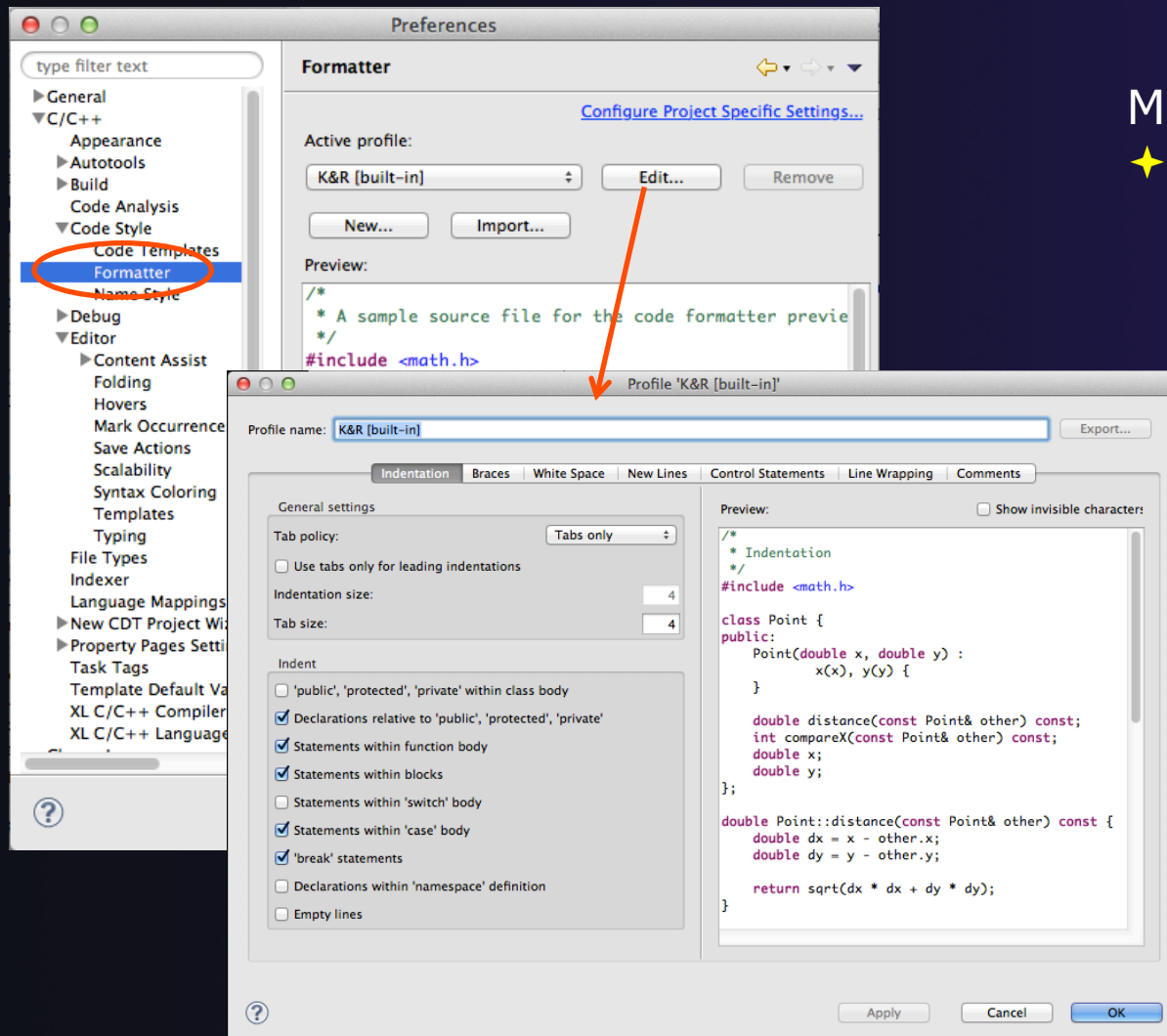


Eclipse Preferences



- ✦ Eclipse Preferences allow customization of almost everything
- ✦ To open use
 - ✦ Mac: **Eclipse>Preferences...**
 - ✦ Others: **Window>Preferences...**
- ✦ The C/C++ preferences allow many options to be altered
- ✦ In this example you can adjust what happens in the editor as you type.

Preferences Example



More C/C++ preferences:

- ✦ In this example the Code Style preferences are shown
- ✦ These allow code to be automatically formatted in different ways



Exercise

1. Change to a different perspective
2. Experiment with moving and resizing views
 - ✦ Move a view from a stack to beside another view
 - ✦ Expand a view to maximize it; return to original size
3. Save the perspective
4. Reset the perspective
5. Open Eclipse preferences
6. Search for “Launching”
7. Make sure the “Build (if required) before launching” setting is *disabled*



Optional Exercise

Best performed *after* learning about projects, CVS, and editors

1. Use source code formatting to format a source file, or a region of a source file
 - ✦ Use Source>Format menu
2. In Eclipse Preferences, change the C/C++ source code style formatter, e.g.
 - ✦ Change the indentation from 4 to 6
 - ✦ Make line wrapping not take effect until a line has a maximum line width of 120, instead of the default 80
 - ✦ Save a (new) profile with these settings
 - ✦ Format a source file with these settings
3. Revert the file back to the original – experiment with
 - ✦ Replace with HEAD, replace with previous from local history, or reformat using original style

Creating a Synchronized Project

★ Objective

- ★ Learn how to create and use synchronized projects
- ★ Learn how to create a sync project
 - ★ From a source code repository in Git

★ Contents

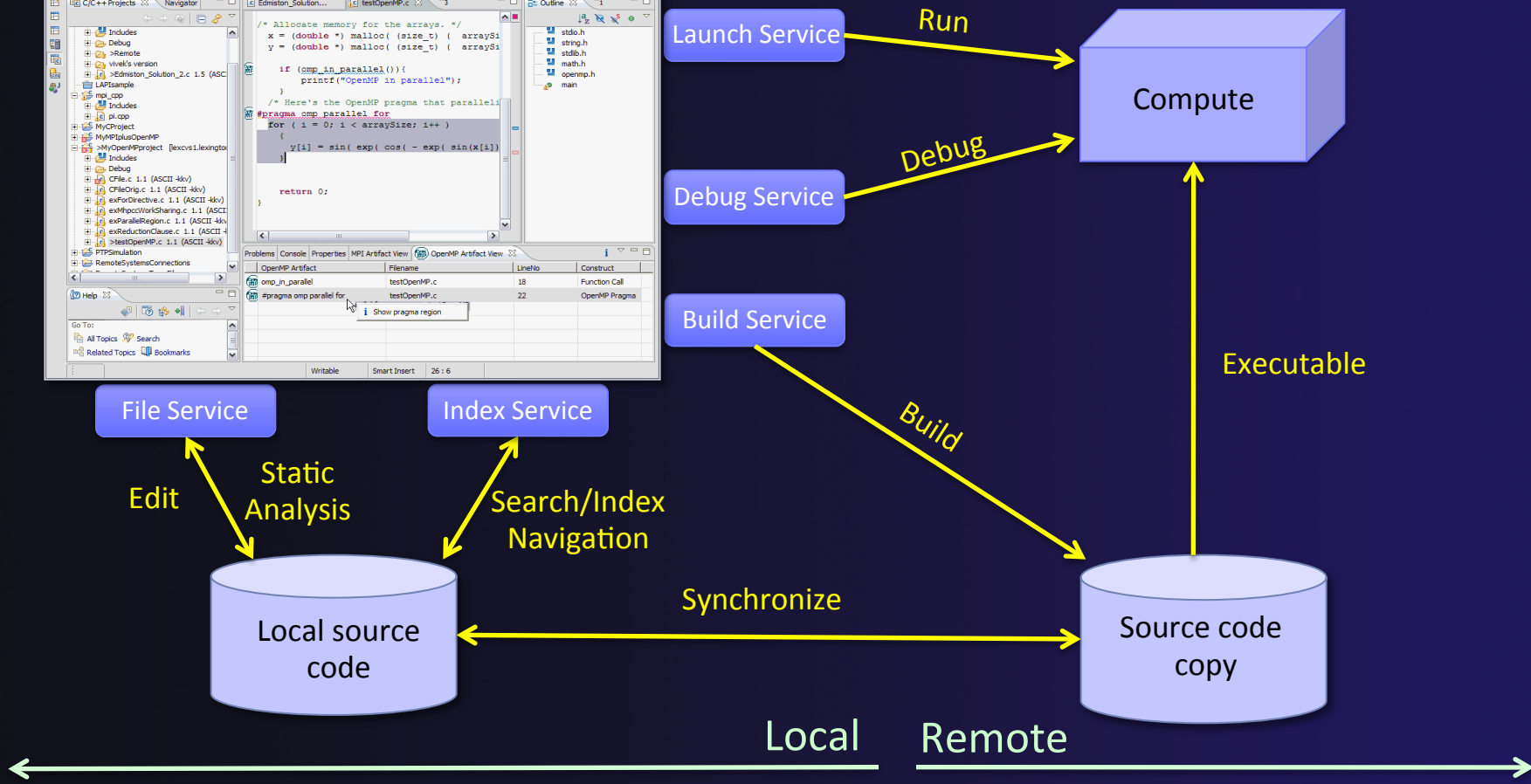
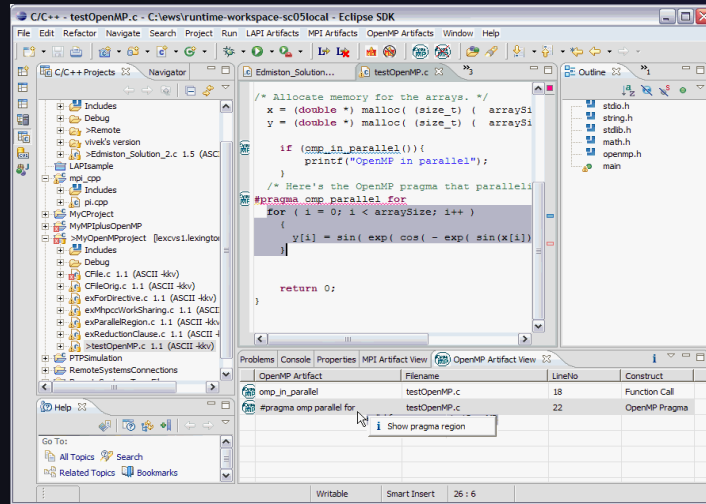
- ★ Eclipse project types
- ★ Clone a git repository; create a synchronized project

- ★ Using synchronize filters
- ★ Remote Terminal view
- ★ Converting an existing project to synchronized

Project Location

- ★ Local
 - ★ Source is located on local machine, builds happen locally
 - ★ This is the default Eclipse model
- ★ Synchronized
 - ★ Source is located on both local and remote machine(s), then kept in synchronization by Eclipse
 - ★ Building and launching happens remotely (can also happen locally)
 - ★ Used mainly for scientific and supercomputing applications
- ★ There are also remote-only projects, but these have limitations and are not covered here

Synchronized Projects



Synchronized Projects

Sync-2

Revision Control Systems

(Source Code Repositories)

- ★ Eclipse supports a range of revision control systems, such as CVS, Git, and Subversion (and others)
- ★ These are distinct from synchronized projects
- ★ Revision control systems can be used in conjunction with synchronized projects
- ★ Synchronized projects are typically *not* used for revision control

Synchronized Project Creation

★ Local -> Remote

- ★ Projects start out local then are synchronized to a remote machine
- ★ Three options
 - ★ Created from scratch
 - ★ Imported from local filesystem
 - ★ Imported from source code repository (Git) <- this tutorial

★ Remote -> Local

- ★ Projects start out on remote machine then are synchronized to the local system
- ★ Two options
 - ★ Already on remote system
 - ★ Checked out from source code repository

C, C++, and Fortran Projects

Build types

- ★ Makefile-based
 - ★ Project contains its own build command – typically a makefile (or makefiles) for building the application – but can be any build scripts, etc.
- ★ Managed
 - ★ Eclipse manages the build process, no makefile required by the user

Check out source code from Git repository

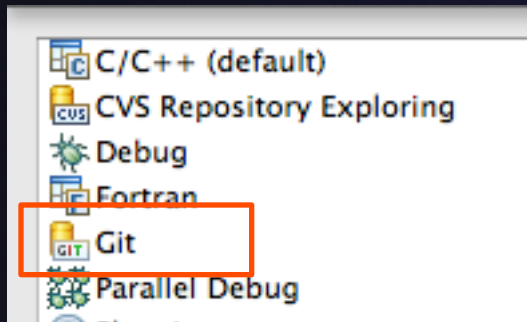
Create Synchronized project on the local machine
at the same time.

Two steps:

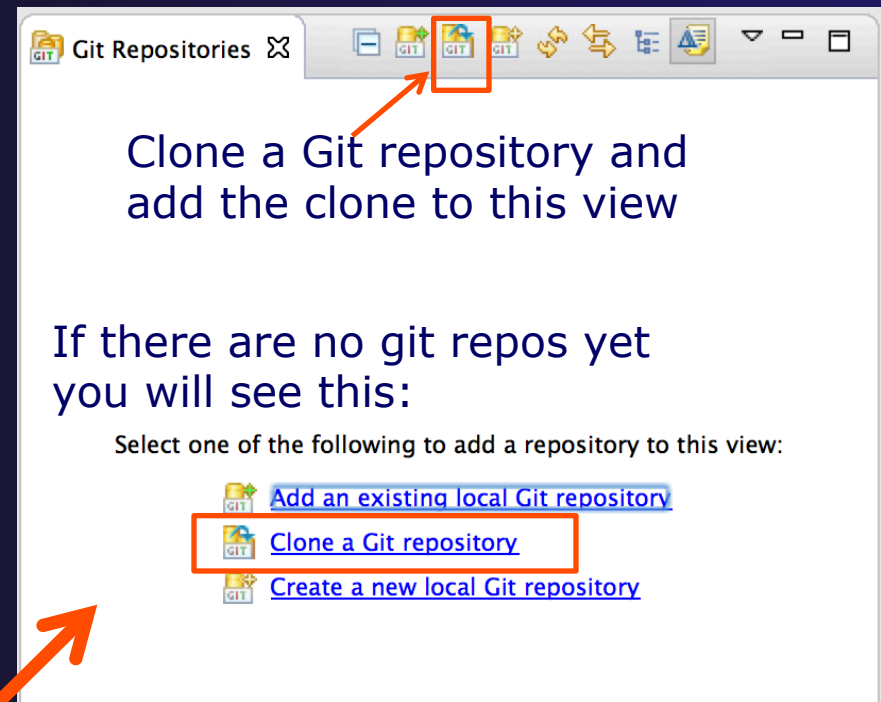
- ★ Clone Git Repo
- ★ Create project files from within the clone

Clone the git repo

- ✦ Open Git perspective
 - ✦ Window > Open Perspective > Other
- ✦ Select Git



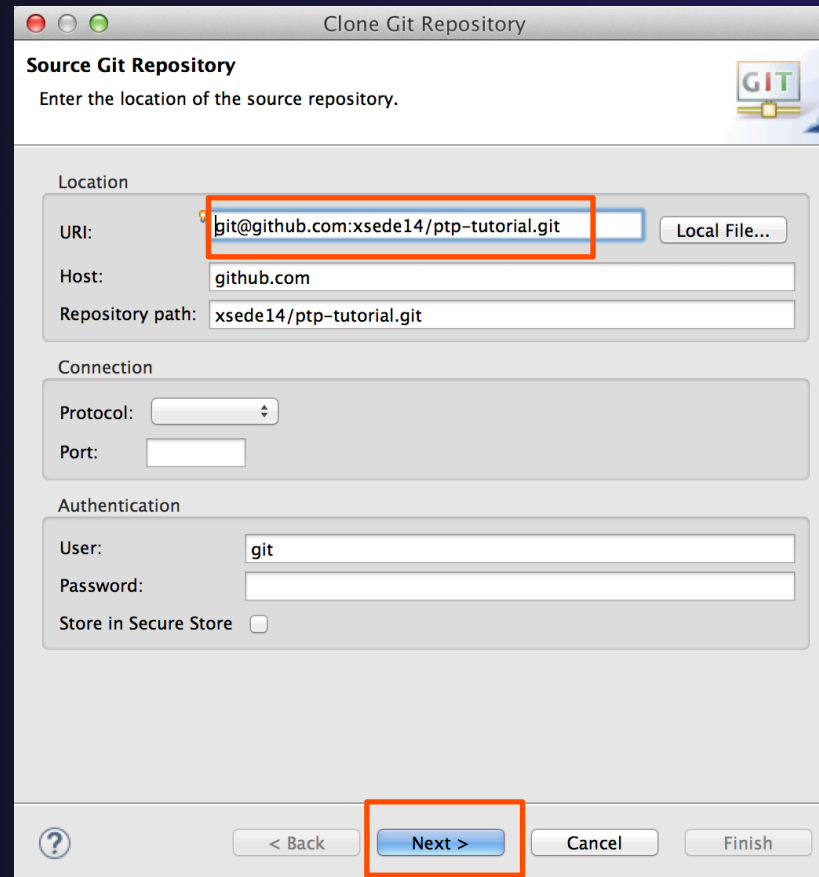
- ✦ In the view, select **Clone a Git repository** one of two ways



Specify remote git repo location

- ★ URI: <git@github.com:xsede14/ptp-tutorial.git>
- ★ If this doesn't work for you, try:
<https://github.com/xsede14/ptp-tutorial.git>

- ★ Fill in URI and other fields fill themselves
- ★ Select **Next>**



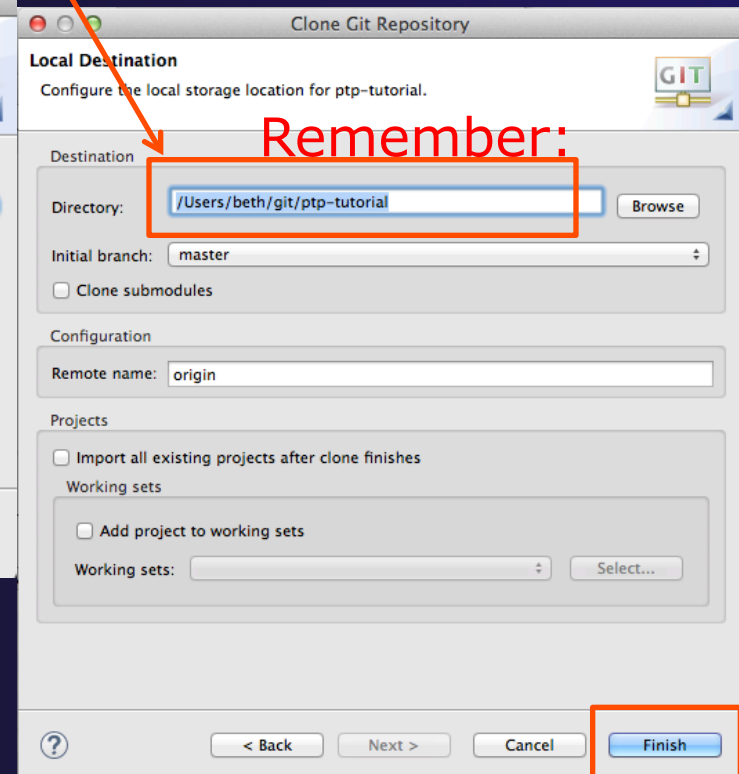
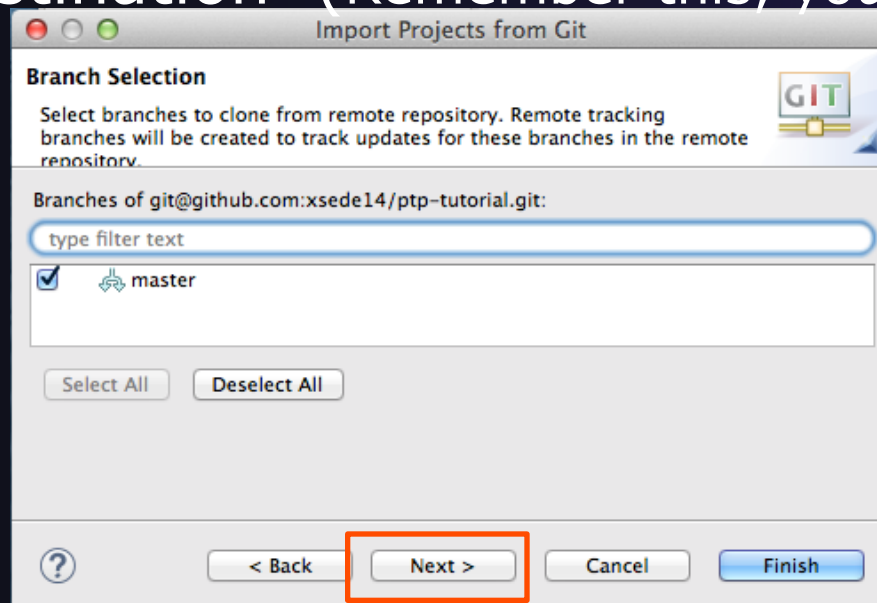
The screenshot shows a 'Clone Git Repository' dialog box with the following fields and values:

- Location:**
 - URI: `git@github.com:xsede14/ptp-tutorial.git` (highlighted with a red box)
 - Host: `github.com`
 - Repository path: `xsede14/ptp-tutorial.git`
- Connection:**
 - Protocol: (dropdown menu)
 - Port: (text field)
- Authentication:**
 - User: `git`
 - Password: (text field)
 - Store in Secure Store:

At the bottom, the **Next >** button is highlighted with a red box.

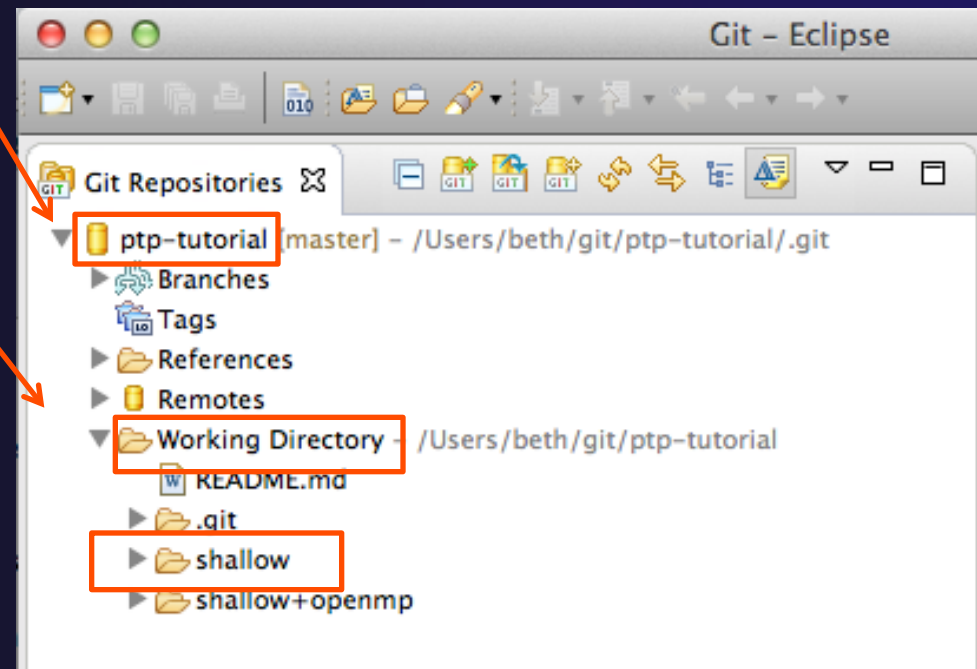
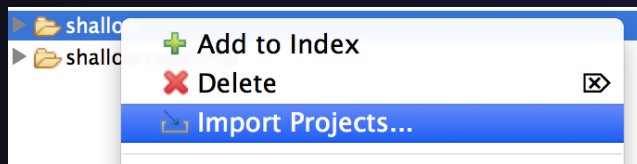
Finish git cloning

- ★ Select **Next>** to choose the (only) branch
- ★ Then select **Finish>** to use the default git destination (Remember this, you'll need it later)



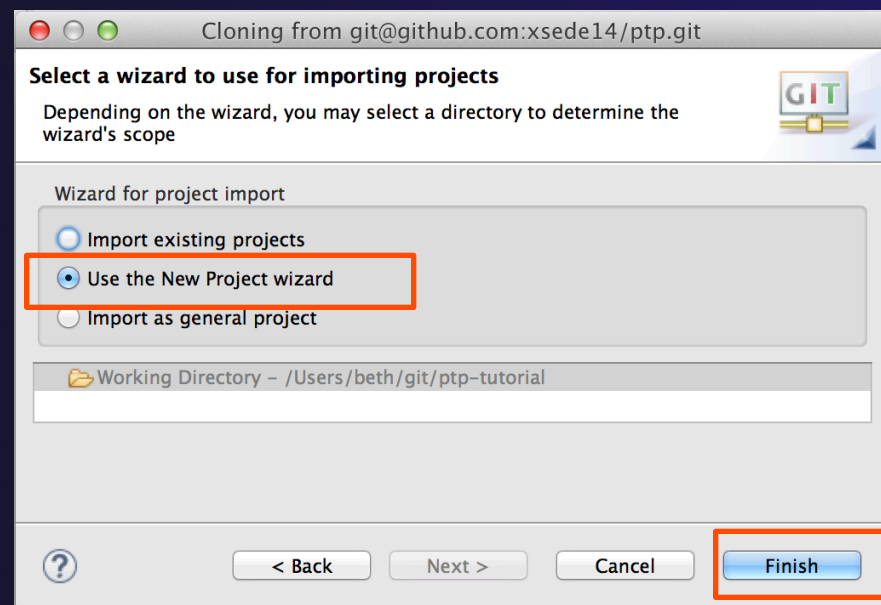
Import project from cloned repo

- ★ After repo is cloned, expand ptp-tutorial and Working Directory
- ★ We are importing only one project
- ★ Select **shallow**
- ★ Right mouse, **Import Projects...**



Create new project with wizard

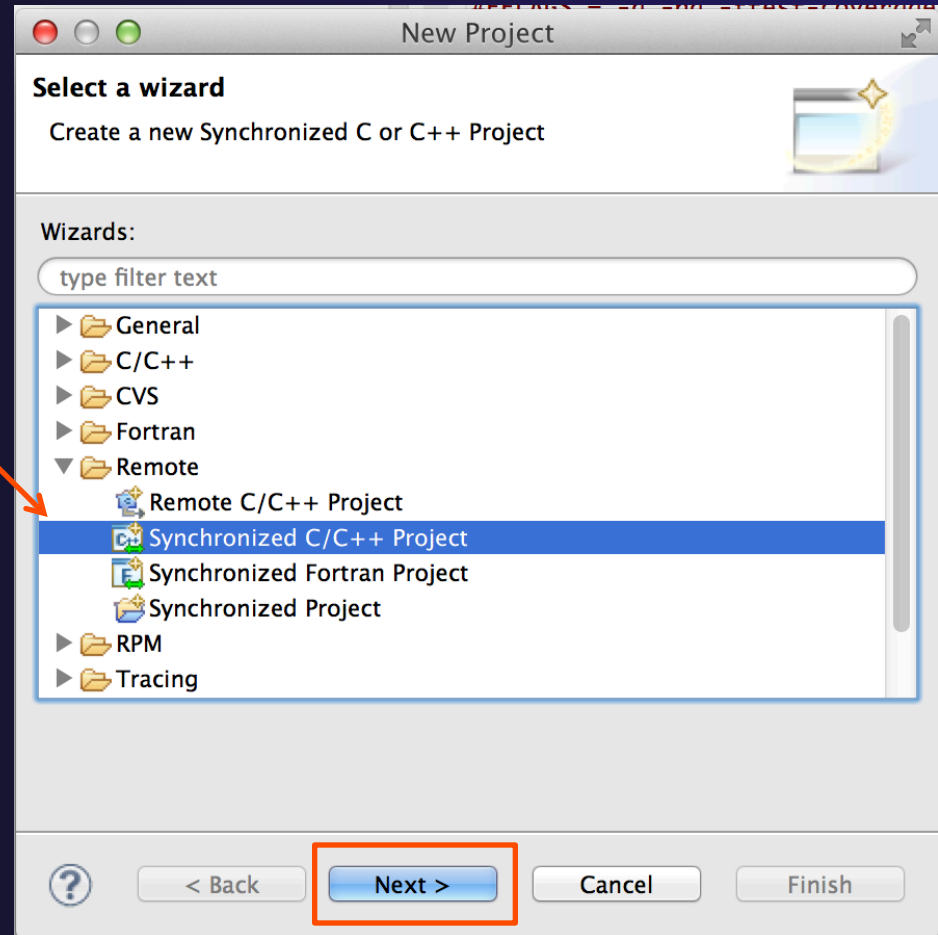
- ★ Select **Use the New Project Wizard** to be able to create the project as a Synchronized C/C++ project at creation
- ★ Select **Finish** to finish the git cloning, and you will be taken to Sync project info next.



New Project Wizard

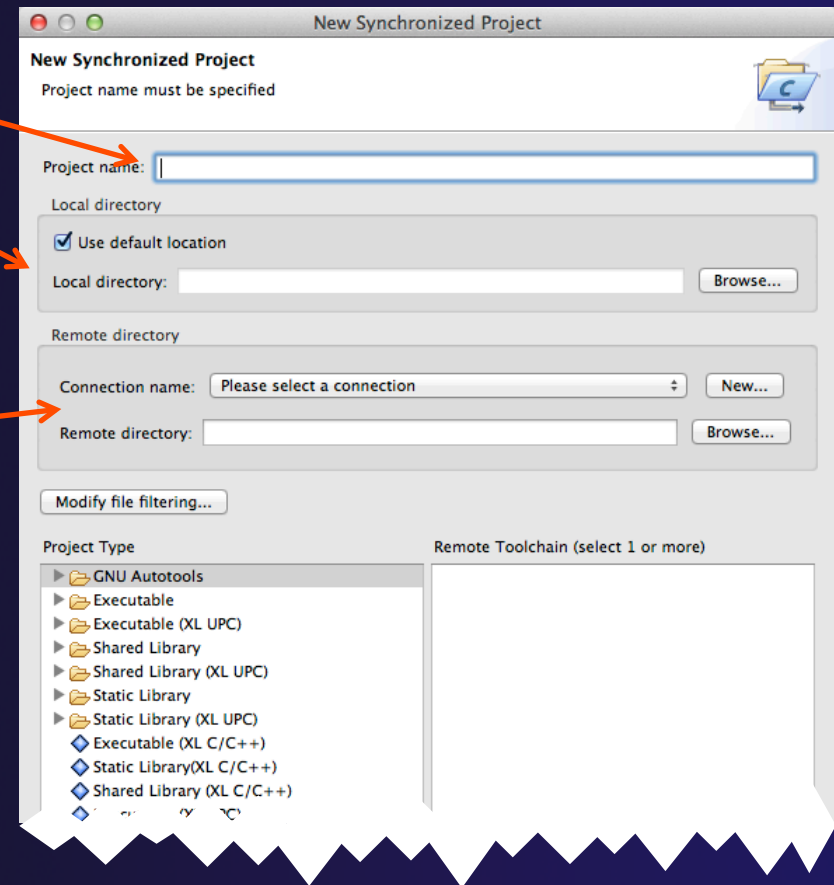
We are creating the project directly as a Synchronized C/C++ project

- ✦ Expand **Remote**
- ✦ Select **Synchronized C/C++ Project**
- ✦ Select **Next>**



New Synchronized Project Wizard

- ✦ Enter the **Project Name**
 - ✦ E.g. “shallow”
- ✦ Next we will specify the **Local Directory** where the local files are located (cloned from git)
 - ✦ Files are synchronized here, and we will edit them locally
- ✦ ...and the **Remote Directory** where the remote files are located
 - ✦ Our remote target machine, where we will build, run, & debug
- ✦ Use **Modify File Filtering...** if required (see later slide)



See Next slides...

Local and remote directories

1. For Local directory,
NOTE: Uncheck **Use default location**
and browse to the location you chose for git repo
- **the shallow dir beneath that**
2. To specify the Remote directory, first Create a connection to the remote target machine by selecting **New...**

New Synchronized Project

Please select a project type to continue

Project name:

Local directory

Use default location

Local directory:

Remote directory

Connection name:

Remote directory:

Creating a Connection

- ★ In the **New Connection** dialog
 - ★ Enter a **Connection name** for the remote host
 - ★ Enter host name, user name, and user password or other credentials
 - ★ Select **Finish**

New Connection
Please enter a name for the connection

Connection name:

Host information

Host:

User:

Password based authentication

Password:

Public key based authentication

File with private key:

Passphrase:

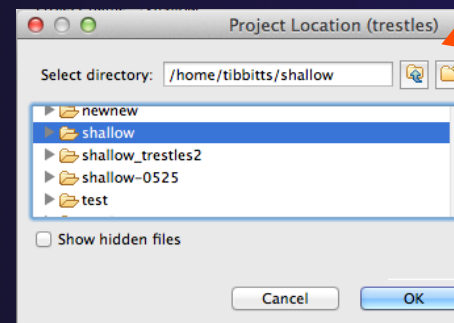
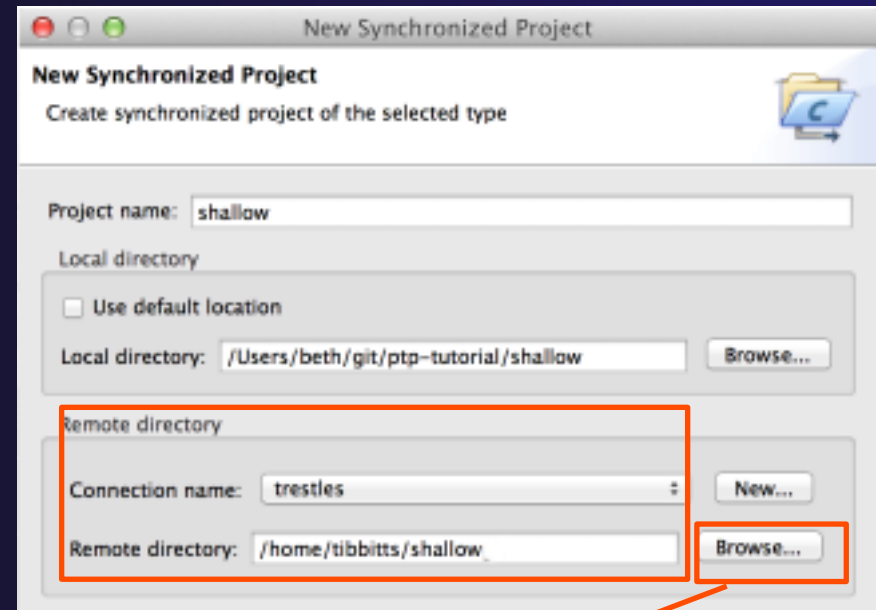
Specifying the remote directory

★ After the connection has been specified, back in the **New Synchronized Project** window..

★ For Remote directory, you can enter its location. If it does not exist, it will be created.

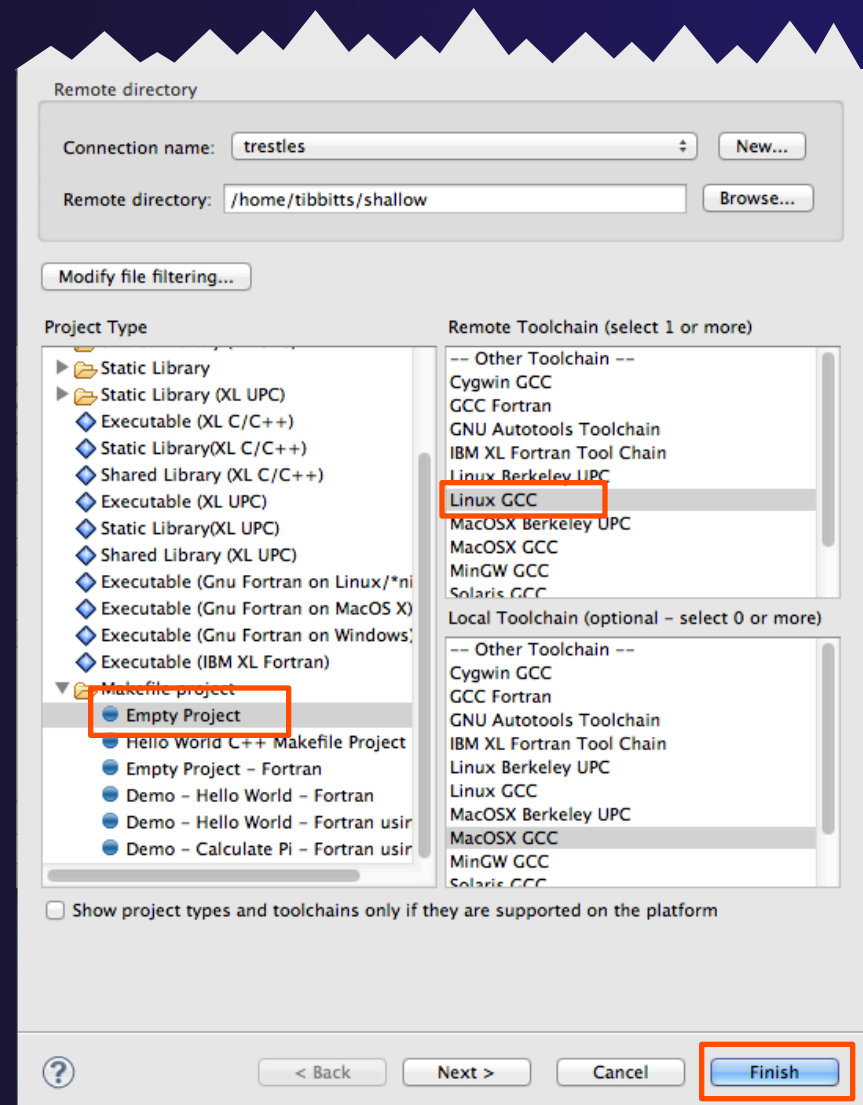
★ If the remote dir exists, you can select it with the **Browse...** Note that this is the first time that the Connection information is utilized.

★ *Later slides in this section show how to fix Connection if e.g. password or userid are entered incorrectly*



Project Type & Toolchain

- ★ Choose the **Project Type**
 - ★ This tutorial's code has its own makefile, so use **Makefile Project>Empty Project**
 - ★ Otherwise, choose the type of project you want to create
- ★ Choose toolchain for remote build
 - ★ Use a toolchain that most closely matches the remote system
- ★ Choose a toolchain for the local build (OPTIONAL)
 - ★ This is optional if you don't plan to build on the local machine
 - ★ This is used for advanced editing/searching
- ★ Click **Finish** to create the project



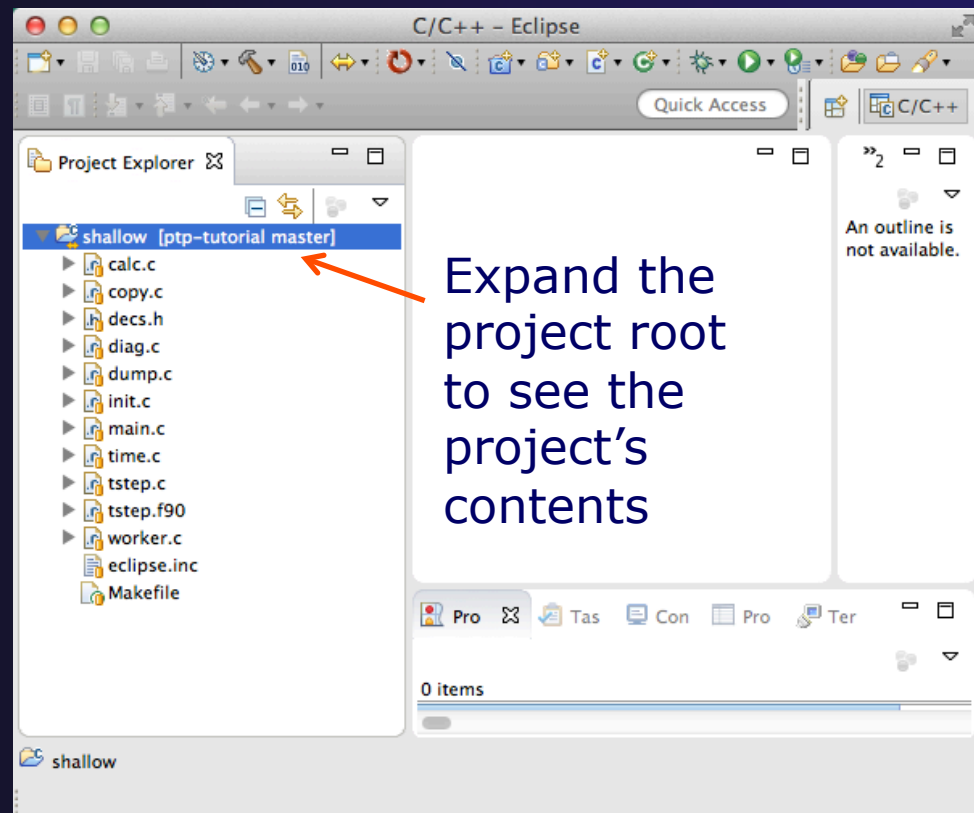
Project successfully created



- ★ You should now see the “shallow” project in your workspace
- ★ Project is synchronized with remote host

Status area in lower right shows Synchronization progress:

Remote Synchronization: (19%)






Synchronized Project

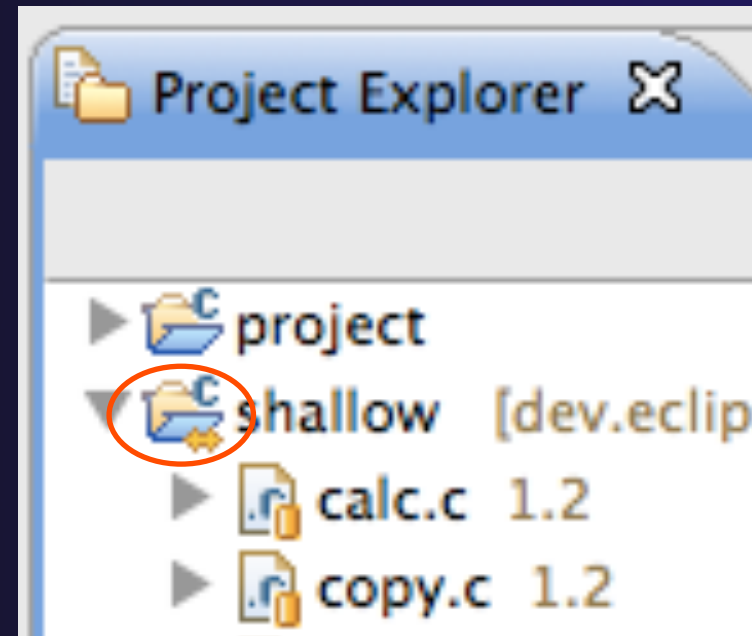
- ★ Back in the Project Explorer, decorator on project icon indicates synchronized project
- ★ Double-+ icon

- ★ C Project w/o Sync

▼  shallow [dev.eclipse.org]

- ★ Synchronized Project

▼  shallow [dev.eclipse.org]

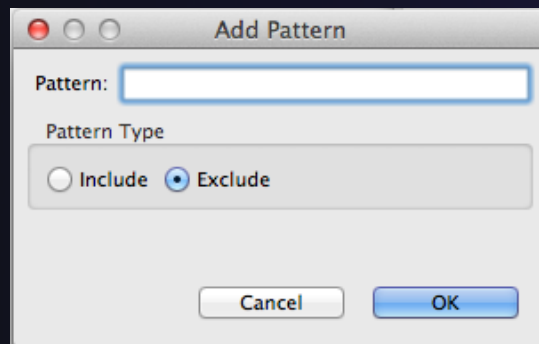


Synchronize Filters

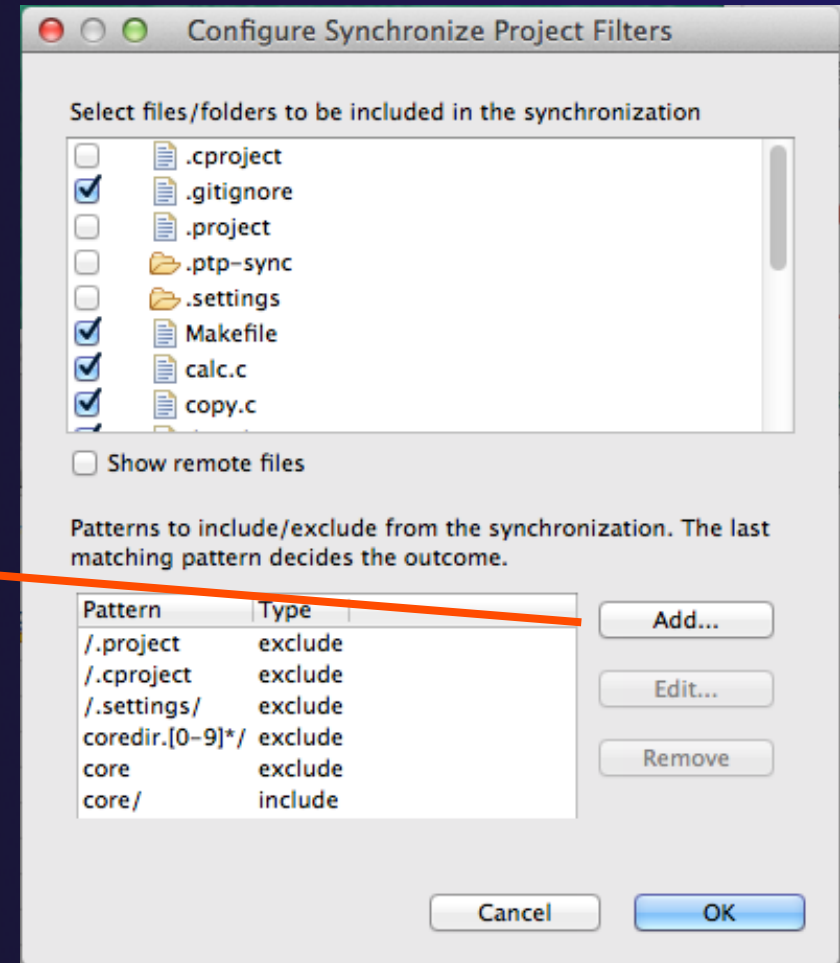
- ✦ If not all files in the remote project should be synchronized, a filter can be set up
 - ✦ For example, it may not be desirable to synchronize binary files, or large data files
- ✦ Filters can be created at the same time as the project is created
 - ✦ Click on the **Modify File Filtering...** button in the New Project wizard
- ✦ Filters can be added later
 - ✦ Right click on the project and select **Synchronize>Filter...**

Synchronize Filter Dialog

- ★ Files can be filtered individually by selecting/unselecting them in the **File View** at the top
- ★ Include or exclude files based on paths and expressions

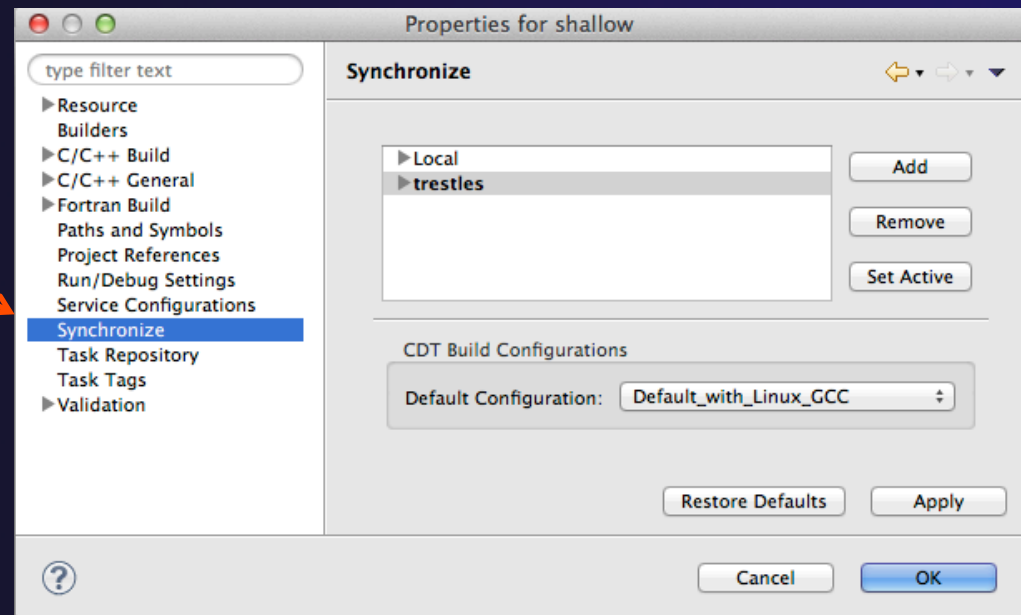


- ★ Suggestion: add filter for 'shallow' so the executable, built on remote machine, doesn't get synced back



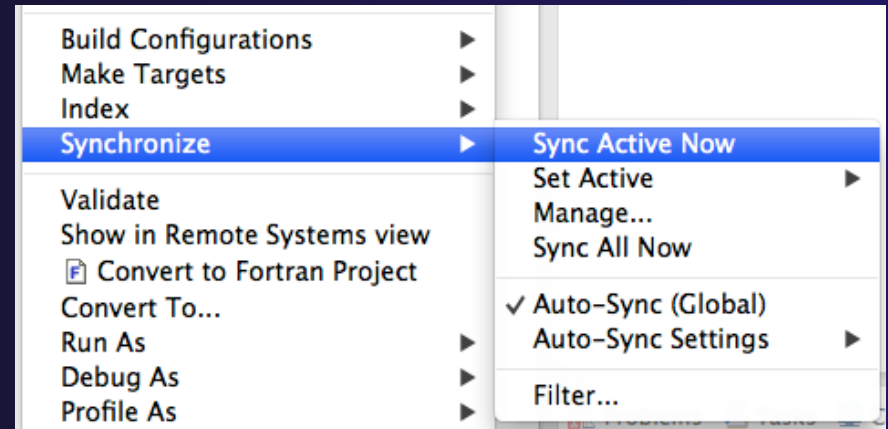
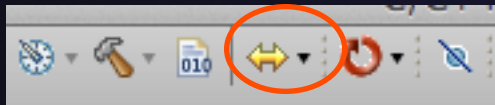
Synchronized Project Properties

- ★ Synchronized configurations can be managed through the project properties
- ★ Open the project properties by right-clicking on the project and selecting **Properties**
 - ★ Select **Synchronize**
- ★ This is the same as using the **Synchronize>Manage...** menu



Forcing a Resync

- ★ If Auto-sync is set, the project should automatically resync with remote system when things change (e.g. after build)
- ★ Sometimes you may need to do it explicitly
- ★ Right click on project and select **Synchronization>Sync Active Now**
 - or use the toolbar icon

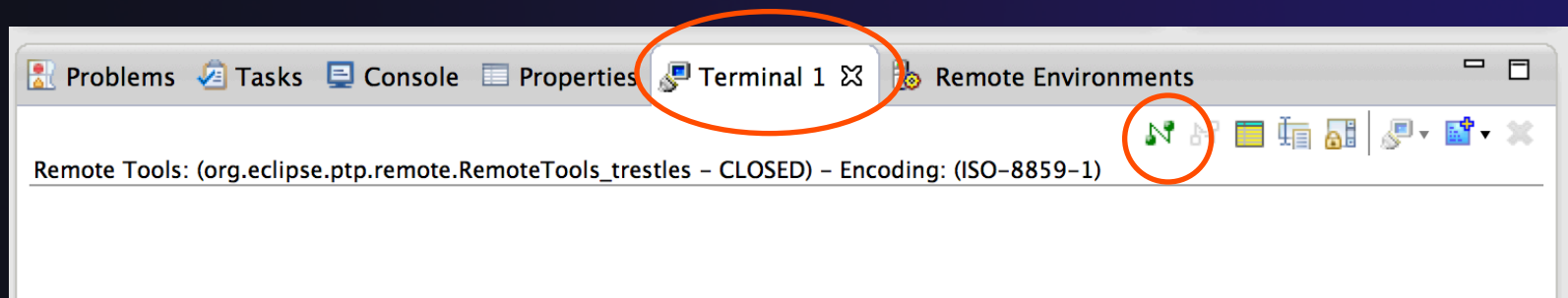
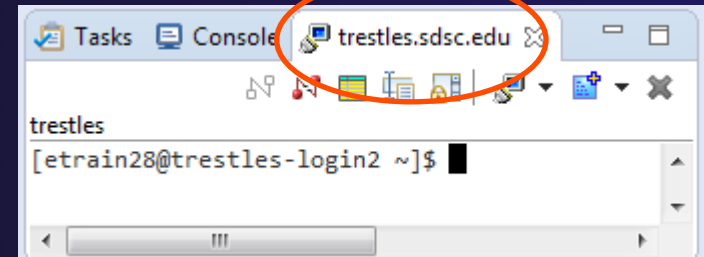


- ★ Status area in lower right shows when Synchronization occurs



Remote Terminal

- ✦ There is a remote terminal that can provide a shell from within Eclipse using the connection you created for your synchronized project
- ✦ Right-Click on your synchronized project and select "Show Terminal"
Or
- ✦ If view is not in your workbench:
 - ✦ Select **Window>Show View>Other...**
 - ✦ Choose **Terminal** from the Terminal folder
- ✦ In the **Terminal** view, click on the **Connect** button
- ✦ It will use the previously configured connection from the dropdown, or create a new one *...more in Advanced Features section...*



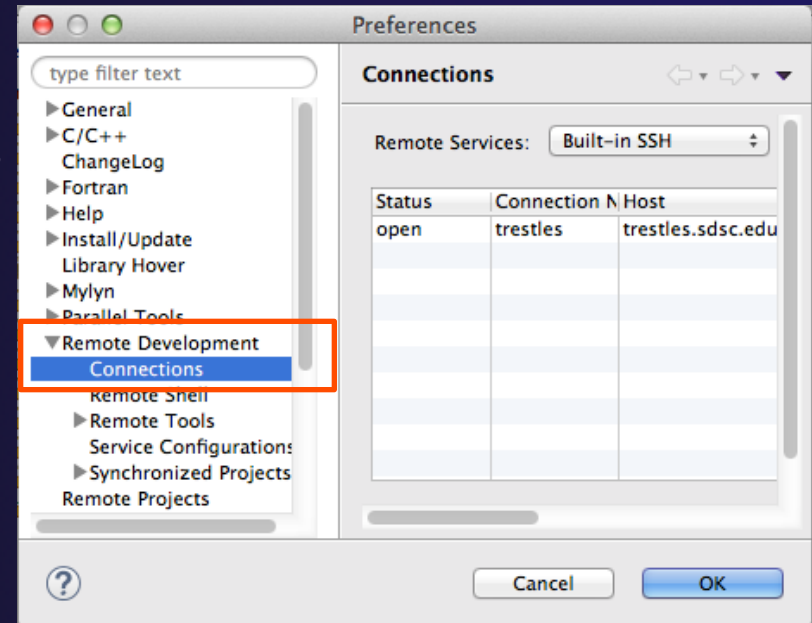
Changing Remote Connection Information

★ If you need to change remote connection information (such as username or password), open **Preferences**

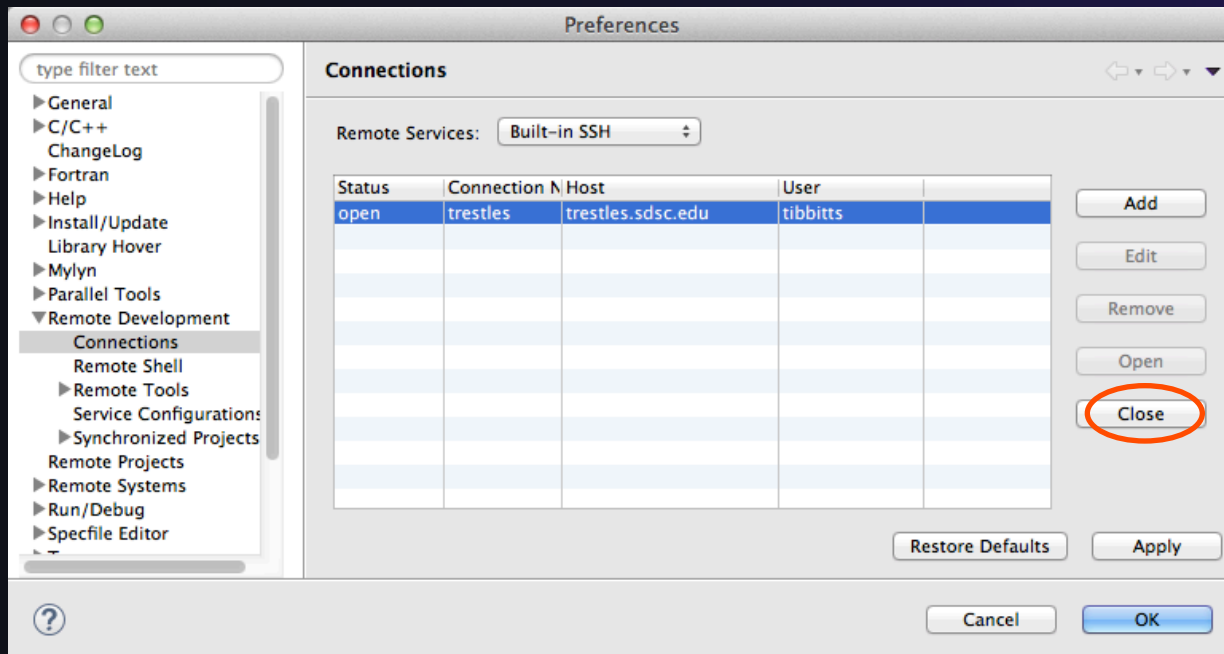
★ **Win/Linux: Window > Preferences**

★ **Mac: Eclipse > Preferences**

and use **Remote Development > Connections**



Remote Connections



To Edit a connection:

- ✦ Close the remote connection first
- ✦ Right-click and select **Edit**
 - ✦ Change host, userid, password, etc.

- ✦ Note: Remote Host may be closed/stopped
 - ✦ Any remote interaction starts it
 - ✦ No need to restart it explicitly



Exercise

1. Create a synchronized project
 - ✦ Your login information and source directory will be provided by the tutorial instructor
2. Observe that the project files are copied to your workspace
3. Open a file in an editor, add a comment, and save the file
4. Observe that the file is synchronized when you save the file
 - ✦ Watch lower-right status area; confirm on host system



Optional Exercise

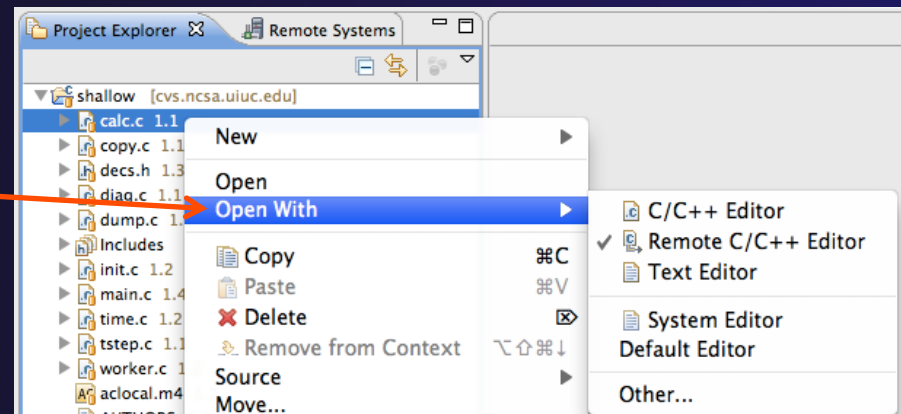
1. Modify Sync filters to not bring the *.o files and your executable back from the remote host
 - ★ Rebuild and confirm the files don't get copied

Editor Features

- ★ Objective
 - ★ Learn about Eclipse editor features
- ★ Contents
 - ★ Saving
 - ★ Editor markers
 - ★ Setting up include paths
 - ★ Code analysis
 - ★ Content assistance and templates

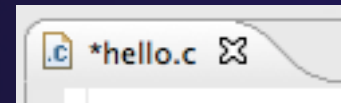
Editors

- ★ An editor for a resource (e.g. a file) opens when you double-click on a resource
- ★ The type of editor depends on the type of the resource
 - ★ .c files are opened with the C/C++ editor by default
 - ★ You can use **Open With** to use another editor
 - ★ In this case the default editor is fine (double-click)
 - ★ Some editors do not just edit raw text
- ★ When an editor opens on a resource, it stays open across different perspectives
- ★ An active editor contains menus and toolbars specific to that editor



Saving File in Editor

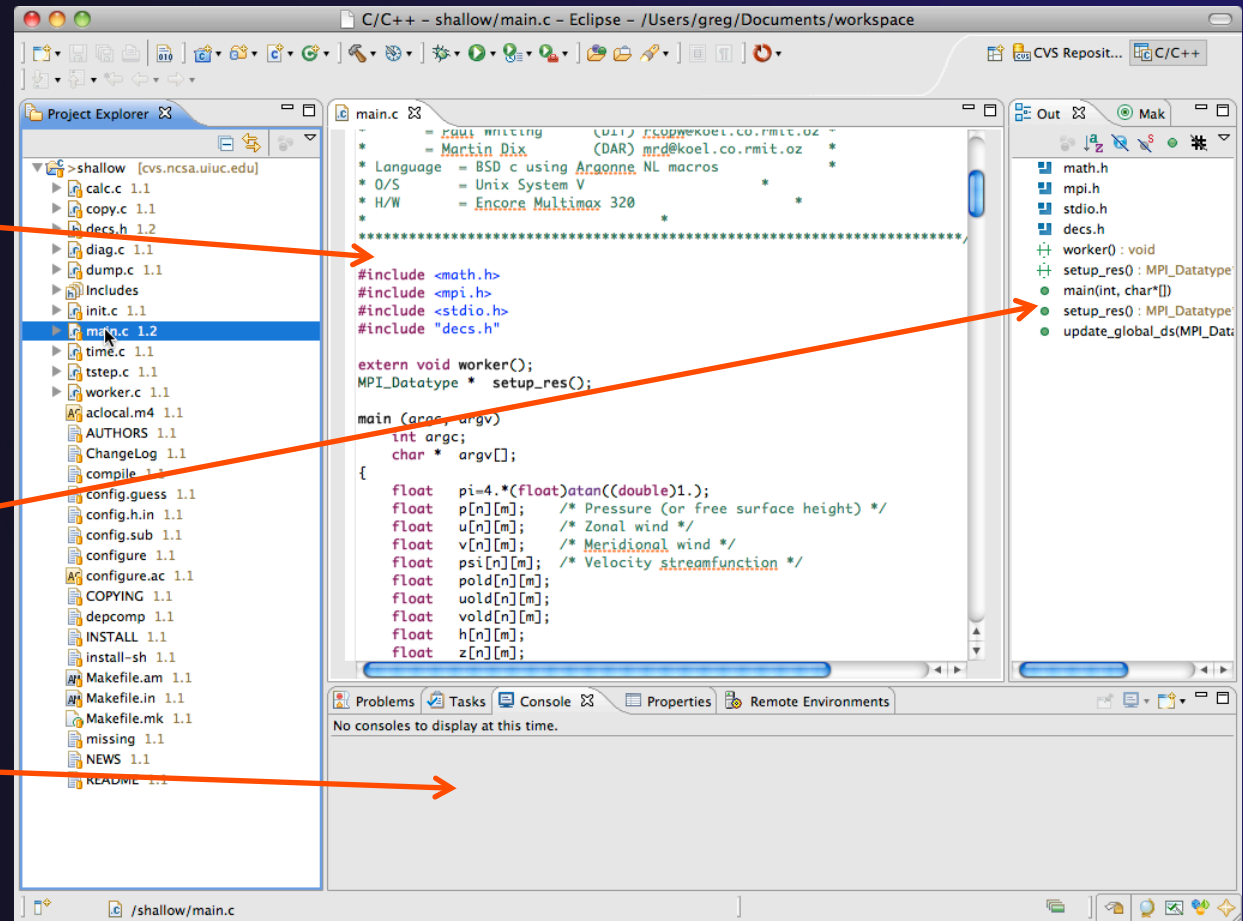
- ★ When you change a file in the editor, an asterisk on the editor's title bar indicates unsaved changes



- ★ Save the changes by using Command/Ctrl-S or **File>Save**
- ★ Undo last change using **Command/Ctrl Z**

Editor and Outline View

- ★ Double-click on source file
- ★ Editor will open in main view
- ★ Outline view is shown for file in editor
- ★ Console shows results of build, local runs, etc.

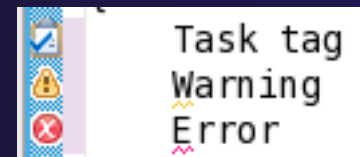


Source Code Editors & Markers

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

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

Icons:



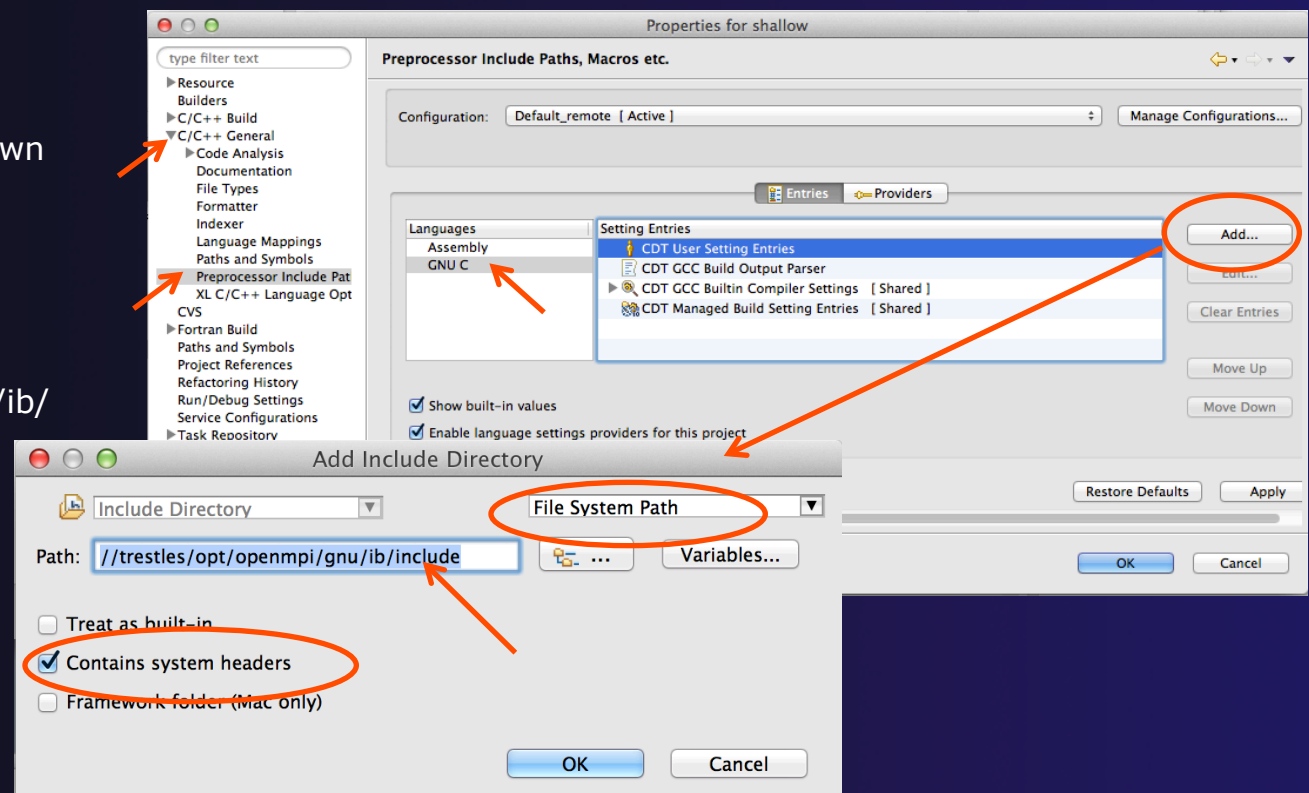
Remote Include Paths

- ★ In order for editor and build features to work properly, *Eclipse needs to know* where your include files are located
 - ★ The build environment on the remote host knows your include files etc., and will work fine without additional information
- ★ But if we tell Eclipse also,
 - ★ Then indexing, search, completion, etc. will know where things are
- ★ Two methods: **A** manual and **B** discover

A

Set Include Paths manually

- ✦ Open Project Properties
- ✦ Expand C/C++ General
- ✦ Select **Preprocessor Include Paths**
- ✦ Click **GNU C**, then **CDT User Setting Entries**, then click **Add...**
- ✦ In upper right, select **File System Path** in pulldown
- ✦ Check **Contains System Headers**
- ✦ A UNC-style path specifies `//<connection>/<path>`
- ✦ Enter Path `//trestles/opt/openmpi/gnu/ib/include`
- ✦ Select **OK**



A

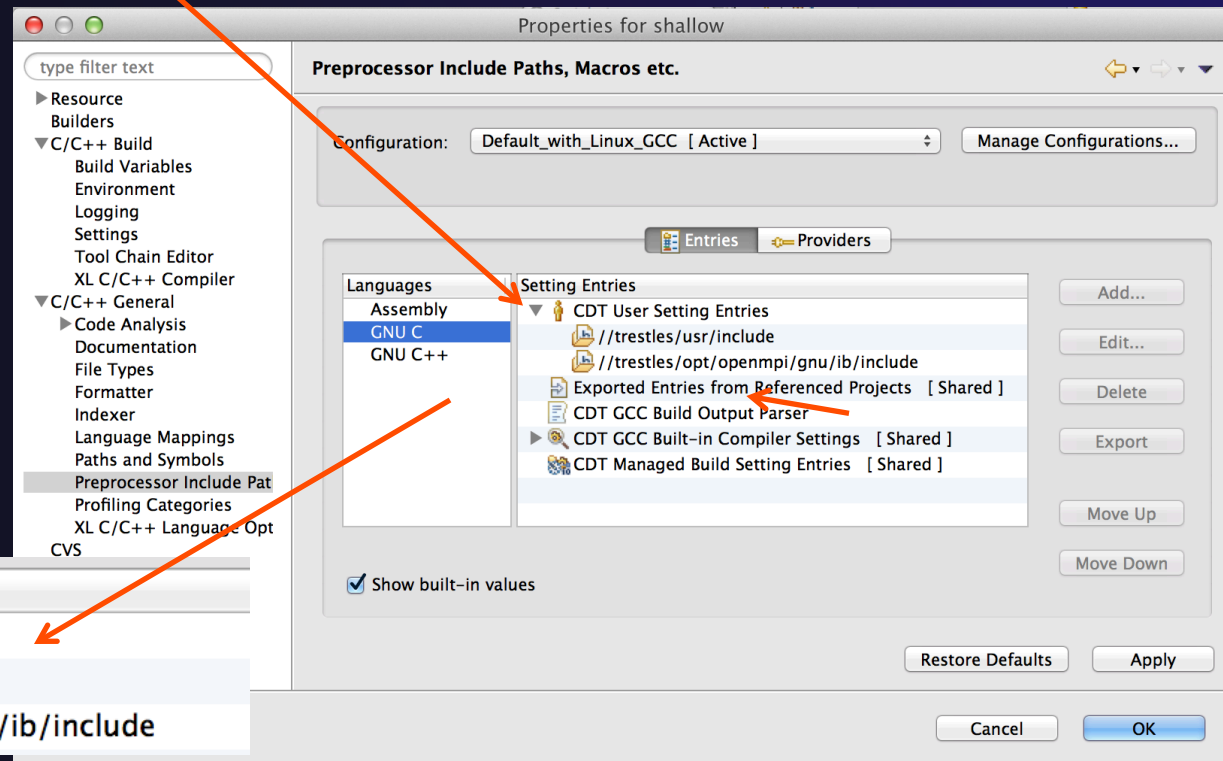
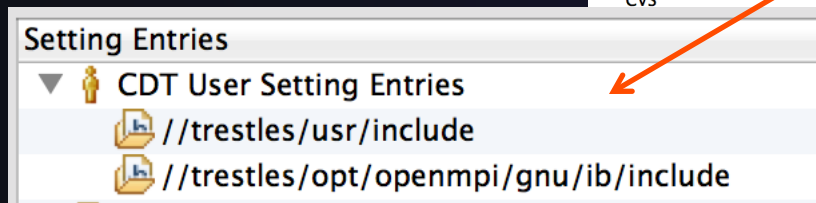
Include Paths con't

- ★ After adding include directory, it should appear in the list

- ★ Add second value:

//trestles/usr/include
... the same way

You should have
two entries:



A Include Paths con't (3)

- ✦ Select **OK**
- ✦ The C/C++ Indexer should run
 - ✦ Lower right status area indicates it



- ✦ If not force it via Project Properties>Index>Rebuild

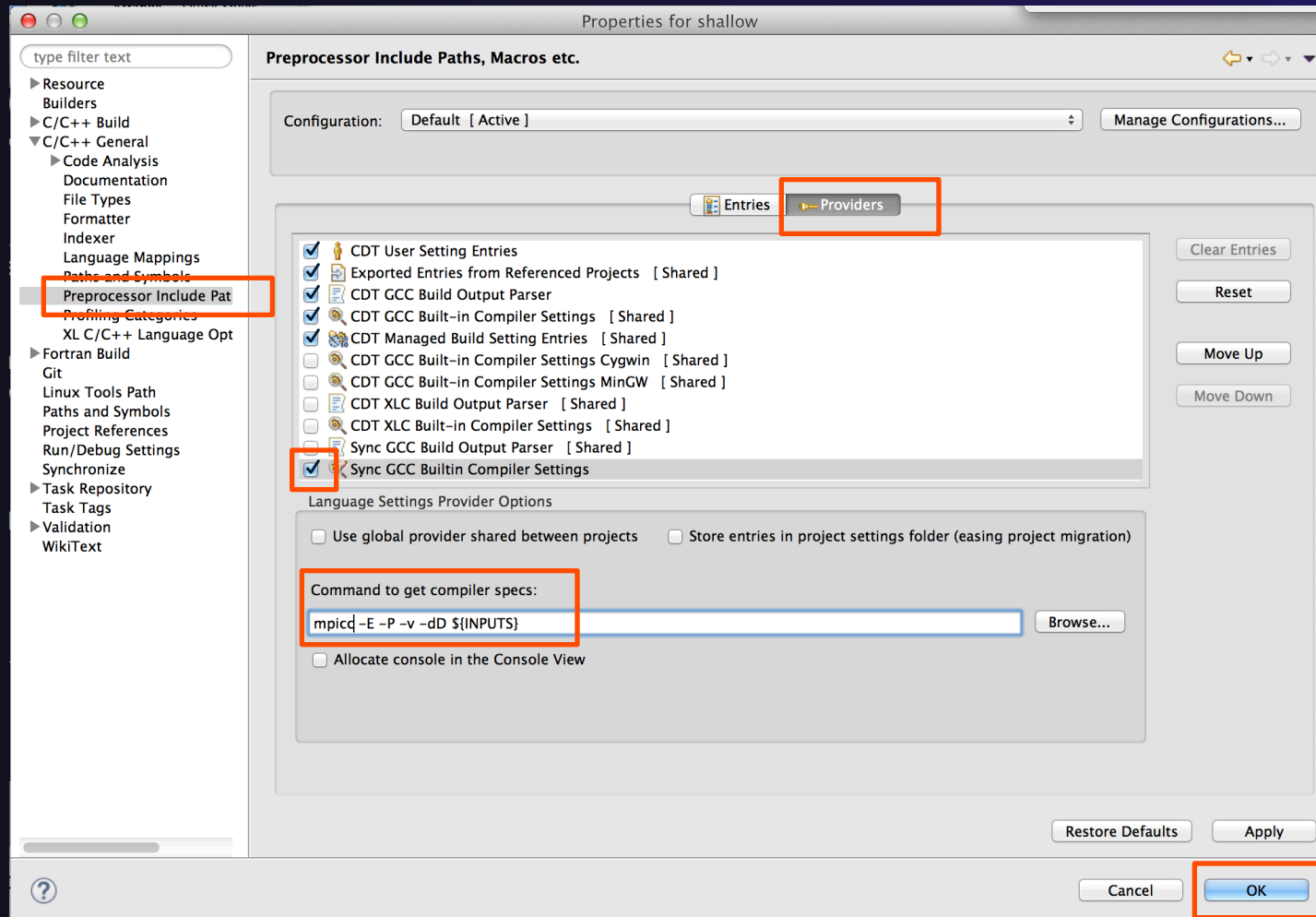
B

Set Include Paths automatically

1. Project Properties > C/C++ General > Preprocessor Include Paths, Macros etc.
2. Select the "Providers" tab
3. Click on the checkbox for "Sync GCC Builtin Compiler Settings"
4. Open the window wider. You'll see a text box with "Command to get compiler specs"
 - ✦ It will read
 - ✦ `${COMMAND} -E -P -v -dD ${INPUTS}`
 - ✦ Change `${COMMAND}` to `mpicc`, and click OK
5. Rebuild the index
 - ✦ Right click on project, Index > Rebuild
6. `mpi.h` and its symbols should now be resolved.

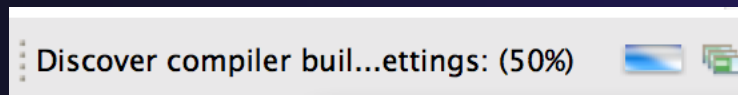
B

Set include paths automatically (con't)

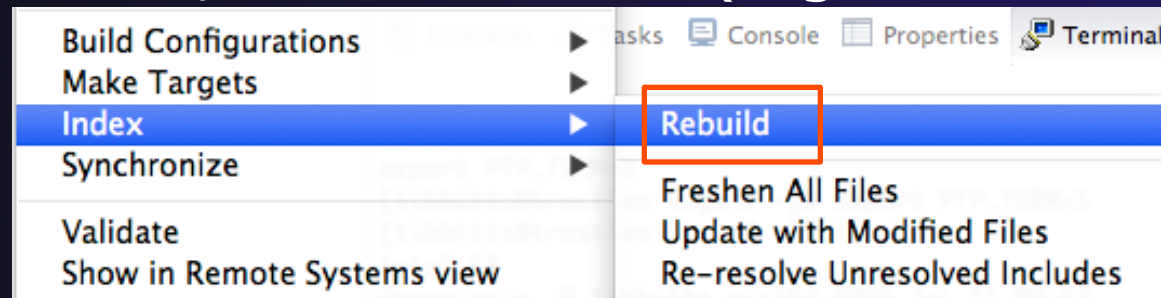


B Set include paths automatically (con't)

- ★ You may see in lower right:



- ★ When it's done, Rebuild Index (Rightmouse on project)



- ★ The C/C++ Indexer should run
 - ★ Lower right status area indicates it



Code Analysis (Codan)

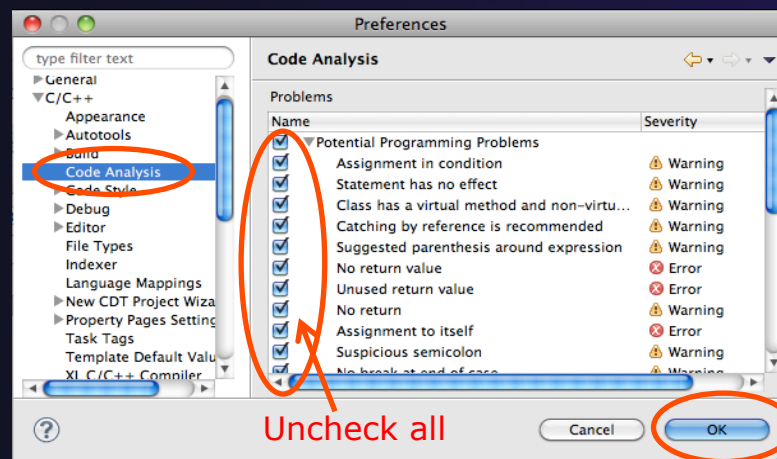
- ★ If you see bug icons in the editor marker bar, they are likely suggestions from Codan
 - ★ If include files are set correctly, they *should* not appear.
- ★ Code checkers can flag possible errors, even if code is technically correct
- ★ To turn them off, use Preferences

Window > Preferences or Mac: Eclipse > Preferences

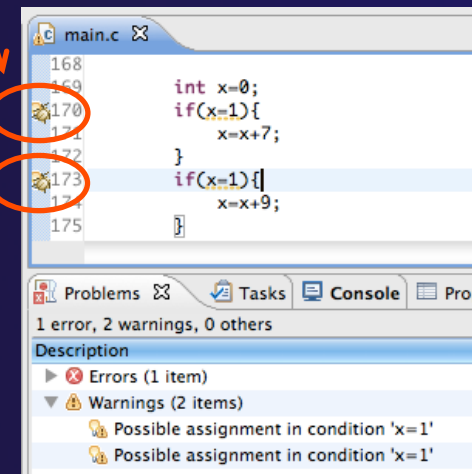
C/C++ > Code Analysis

and uncheck
all problems

- ★ Select OK to close Preferences



Uncheck all



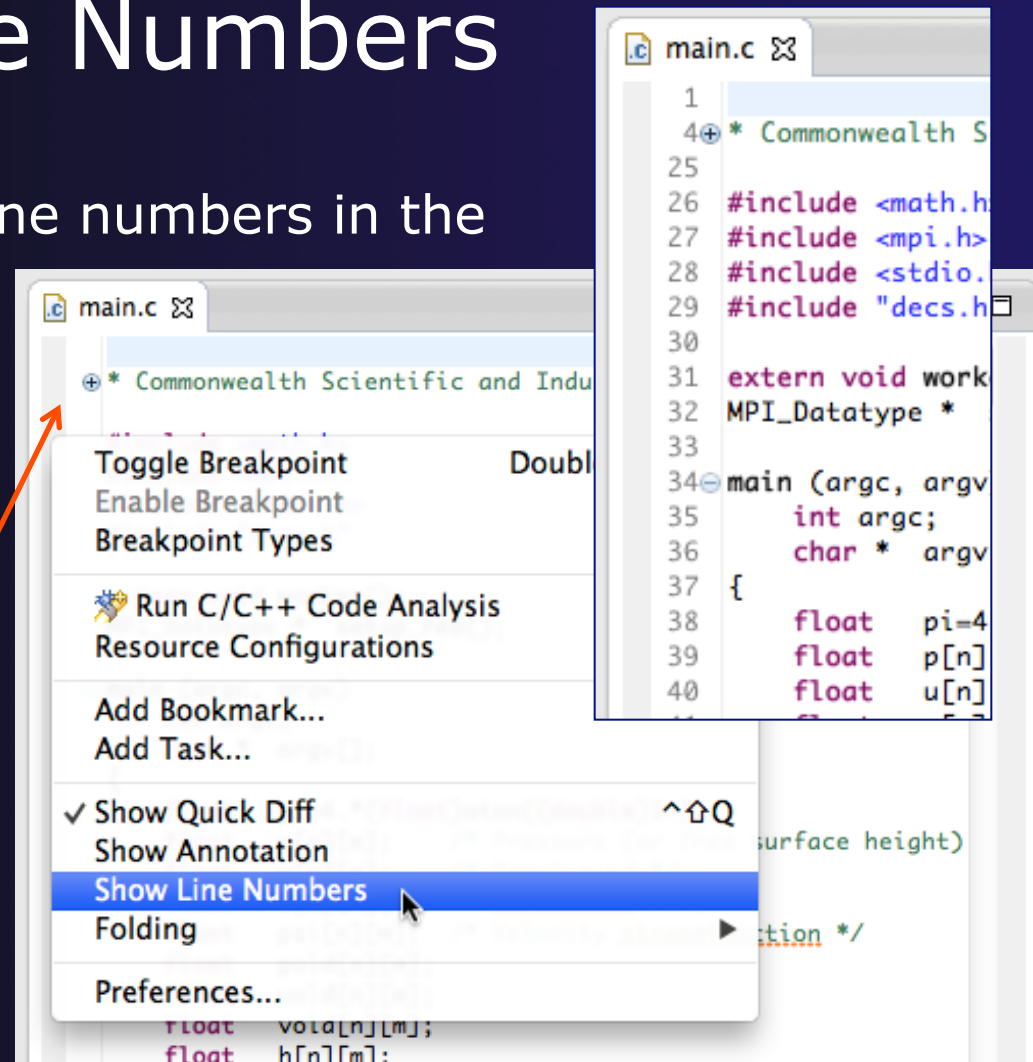
- ★ If icons don't disappear: Right mouse on Project > **Run C/C++ Code Analysis**
- ★ You can also enable/disable this per project in Project Properties

Editor Features

Editor-12

Line Numbers

- ★ Text editors can show line numbers in the left column
- ★ To turn on line numbering:
 - ★ Right-mouse click in the editor marker bar (at editor left edge)
 - ★ Click on **Show Line Numbers**



Navigating to Other Files

- ★ On demand hyperlink
 - ★ In main.c line 135:
 - ★ Hold down Command/Ctrl key e.g. on call to `initialise`
 - ★ Click on `initialise` to navigate to its definition in the header file (Exact key combination depends on your OS)
 - ★ E.g. Command/Ctrl and click on `initialise`

```

128 }
129
130
131 /*
132  initialise data structures and construct packets to be sent to workers
133  */
134
135 initialise(p, u, v, psi, pold, uold, vold, di, dj, z);
136 diag(1, 0, p, u, v, h, z);
137
138 for (i = 1; i < proc_cnt; i++) {
139     for (j = 0; j < n; j++) {

```

```

26 #include <math.h>
27 #include "dss.h"
28
29 void initialise(p, u, v, psi, pold, uold, vold, di, dj, z)
30 float p[n][m];
31 float u[n][m];
32 float v[n][m];
33 float psi[n][m];

```

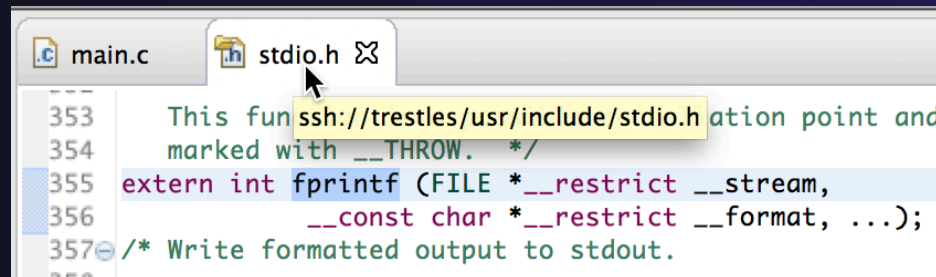
- ★ Open declaration
 - ★ Right-click and select **Open Declaration** will also open the file in which the element is declared
 - ★ E.g. in main.c line 29 right-click on `decs.h` and select **Open Declaration**

Open Declaration	F3
Open Type Hierarchy	F4
Open Call Hierarchy	^⌘H
Quick Outline	⌘O
Quick Type Hierarchy	⌘T
Explore Macro Expansion	⌘=
Toggle Source/Header	^Tab

Note: may need to left-click before right-click works

Navigating to Remote Files

- ★ Note: remote includes must be set up correctly for this to work
- ★ On demand hyperlink
 - ★ In main.c line 73:
 - ★ Ctrl-click on fprintf
 - ★ stdio.h on remote system opens
- ★ Open declaration (or F3)
 - ★ In main.c, right-click and select **Open Declaration** e.g on <stdio.h>
 - ★ File from remote system is opened.
- ★ Hover over editor name tab to see remote location.

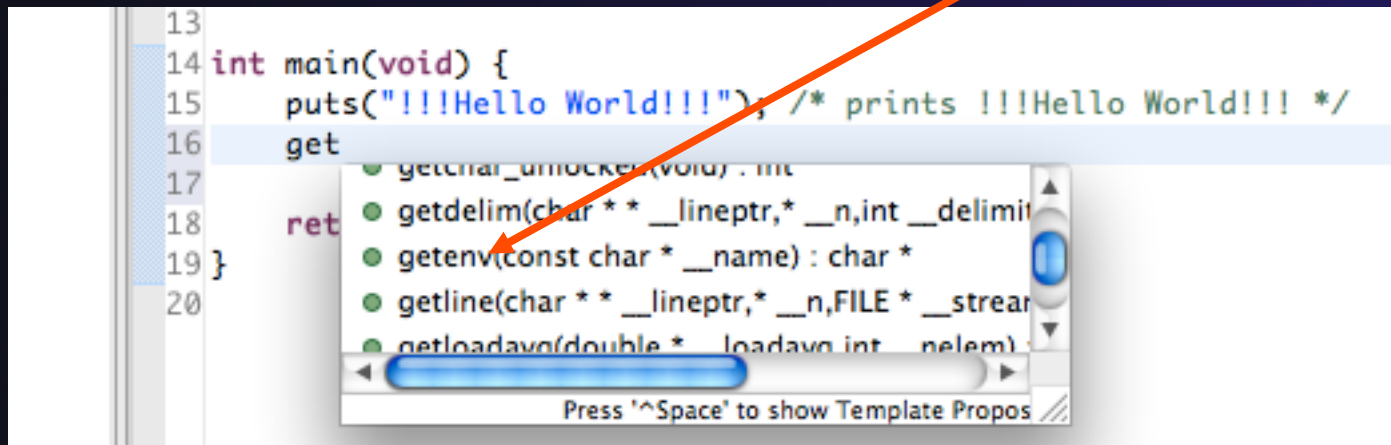


The screenshot shows an IDE window with two tabs: 'main.c' and 'stdio.h'. The 'stdio.h' tab is active and has a tooltip showing the remote file path: 'ssh://trestles/usr/include/stdio.h'. The code in the editor is as follows:

```
353 This function is a declaration point and
354 marked with __THROW. */
355 extern int fprintf (FILE *__restrict __stream,
356                    __const char *__restrict __format, ...);
357 /* Write formatted output to stdout.
```

Content Assist & Templates

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



A screenshot of a code editor showing a function completion list. The code in the background is:

```
13  
14 int main(void) {  
15     puts("!!!Hello World!!!"); /* prints !!!Hello World!!! */  
16     get  
17  
18     ret  
19 }  
20
```

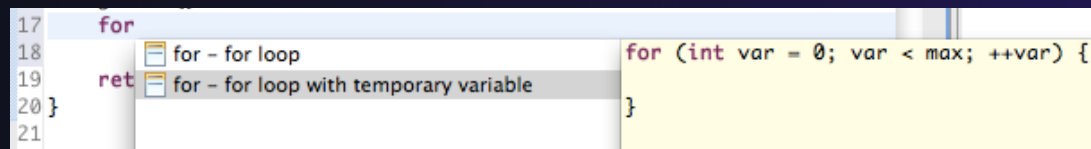
The completion list is open, showing several functions with green circular icons next to them:

- getchar_unlocked(void) : int
- getdelim(char ** __lineptr, * __n, int __delimit
- getenv(const char * __name) : char *
- getline(char ** __lineptr, * __n, FILE * __stream
- getloadavg(double * __loadavg, int __nelem)

An orange arrow points from the text "Select desired completion value with cursor or mouse" to the first item in the list. A blue highlight is visible under the 'get' text in the code. At the bottom of the list, it says "Press '^Space' to show Template Propos".

- ✦ Code Templates: type ‘for’ and Ctrl-space

Hit ctrl-space again
for code templates



A screenshot of a code editor showing code templates. The code in the background is:

```
17     for  
18  
19     ret  
20 }  
21
```

The completion list is open, showing two templates:

- for - for loop
- for - for loop with temporary variable

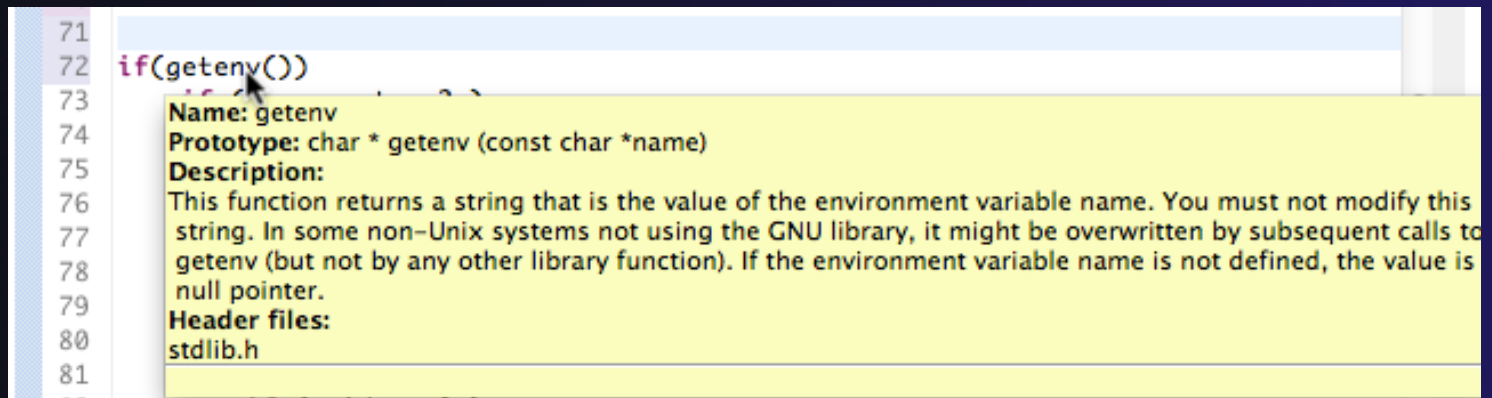
The second template is selected, and its content is shown in a yellow box:

```
for (int var = 0; var < max; ++var) {  
}
```

More info on code templates later

Hover Help

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



The screenshot shows a code editor window with a yellow tooltip displayed over the `if(getenv())` line. The tooltip contains the following information:

```
71  
72 if(getenv())  
73  
74  
75  
76  
77  
78  
79  
80  
81
```

Name: getenv
Prototype: char * getenv (const char *name)
Description:
This function returns a string that is the value of the environment variable name. You must not modify this string. In some non-Unix systems not using the GNU library, it might be overwritten by subsequent calls to getenv (but not by any other library function). If the environment variable name is not defined, the value is null pointer.
Header files:
stdlib.h

Inactive code

- ★ Inactive code will appear grayed out in the CDT editor

```
260 #define VAL
261 #ifdef VAL
262     acopy_one_to_two(VAL, ds, res.indx);
263 #else
264     acopy_one_to_two(res.row, ds, res.indx);
265 #endif
```

```
260 //#define VAL
261 #ifdef VAL
262     acopy_one_to_two(VAL, ds, res.indx);
263 #else
264     acopy_one_to_two(res.row, ds, res.indx);
265 #endif
```



Exercise

1. Open an editor by double clicking on a source file in the **Project Explorer**
2. Use the **Outline View** to navigate to a different line in the editor
3. Back in main.c, turn on line numbering
4. In main.c, ctrl-click on line 99, master_packet, should navigate to its definition in the file
5. In worker.c, line 132, hover over variable p to see info
6. Try the exercises at the end of the "Basics" section, if you haven't already, since you now have some project/source files to play with.



Optional Exercise

1. Type “for”, then activate content assist
 - ✦ Select the **for loop with temporary variable** template, insert it, then modify the template variable
 - ✦ Surround the code you just inserted with “#if 0” and “#endif” and observe that it is marked as inactive
 - ✦ Save the file
2. What do these keys do in the editor?
 - ✦ Ctrl+L; Ctrl+Shift+P (do it near some brackets)
 - ✦ Ctrl+Shift+;/
 - ✦ Ctrl+Shift+Y and Ctrl+Shift+X (do it on a word or variable name e.g.)
 - ✦ Alt+Down; Alt+Up
3. To make sure you didn't do any damage,
 - ✦ Select any source files you changed and do rightmouse > replace with ..
 - ✦ (if you made project from CVS)Latest from HEAD
 - ✦ (If you made project from remote files) ... Local History ...
 - ✦ Observe that your changes are gone.

MPI Programming

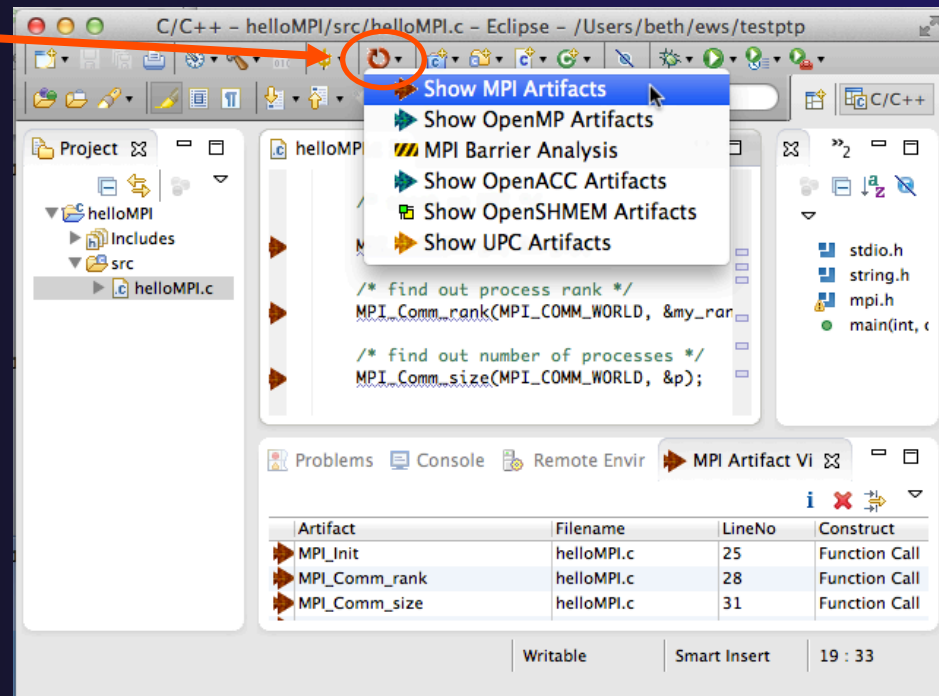
- ★ Objective
 - ★ Learn about MPI features for your source files
- ★ Contents
 - ★ Using Editor features for MPI
 - ★ MPI Help features
 - ★ Finding MPI Artifacts
 - ★ MPI New Project Wizards
 - ★ MPI Barrier Analysis

MPI-Specific Features


- ★ PTP's Parallel Language Development Tools (PLDT) has several features specifically for developing MPI code
 - ★ Show MPI Artifacts
 - ★ Code completion / Content Assist
 - ★ Context Sensitive Help for MPI
 - ★ Hover Help
 - ★ MPI Templates in the editor
 - ★ MPI Barrier Analysis
- ★ PLDT has similar features for OpenMP, UPC, OpenSHMEM, OpenACC

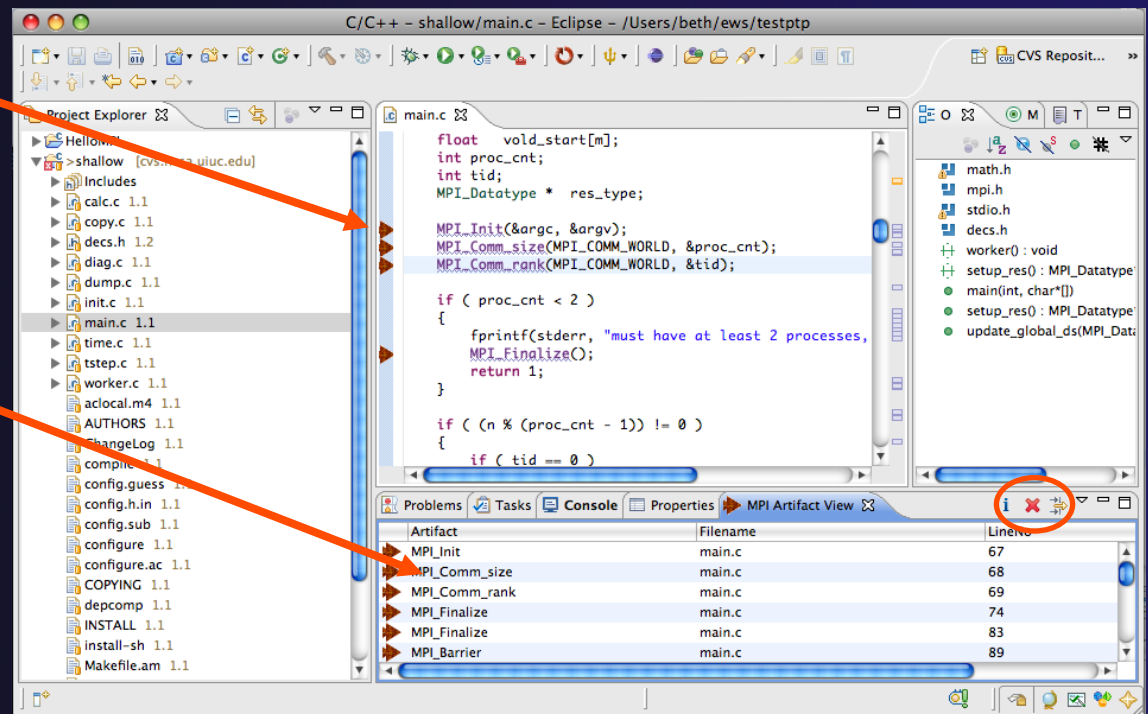
Show MPI Artifacts

- ★ In Project Explorer, select a project, folder, or a single source file
 - ★ The analysis will be run on the selected resource(s)
- ★ Run the analysis by clicking on drop-down menu next to the analysis button
- ★ Select **Show MPI Artifacts**



MPI Artifact View

- ★ Markers indicate the location of artifacts in editor
- ★ The **MPI Artifact View** lists the type and location of each artifact
- ★ Navigate to source code line by double-clicking on the artifact
- ★ Run the analysis on another file (or entire project!) and its markers will be added to the view
- ★ Click on column headings to sort
- ★ Remove markers via 



MPI Editor Features

A screenshot of an MPI Editor window showing a C program. The code includes MPI_Init, MPI_Comm_size, and MPI_Comm_rank. The cursor is at the start of a new line with 'MPI_B'. A dropdown menu shows a list of MPI functions starting with 'MPI_B', such as MPI_Barrier, MPI_Bcast, MPI_Bsend, MPI_Buffer_attach, MPI_Buffer_detach, MPI_Barrier, MPI_Bcast, MPI_Bsend, and MPI_Bsend_init. An orange arrow points from the text 'Code completion will show all the possible MPI keyword completions' to this dropdown menu.

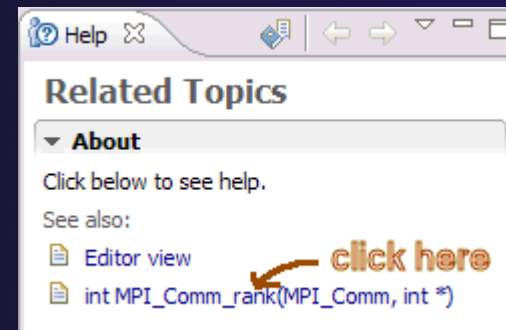
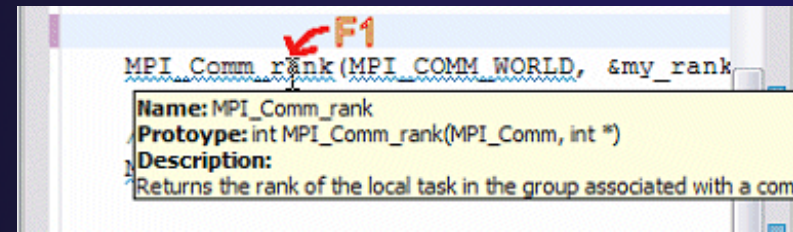
- ★ Code completion will show all the possible MPI keyword completions
- ★ Enter the start of a keyword then press <ctrl-space>

- ★ Hover over MPI API
- ★ Displays the function prototype and a description

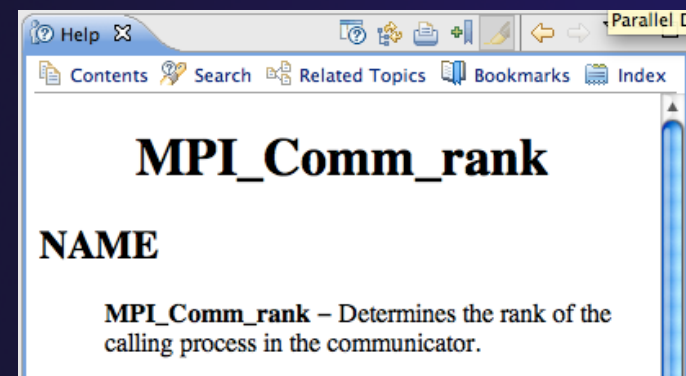
A screenshot of the same MPI Editor window. The cursor is over the MPI_Comm_rank function call. A tooltip is displayed, showing the function's name, prototype, and description. The tooltip text is: 'Name: MPI_Comm_rank', 'Prototype: int MPI_Comm_rank(MPI_Comm, int *)', and 'Description: Returns the rank of the local task in the group associated with a communicator.' An orange arrow points from the text 'Hover over MPI API' to the tooltip.

Context Sensitive Help

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



Some special info has been added for MPI APIs



MPI Templates

✦ Allows quick entry of common patterns in MPI programming

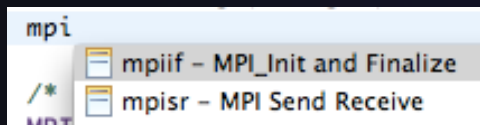
✦ Example:
MPI send-receive

✦ Enter:
mpisr <ctrl-space>

✦ Expands to a send-receive pattern

✦ Highlighted variable names can all be changed at once

✦ Type mpi <ctrl-space> <ctrl-space> to see all templates



```

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

```

Add more templates using Eclipse preferences!
C/C++>Editor>Templates
 Extend to other common patterns

MPI Barrier Analysis

The screenshot shows the Eclipse IDE interface for a C/C++ project named 'MyBarrier'. The main editor displays the source code for 'MyBarrier.c', which includes MPI barrier calls. The 'Barrier Matches' window at the bottom shows a table of barrier sets, and the 'Barrier Errors' window shows a tree view of errors.

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

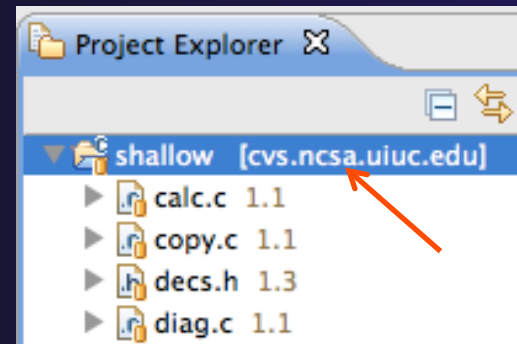
- ✦ Verify barrier synchronization in C/MPI programs
- ✦ For verified programs, lists barrier statements that synchronize together (match)
- ✦ For synchronization errors, reports counter example that illustrates and explains the error

Local files only

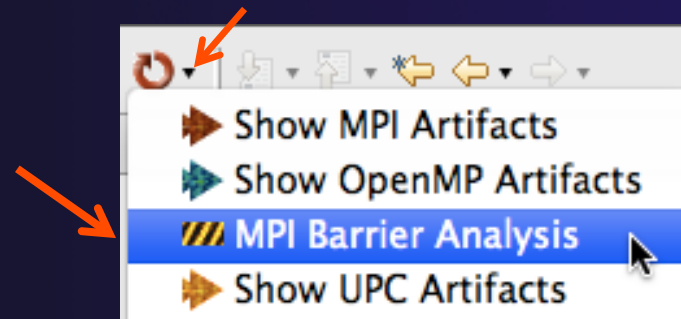
MPI Barrier Analysis (2)

Run the Analysis:

- ★ In the Project Explorer, select the project (or directory, or file) to analyze



- ★ Select the MPI Barrier Analysis action in the pull-down menu



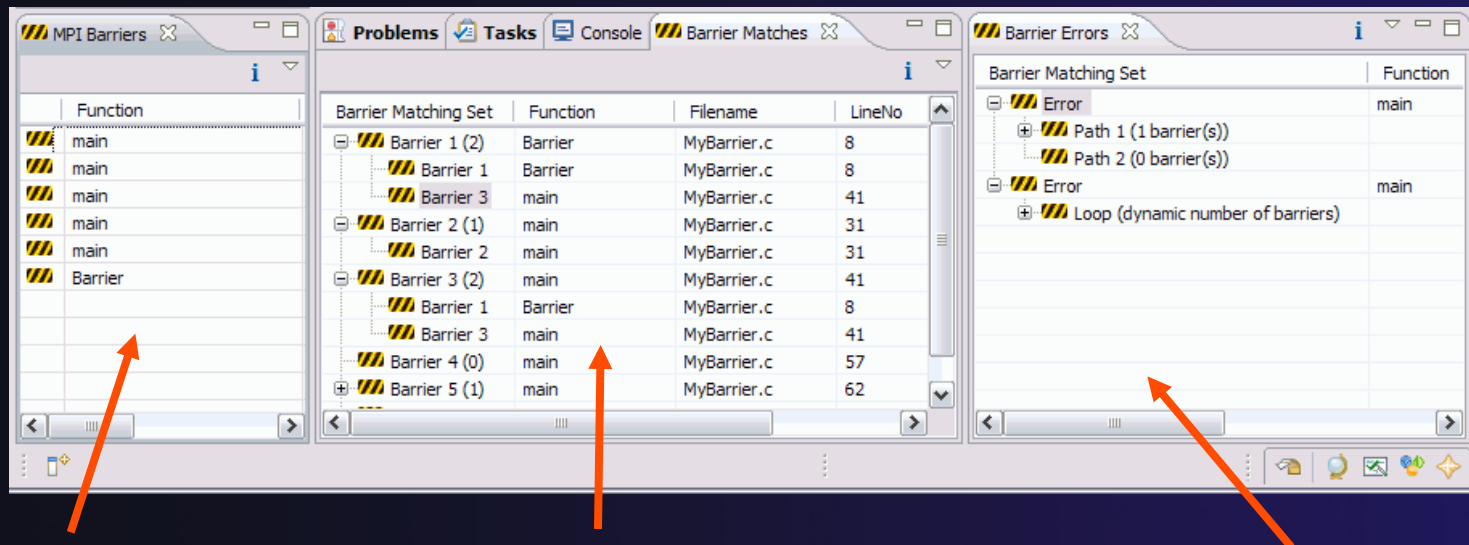
MPI Barrier Analysis (3)

- ★ No Barrier Errors are found (no pop-up indicating error)
- ★ Two barriers are found

```
83     MPI_Finalize();
84     return 1;
85 }
86
87 if (tid != 0) {
88     worker();
89     MPI_Barrier(MPI_COMM_WORLD);
90     MPI_Finalize();
91 } else {
92
93     /* master process */
94
95     struct size_t {
96         int n;
97     };
98     size_t s;
99     s.n = 1;
100    MPI_Bcast(&s, 1, MPI_INT, 0, MPI_COMM_WORLD);
101    MPI_Barrier(MPI_COMM_WORLD);
102    MPI_Finalize();
103    return 0;
104 }
```

Function	Filename	LineNo	IndexNo
main	main.c	89	1
main	main.c	206	2

MPI Barrier Analysis Views



MPI Barriers view

Simply lists the barriers

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

Barrier Matches view

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

Barrier Errors view

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

Barrier Errors

- ✦ Let's cause a barrier mismatch error
- ✦ Open worker.c in the editor by double-clicking on it in Project Explorer
- ✦ At about line 125, enter a barrier:
 - ✦ Type MPI_B
 - ✦ Hit Ctl-space
 - ✦ Select MPI_Barrier
 - ✦ Add communicator arg MPI_COMM_WORLD and closing semicolon

```
120 prv = worker[PREV];
121 nxt = worker[NEXT];
122 jstart = worker[JSTART];
123 jend = worker[JEND];
124
125 MPI_B
126 /*
127 MPI_Barrier(MPI_Comm) int
128 MPI_Bcast(void*, int, MPI_Datatype, int, MPI_Recv_status) int
129 MPI_Bsend(void*, int, MPI_Datatype, int, int, MPI_Recv_status) int
130 MPI_Bsend_init(void*, int, MPI_Datatype, int, MPI_Recv_status) int
131 MPI_Buffer_attach(void*, int) int
132 MPI_Buffer_detach(void*, int*) int
```

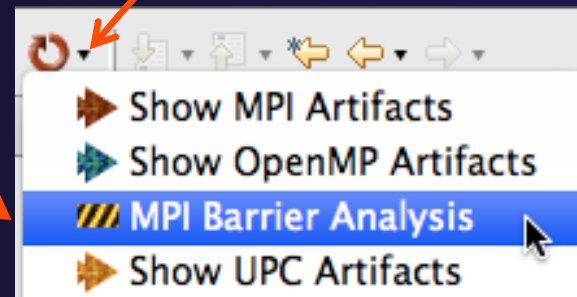
```
124
125 MPI_Barrier(MPI_COMM_WORLD);
126
```

Barrier Errors (2)

- ★ Save the file
 - ★ Ctl-S (Mac Command-S) or File > Save
 - ★ Tab should lose asterisk indicating file saved

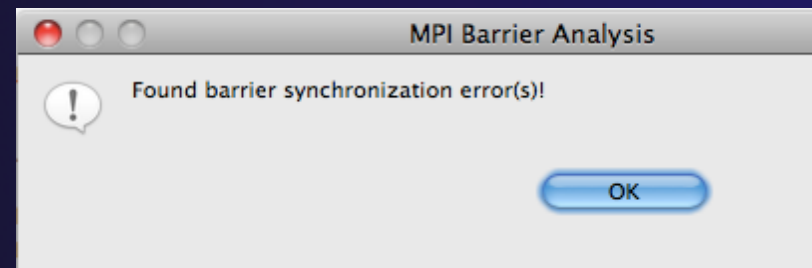


- ★ Run barrier analysis on shallow project again
 - ★ Select shallow project in Project Explorer first



Barrier Errors (3)

- ★ Barrier Error is found
- ★ Hit OK to dismiss dialog
- ★ Code diverges on line 87
 - ★ One path has 2 barriers, other has 1



Barrier Matching Set	Function	Filename	LineNo	IndexNo
▼ Error	main	main.c	87	0
▼ Path 1 (2 barrier(s))			0	0
Barrier 1	main	main.c	89	1
Barrier 3	worker	worker.c	125	3
▼ Path 2 (1 barrier(s))			0	0
Barrier 2	main	main.c	206	2

Double-click on a row in Barrier Errors view to find the line it references in the code

Fix Barrier Error

- ★ Fix the Barrier Error before continuing
- ★ Double-click on the barrier in worker.c to quickly navigate to it
- ★ Remove the line and save the file
- ★ Re-run the barrier analysis to check that it has been fixed

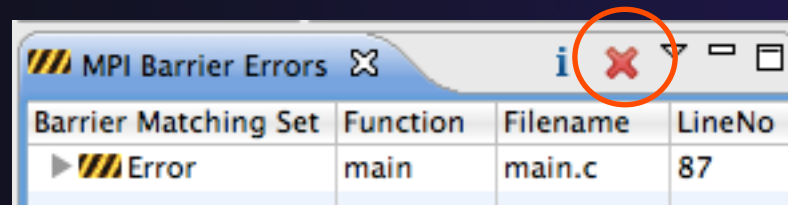
The screenshot shows an IDE with two tabs: worker.c and main.c. The worker.c tab is active, showing lines 103, 104, and 105. Line 104 contains the code `MPI_Barrier(MPI_COMM_WORLD);`. Below the code editor, there is a table titled 'MPI Barrier Errors'. The table has columns for Barrier Matching Set, Function, Filename, LineNo, and IndexNo. The table contains the following data:

Barrier Matching Set	Function	Filename	LineNo	IndexNo
▼ Error	main	main.c	87	0
▼ Path 1 (2 barrier(s))			0	0
Barrier 1	main	main.c	89	1
Barrier 3	worker	worker.c	104	3
▼ Path 2 (1 barrier(s))			0	0
Barrier 2	main	main.c	206	2

Red arrows indicate the flow of information: one arrow points from the 'Barrier 3' row in the table to the corresponding line in the code editor, and another arrow points from the code editor back to the 'Barrier 3' row in the table. The 'Barrier 3' row in the table is highlighted with a blue background, and the 'worker.c' and '104' cells are circled in red.

Remove Barrier Markers

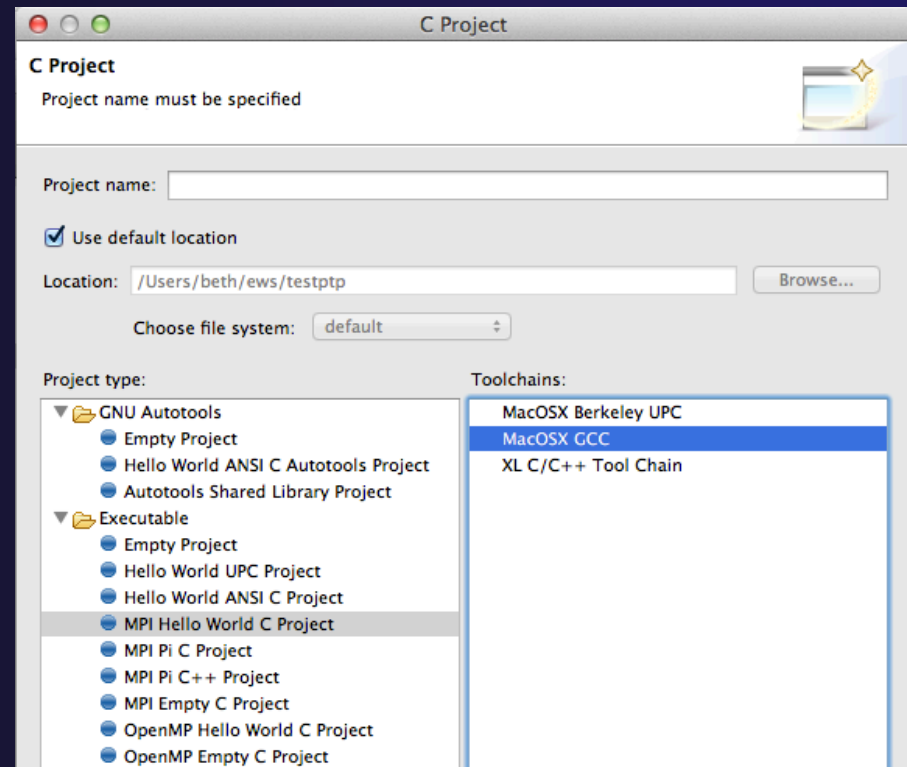
- ✦ Run Barrier Analysis again to remove the error
- ✦ Remove the Barrier Markers via the “X” in one of the MPI Barrier views



Barrier Matching Set	Function	Filename	LineNo
▶ Error	main	main.c	87

MPI New Project Wizards

- ★ Quick way to make a simple MPI project
- ★ File > New > C Project
- ★ “MPI Hello World” is good for trying out Eclipse for MPI



MPI New Project Wizards (2)

★ Next> and fill in (optional) Basic Settings

Basic Settings
Basic properties of a project

Author: Polly Parallel

Copyright notice: Your copyright notice

Hello world greeting: Hello MPI World

Source: src

< Back Next > Cancel Finish

★ Next> and fill in MPI Project Settings

★ Include path set in MPI Preferences can be added to project

MPI Project Settings
Select the MPI include path, lib name, library search path, and build command information to be automatically added to the new project.

Add MPI project settings to this project

Use default information

Include path: Browse...

Library name:

Library search path: Browse...

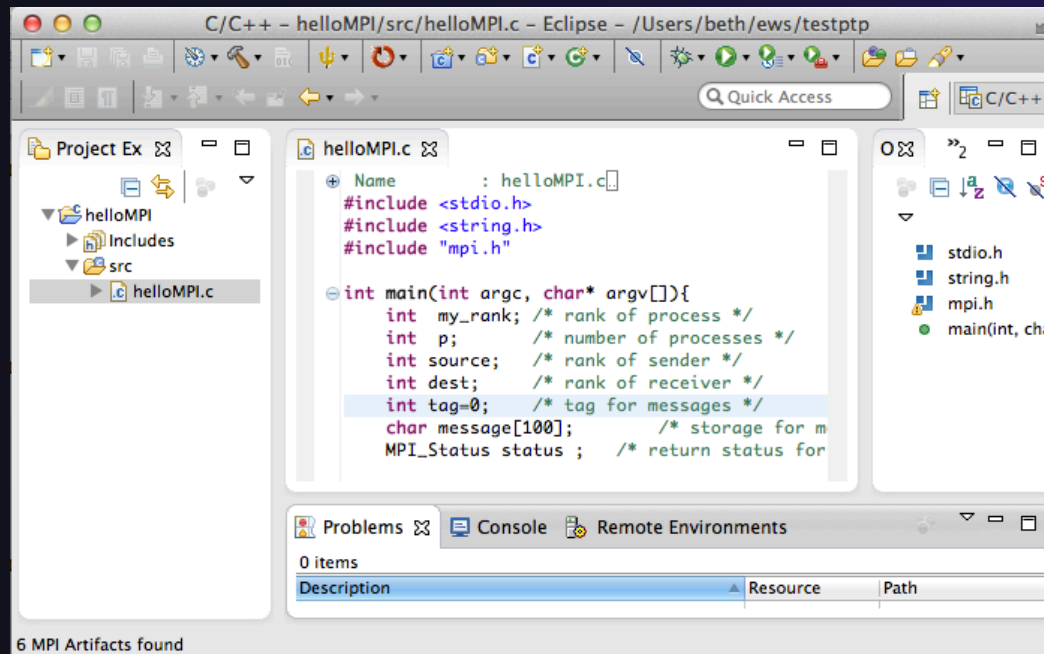
MPI compile command: mpicc

MPI link command: mpicc

< Back Next > Cancel Finish

MPI New Project Wizards (3)

- ★ Select **Finish** and “MPI Hello World” project is created

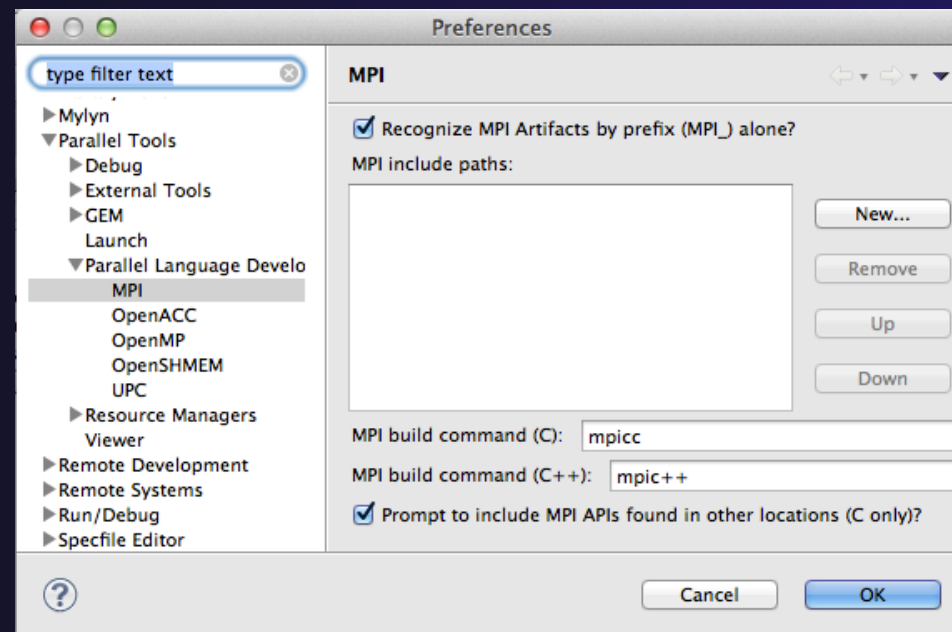


```
helloMPI.c
#include <stdio.h>
#include <string.h>
#include "mpi.h"

int main(int argc, char* argv[]){
    int my_rank; /* rank of process */
    int p; /* number of processes */
    int source; /* rank of sender */
    int dest; /* rank of receiver */
    int tag=0; /* tag for messages */
    char message[100]; /* storage for m
MPI_Status status; /* return status for
```

MPI Preferences

- ★ Settings for MPI New Project wizards
- ★ MPI Include paths, if set in MPI Preferences, are added in MPI New Project Wizard





Exercise

1. Find MPI artifacts in 'shallow' project
 - ✦ Locate all the MPI communication (send/receive) calls
2. Use content assist to add an api call
 - ✦ E.g., Type `MPI_S`, hit `ctl-space`
3. Use hover help
4. Use a template to add an MPI code template
 - ✦ On a new line, type `mpisr` and `ctl-space`...



Optional Exercise

1. Insert an `MPI_Barrier` function call into one of your source files using content assist
 - ✦ E.g. Line 125 of `worker.c`
2. Save the file
3. Run Barrier Analysis on the project
4. Locate the source of the barrier error and remove the statement
5. Re-run barrier analysis to observe that the problem has been fixed

Building a Project

★ Objective

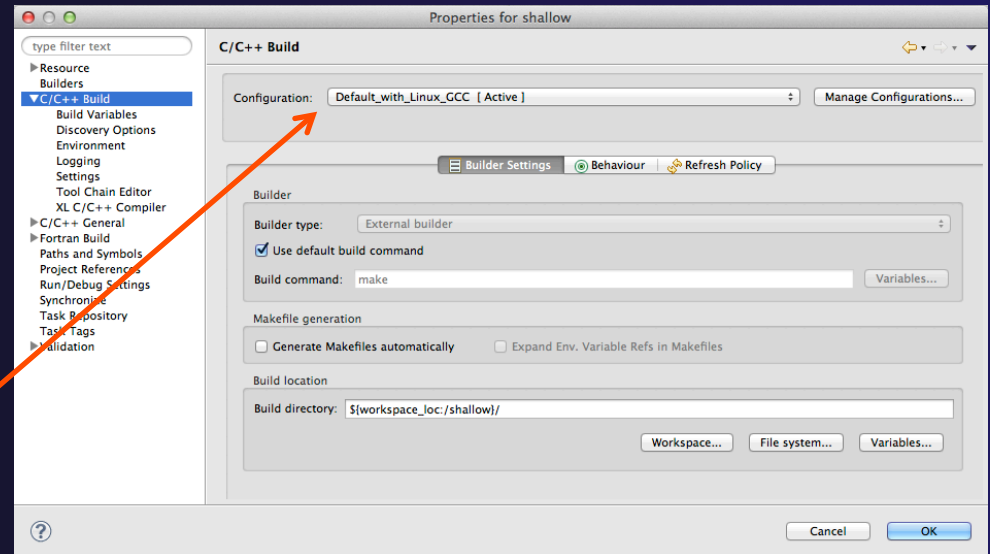
- ★ Learn how to build an MPI program on a remote system

★ Contents

- ★ How to change build settings
- ★ How to start a build and view build output
- ★ How to clean and rebuild a project
- ★ How to do environment configuration with **modules**
- ★ How to create build targets

Build Configurations

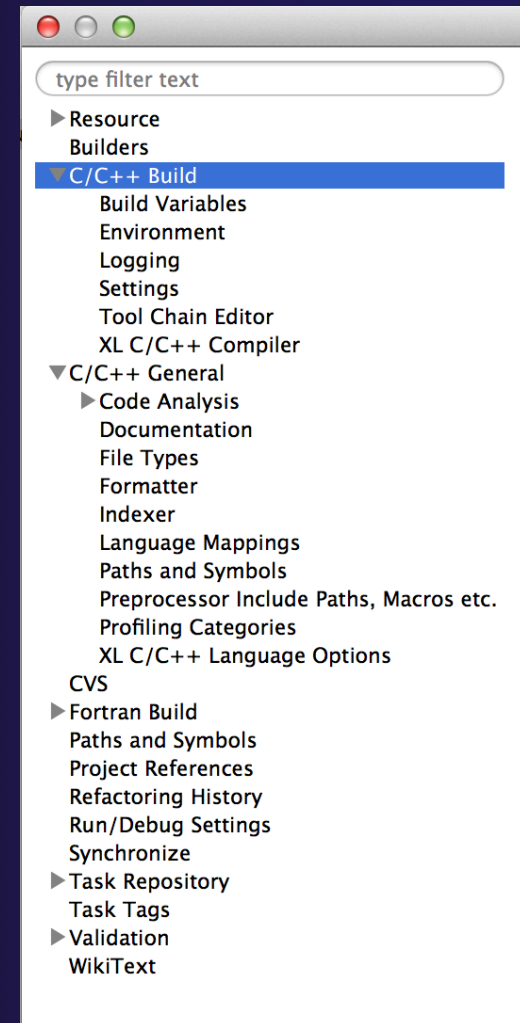
- ★ A build configuration provides the necessary information to build the project
- ★ The build configuration information is specified in the project properties
- ★ Projects can have multiple build configurations, each configuration specifies a different set of options for a build
- ★ Open the properties by right-clicking on the project name in the **Project Explorer** view and selecting **Properties** (bottom of the context menu list)



Note: Fortran projects are a superset of C/C++ projects, so they have properties for both

Build Properties (1)

- ★ **C/C++ Build**
 - ★ Main properties page
 - ★ Configure the build command
 - ★ Default is “make” but this can be changed to anything
- ★ **Build Variables**
 - ★ Create/manage variables that can be used in other build configuration pages
- ★ **Environment**
 - ★ Modify/add environment variables passed to build
- ★ **Logging**
 - ★ Enable/disable build logging



Build Properties (2)

★ Settings

- ★ Binary parser selection (used to display binaries in Project Explorer)
- ★ Error parser selection (used to parse the output from compiler commands)
- ★ Tool Chain settings (managed projects only)

★ Tool Chain Editor

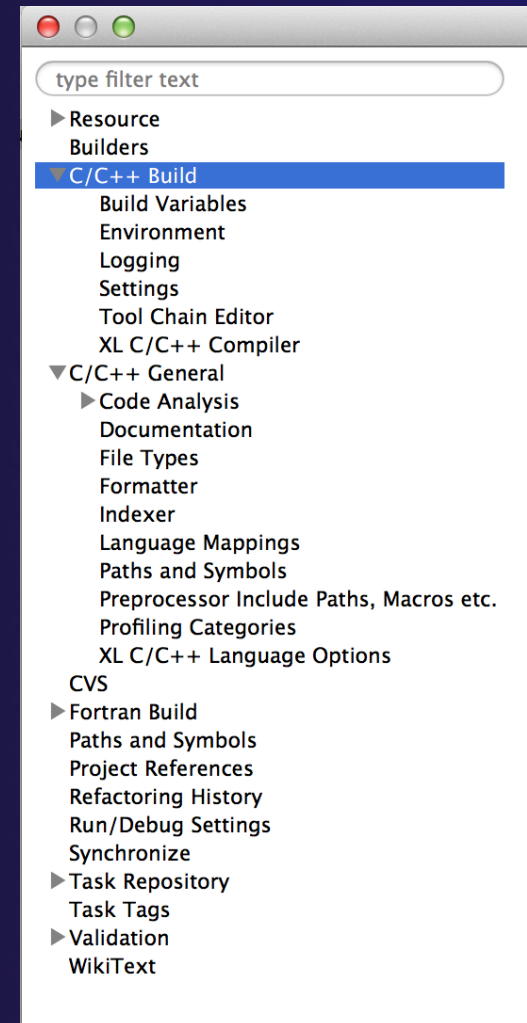
- ★ Allows the tools in a particular tool chain to be modified

★ XL C/C++ Compiler


- ★ Compiler settings for XL C/C++ compilers (if installed)

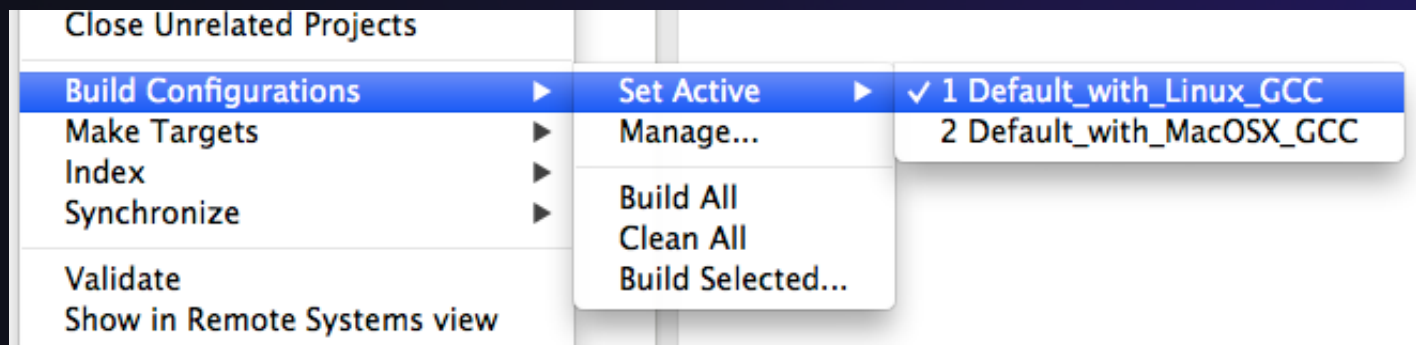
★ C/C++ General/Preprocessor Include Paths...

- ★ Set include paths here



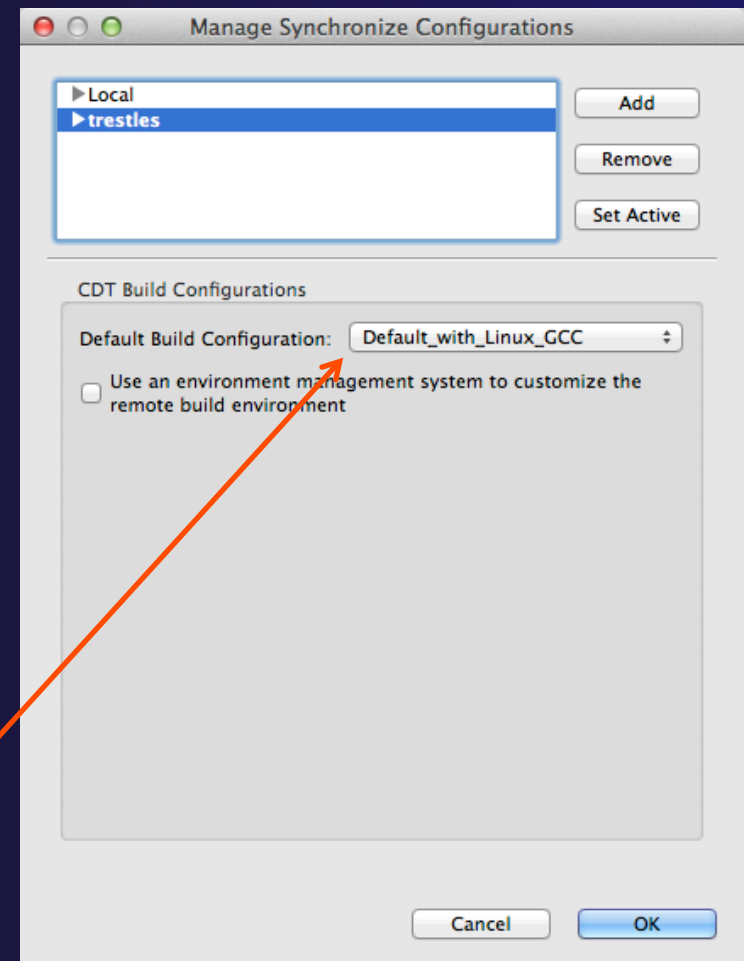
Selecting Build Configuration

- ★ Multiple build configurations may be available
 - ★ Synchronized projects will usually have a remote and a local build configuration
 - ★ Build configurations for different architectures
- ★ The active build configuration will be used when the build button  is selected
- ★ The **Build Configurations** project context menu can be used to change the active configuration
 - ★ Right click on project, then select the build configuration from the **Build Configurations > Set Active** menu



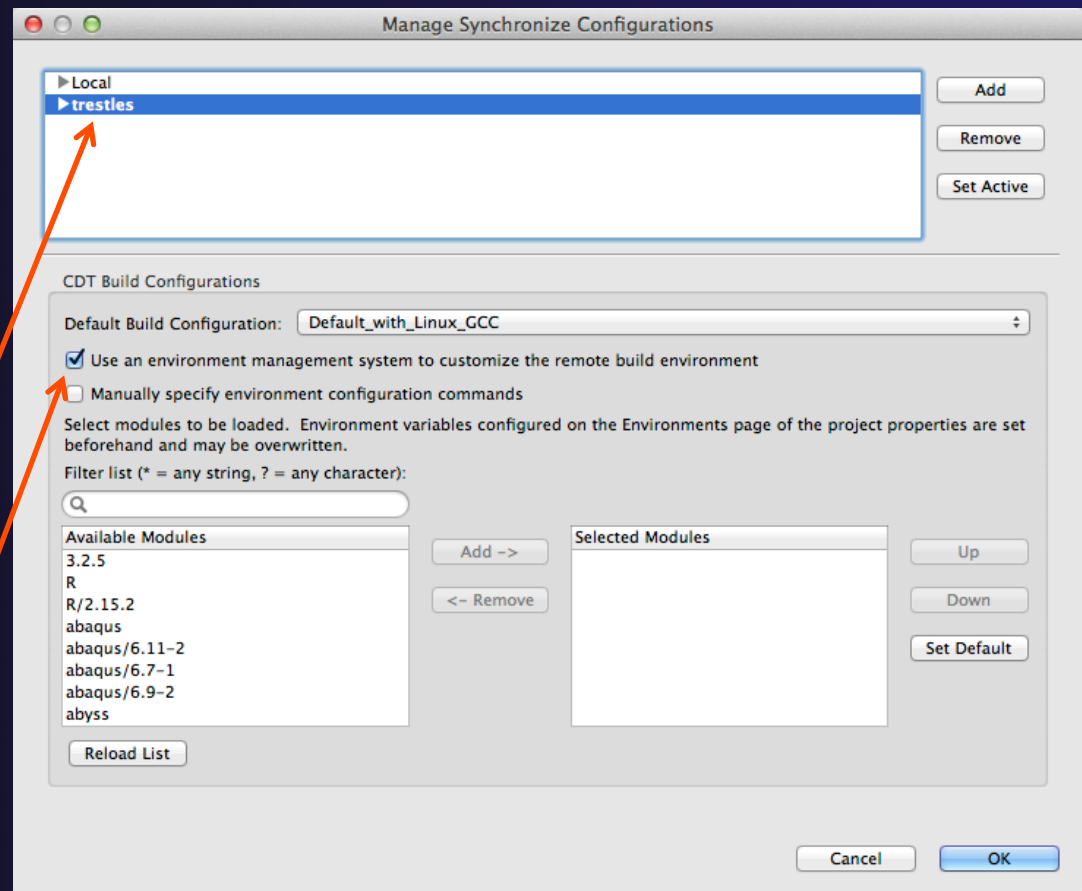
Building Synchronized Projects

- ★ When the build button is selected, the “active” build configuration will be built on the remote system specified by the “active” synchronize configuration
- ★ The build and synchronize configurations are independent
 - ★ It is possible to change which build configuration is active, but make sure this makes sense on the remote system specified in the synchronize configuration
- ★ Right mouse on Project, **Synchronize > Manage...**
- ★ A build configuration can be associated with a synchronize configuration, so that it is automatically selected when the synchronize configuration is changed



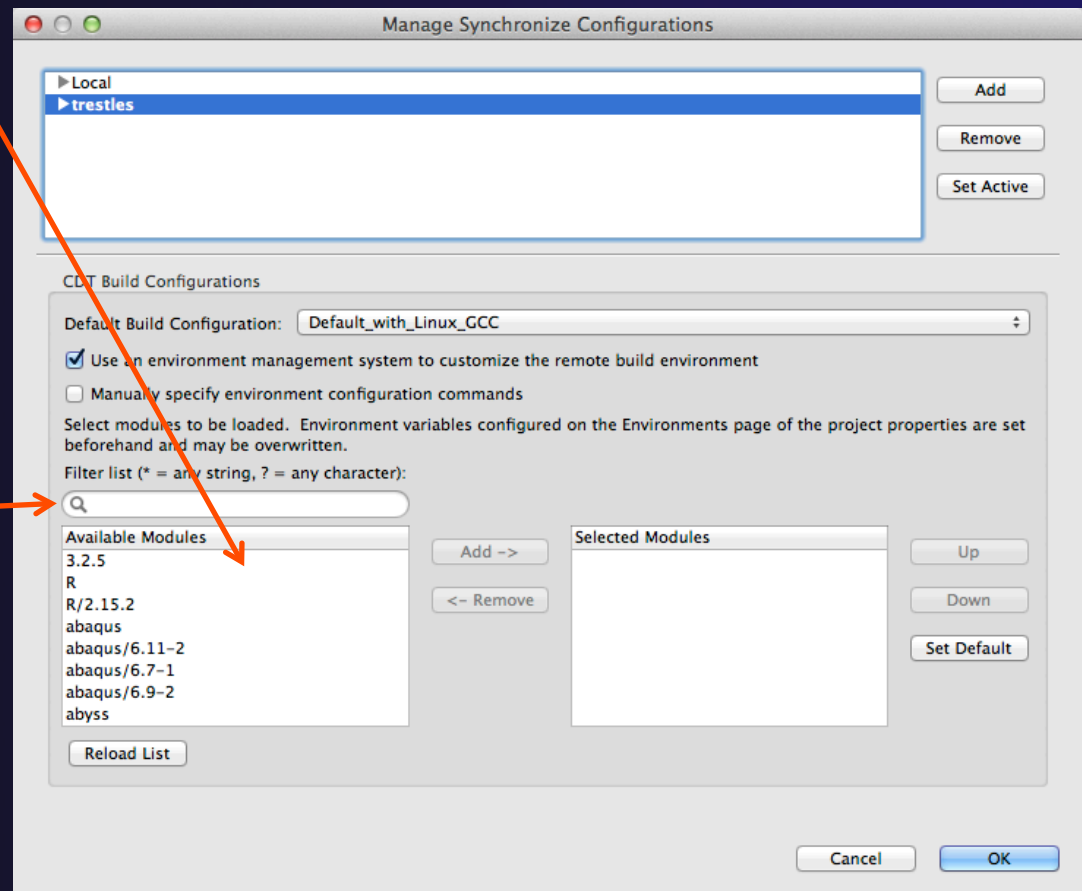
Configuring the Build Environment

- ✦ If the remote system has an environment system (such as Modules) installed, a custom set of modules can be configured for building C/C++ projects
- ✦ In the **Manage Synchronize Configurations** dialog, select the configuration you wish to change
- ✦ Check **Use an environment management system to customize the remote build environment**



Build Environment (2)

- ★ Select a module from the **Available Modules** list and click the **Add->** button to add them to the **Selected Modules** list
- ★ Use the **<-Remove** button to remove modules from the Selected Modules list
- ★ Use the **Filter list** field to quickly find modules with a given name
- ★ Use the **Up** and **Down** buttons to change the order of the **Selected Modules**
- ★ Click **Select Defaults** to load only those modules that are present in a new login shell



We'll do this for tutorial in a few slides...

Build Environment (3)

- ★ When you build the project, Eclipse will
 - ★ Open a new Bash login shell
 - ★ Execute *module purge*
 - ★ Execute *module load* for each selected module
 - ★ Run *make*
- ★ Module commands are displayed in the Console view during build
- ★ Beware of modules that must be loaded in a particular order, or that contain common paths like */bin* or */usr/bin*

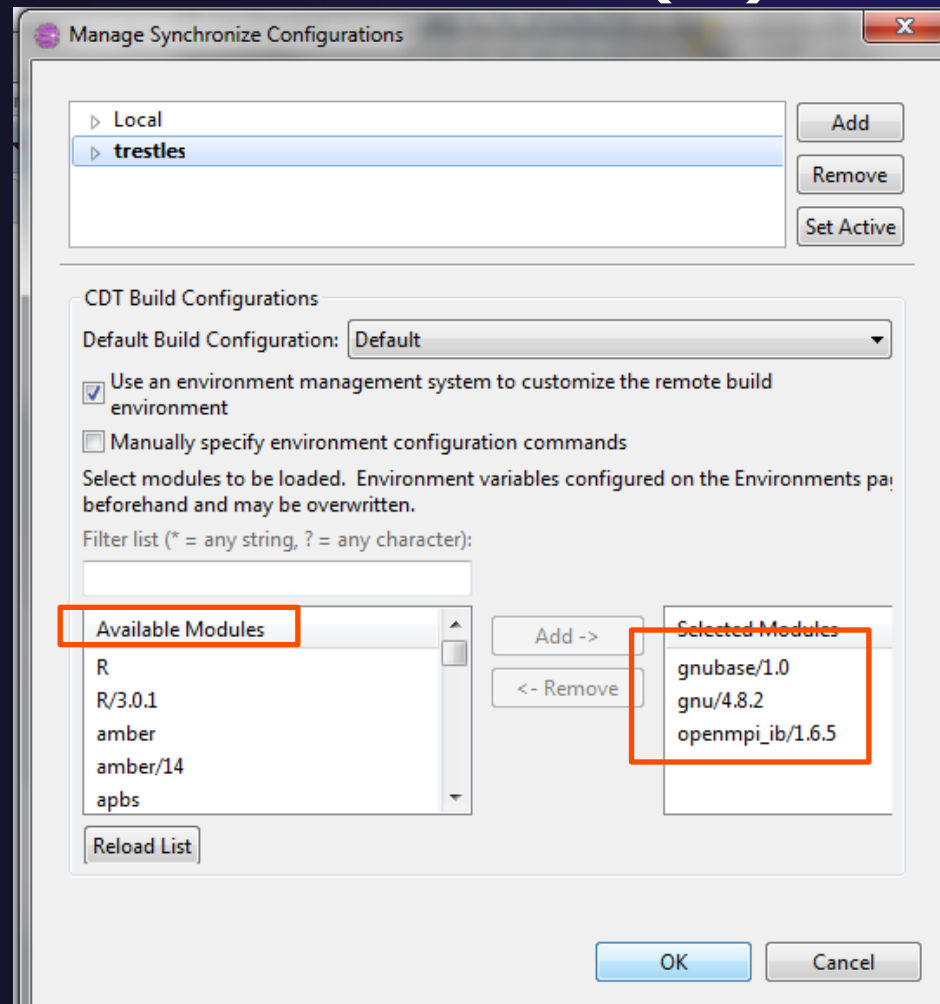


```
Console [X]
CDT Build Console [shallow]
17:53:20 **** Build of configuration Default_remote for project shallow ****
make all

**** Environment configuration script temporarily stored in /tmp/ptpscript_rhMesG ****
module purge >/dev/null 2>&1
module load cuda-4.0.17
module load cupti/4.0.17
module load ctkub 5.0.4.01
```

Build Environment (4)

- ✦ For this tutorial, we want to use gcc and Open MPI
- ✦ To get to this dialog: Right mouse on Project, **Synchronize > Manage...**
- ✦ Navigate to **gnu** in **Available Modules** and select **Add ->**
- ✦ Navigate to **openmpi_ib** and select **Add ->**
- ✦ Assure the order matches this
 - ✦ If not, use **Up/Down** buttons

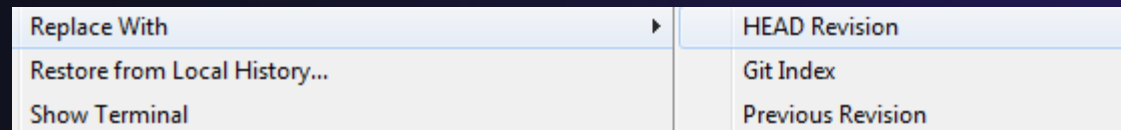
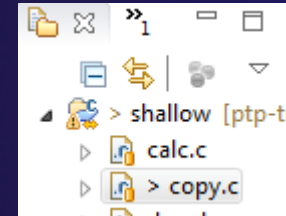


Start with original 'shallow'

- ★ Start with original 'shallow' code:

- ★ Project checked out from git:

- ★ Right mouse on project,
Replace With > HEAD Revision



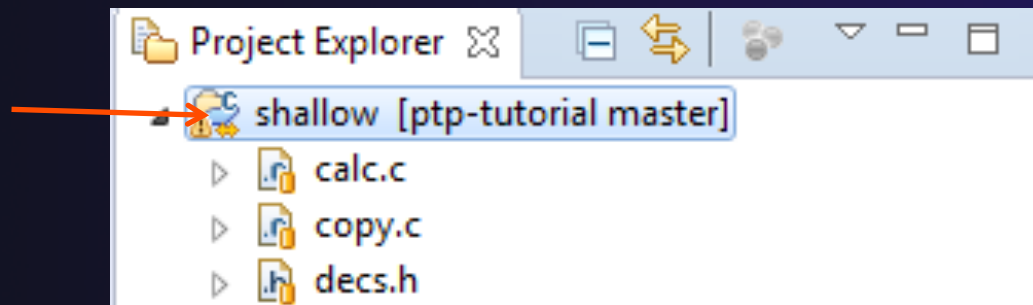
Also see Compare With ...

- ★ Other project:

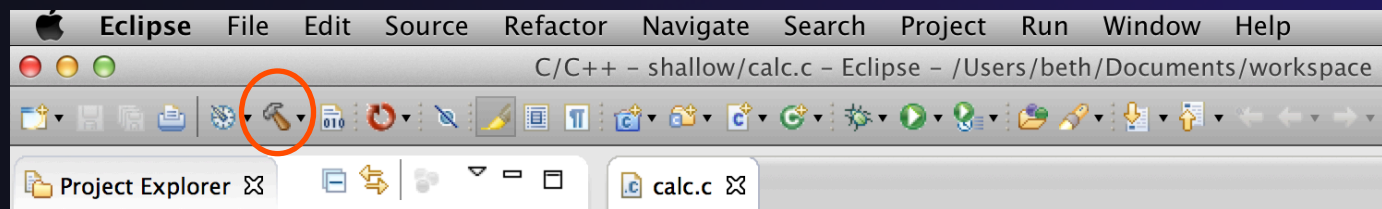
- ★ Right mouse on project,
Restore from local history – finds deleted files
 - ★ Right mouse on file, **Compare With**
or **Replace With**

Starting the Build

- ✦ Select the project in Project Explorer



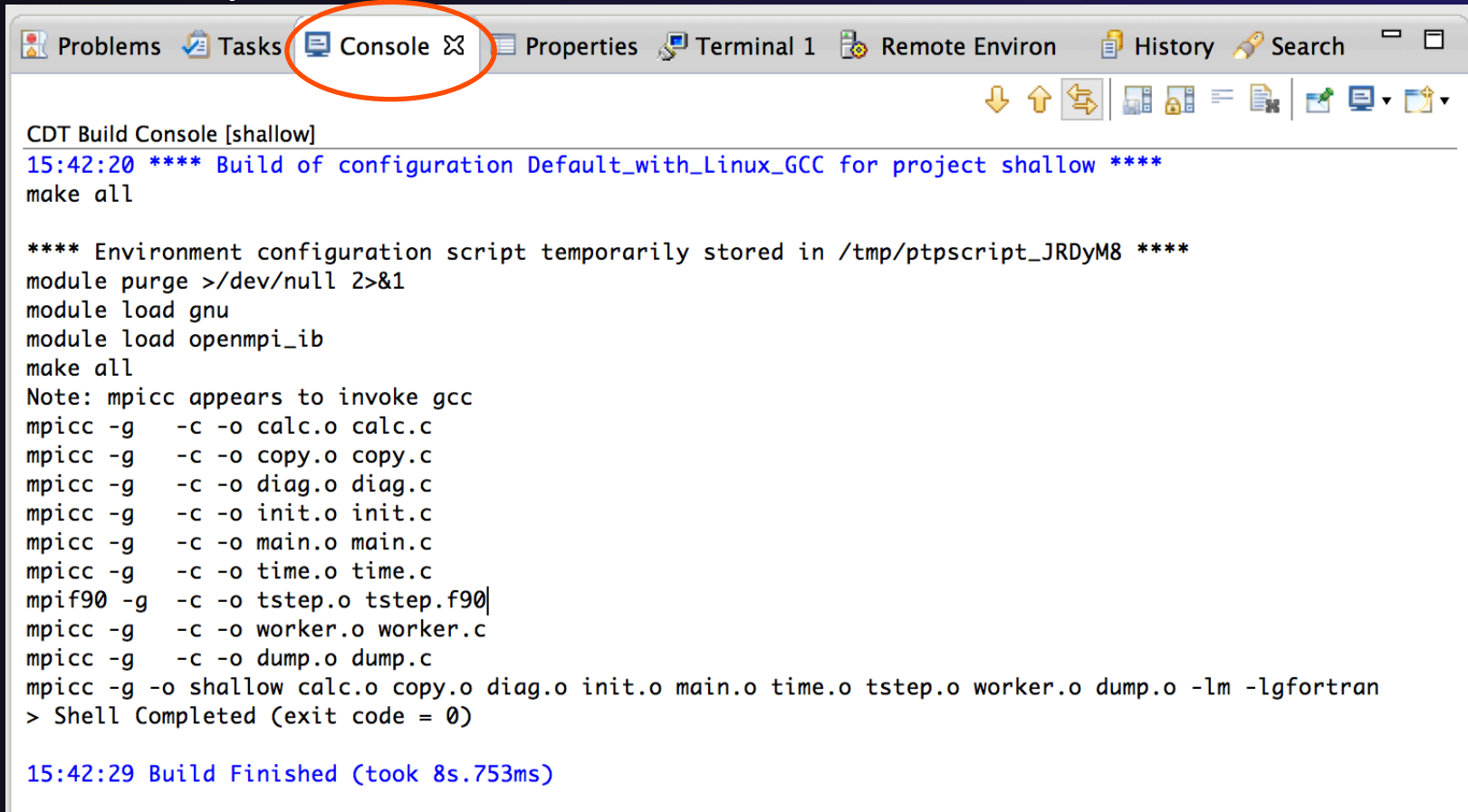
- ✦ Click on the  hammer button in toolbar to run a build using the active build configuration



- ✦ By default, the Build Configuration assumes there is a Makefile (or makefile) for the project

Viewing the Build Output

- ★ Build output will be visible in console



```
CDT Build Console [shallow]
15:42:20 **** Build of configuration Default_with_Linux_GCC for project shallow ****
make all

**** Environment configuration script temporarily stored in /tmp/ptpscript_JRDyM8 ****
module purge >/dev/null 2>&1
module load gnu
module load openmpi_ib
make all
Note: mpicc appears to invoke gcc
mpicc -g -c -o calc.o calc.c
mpicc -g -c -o copy.o copy.c
mpicc -g -c -o diag.o diag.c
mpicc -g -c -o init.o init.c
mpicc -g -c -o main.o main.c
mpicc -g -c -o time.o time.c
mpif90 -g -c -o tstep.o tstep.f90
mpicc -g -c -o worker.o worker.c
mpicc -g -c -o dump.o dump.c
mpicc -g -o shallow calc.o copy.o diag.o init.o main.o time.o tstep.o worker.o dump.o -lm -lgfortran
> Shell Completed (exit code = 0)

15:42:29 Build Finished (took 8s.753ms)
```

Build Problems

★ Build problems will be shown in a variety of ways

- ★ Marker on file
- ★ Marker on editor line
- ★ Line is highlighted
- ★ Marker on overview ruler
- ★ Listed in the **Problems view**

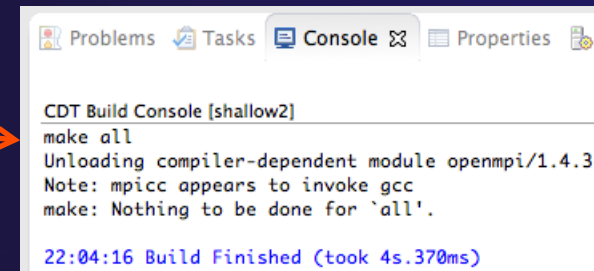
★ Double-click on line in **Problems view** to go to location of error in the editor

The screenshot shows the Eclipse IDE interface. The Project Explorer on the left displays a project named 'shallow' with various source files. The main editor window shows the code for 'main.c', with line 97 highlighted. The Problems view at the bottom shows three errors:

Description	Resource	Path	Location	Type
✘ syntax error before ';' token	main.c	/shallow	line 97	C/C++ Problem
✘ syntax error before ')' token	main.c	/shallow	line 97	C/C++ Problem
✘ syntax error before "return"	main.c	/shallow	line 212	C/C++ Problem

Forcing a Rebuild

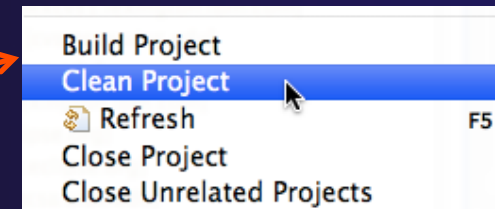
- ★ If no changes have been made, make doesn't think a build is needed e.g. if you only change the Makefile



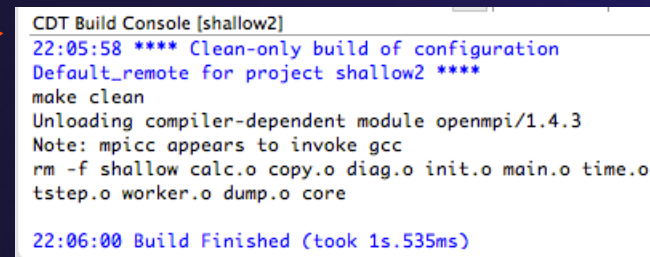
```
CDT Build Console [shallow2]
make all
Unloading compiler-dependent module openmpi/1.4.3
Note: mpicc appears to invoke gcc
make: Nothing to be done for `all'.

22:04:16 Build Finished (took 4s.370ms)
```

- ★ In **Project Explorer**, right click on project; Select **Clean Project**



- ★ Build console will display results



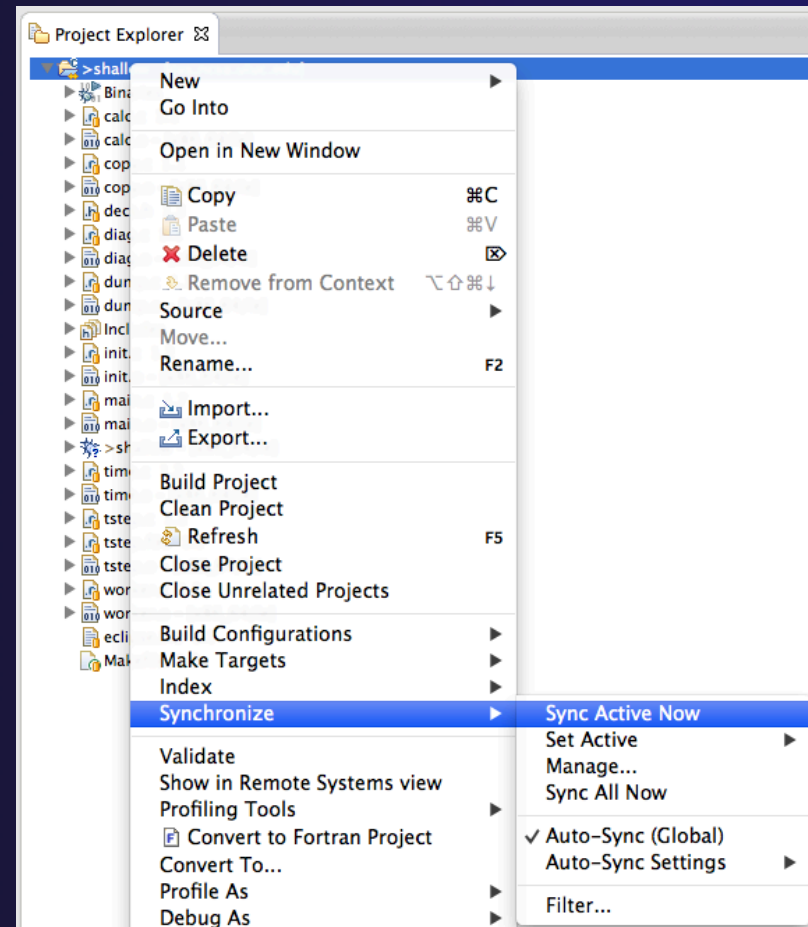
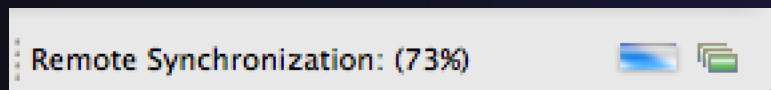
```
CDT Build Console [shallow2]
22:05:58 **** Clean-only build of configuration
Default_remote for project shallow2 ****
make clean
Unloading compiler-dependent module openmpi/1.4.3
Note: mpicc appears to invoke gcc
rm -f shallow calc.o copy.o diag.o init.o main.o time.o
tstep.o worker.o dump.o core

22:06:00 Build Finished (took 1s.535ms)
```

- ★ Rebuild project by clicking on build button again 

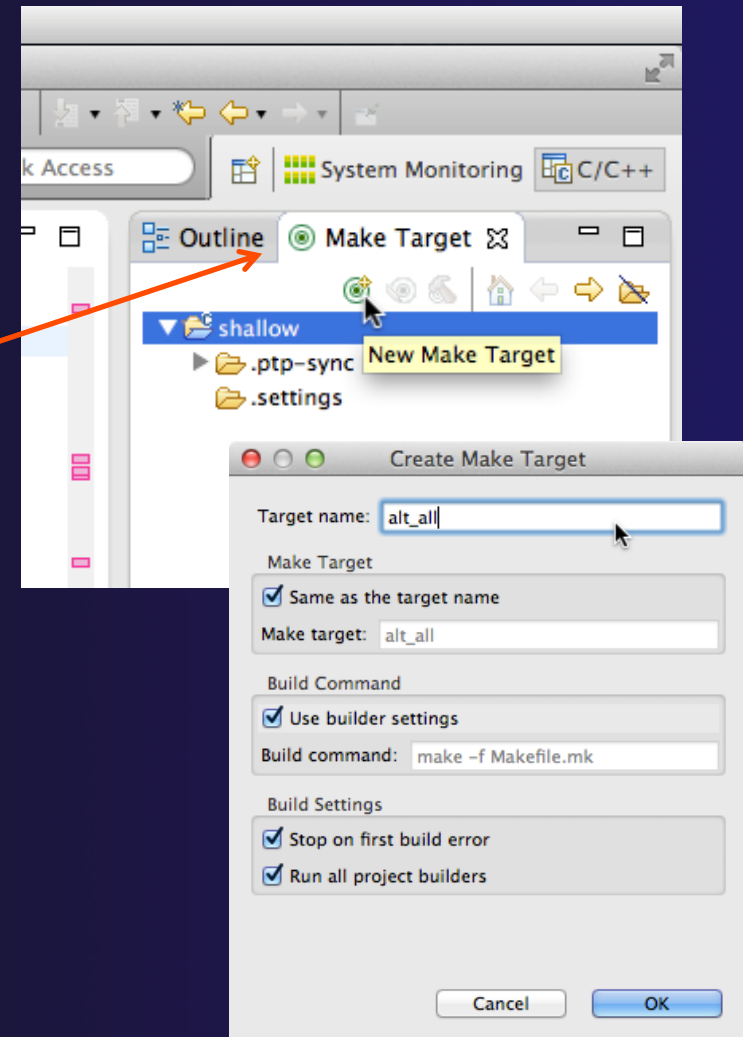
Forcing a Resync

- ✦ Project should resync with remote system when things change
- ✦ Sometimes you may need to do it explicitly
- ✦ Right mouse on project, **Synchronize>Sync Active Now**
- ✦ Status area in lower right shows when Synchronization occurs



Creating Make Targets

- ✦ By default
 - ✦ The build button will run “make all”
 - ✦ Cleaning a project will run “make clean”
- ✦ Sometimes, other build targets are required
- ✦ Open **Make Target** view
- ✦ Select project and click on **New Make Target** button
- ✦ Enter new target name
- ✦ Modify build command if desired
- ✦ New target will appear in view
- ✦ Double click on target to activate





Exercise

1. Start with your 'shallow' project
2. Build the project
3. Edit a source file and introduce a compile error
 - ✦ In main.c, line 97, change ';' to ':'
 - ✦ Save, rebuild, and watch the Console view
 - ✦ Use the Problems view to locate the error
 - ✦ Locate the error in the source code by double clicking on the error in the **Problems** view
 - ✦ Fix the error
4. Rebuild the project and verify there are no build errors



Optional Exercise

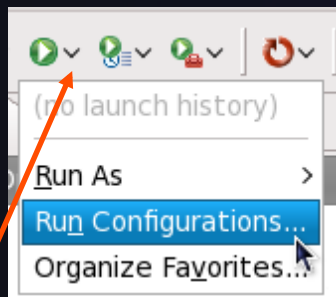
1. Open the Makefile in Eclipse. Note the line starting with “tags:” – this defines a make target named **tags**.
2. Open the **Outline** view while the Makefile is open. What icon is used to denote make targets in the Outline?
3. Right-click the **tags** entry in the Outline view. Add a Make Target for **tags**.
4. Open the **Make Target** view, and build the **tags** target.


5. Rename Makefile to Makefile.mk
6. Attempt to build the project; it will fail
7. In the project properties (under the C/C++ Build category), change the build command to: `make -f Makefile.mk`
8. Build the project; it should succeed

Running an Application

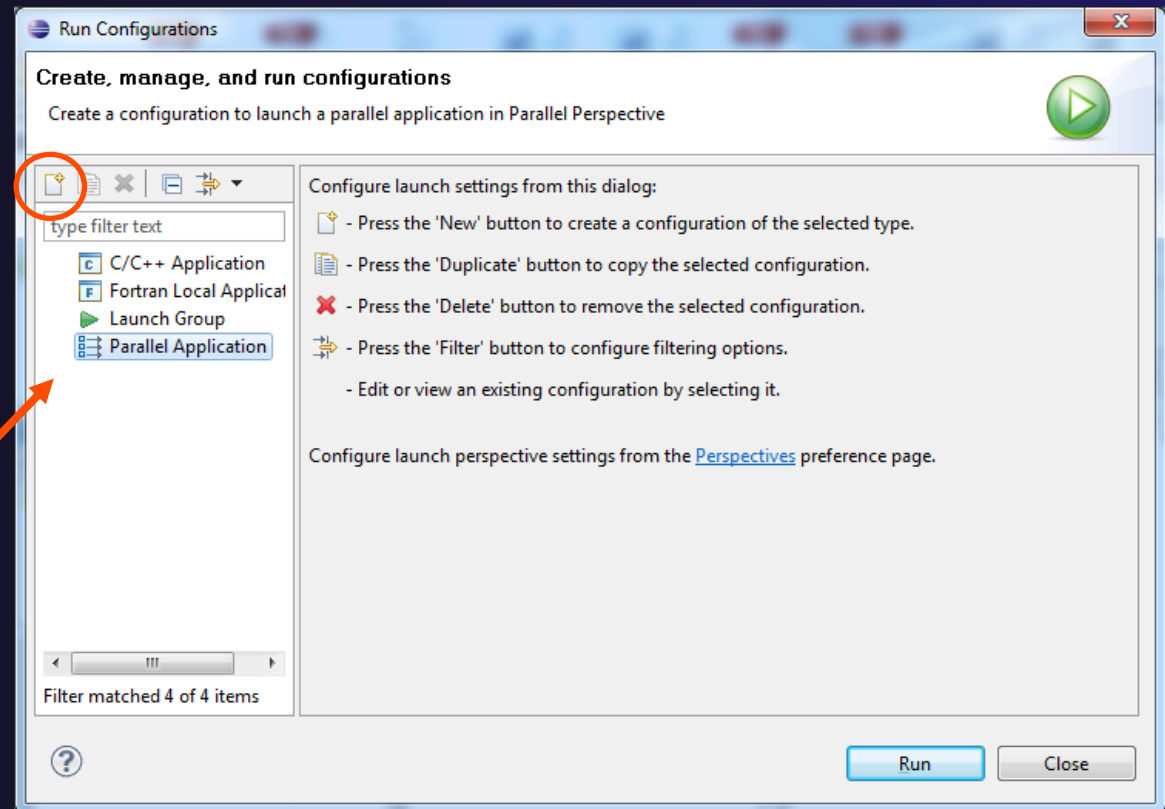
- ★ Objective
 - ★ Learn how to run an MPI program on a remote system
- ★ Contents
 - ★ Creating a run configuration
 - ★ Configuring the application run
 - ★ Monitoring the system and jobs
 - ★ Controlling jobs
 - ★ Obtaining job output

Creating a Run Configuration



- ★ Open the run configuration dialog **Run>Run Configurations...**
- ★ Select **Parallel Application**
- ★ Select the **New** button 

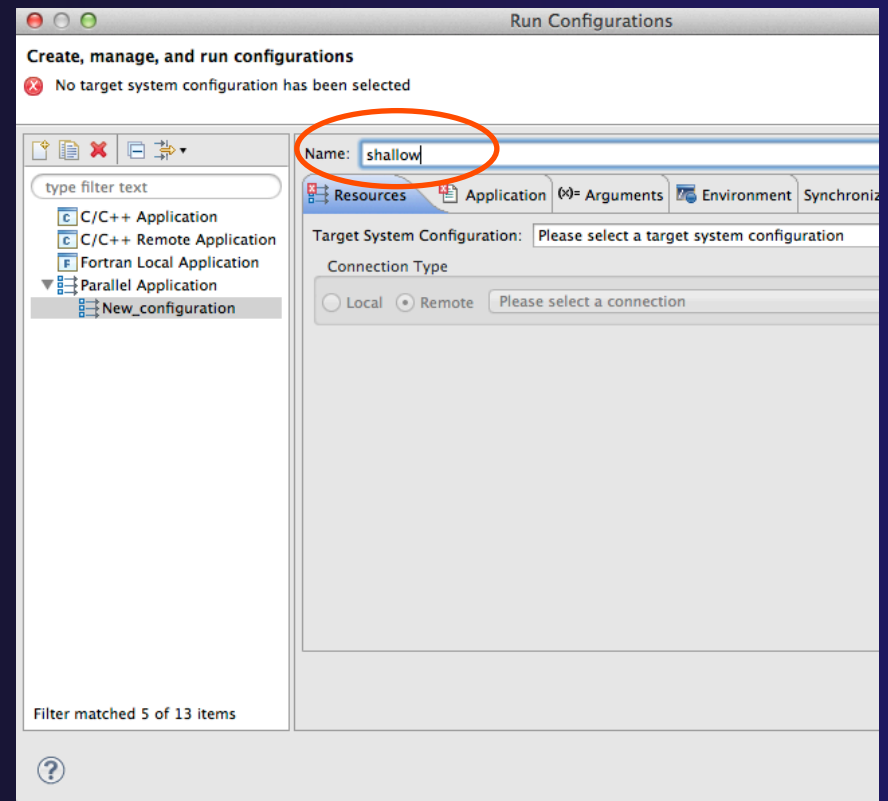
Or, just double-click on **Parallel Application** to create a new one



Note: We use "Launch Configuration" as a generic term to refer to either a "Run Configuration" or a "Debug Configuration", which is used for debugging.

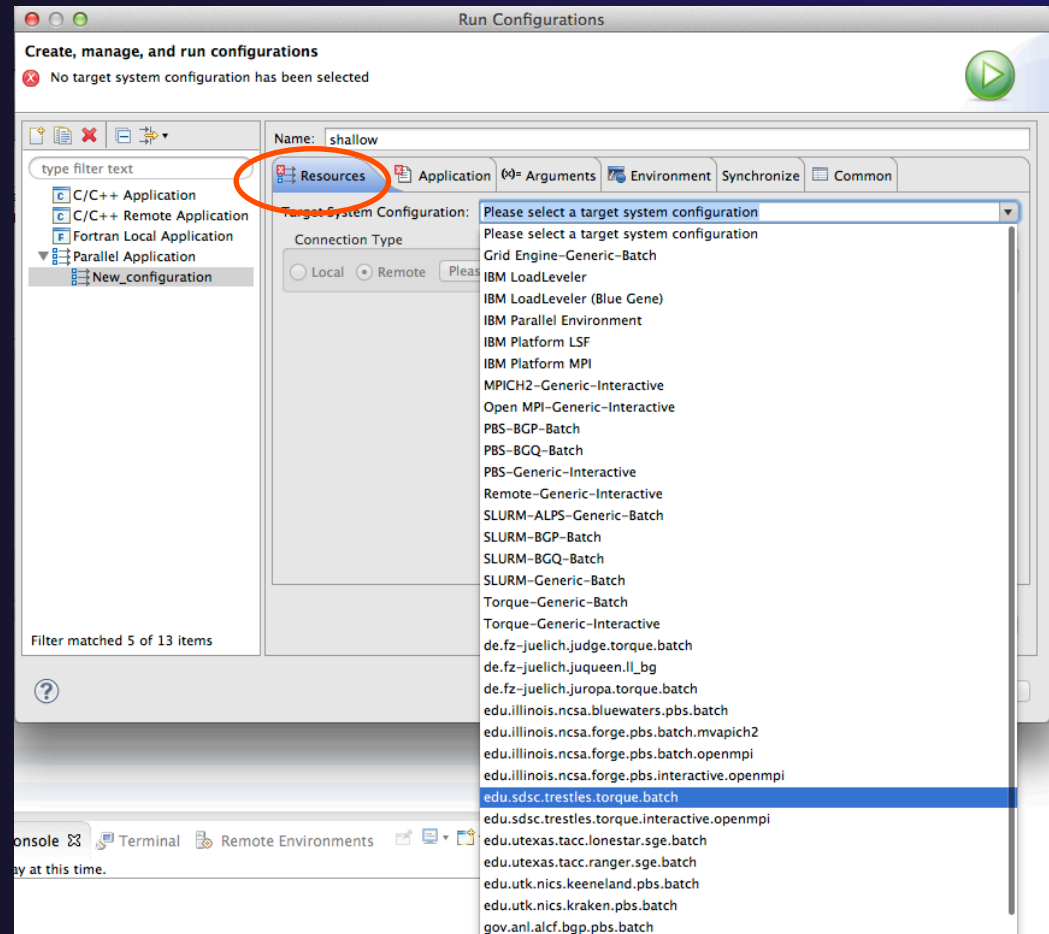
Set Run Configuration Name

- ✦ Enter a name for this run configuration
 - ✦ E.g. "shallow"
- ✦ This allows you to easily re-run the same application
- ✦ If the "shallow" project was selected when the dialog was opened, its name will be automatically entered



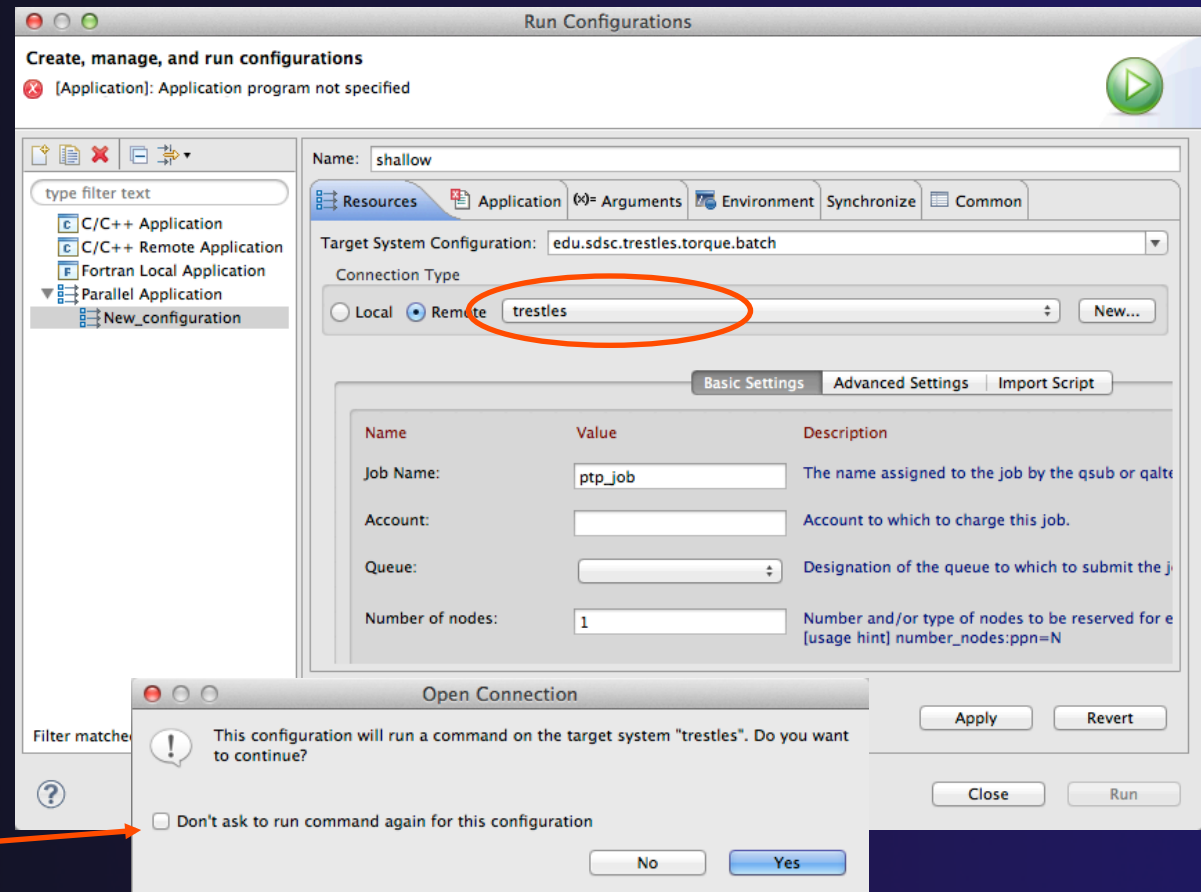
Configuring the Target System

- ★ In **Resources** tab, select a **Target System Configuration** that corresponds to your target system
 - ★ Use `edu.sdsc.trestles.torque.batch`
- ★ Target system configurations can be *generic* or can be specific to a particular system
- ★ Use the specific configuration if available, or the generic configuration that most closely matches your system
- ★ You can type text in the box to filter the configurations in the list



Configure the Connection

- ★ Choose a connection to use to communicate with the target system
- ★ If no connection has been configured, click on the **New** button to create a new one
 - ★ Fill in connection information, then click ok
- ★ The new connection should appear in the dropdown list
- ★ Select the connection you already have to *trestles.sdsc.edu*
- ★ Select toggle if you don't want to see popup again



Resources Tab

- ★ The content of the **Resources** tab will vary depending on the target system configuration selected
- ★ This example shows the TORQUE configuration
- ★ For TORQUE, you will normally need to select the *Queue* and the *Number of nodes*
- ★ For parallel jobs, choose the *MPI Command* and the *MPI Number of Processes*

Name: shallow

Resources Application Arguments Environment Synchronize Common

Target System Configuration: edu.sdsc.trestles.torque.batch

Connection Type
 Local Remote trestles

Basic Settings Advanced Settings Import Script

Name	Value	Description
Job Name:	ptp_job	The name assigned to the job by the qsub or qalter command.
Account:		Account to which to charge this job.
Queue:		Designation of the queue.
Number of nodes:	1	Number and/or type of nodes [usage hint] number of nodes
Total Memory Needed:		Maximum amount of memory needed.
Wallclock Time:	00:30:00	Maximum amount of wallclock time.
MPI Command:		Which mpi command to use.
MPI Number of Processes:	1	the '-np' value [usually equals Nodes*ppn]
Export Environment:	<input checked="" type="checkbox"/>	All variables in the qsub command's environment are to be exported to the batch job.
Modules to Load:	Configure...	Modules that will be loaded inside the job script.

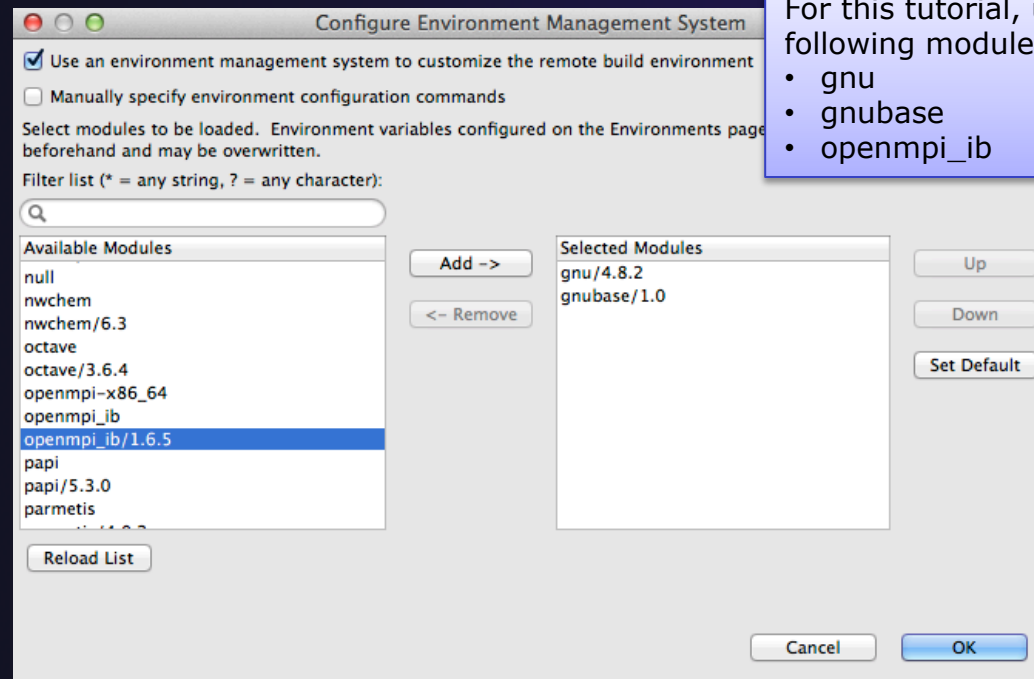
View Script View Configuration Restore Defaults

For this tutorial:

- Queue: **shared**
- Number of nodes: **1:ppn=5**
- MPI Command: **mpirun**
- MPI Number of Processes: **1**
- Leave other fields alone

Configure Environment Modules

- ✦ Click on the *Modules to Load: Configure...* button
- ✦ Check the *Use an environment management system to customize the remote build environment* box if it is not already checked
- ✦ Select the required modules and click **Add ->** (you can either select one at a time, or all at once)
- ✦ Click ok



For this tutorial, use the following modules:

- gnu
- gnubase
- openmpi_ib

Viewing the Job Script

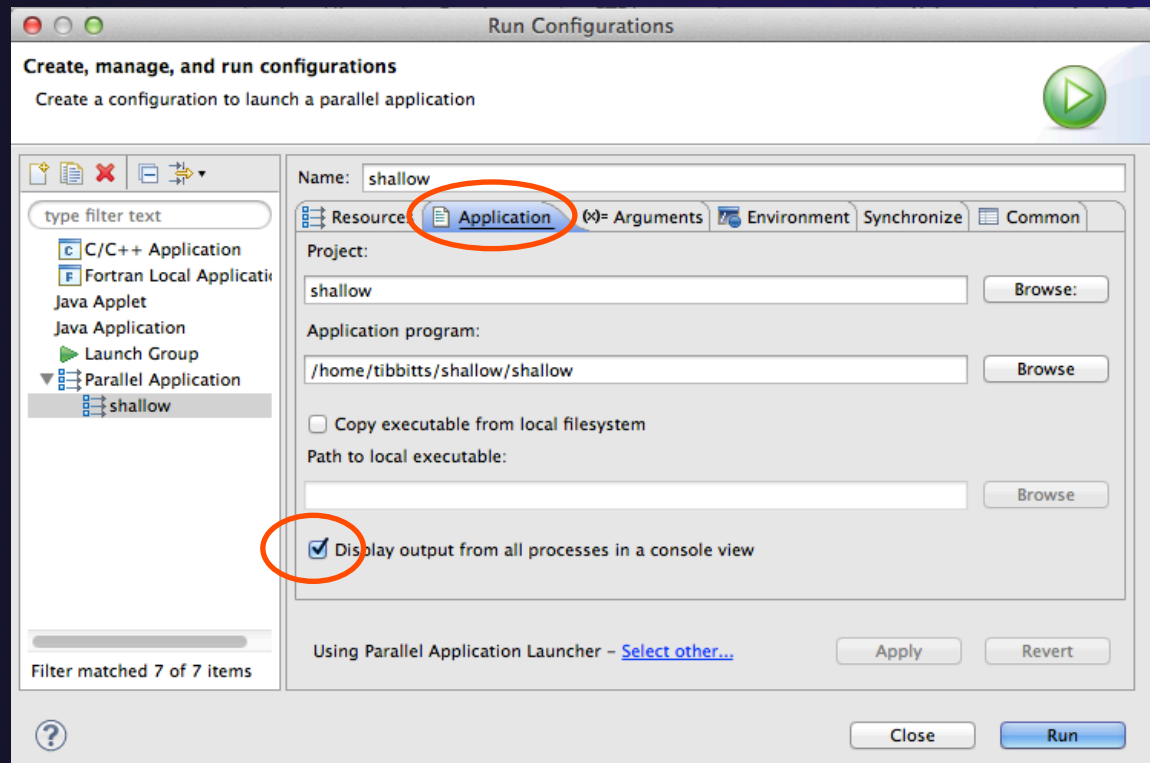
- ★ Some target configurations will provide a **View Script** button
- ★ Click on this to view the job script that will be submitted to the job scheduler
- ★ Batch scheduler configurations should also provide a means of importing a batch script

The screenshot shows a web-based job configuration interface. The 'View Script' button is highlighted with an orange arrow. A window titled 'Script with current values' is open, displaying the following job script:

```
#!/bin/bash --login
#PBS -q shared
#PBS -N ptp_job
#PBS -l nodes=1:ppn=5
#PBS -l walltime=00:30:00
#PBS -V
MPI_ARGS="-np 5"
if [ "-np" == "${MPI_ARGS}" ]; then
  MPI_ARGS=
fi
cd /oasis/scratch/trestles/$USER/$PBS_JOBID
cp /home/tibbitts/shallow/shallow .
MYSCREXE=`basename /home/tibbitts/shallow/shallow`
COMMAND=mpirun
if [ -n "$COMMAND" ]; then
  COMMAND="$COMMAND" ${MPI_ARGS} -hostfile ${PBS_NODEFILE} ${MYSCREXE} "
else
  COMMAND="${MYSCREXE} "
fi
${COMMAND}
```

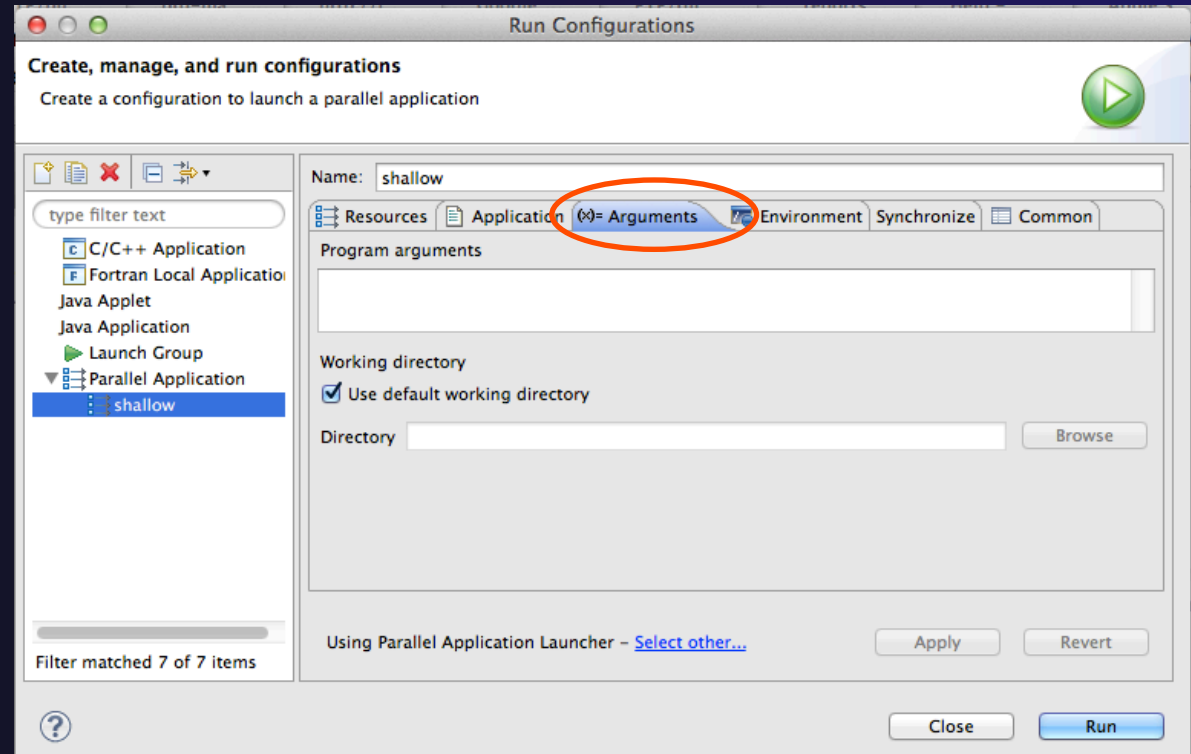
Application Tab

- ✦ Select the **Application** tab
- ✦ Choose the **Application program** by clicking the **Browse** button and locating the executable on the remote machine
 - ✦ Use the same "shallow" executable
- ✦ Select **Display output from all processes in a console view**



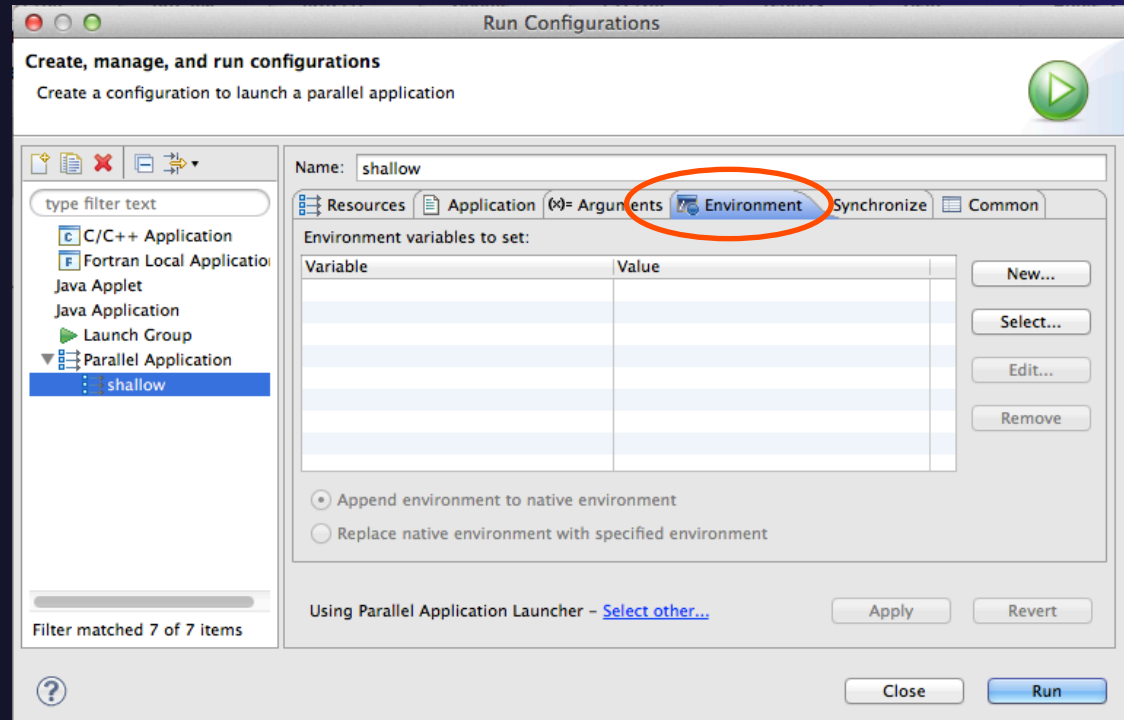
Arguments Tab (Optional)

- ★ The **Arguments** tab lets you supply command-line arguments to the application
- ★ You can also change the default working directory when the application executes



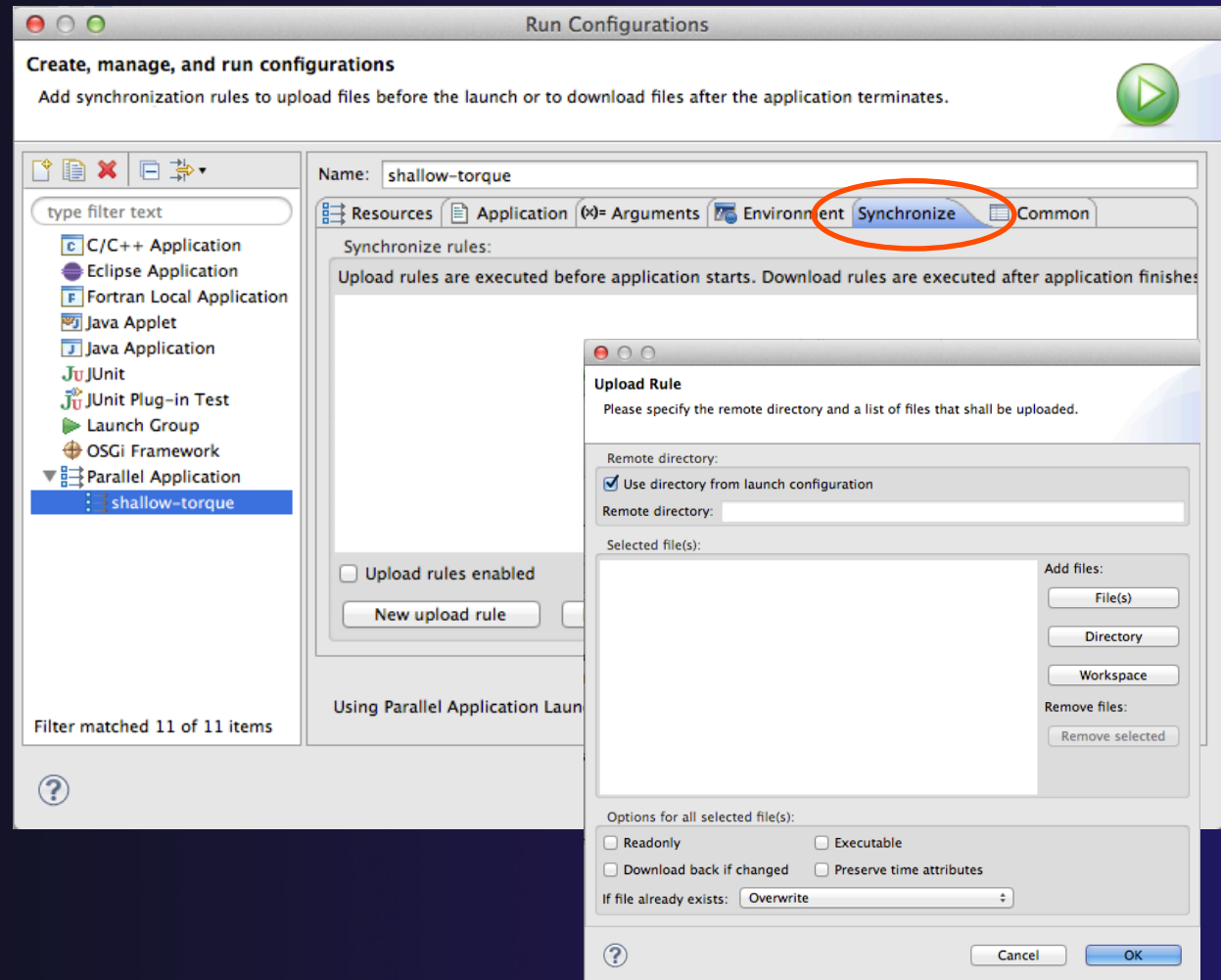
Environment Tab (Optional)

- ★ The **Environment** tab lets you set environment variables that are passed to the job submission command
- ★ This is independent of the Environment Management (module/softenv) support described on previous slide



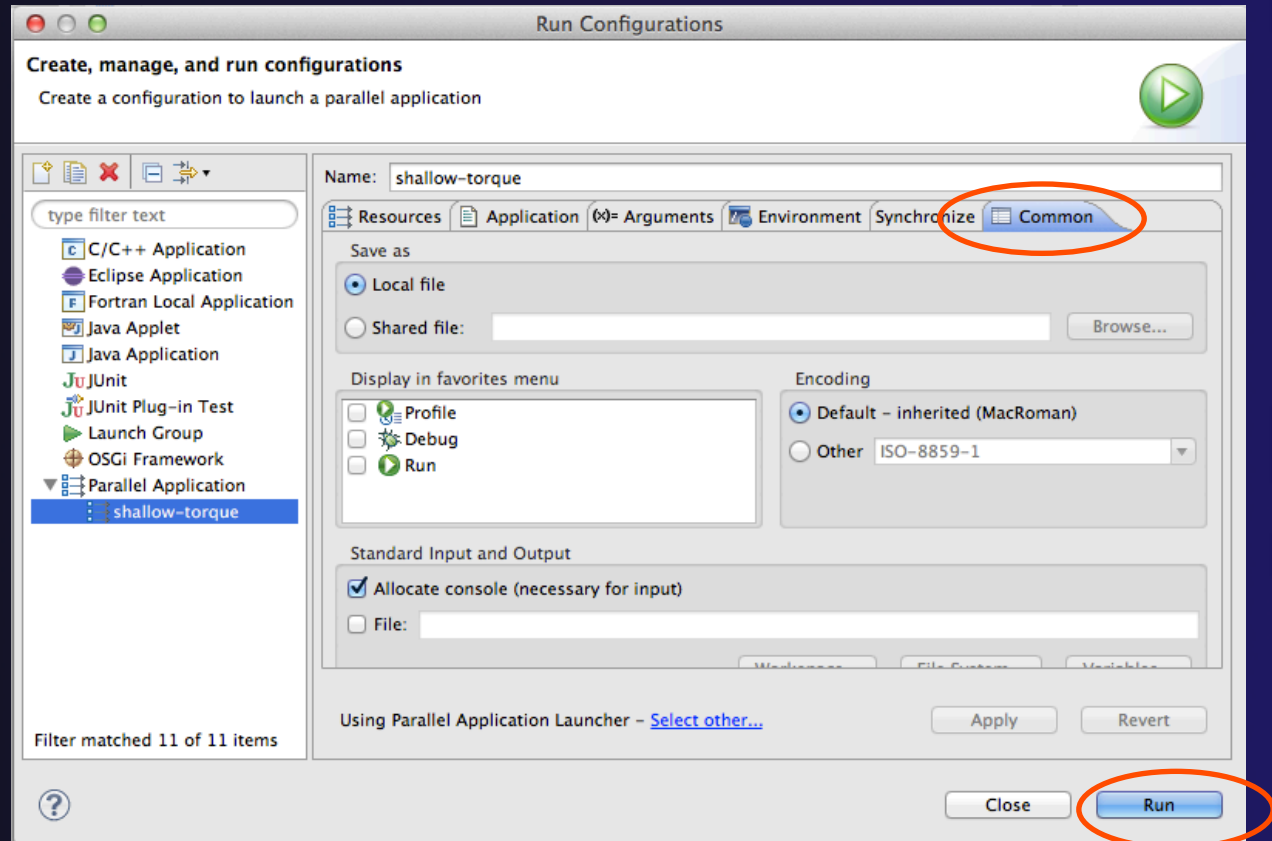
Synchronize Tab (Optional)

- ★ The **Synchronize** tab lets you specify upload/download rules that are execute prior to, and after the job execution
- ★ Click on the **New upload/download rule** buttons to define rules
- ★ The rule defines which file will be uploaded/downloaded and where it will be put
- ★ Can be used in conjunction with program arguments to supply input data to the application



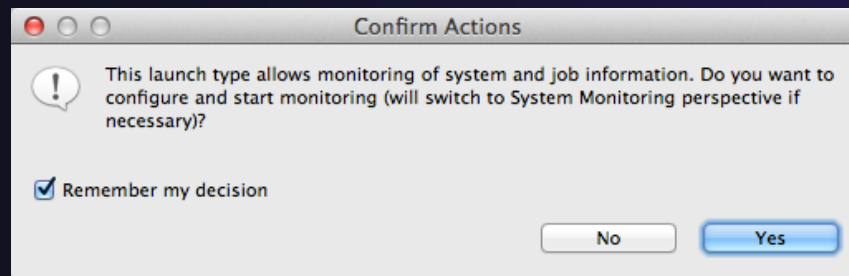
Common Tab (Optional)

- ★ The **Common** tab is available for most launch configuration types (not just Parallel Application)
- ★ Allows the launch configuration to be exported to an external file
- ★ Can add the launch configuration to the favorites menu, which is available on the main Eclipse toolbar
- ★ Select **Run** to launch the job



Run

- ✦ Select **Run** to launch the job
- ✦ You may be asked to switch to the System Monitoring Perspective



- ✦ Select **Remember my decision** so you won't be asked again
- ✦ Select **Yes** to switch and launch the job

System Monitoring Perspective

★ System view

★ Jobs running on system

★ Active jobs

★ Inactive jobs

★ Messages

★ Console

The screenshot shows the Eclipse IDE's System Monitoring perspective. On the left, there are several panels: 'Monitors' showing connection details for 'trestles.sdsc.edu'; 'Active Jobs' and 'Inactive Jobs' tables; and 'Messages' and 'Console' panels. The 'Active Jobs' table contains the following data:

step	owner	queue	wall	queued	dispatch	total
10553...	jmondal	normal	64800	2012...	201	201
10553...	jmondal	normal	64800	2012...	201	201
10553...	jmondal	normal	64800	2012...	201	201
10553...	jmondal	normal	64800	2012...	201	201

The 'Inactive Jobs' table contains the following data:

step	owner	queue	wall	queued	dispatch	total
1056...	tibbitts	shared	1800	201...	201...	5
1056...	tibbitts	shared	?	?	?	?
1056...	tibbitts	shared	?	?	?	?

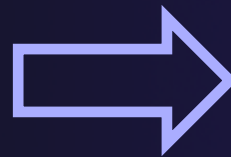
The right side of the interface features a large grid of resource monitors, each represented by a small grid of colored squares (blue, green, red, yellow, purple) indicating the status of different resources. A scroll bar is visible at the bottom of this grid.

Scroll to see more

Moving views

- ★ The System Monitoring Perspective overlaps the **Active Jobs** and **Inactive Jobs** views
- ★ To split them apart and see both at once, *drag* the tab for the **Inactive Jobs** view to the lower half of its area, and let go of mouse

step	owner	queue	wall	queue	dispat	totalcc	status
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED



step	owner	queue	wall	queue	dispat	totalcc	status
509...	alb...	eight	172...	20...	20...	24	RUNNING
509...	alb...	eight	172...	20...	20...	24	RUNNING
509...	rdel...	nor...	172...	20...	20...	4	RUNNING
509...	rdel...	nor...	172...	20...	20...	4	RUNNING

step	owner	queue	wall	queue	dispat	totalcc	status
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED
510...	llev...	nor...	129...	20...	?	16	SUBMITTED

System Monitoring

- ★ **System** view, with abstraction of system configuration
- ★ Hold mouse button down on a job in **Active Jobs** view to see where it is running in **System** view
- ★ Hover over node in **System** view to see job running on node in **Active Jobs** view

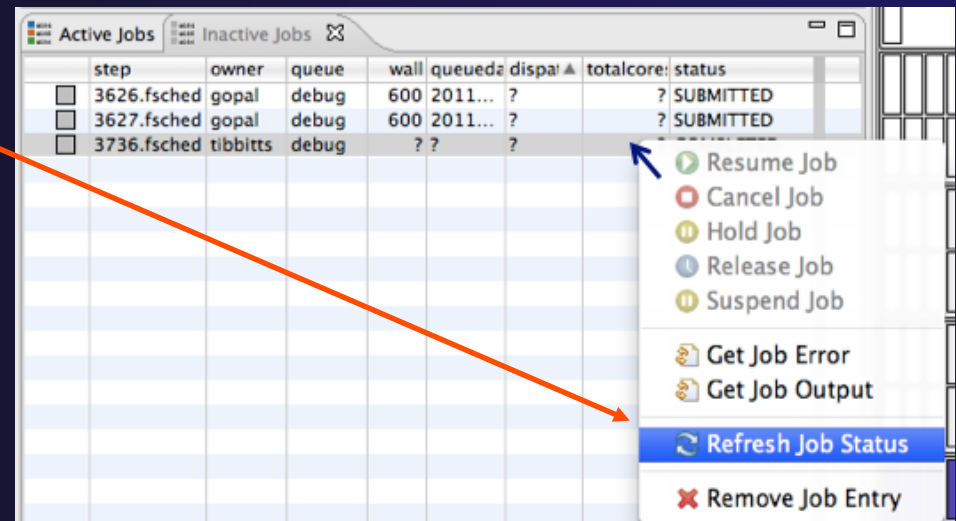
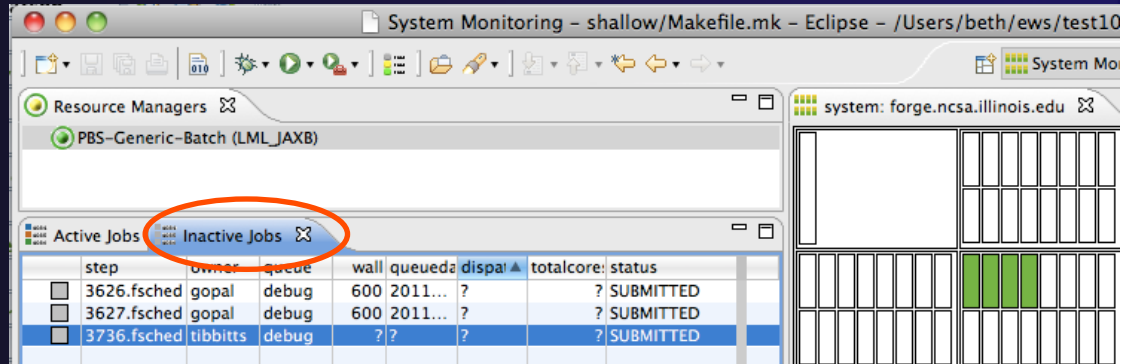
The screenshot displays the System Monitoring interface for the system 'forge.ncsa.illinois.edu'. It features three main panels on the left: 'Monitors', 'Active Jobs', and 'Inactive Jobs'. The 'Active Jobs' panel contains a table with columns for step, owner, queue, wall, queuec, dispatc, totalco, and status. The 'System' view on the right is a grid of nodes, each represented by a set of colored bars indicating job status. Red arrows point from the text annotations to specific elements in the interface: one to the 'forge.ncsa.illinois.edu' entry in the 'Monitors' panel, another to a job entry in the 'Active Jobs' table, and a third to a node in the 'System' view grid.

step	owner	queue	wall	queuec	dispatc	totalco	status
495...	rarijit	normal	172...	201...	201...	12	RUNNING
500...	rarijit	eight	50400	201...	201...	6	RUNNING
500...	rarijit	eight	43200	201...	201...	6	RUNNING
500...	rarijit	eight	43200	201...	201...	6	RUNNING
500...	rarijit	eight	43200	201...	201...	6	RUNNING
500...	rarijit	eight	43200	201...	201...	6	RUNNING
500...	rarijit	eight	82800	201...	201...	6	RUNNING
501...	mkb72	normal	172...	201...	201...	6	RUNNING
501...	mkb72	normal	172...	201...	201...	6	RUNNING

One node with
16 cores

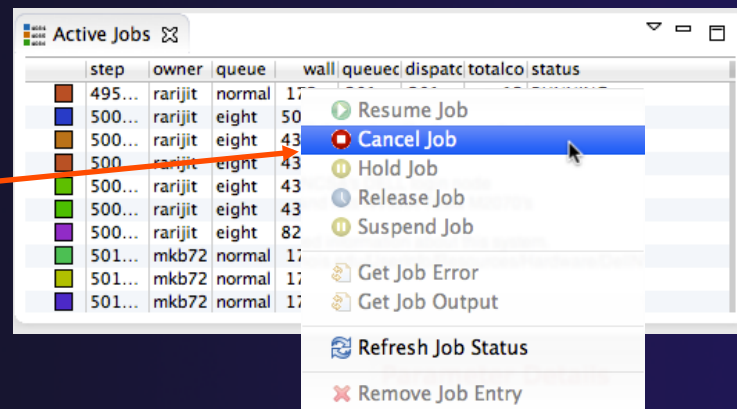
Job Monitoring

- ✦ Job initially appears in **Inactive Jobs** view
- ✦ Moves to the **Active Jobs** view when execution begins
- ✦ Returns to **Inactive Jobs** view on completion
- ✦ Status refreshes automatically every 60 sec
- ✦ Can force refresh with menu



Controlling Jobs

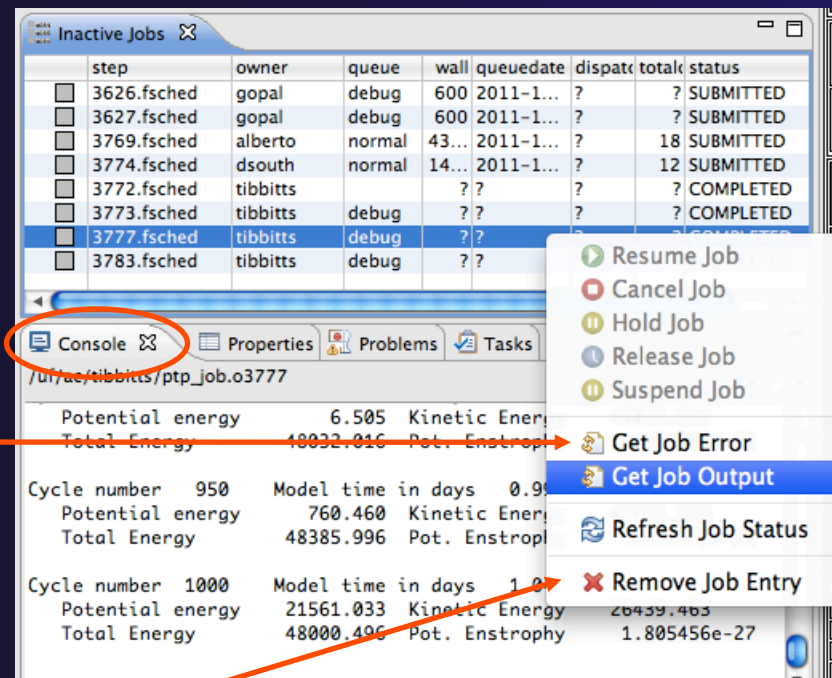
- ★ Right click on a job to open context menu
- ★ Actions will be enabled IFF
 - ★ The job belongs to you
 - ★ The action is available on the target system
 - ★ The job is in the correct state for the action
- ★ When job has COMPLETED, it will remain in the **Inactive Jobs** view



step	owner	queue	wall	queuenc	dispatc	totalco	status
501...	rarijit	eight	79200	201...	?	6	SUBMITTED
501...	rarijit	eight	79200	201...	?	6	SUBMITTED
501...	rarijit	eight	79200	201...	?	6	SUBMITTED
501...	rarijit	eight	79200	201...	?	6	SUBMITTED
502...	nvellor	normal	86400	201...	?	6	SUBMITTED
503...	boxu	normal	28800	201...	?	64	SUBMITTED
503...	boxu	normal	18000	201...	?	64	SUBMITTED
503...	boxu	normal	18000	201...	?	64	SUBMITTED
503...	boxu	normal	28800	201...	?	64	SUBMITTED
504...	alberto	eight	172...	201...	?	24	SUBMITTED
504...	alberto	eight	172...	201...	?	24	SUBMITTED
504...	inca	normal	300	201...	?	4	SUBMITTED
501...	grw		?	?	?	?	COMPLETED

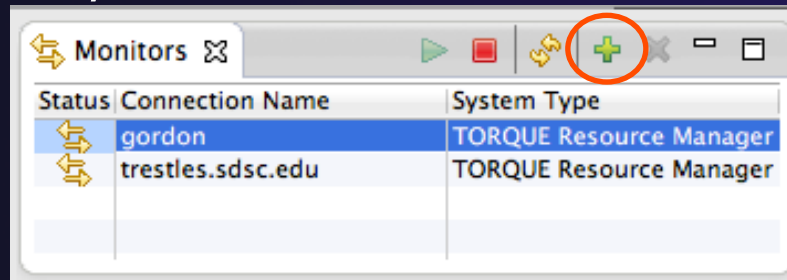
Obtaining Job Output

- ★ After status changes to **COMPLETED**, the output is available
 - ★ Right-click on the job
 - ★ Select **Get Job Output** to display output sent to standard output
 - ★ Select **Get Job Error** to retrieve output sent to standard error
- ★ Output/Error info shows in **Console View**
- ★ Jobs can be removed by selecting **Remove Job Entry**

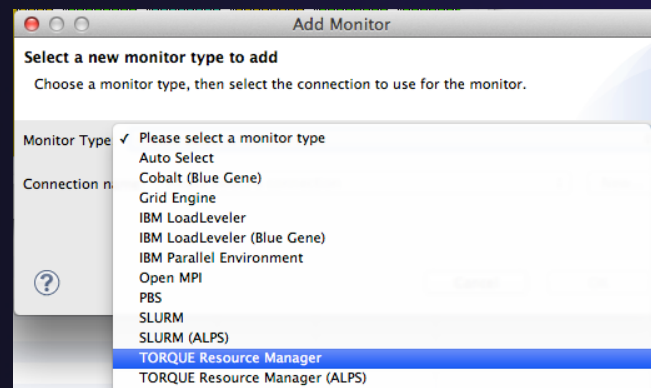


Add a Monitor

- ★ You can monitor other systems too
- ★ In **Monitors** view, select the '+' button to add a monitor



- ★ Choose monitor type and connection; create a new connection if necessary



Double click
new monitor
to start



Exercise

1. Start with your 'shallow' project
2. Create a run configuration
3. Complete the Resources tab
4. Select the executable in the Application tab
5. Submit the job
6. Check the job is visible in the Inactive Jobs view, moves to the Active Jobs view when it starts running (although it may be too quick to show up there), then moves back to the Inactive Jobs view when completed
7. View the job output
8. Remove the job from the Inactive Jobs view

Parallel Debugging

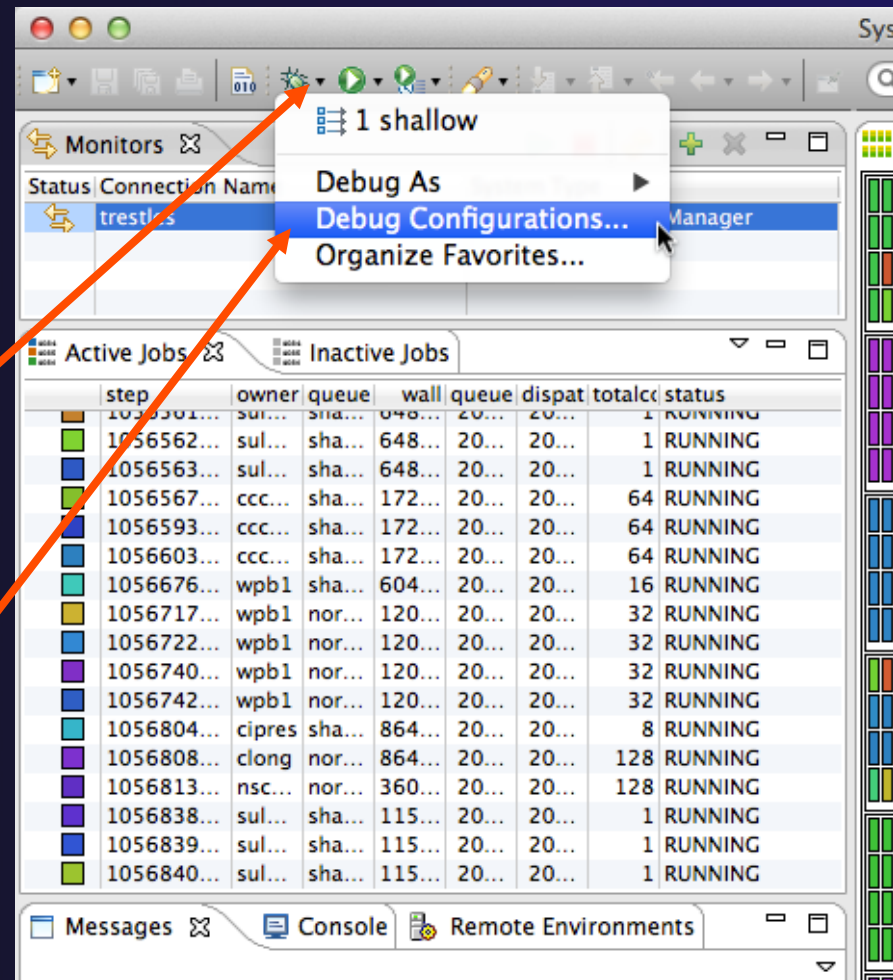
- ✦ Objective
 - ✦ Learn the basics of debugging parallel programs
- ✦ Contents
 - ✦ Launching a debug session
 - ✦ The Parallel Debug Perspective
 - ✦ Controlling sets of processes
 - ✦ Controlling individual processes
 - ✦ Parallel Breakpoints
 - ✦ Terminating processes

Debugging Setup

- ★ Debugging requires interactive access to the application
- ★ Can use any of the *-Interactive* target configurations
 - ★ *Torque-Generic-Interactive*
 - ★ *PBS-Generic-Interactive*
 - ★ *OpenMPI-Generic-Interactive*

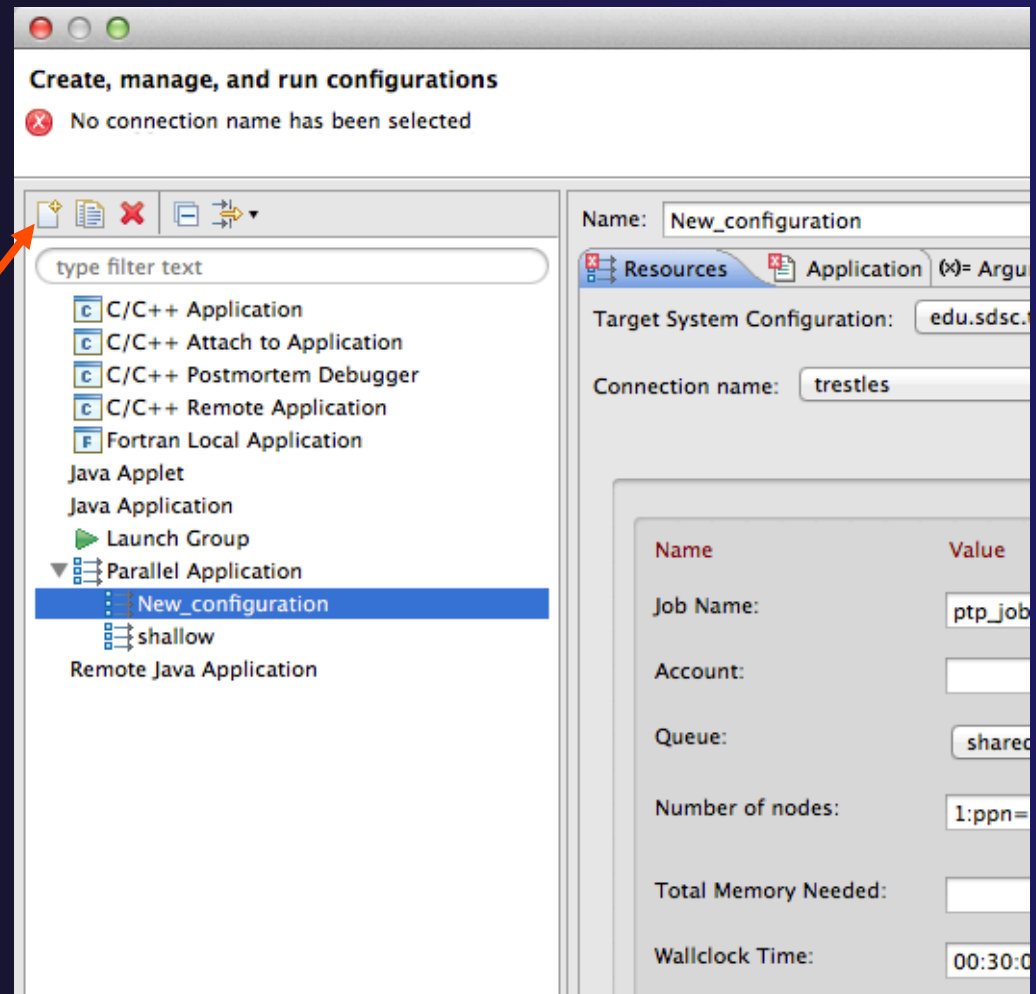
Create a Debug Configuration

- ★ A debug configuration is essentially the same as a run configuration (like we used in the *Running an Application* module)
- ★ It is possible to re-use an existing configuration and add debug information
- ★ Use the drop-down next to the debug button (bug icon) instead of run button
- ★ Select **Debug Configurations...** to open the **Debug Configurations** dialog



Create a New Configuration

- ★ Select the existing configuration
- ★ Click on the **new** button to create a new configuration



Configure the Resources Tab

- ★ Select the new target system configuration
- ★ Choose the queue
- ★ Make sure number of nodes is correct
- ★ Make sure the **mpirun** command is selected
- ★ Select the number of processes
- ★ Configure modules if required

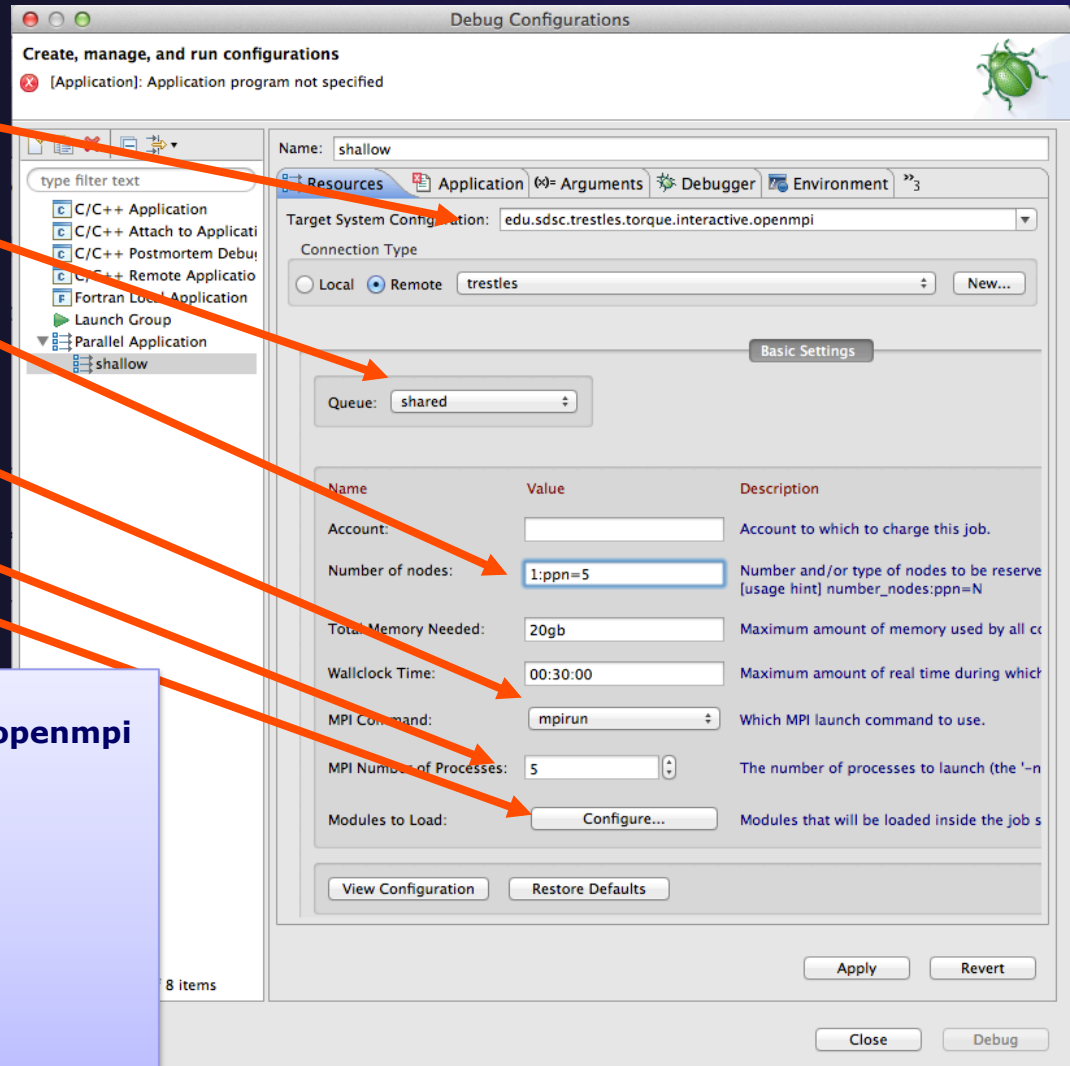
For this tutorial:

- Target: **edu.sdsc.trestles.torque.interactive.openmpi**
- Queue: **shared**
- Number of nodes: **1:ppn=5**
- MPI Command: **mpirun**
- MPI Number of Processes: **5**

Configure modules

- **gnu**
- **gnubase**
- **openmpi_ib**

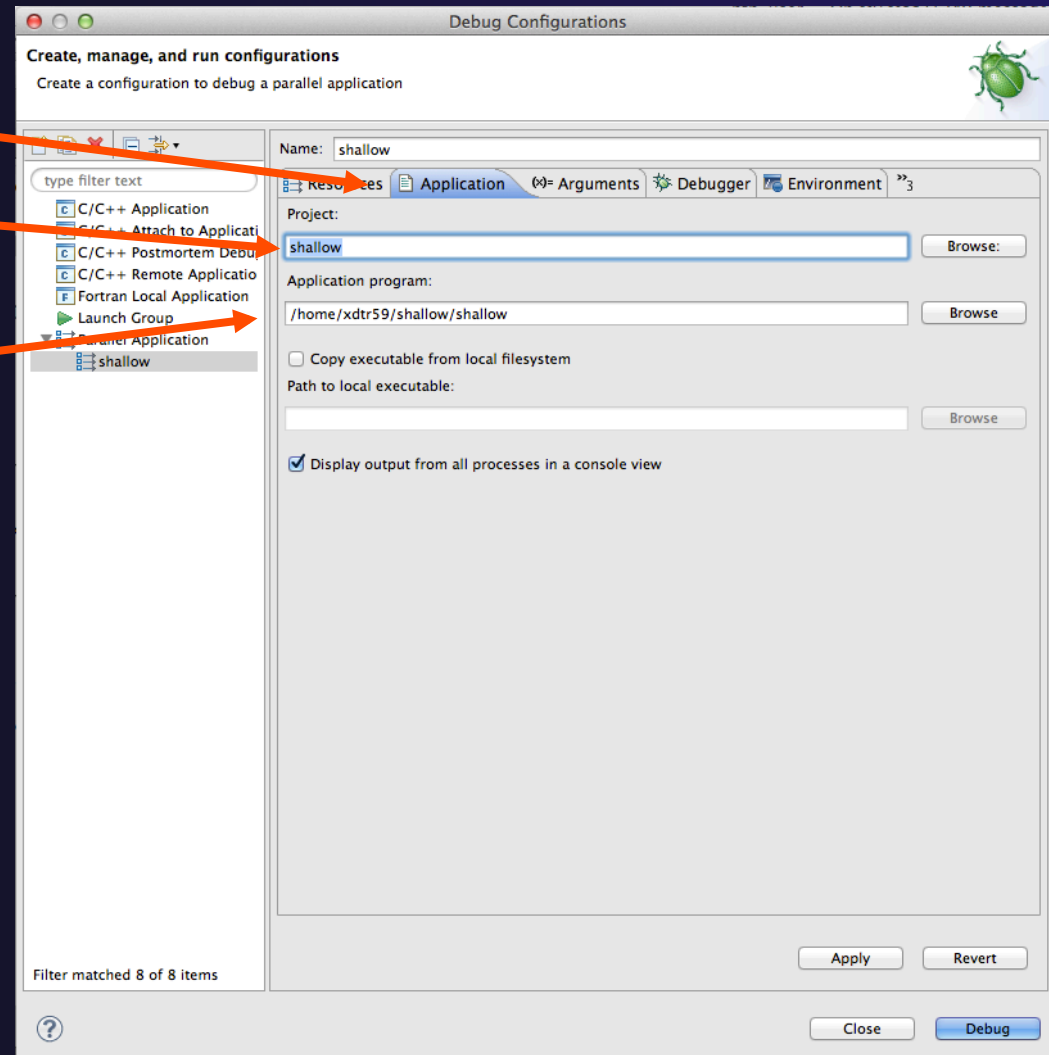
Parallel Debugging



Debug-4

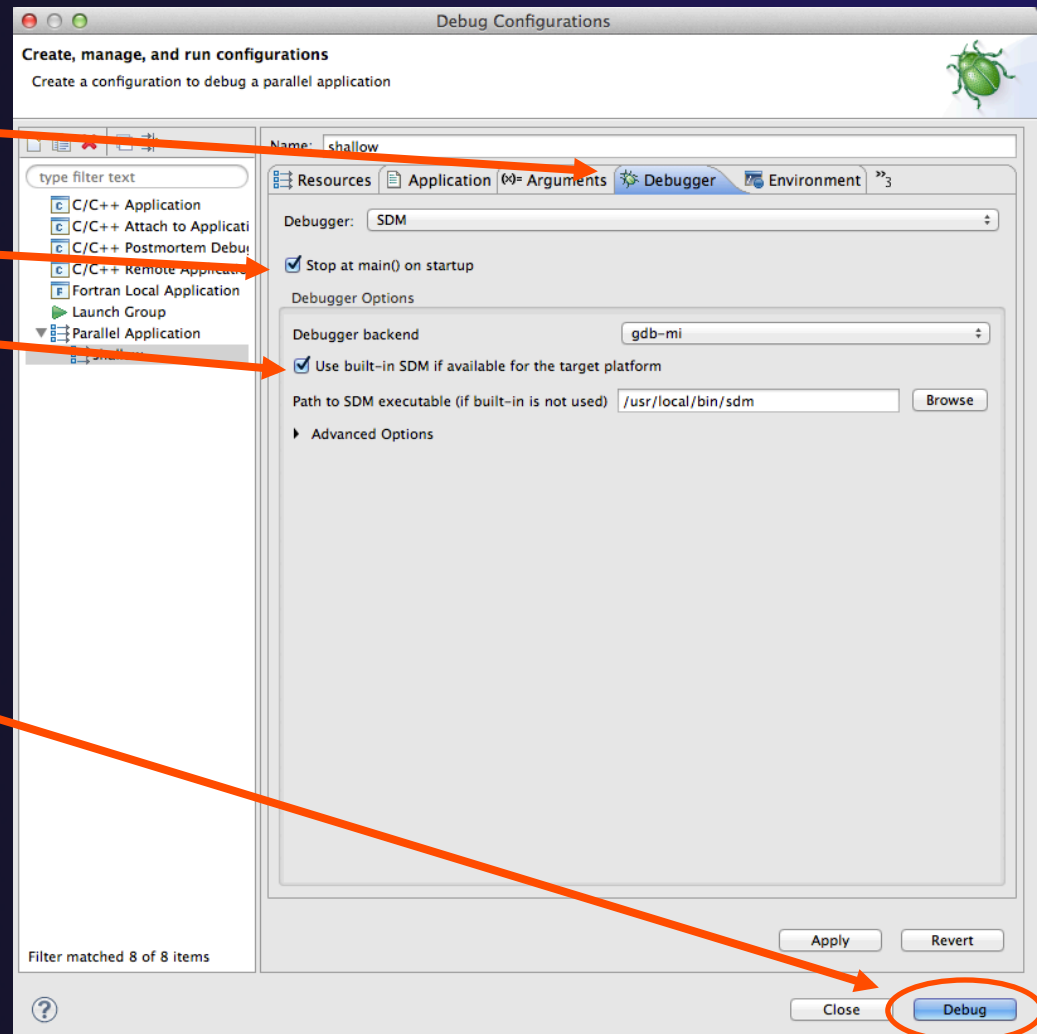
Configure the Application Tab (Optional)

- ✦ Select **Application** tab
- ✦ Make sure the **Project** is correct
- ✦ Select the application executable



Configure the Debug Tab (Optional)

- ★ Select **Debugger** tab
- ★ Debugger will stop at `main()` by default
- ★ By default the built-in SDM will be used
 - ★ Override this if you want to use your own SDM
- ★ Click on **Debug** to launch the program





Exercise

1. Open the debug configuration dialog
2. Create a new configuration
3. Select the *edu.sdsc.trestles.torque.interactive.openmpi* target configuration
4. Configure the **Debug** tab
 - ✦ Queue: *shared*
 - ✦ Number of nodes: *1:ppn=5*
 - ✦ MPI Command: *mpirun*
 - ✦ MPI Number of Processes: *5*
5. Configure the environment modules
 - ✦ *gnu*
 - ✦ *gnubase*
 - ✦ *openmpi_ib*
6. Launch the debugger

The Parallel Debug Perspective (1)

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

The screenshot displays the Eclipse IDE in the Parallel Debug perspective. The top toolbar includes various debugging actions. The main interface is divided into several panes:

- Parallel Debug:** Shows a job with multiple processes. An orange arrow points to this pane from the text "Parallel Debug view shows job and processes being debugged".
- Debug:** Shows a thread with a call stack. An orange arrow points to this pane from the text "Debug view shows threads and call stack for individual processes".
- Source:** Shows the main.c file with a current line marker at line 38. An orange arrow points to this pane from the text "Source view shows a current line marker for all processes".
- Memory:** Shows a table of variables and their values.
- Error Log:** Shows any error messages.
- Problems:** Shows any problem messages.

The console at the bottom shows the output of the application, including the job ID and process ID.

The Parallel 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

The screenshot displays the Eclipse IDE in the Parallel Debug perspective. The top toolbar contains various debugging icons. The Breakpoints view shows a list of breakpoints. The Debug view shows a tree structure of the application and its processes. The Variables view displays a table of variables and their values:

Name	Value
argc	1
argv	7fffffff7f8
pi	0.0
p	[0 - 16]
u	[0 - 16]
v	[0 - 16]
psi	[0 - 16]
pold	[0 - 16]
uold	[0 - 16]

The source code editor shows the following code snippet:

```

1
4 * Commonwealth Scientific and Industrial Research Organisation (CSIRO) *
25
26 #include <math.h>
27 #include <mpi.h>
28 #include <stdio.h>
29 #include "decs.h"
30
31 extern void worker();
32 MPI_Datatype * setup_res();
33
34 main (argc, argv)
35     int argc;
36     char * argv[];
37 {
38     float pi=4.*(float)atan((double)1.);
39     float p[n][m]; /* Pressure (or free surface height) */
40     float u[n][m]; /* Zonal wind */

```

The Outline view shows the project structure and the current file's outline. The Console view shows the output of the application.

Stepping All Processes

- ★ The buttons in the **Parallel Debug View** control groups of processes
- ★ The **Step Over** button will step all processes one line
- ★ The process icons will change to green (running), then back to yellow (suspended)
- ★ The current line marker will move to the next source line

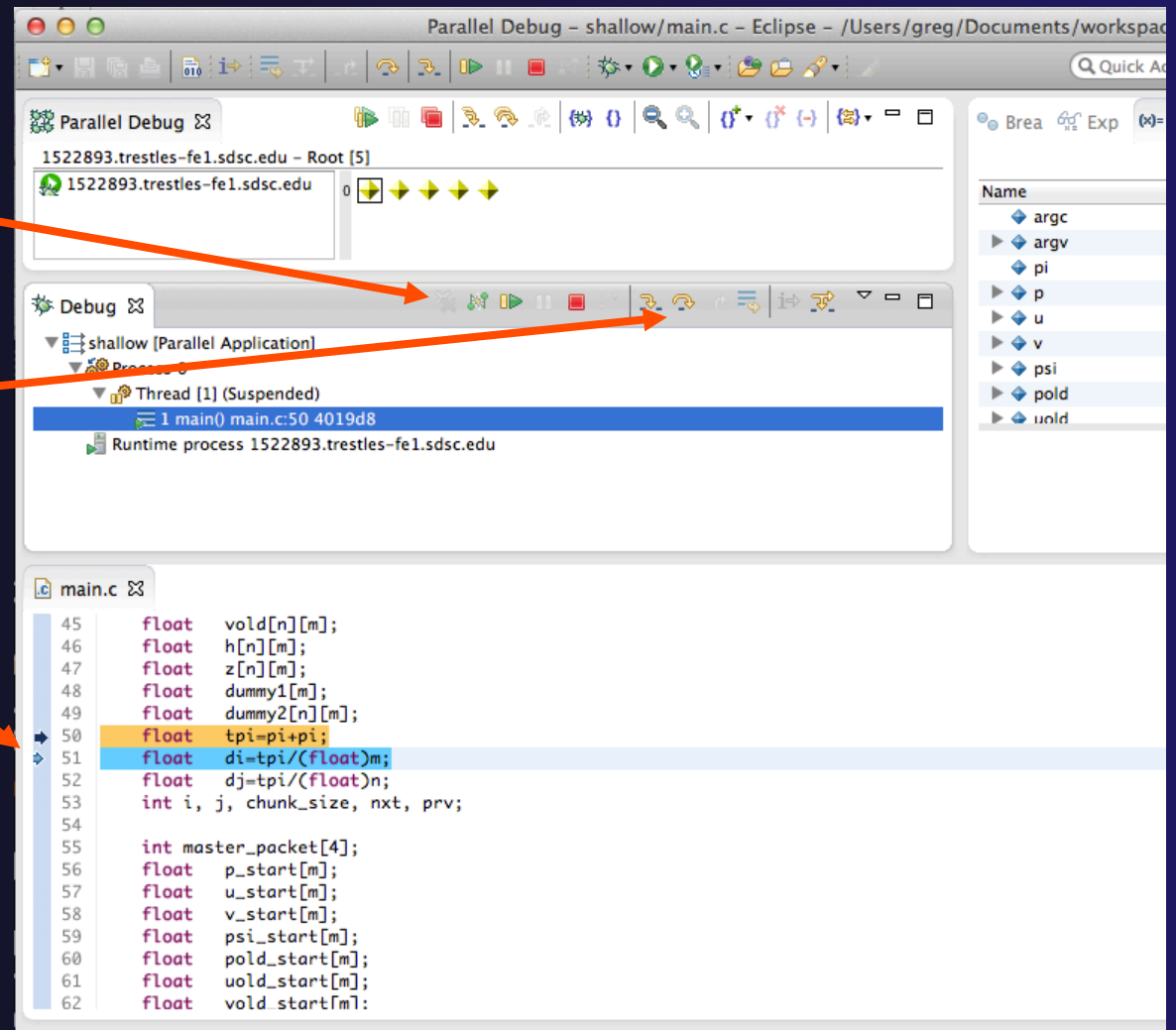
The screenshot shows the Eclipse IDE interface during a parallel debug session. The top toolbar contains various debugging controls. The 'Parallel Debug View' (top left) shows a tree structure with process icons. The 'Debug View' (middle) shows the current thread and process. The 'Source Editor' (bottom) shows the source code for 'main.c' with line 50 highlighted. Red arrows point from the text on the left to the corresponding UI elements in the screenshot.

```

45 float vold[n][m];
46 float h[n][m];
47 float z[n][m];
48 float dummy1[m];
49 float dummy2[n][m];
50 float tpi=pi+pi;
51 float di=tpi/(float)m;
52 float dj=tpi/(float)n;
53 int i, j, chunk_size, nxt, prv;
54
55 int master_packet[4];
56 float p_start[m];
57 float u_start[m];
58 float v_start[m];
59 float psi_start[m];
60 float pold_start[m];
61 float uold_start[m];
62 float vold_start[m];
  
```

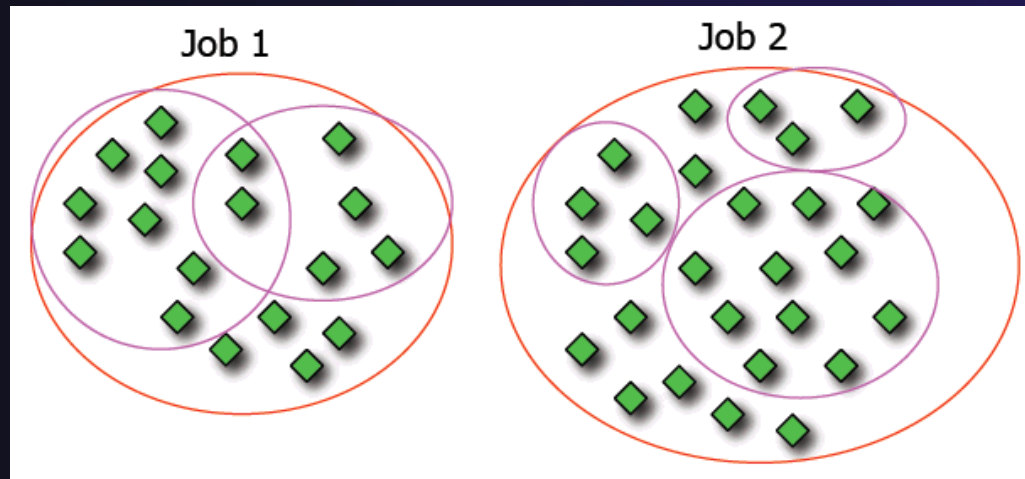
Stepping An Individual Process

- ★ The buttons in the **Debug view** are used to control an individual process, in this case process 0
- ★ The **Step Over** button will control just the one process
- ★ There are now two current line markers, the first shows the position of process 0, the second shows the positions of processes 1-4



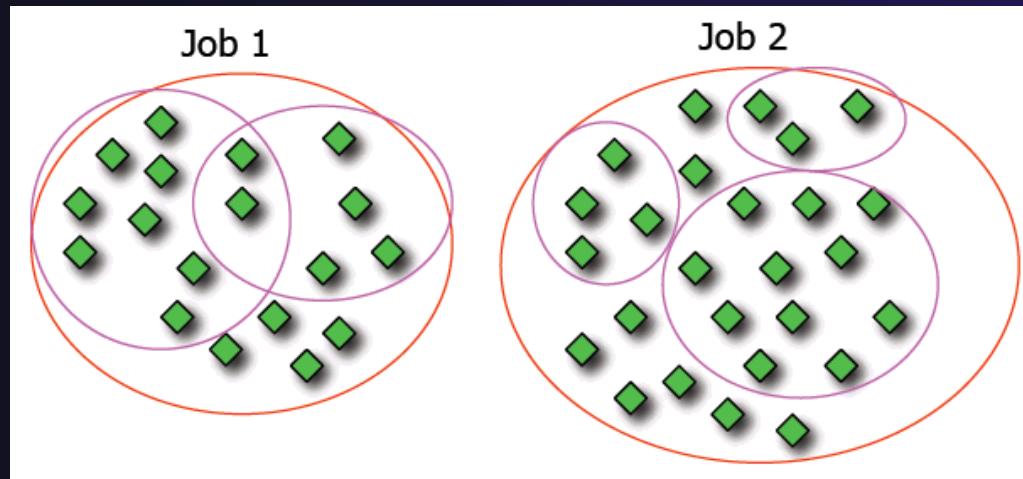
Process Sets (1)

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



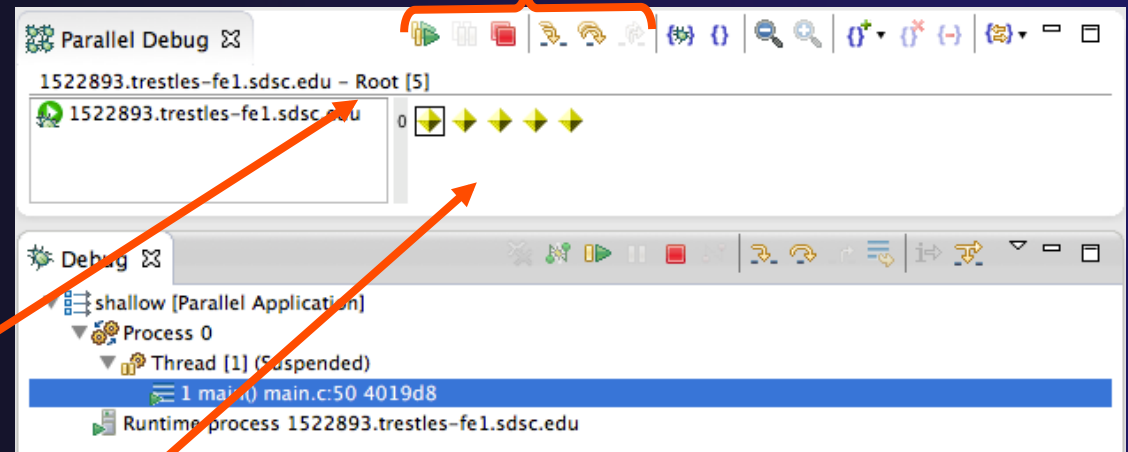
Process Sets (2)

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



Operations On Process Sets

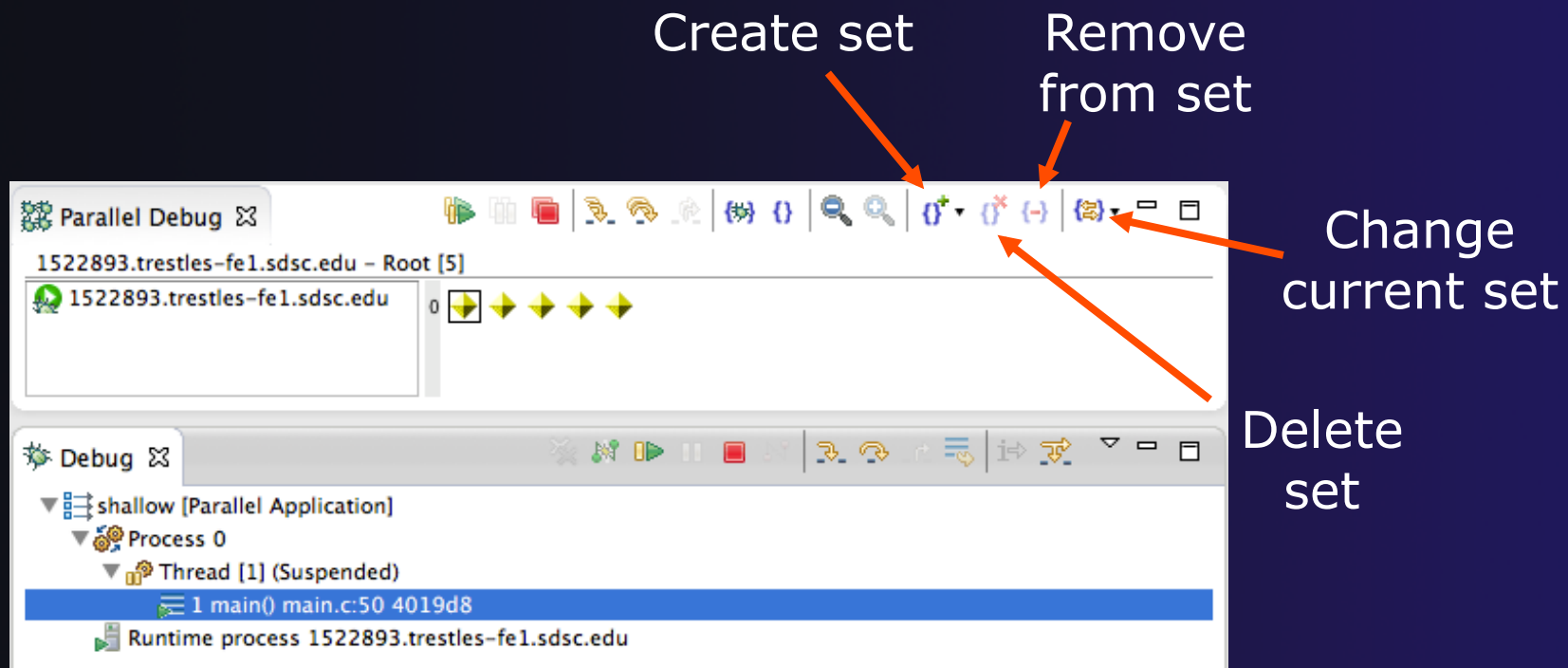
- ★ Debug operations on the **Parallel Debug view** toolbar always apply to the current set:
 - ★ Resume, suspend, stop, step into, step over, step return
- ★ The current process set is listed next to job name along with number of processes in the set
- ★ The processes in process set are visible in right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



Root set = all processes

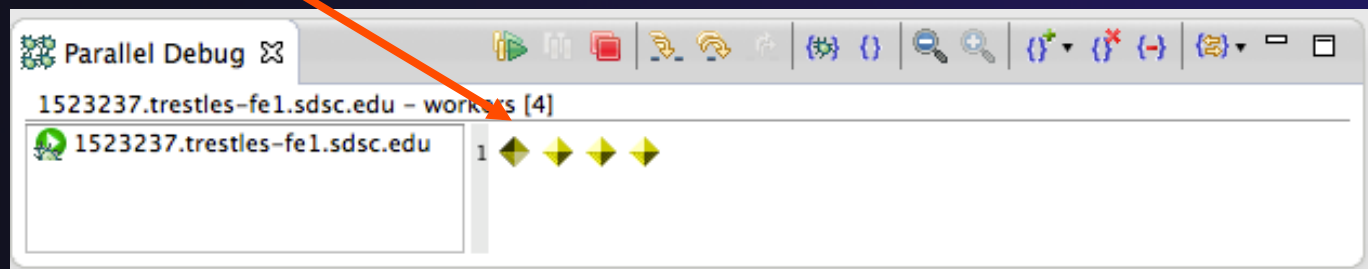
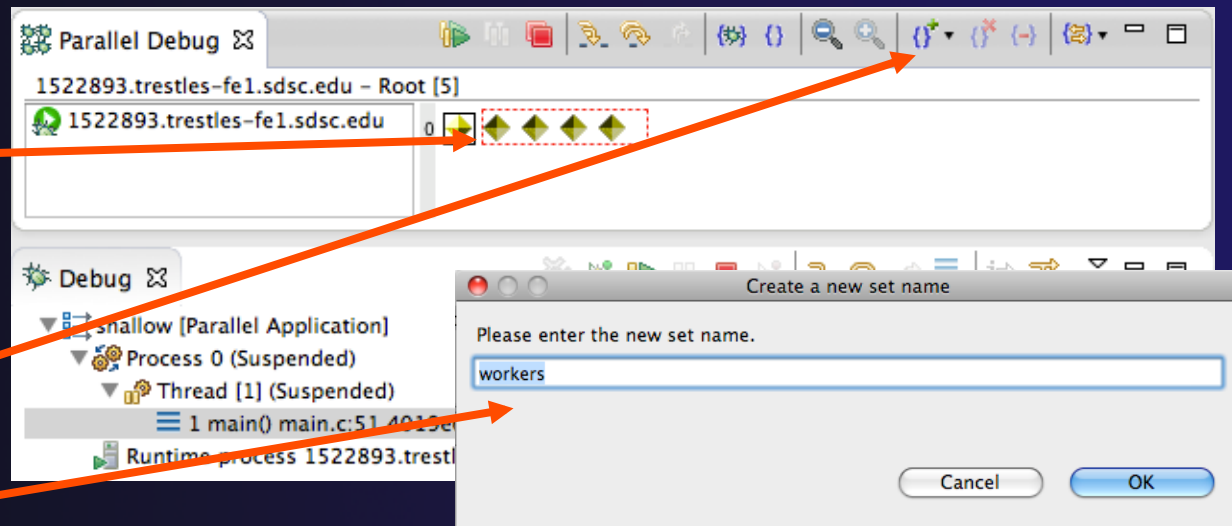
Managing Process Sets

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



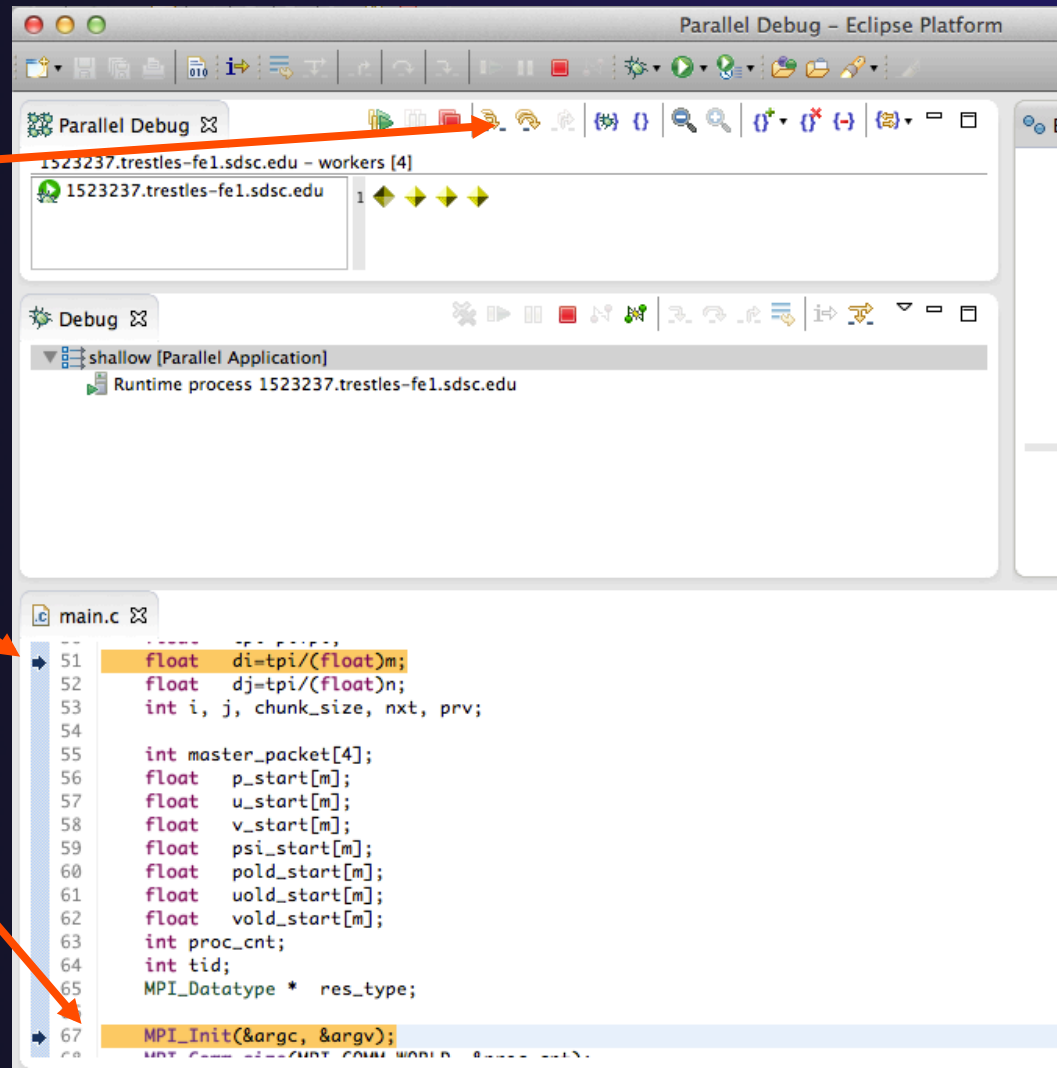
Creating A New Process Set

- ★ Select the processes in the set by clicking and dragging, in this case, the last three
- ★ The **Create Set** button enables a new process set to be created
- ★ The set can be given a name, in this case **workers**
- ★ The view is changed to display only the selected processes



Stepping Using New Process Set

- ★ With the **workers** set active, the **Step Over** button will now operated on only these processes
- ★ Only the first line marker will move
- ★ After stepping a couple more times, two line markers will be visible, one for the single master process, and one for the 4 worker processes

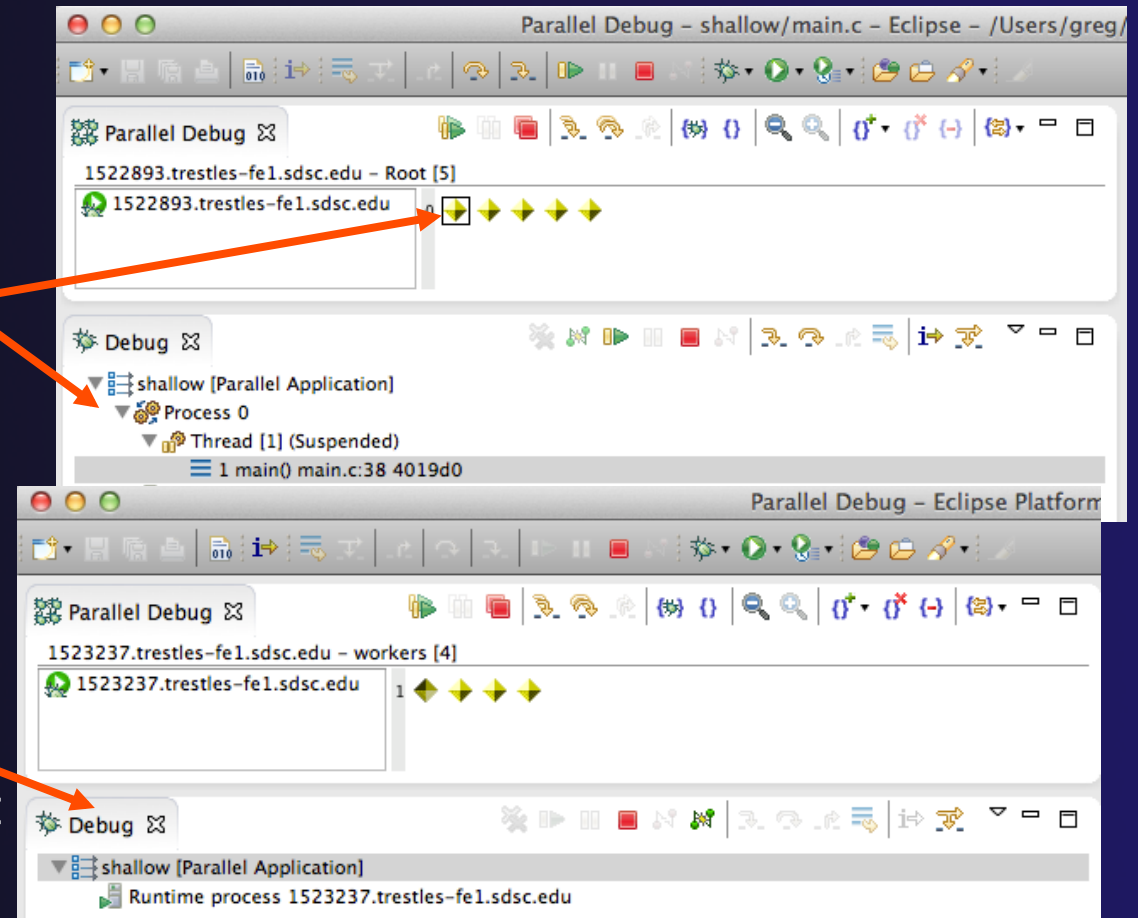


Process Registration

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

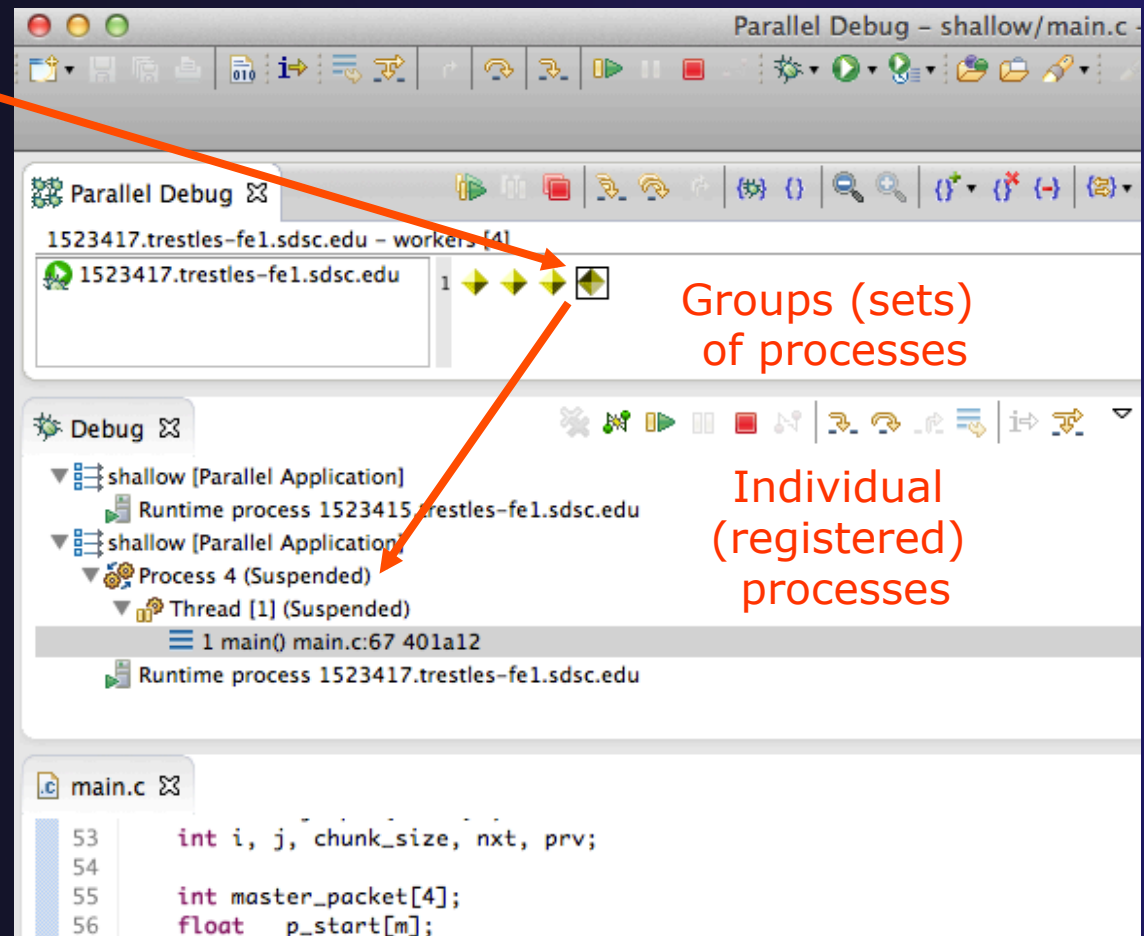
Process Registration (2)

- ✦ By default, process 0 was registered when the debug session was launched
- ✦ Registered processes are surrounded by a box and shown in the Debug view
- ✦ The Debug view only shows registered processes in the current set
- ✦ Since the “workers” set doesn't include process 0, it is no longer displayed in the Debug view



Registering A Process

- ★ To register a process, double-click its process icon in the **Parallel Debug view** or select a number of processes and click on the **register** button
- ★ To un-register a process, double-click on the process icon or select a number of processes and click on the **unregister** button



Current Line Marker

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

Colors And Markers

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

```

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

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

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



Multiple processes marker



Registered process marker



Un-registered process marker



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

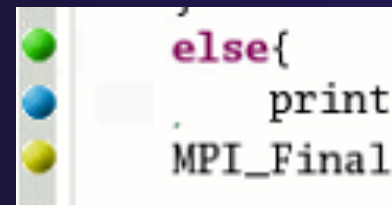


Exercise

1. From the initial debugger session, step all processes until the current line is just after MPI_Init (line 68)
2. Create a process set called “workers” containing processes 1-4
3. Step the “worker” processes twice, observe two line markers
4. Hover over markers to see properties
5. Switch to the “root” set
6. Step only process 0 twice so that all processes are now at line 71 (hint – use the debug view)

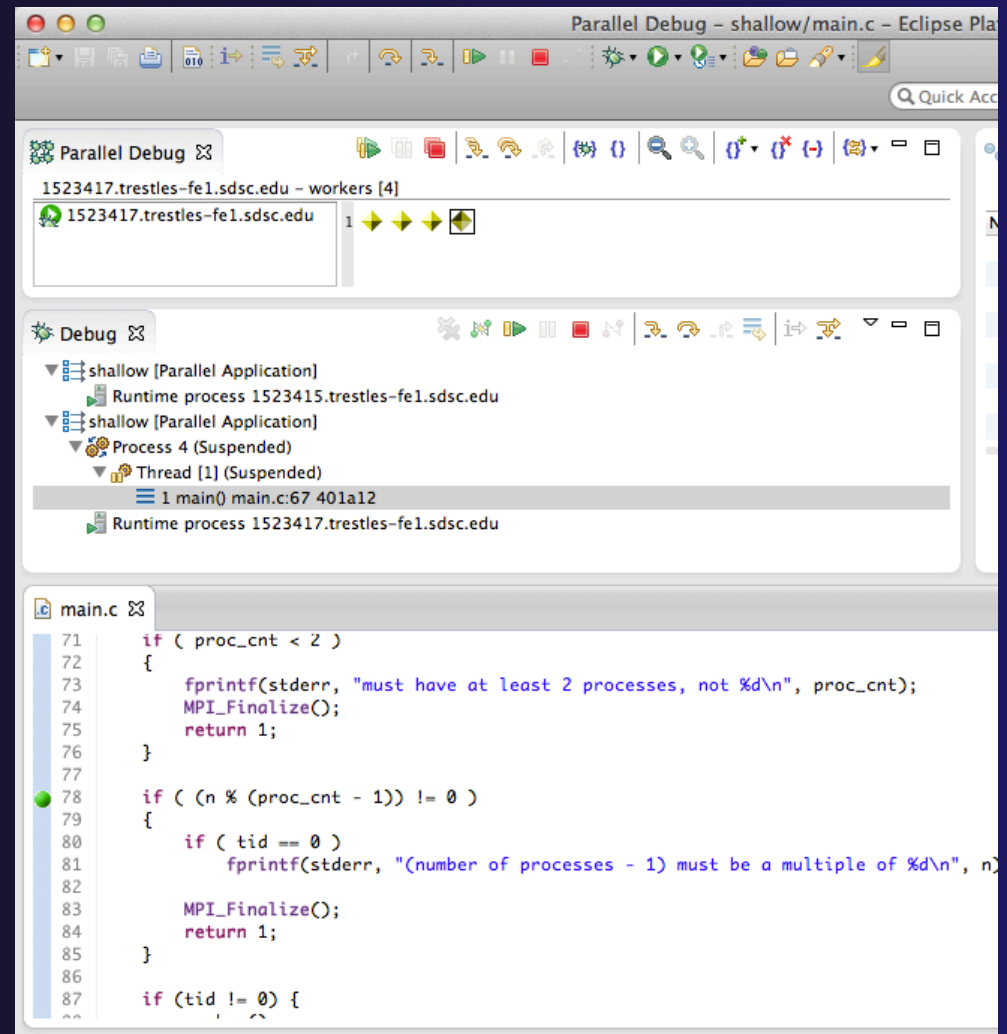
Breakpoints

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



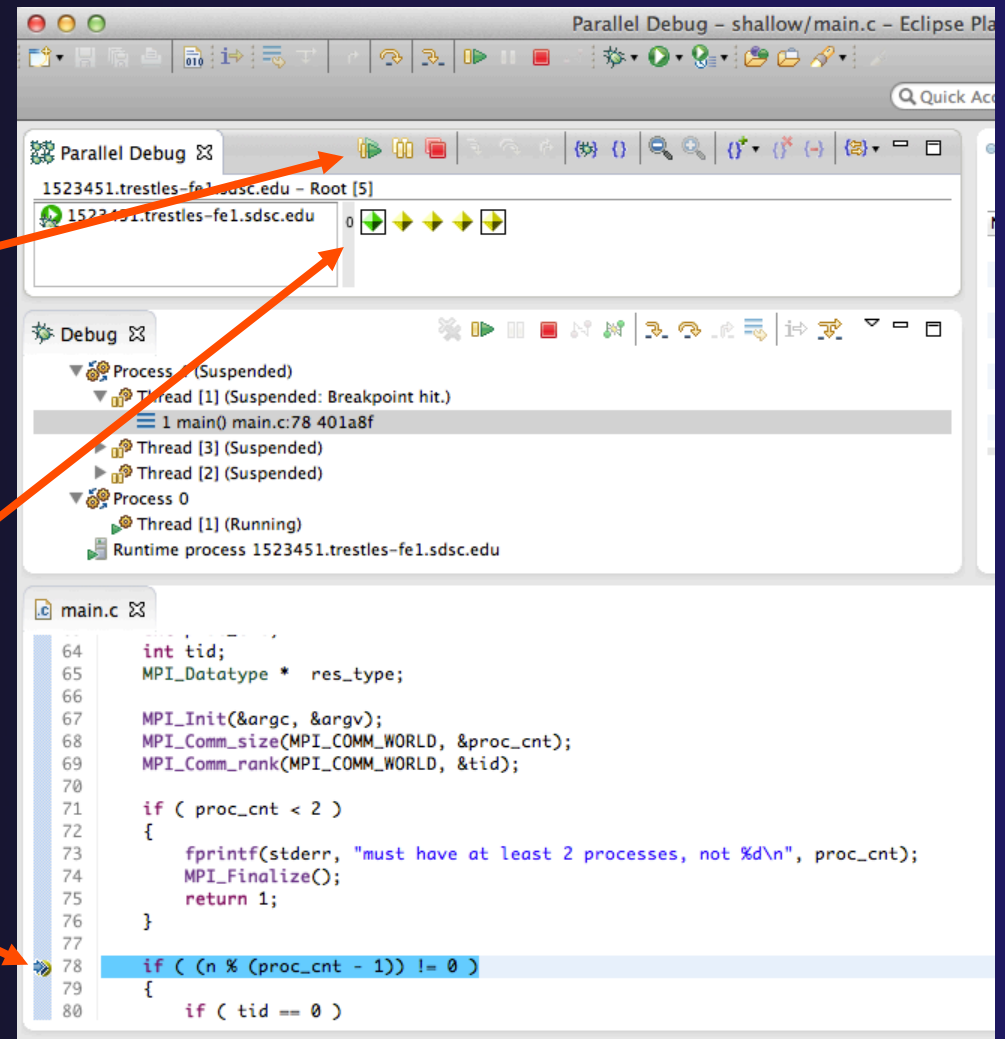
Creating A Breakpoint

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



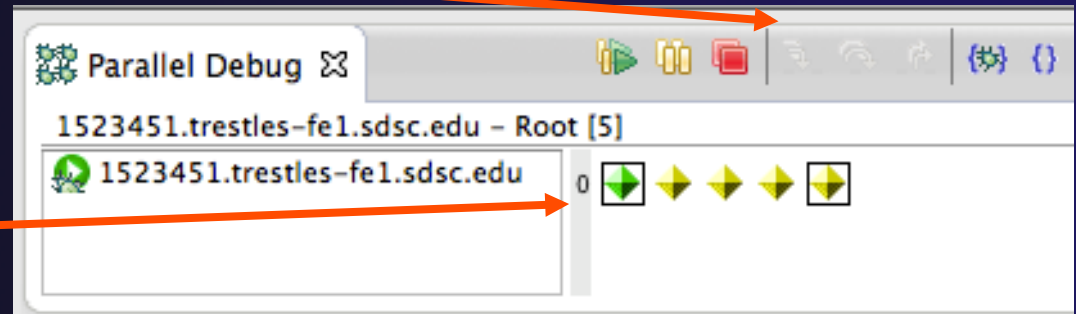
Hitting the Breakpoint

- ✦ Switch back to the **Root** set by clicking on the **Change Set** button
- ✦ Click on the **Resume** button in the **Parallel Debug view**
- ✦ In this example, the three worker processes have hit the breakpoint, as indicated by the yellow process icons and the current line marker
- ✦ Process 0 is still running as its icon is green
- ✦ Processes 1-4 are suspended on the breakpoint

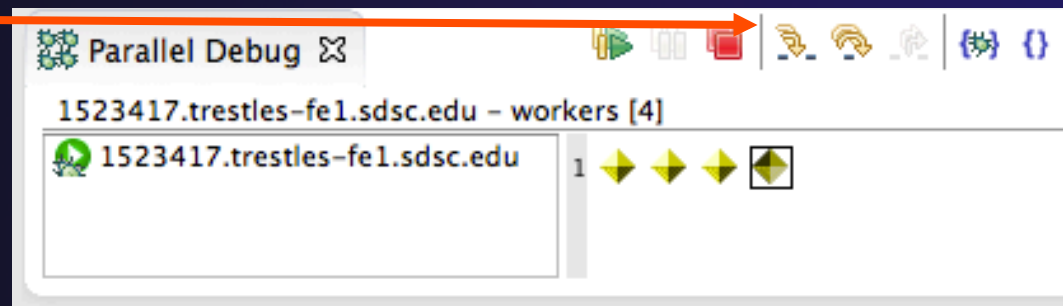


More On Stepping

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

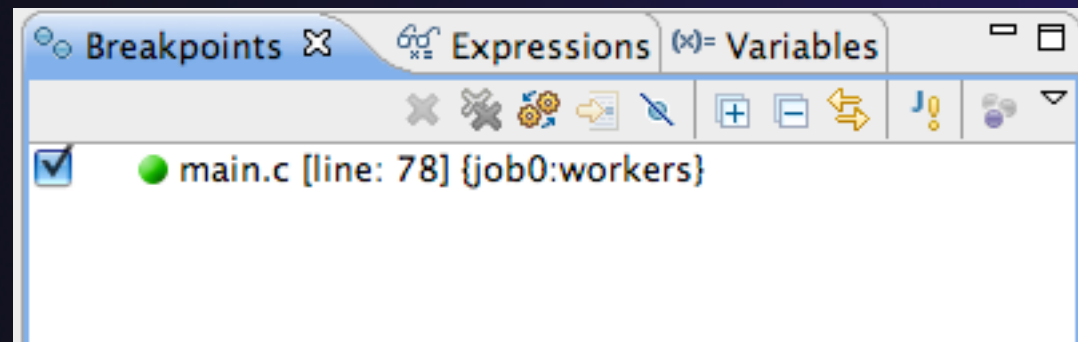


- ★ Switch to the set of suspended processes (the **workers** set)
- ★ You will now see the **Step** buttons become enabled



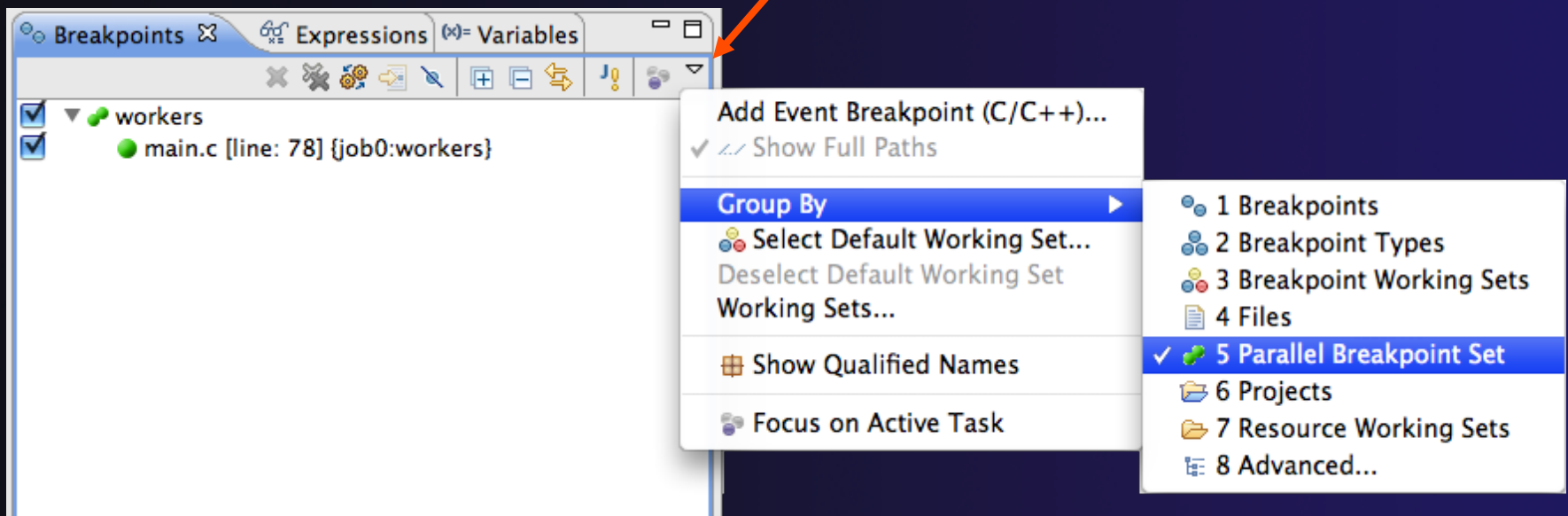
Breakpoint Information

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



Breakpoints View

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



Global Breakpoints

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

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

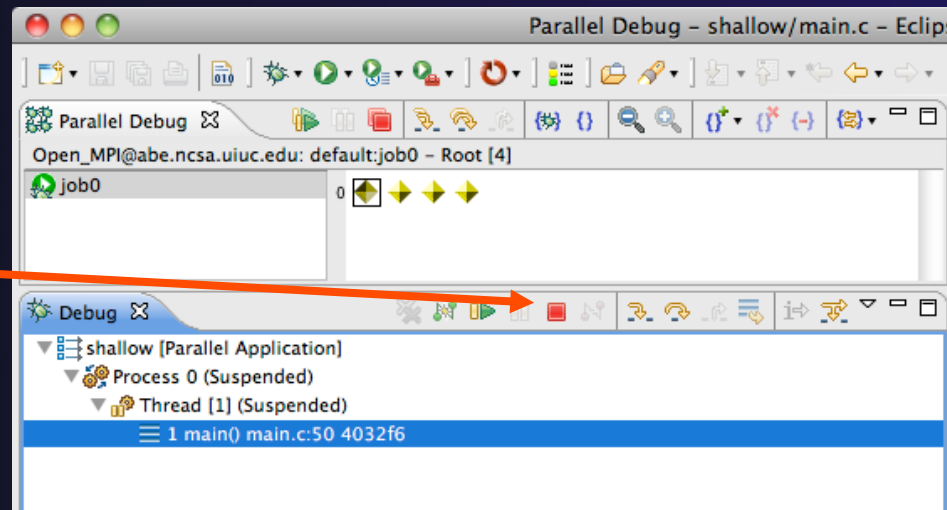
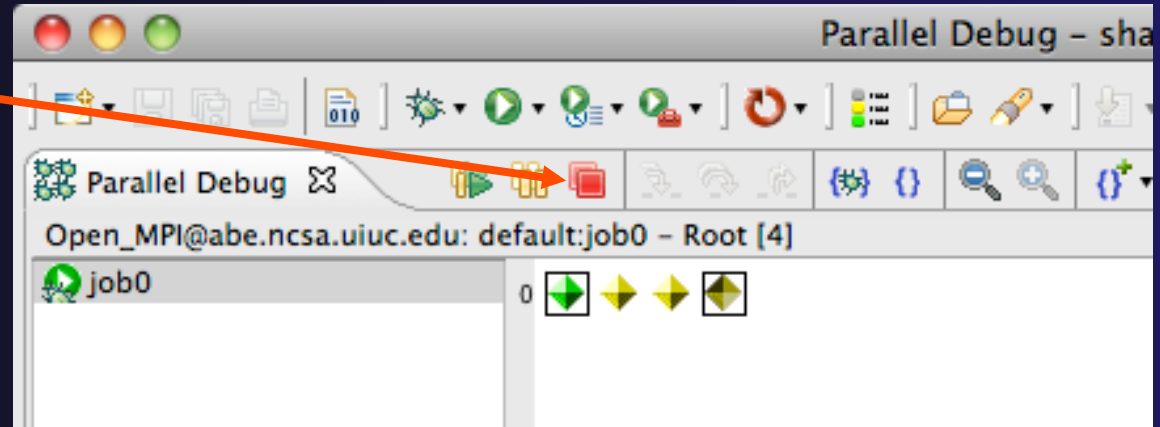


Exercise

1. Select the “worker” process set
2. Create a breakpoint by double-clicking on right hand bar at line 88 (worker function)
3. Hover over breakpoint to see properties
4. Switch to “root” process set
5. Observer breakpoint color changes to blue
6. Resume all processes
7. Observe “worker” processes at breakpoint, and process 0 still running (green icon)
8. Switch to “worker” process set
9. Step “worker” processes over worker() function
10. Observe output from program

Terminating A Debug Session

- ★ Click on the **Terminate** icon in the **Parallel Debug view** to terminate all processes in the active set
- ★ Make sure the **Root** set is active if you want to terminate all processes
- ★ You can also use the terminate icon in the **Debug view** to terminate the currently selected process



Cancelling The Job

- ✦ Interactive jobs will continue until the reservation time has expired
- ✦ You can cancel the job once the debug session is finished
- ✦ Switch back to the **System Monitoring** perspective
- ✦ Locate the job in the **Active Jobs** view
- ✦ Use the view menu to filter for only your jobs if there are too many
- ✦ Right click on the job and select **Cancel Job**

System Monitoring – shallow/mair

Monitors

Status	Connection Name	System Type
🔗	trestles	TORQUE Resource Manager

Active Jobs

step	owner	queue	wall	queue	dispat	totalcc	status
1059550...	cipres	sha...	720...	20...	20...	8	RUNNING
1059552...	cipres	sha...	259...	20...	20...	8	RUNNING
1059553...	cipres	sha...	604...	20...	20...	8	RUNNING
1059565...	gha...	sha...	180...	20...	20...	1	RUNNING
1059569...	cipres	sha...	126...	20...	20...	8	RUNNING
1059572...	cipres	sha...	126...	20...	20...	8	RUNNING
1059573...	tib...	sha...	1800...	20...	20...	5	RUNNING
1059578...	arm...	sha...	432...	20...	20...	6	RUNNING
1059579...	arm...	sha...	432...	20...	20...	6	RUNNING
1059588...	ccole	sha...	864...	20...	20...	8	RUNNING
1059590...	cipres	sha...	108...	20...	20...	8	RUNNING
1059592...	cipres	sha...	108...	20...	20...	8	RUNNING
1059594...	ram...	sha...	306...	20...	20...	16	RUNNING
1059595...	cipres	nor...	604...	20...	20...	10	RUNNING
1059599...	grw	sha...	180...	20...	20...	8	RUNNING
1059600...	cipres	nor...	720...	20...	20...	8	RUNNING

Resource Configurations

- ▶ Resume Job
- ▶ **Cancel Job**
- ▶ Hold Job
- ▶ Release Job
- ▶ Suspend Job
- ▶ Get Job Error
- ▶ Get Job Output
- ▶ Refresh Job Status
- ▶ Remove Job Entry

Messages Console

```
shallow (1) [Parallel Application] Runtime
qsub: waiting for job 1059599
qsub: job 1059599.trestles-fe
[grw@trestles-6-5 ~]$ module
esettings/rms/OPENMPI/start_j
[grw@trestles-6-5 ~]$ module
[grw@trestles-6-5 ~]$ module
[grw@trestles-6-5 ~]$ module
[grw@trestles-6-5 ~]$ module
ome/grw/.eclipsesettings/rms/
DEFILE
#DTP_job_id=15105
```



Exercise

1. Switch to the “root” set
2. Terminate all processes
3. Switch to the System Monitoring perspective
4. Right-click on your running job and select **Cancel**



Optional Exercise

1. Launch another debug job
2. Create a breakpoint at line 71 in main.c
3. Resume all processes
4. Select the Variables view tab if not already selected
5. Observe value of the “tid” variable
6. Register one of the worker processes
7. Select stack frame of worker process in Debug view
8. Observe value of the “tid” variable matches worker process
9. Switch to the breakpoints view, change grouping
10. Terminate all processes
11. Switch to the System Monitoring perspective and cancel the job

Fortran

★ Objectives

- ★ Learn how to create and convert Fortran projects
- ★ Learn to use Fortran-specific editing features
- ★ Learn about Fortran-specific properties/preferences

★ Contents

- ★ Fortran projects
- ★ Using the Fortran editor
- ★ Fortran project properties and workbench preferences

★ Prerequisites

- ★ Basics (for exercises)



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

PHOTRAN

eclipse

IDE for Fortran

eclipse

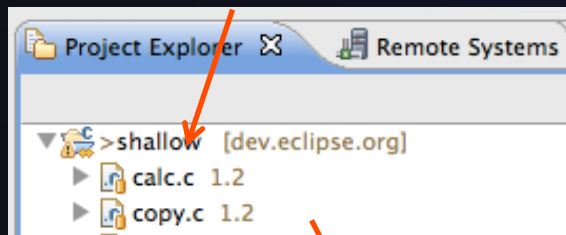
TAKE OUT 365-0051

...which became the name of their Fortran IDE.

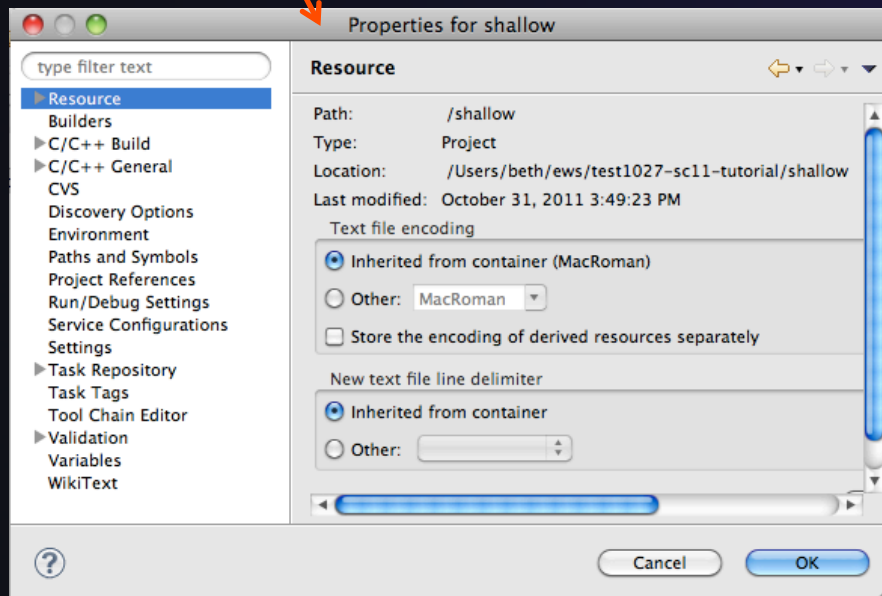
Configuring Fortran Projects

Project Properties

- ★ Right-click Project
- ★ Select **Properties...**



- ★ *Project properties* are settings that can be changed for each project

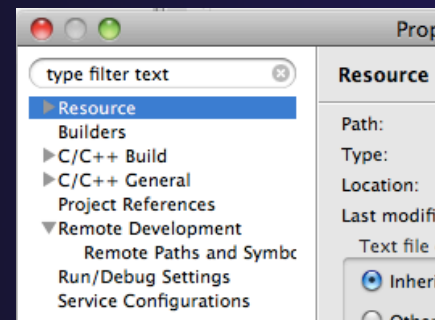


- ★ Contrast with *workspace preferences*, which are the same regardless of what project is being edited
 - ★ e.g., editor colors
 - ★ Set in **Window ► Preferences** (on Mac, **Eclipse ► Preferences**)
 - ★ Careful! Dialog is very similar

Converting to a Fortran Project

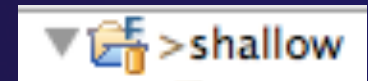
- ★ Are there categories labeled **Fortran General** and **Fortran Build** in the project properties?

No Fortran categories →



Do this once

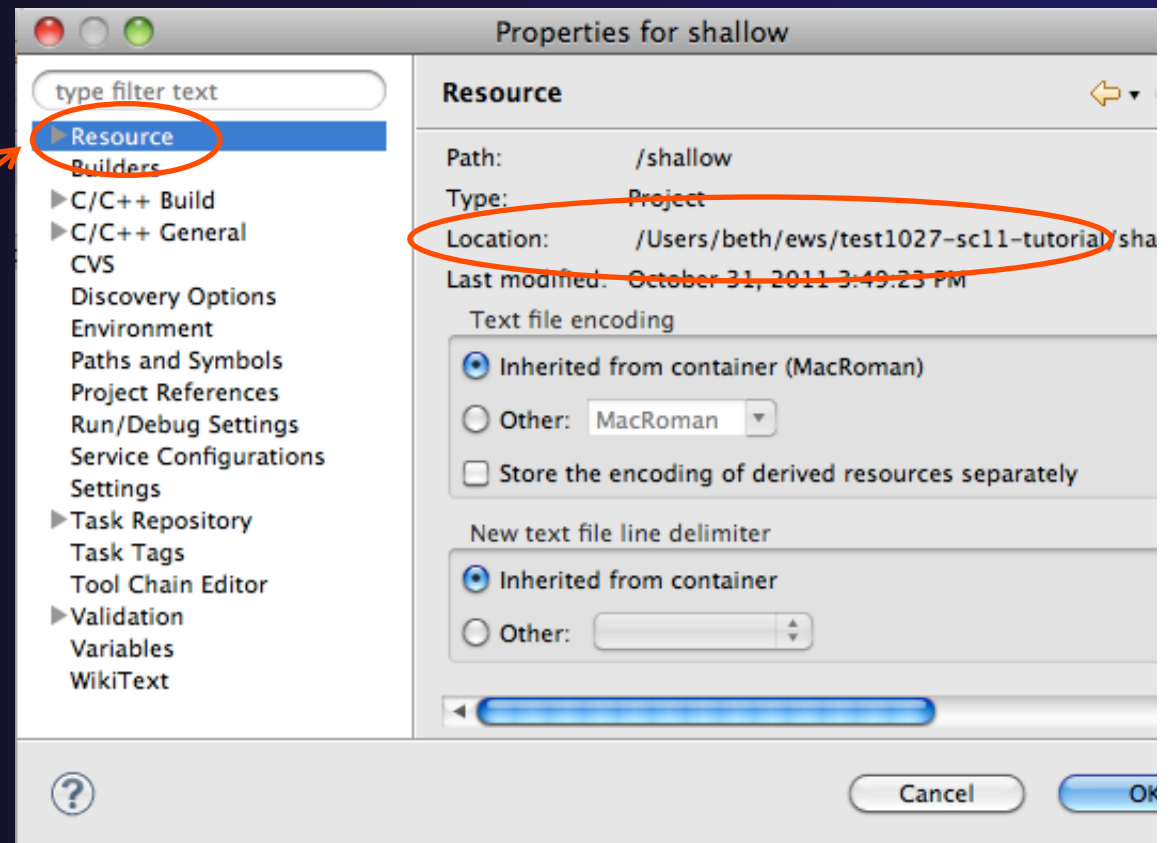
- ★ If not, the project is not a Fortran Project
 - ★ Switch to the Fortran Perspective
 - ★ In the Fortran Projects view, right-click on the project, and click **Convert to Fortran Project**
 - ★ Don't worry; it's still a C/C++ project, too
- ★ *Every* Fortran project is also a C/C++ Project



Project Location

★ How to tell where a project resides?

★ In the project properties dialog, select the **Resource** category

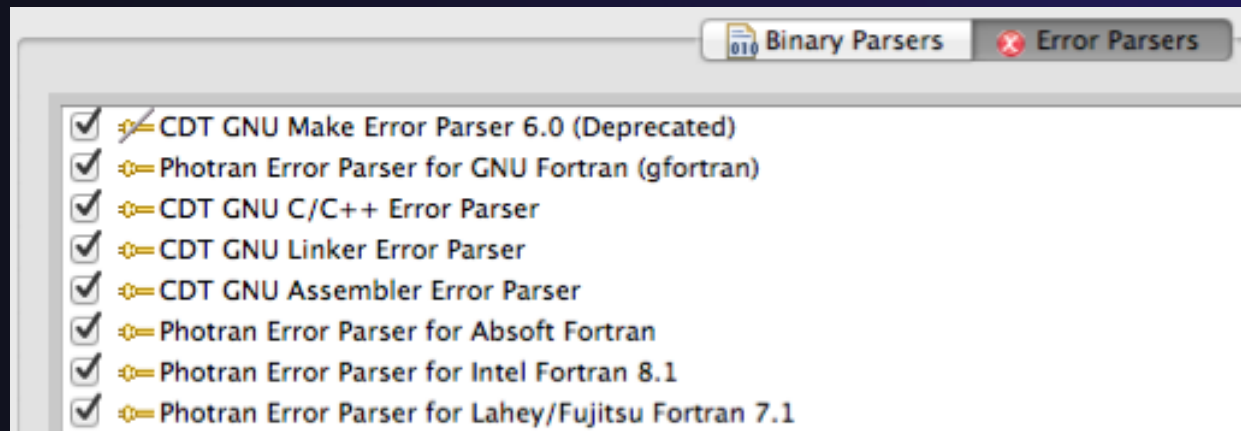


Error Parsers

- ★ Are compiler errors not appearing in the Problems view?
 - ★ Make sure the correct *error parser* is enabled
 - ★ In the project properties, navigate to **C++ Build ► Settings** or **Fortran Build ► Settings**
 - ★ Switch to the **Error Parsers** tab
 - ★ Check the error parser(s) for your compiler(s)



Do this
once



Fortran Source Form Settings

- ★ Fortran files are either *free form* or *fixed form*; some Fortran files are *preprocessed* (#define, #ifdef, etc.)

- ★ Source form determined by filename extension

- ★ Defaults are similar to most Fortran compilers:

Fixed form:	.f	.fix	.for	.fpp	.ftn	.f77	
Free form:	.f08	.f03	.f95	.f90			< unpreprocessed
	.F08	.F03	.F95	.F90			< preprocessed

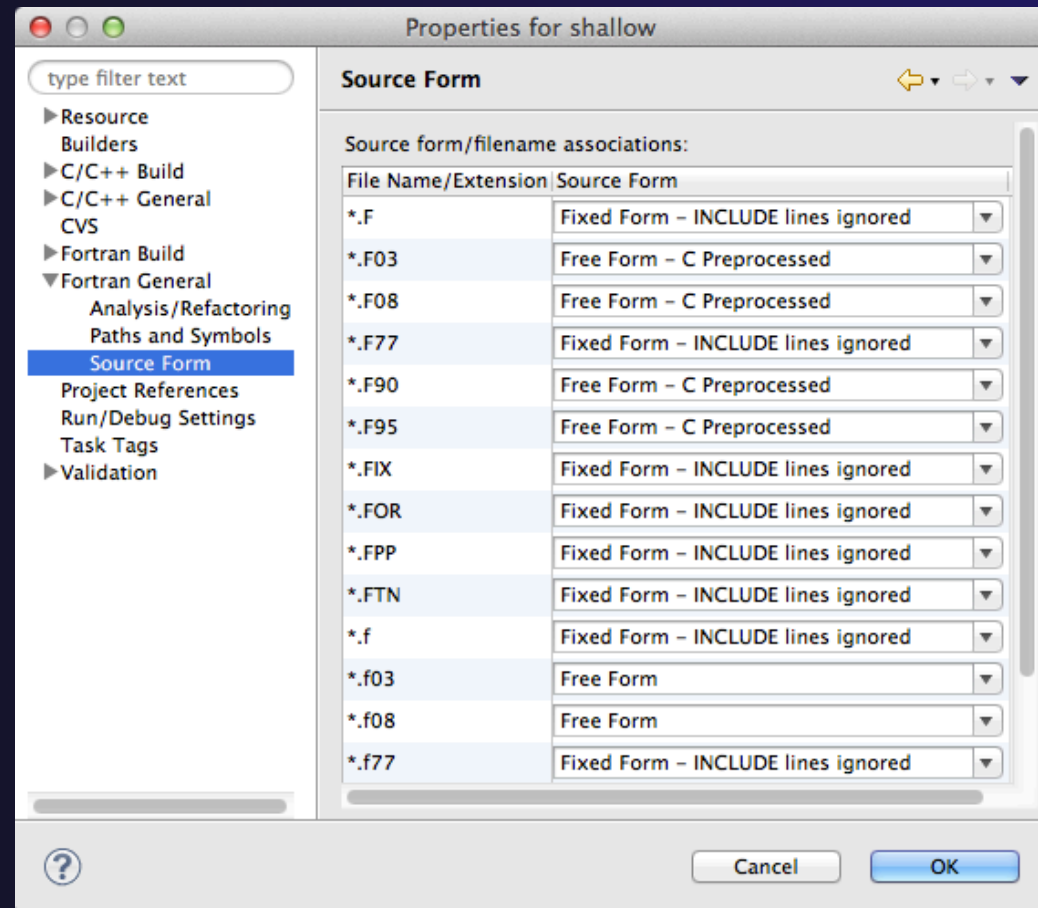
- ★ Many features *will not work* if filename extensions are associated with the wrong source form (outline view, content assist, search, refactorings, etc.)

Fortran Source Form Settings



Do this
once

- ★ In the project properties, select **Fortran General** ▶ **Source Form**
- ★ Select source form for each filename extension
- ★ Click **OK**

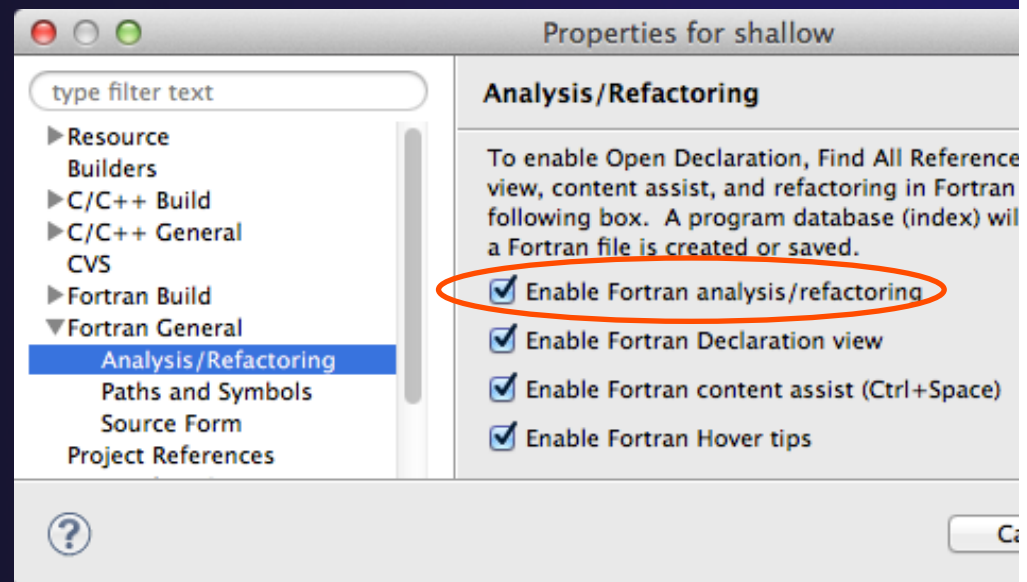


Enabling Fortran Advanced Features

- ★ Some Fortran features are *disabled* by default
- ★ Must be explicitly enabled
 - ★ In the project properties dialog, select **Fortran General** ▶ **Analysis/Refactoring**
 - ★ Click **Enable Analysis/Refactoring**
 - ★ Close and re-open any Fortran editors
- ★ This turns on the “Photran Indexer”
 - ★ Turn it off if it’s slow



Do this once





Exercise

1. Convert shallow to a Fortran project
2. Make sure errors from the GNU Fortran compiler will be recognized
3. Make sure *.f90 files are treated as “Free Form” which is unpreprocessed
4. Make sure search and refactoring will work in Fortran

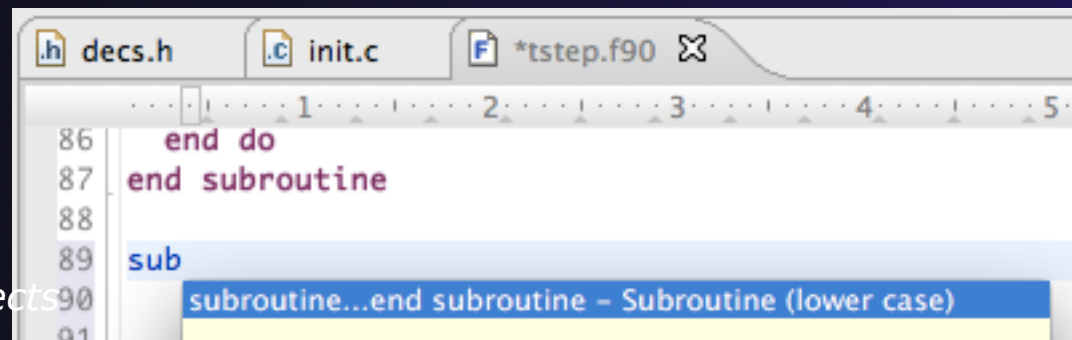
Advanced Editing

Code Templates

Code Templates

(C/C++ and Fortran)

- ★ Auto-complete common code patterns
 - ★ For loops/do loops, if constructs, etc.
 - ★ Also MPI code templates
- ★ Included with content assist proposals (when **Ctrl-Space** is pressed)
 - ★ E.g., after the last line in tstep.f90, type “sub” and press **Ctrl-Space**
 - ★ Press **Enter** to insert the template



The screenshot shows an IDE window with three tabs: 'decs.h', 'init.c', and '*tstep.f90'. The code in the window is as follows:

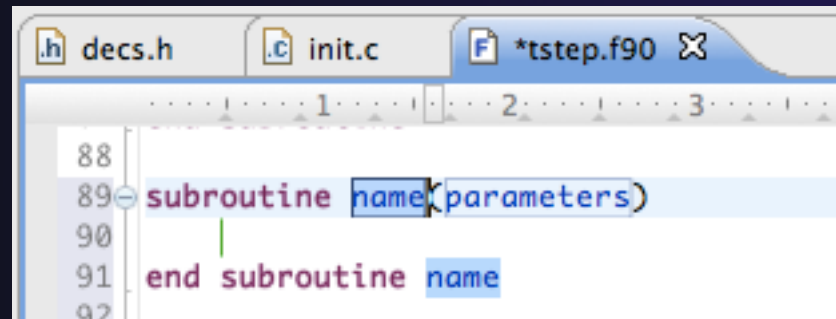
```
86   end do
87   end subroutine
88
89   sub
90   subroutine...end subroutine - Subroutine (lower case)
91
```

A dropdown menu is visible below line 90, showing the suggestion 'subroutine...end subroutine - Subroutine (lower case)'. The text 'Fortran Projects' is partially visible on the left side of the image.

Code Templates (2)

(C/C++ and Fortran)

- ★ After pressing enter to insert the code template, completion fields are highlighted



The screenshot shows an IDE window with three tabs: 'decs.h', 'init.c', and '*tstep.f90'. The active window is '*tstep.f90'. The code is as follows:

```
88  
89 subroutine name(parameters)  
90  
91 end subroutine name  
92
```

The text 'name' in the subroutine signature and 'name' in the 'end subroutine' statement are highlighted in blue. The text 'parameters' in the parentheses is highlighted in light blue. A vertical cursor is positioned at the end of the first line of the subroutine body.

- ★ Press **Tab** to move between completion fields
- ★ Changing one instance of a field changes all occurrences



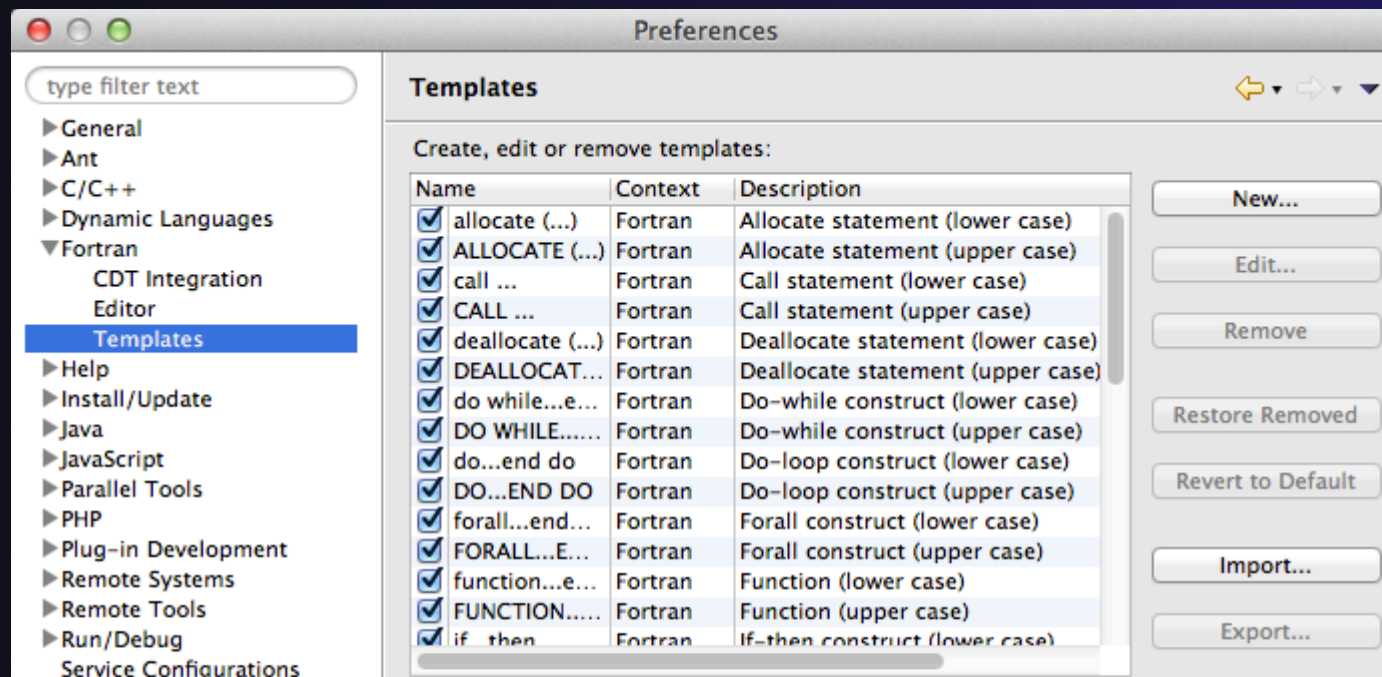
Exercise

- ✦ Open `tstep.f90` and retype the last loop nest
 - ✦ Use the code template to complete the do-loops
 - ✦ Use content assist to complete variable names

Custom Code Templates

(Fortran)

- ★ Customize code templates in **Window ► Preferences ► Fortran ► Templates**



- ★ Can import/export templates to XML files

Advanced Features

Search, Refactoring, & GUI Terminal

★ Objectives

- ★ Develop proficiency using Eclipse's textual and language-based search and navigation capabilities
- ★ Introduce common automated refactorings
- ★ Develop proficiency using the PTP GUI Terminal

★ Contents

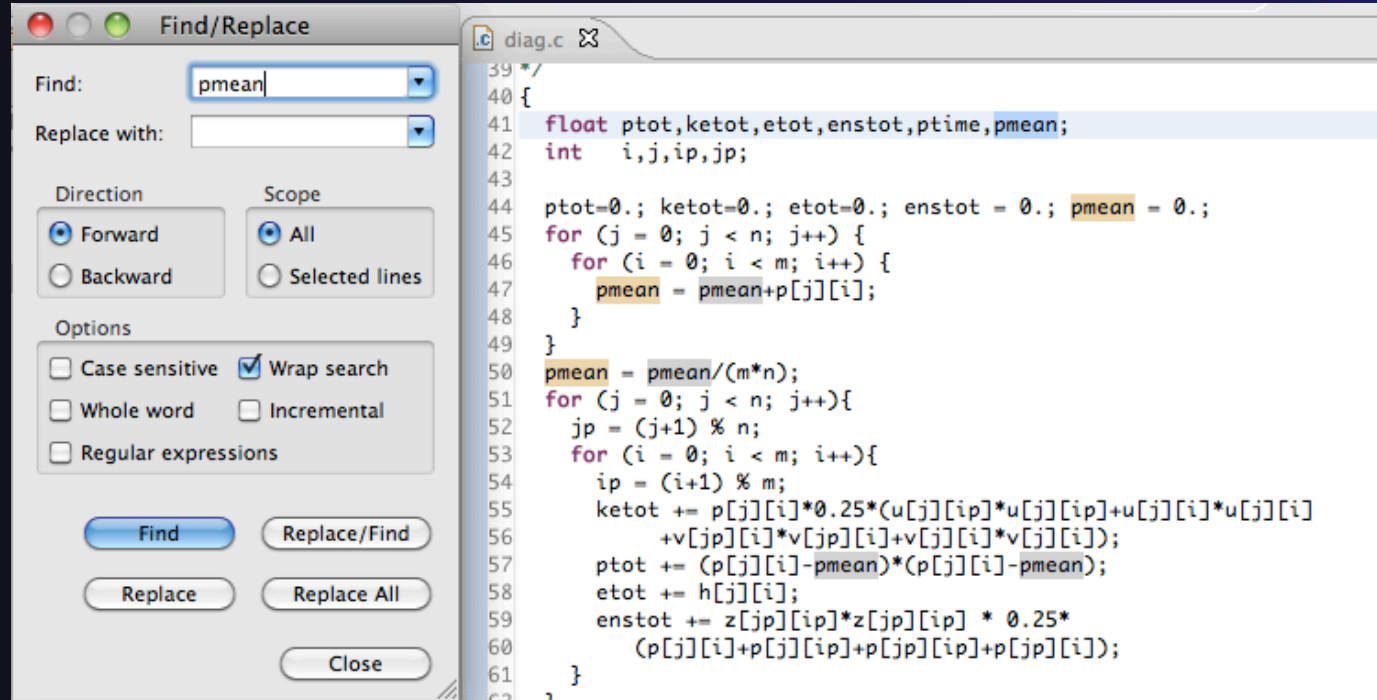
- ★ Searching
- ★ Refactoring and Transformation
- ★ GUI Terminal

★ Prerequisites

- ★ Basics
- ★ Fortran

Find/Replace within Editor

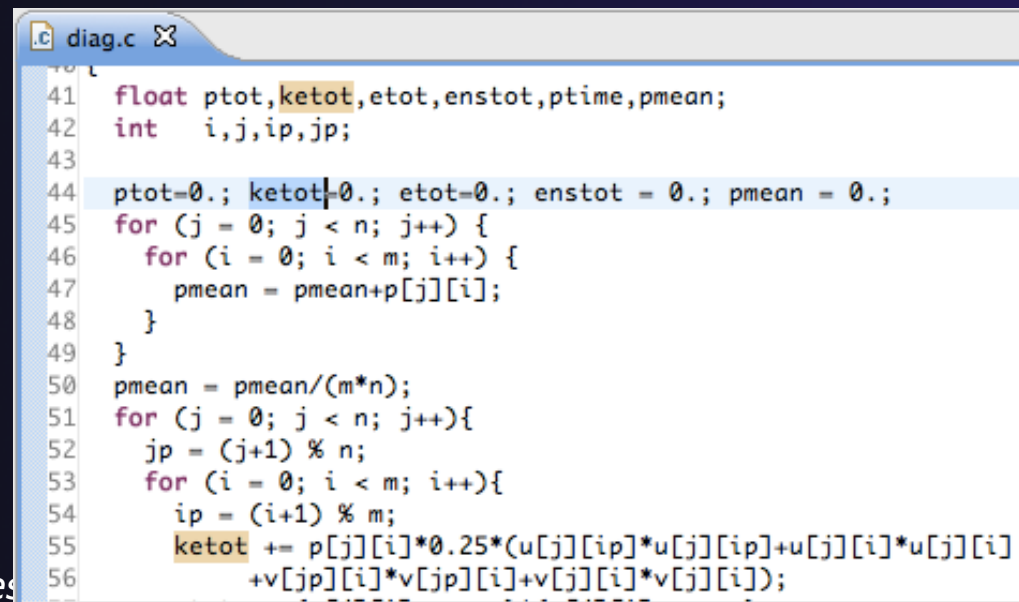
- ✦ Simple Find within editor buffer
- ✦ Ctrl-F (Mac: Command-F)



Mark Occurrences

(C/C++ Only)

- ★ Double-click on a variable in the CDT editor
- ★ All occurrences in the source file are highlighted to make locating the variable easier
- ★ Alt-shift-O to turn off (Mac: Alt-Command-O)



The screenshot shows a CDT editor window titled 'diag.c'. The code is as follows:

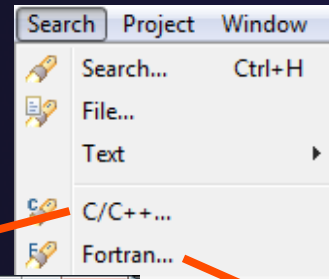
```
41 float ptot, ketot, etot, enstot, ptime, pmean;  
42 int i, j, ip, jp;  
43  
44 ptot=0.; ketot=0.; etot=0.; enstot = 0.; pmean = 0.;  
45 for (j = 0; j < n; j++) {  
46     for (i = 0; i < m; i++) {  
47         pmean = pmean+p[j][i];  
48     }  
49 }  
50 pmean = pmean/(m*n);  
51 for (j = 0; j < n; j++){  
52     jp = (j+1) % n;  
53     for (i = 0; i < m; i++){  
54         ip = (i+1) % m;  
55         ketot += p[j][i]*0.25*(u[j][ip]*u[j][ip]+u[j][i]*u[j][i]  
56             +v[jp][i]*v[jp][i]+v[j][i]*v[j][i]);
```

In the screenshot, the variable 'ketot' is highlighted in yellow in the declaration on line 41 and in the assignment on line 55. The entire line 44 is highlighted in light blue.

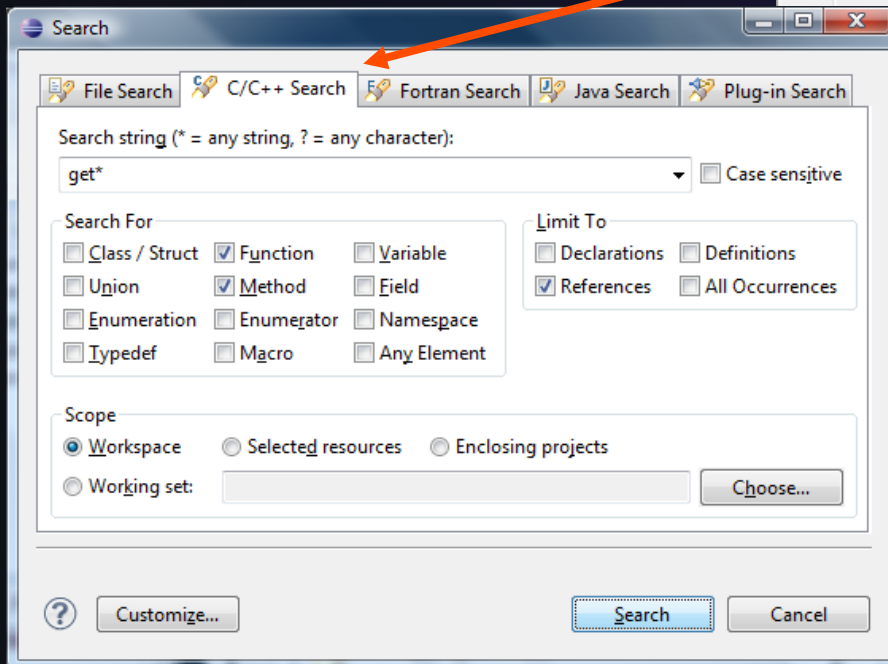
Language-Based Searching

(C/C++ and Fortran)

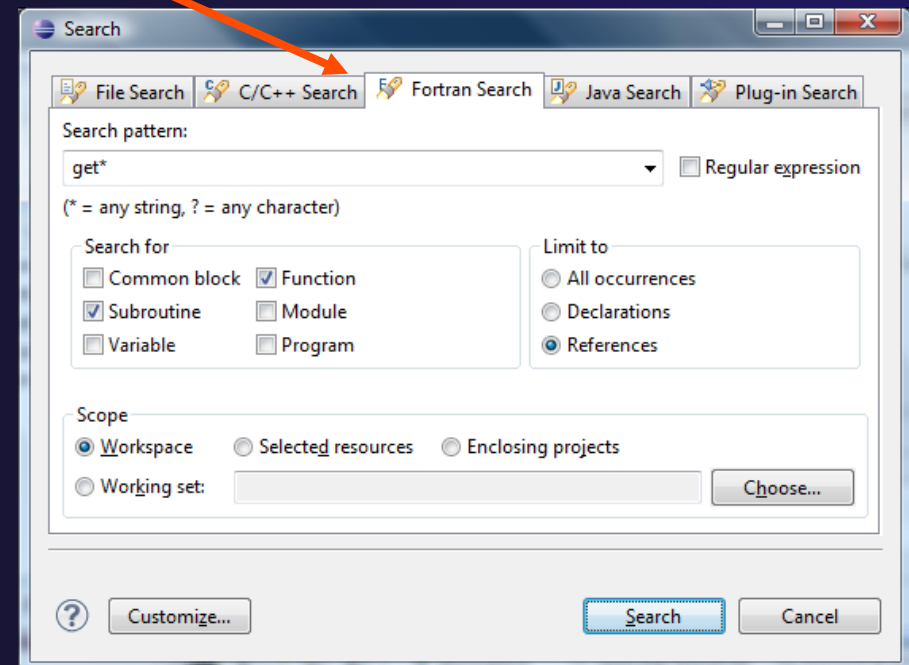
- ★ “Knows” what things can be declared in each language (functions, variables, classes, modules, etc.)



- ★ E.g., search for every call to a function whose name starts with “get”
- ★ Search can be project- or workspace-wide



Advanced Features



Advanced-3

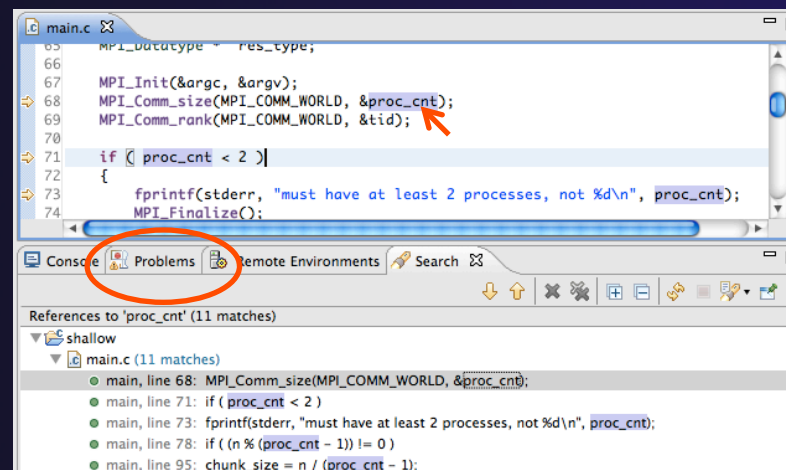
Find References

(C/C++ and Fortran)

- ★ Finds all of the places where a variable, function, etc., is used
 - ★ Right-click on an identifier in the editor
 - ★ Click **References** ▶ **Workspace** or **References** ▶ **Project**



- ★ **Search** view shows matches



Open Declaration

(C/C++ and Fortran)

- ★ Jumps to the declaration of a variable, function, etc., even if it's in a different file
- ★ Left-click to select identifier
- ★ Right-click on identifier
- ★ Click **Open Declaration**
- ★ C/C++ only:
Can also Ctrl-click (Mac: Cmd-click) on an identifier to “hyperlink” to its declaration

```

134
135 initialise(p, u, v, psi, pold, uold, vold, di, dj, z);
136 diag(1, 0., p, u, v, h, z);
137
138 for (i = 1; i < proc_cnt; i++)
139     for (j = 0; j < n; j++)
140         acopy_two_to_one(i, j, p, u, v, h, z);
141     MPI_Send(&star, 1, MPI_COMM_WORLD, j, 0, MPI_COMM_WORLD);
142
143
144     acopy_two_to_one(i, j, p, u, v, h, z);
145     MPI_Send(&star, 1, MPI_COMM_WORLD, j, 0, MPI_COMM_WORLD);
  
```

Goes to its declaration in copy.c

```

59 bcopy(src[column], dest[column], sizeof(src[column]));
60 }
61
62 acopy_two_to_one(twodim, onedim, column)
63 float twodim[n][m];
64 float onedim[m];
65 int column;
  
```



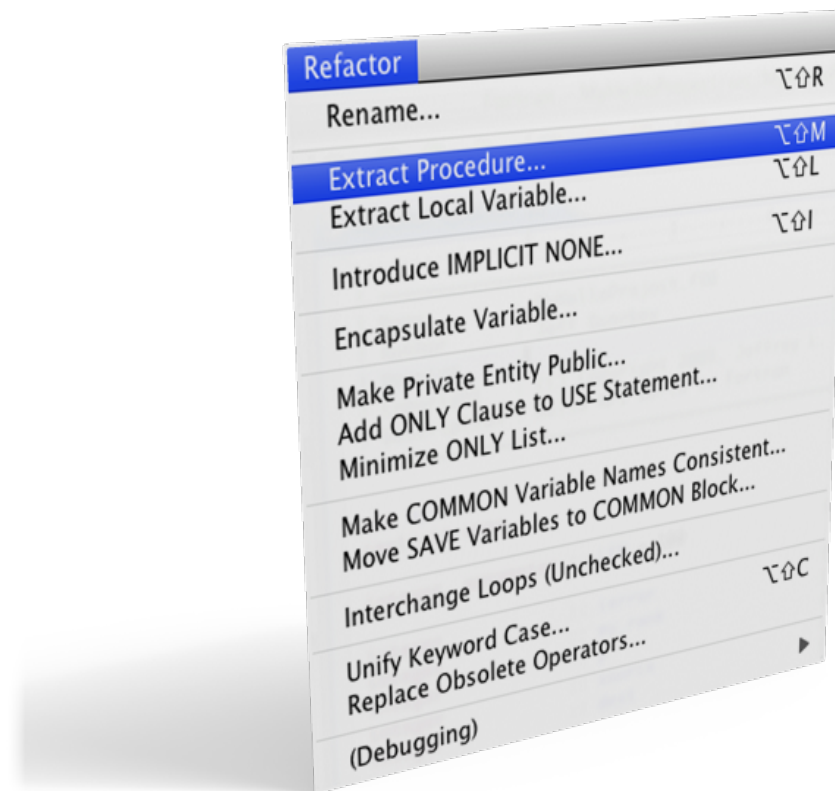
Search – Try It!

1. Find every call to `MPI_Recv` in Shallow.
2. In `worker.c`, on line 42, there is a declaration `float p[n][m]`.
 - a) What is `m` (local? global? function parameter?)
 - b) Where is `m` defined?
 - c) How many times is `m` used in the project?
3. Find every C function in Shallow whose name contains the word `time`

Refactoring and Transformation

Refactoring

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



★ 39 automated refactorings in Photran

Refactoring Caveats

- ★ Photran can only refactor free form code that is *not* preprocessed

- ★ Determined by Source Form settings

(recall from earlier that these are configured in
Project Properties: Fortran General ▶ Source Form)

✓	Free Form, Unpreprocessed:	.f08	.f03	.f95	.f90		
✗	Free Form, Preprocessed:	.F08	.F03	.F95	.F90		
✗	Fixed Form:	.f	.fix	.for	.fpp	.ftn	.f77

- ★ Refactor menu will be empty if

- ★ Refactoring not enabled in project properties

(recall from earlier that it is enabled in

Project Properties: Fortran General ▶ Analysis/Refactoring)

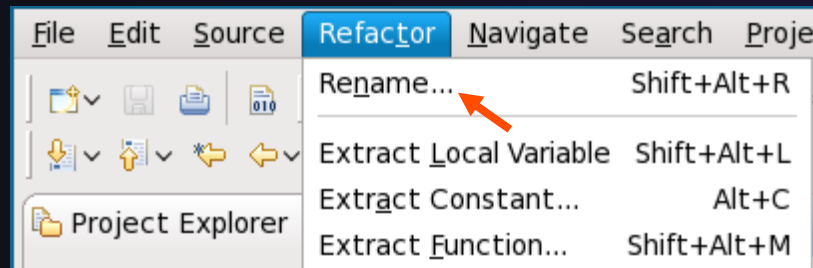
- ★ The file in the active editor is fixed form

- ★ The file in the active editor is preprocessed

Rename Refactoring

(also available in Fortran)

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



In Java (Murphy-Hill et al., ICSE 2008):

Refactoring	Uses	Percentage
Rename	179,871	74.8%
Extract Local Variable	13,523	5.6%
Move	13,208	5.5%
Extract Method	10,581	4.4%
Change Method Signature	4,764	2.0%
Inline	4,102	1.7%
Extract Constant	3,363	1.4%
(16 Other Refactorings)	10,924	4.5%

Advanced

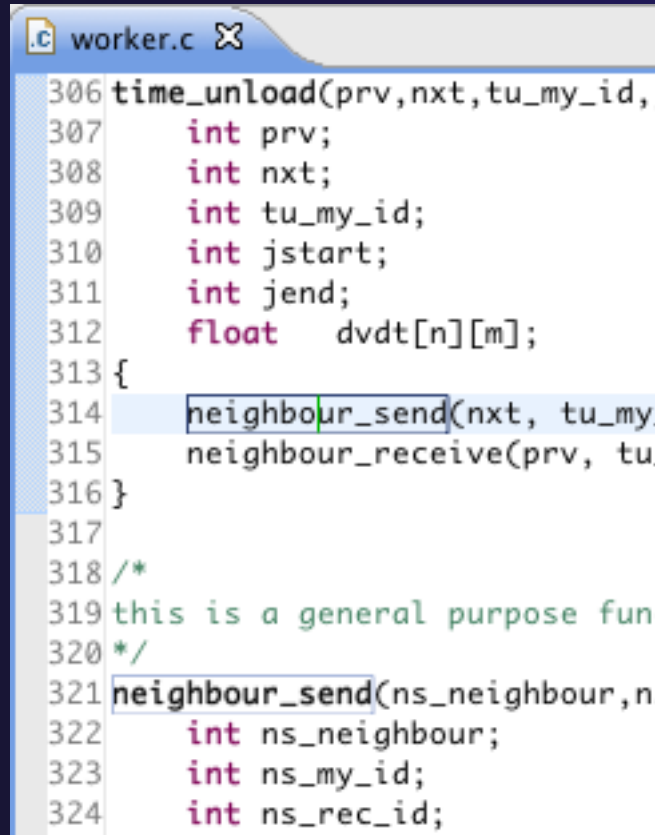
- ✦ Switch to C/C++ Perspective
- ✦ Open a source file
- ✦ In the editor, click on a variable or function name
- ✦ Select menu item **Refactor ▶ Rename**
 - ✦ Or use context menu
- ✦ Enter new name

Advanced-10

Rename in File

(C/C++ Only)

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

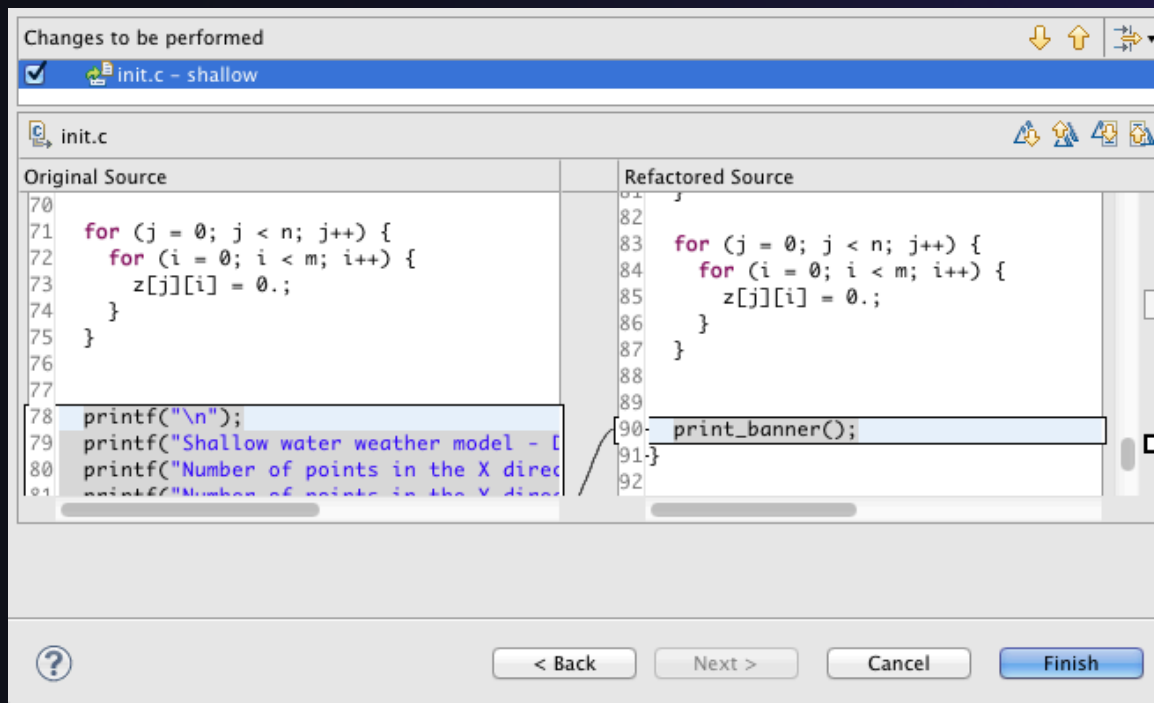


```
worker.c X
306 time_unload(prv,nxt,tu_my_id,
307     int prv;
308     int nxt;
309     int tu_my_id;
310     int jstart;
311     int jend;
312     float  dvdt[n][m];
313 {
314     neighbour_send(nxt, tu_my.
315     neighbour_receive(prv, tu.
316 }
317
318 /*
319 this is a general purpose fun
320 */
321 neighbour_send(ns_neighbour,n
322     int ns_neighbour;
323     int ns_my_id;
324     int ns_rec_id;
```


Extract Function Refactoring

(also available in Fortran - "Extract Procedure")

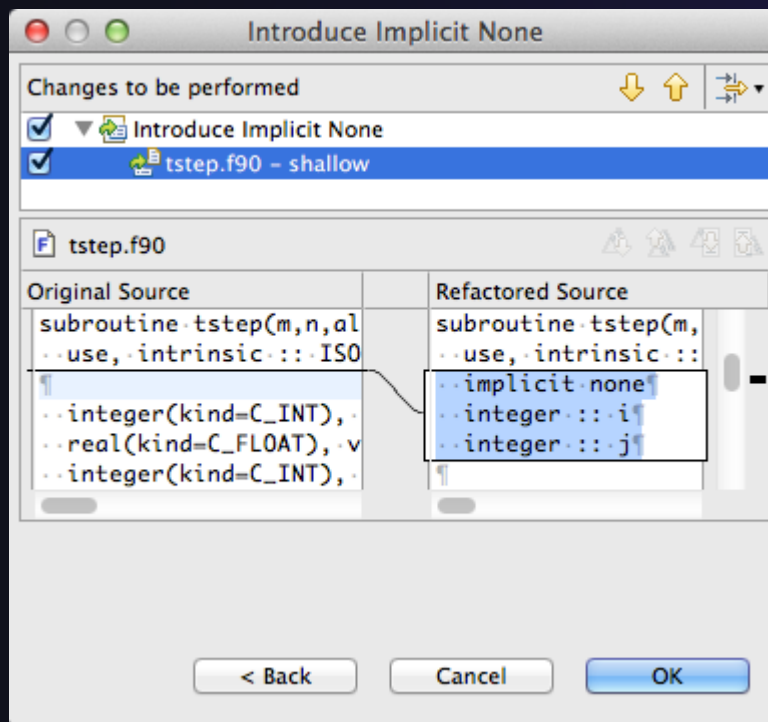
- ✦ Moves statements into a new function, replacing the statements with a call to that function
- ✦ Local variables are passed as arguments



- ✦ Select a sequence of statements
- ✦ Select menu item **Refactor** ► **Extract Function...**
- ✦ Enter new name

Introduce IMPLICIT NONE Refactoring

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

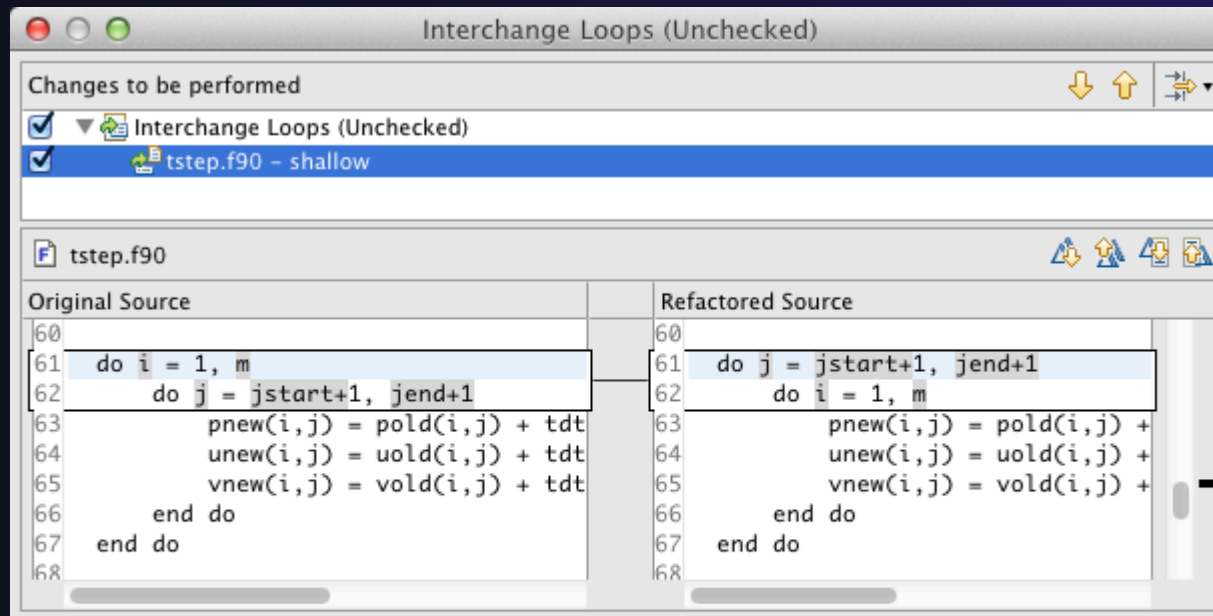


- ★ Introduce in a single file by opening the file and selecting **Refactor ▶ Coding Style ▶ Introduce IMPLICIT NONE...**
- ★ Introduce in multiple files by selecting them in the Fortran Projects view, right-clicking on the selection, and choosing **Refactor ▶ Coding Style ▶ Introduce IMPLICIT NONE...**

Loop Transformations

(Fortran only)

- ✦ **Interchange Loops** **CAUTION:** No check for behavior preservation
 - ✦ Swaps the loop headers in a two-loop nest
 - ✦ Select the loop nest, click menu item **Refactor ▶ Do Loop ▶ Interchange Loops (Unchecked)...**



Old version traverses
matrices in row-major order

Advanced Features

New version traverses
in column-major order
(better cache performance)

Advanced-14

Loop Transformations

(Fortran only)

★ Unroll Loop

★ Select a loop, click **Refactor** ▶ **Do Loop** ▶ **Unroll Loop...**

```
do i = 1, 12
  print *, 10*i
end do
```



Unroll 4x

```
do i = 1, 12, 4
  print *, 10*i
  print *, 10*(i+1)
  print *, 10*(i+2)
  print *, 10*(i+3)
end do
```

Original Source	Refactored Source
68	78
69 ! Don't apply time filter on first	79 end if
70 if (firststep == 0) then	80
71 do j = jstart+1, jend+1	81 do j = jstart+1, jend+1
72 do i = 1, m	82 LoopUpperBound = m
73 pold(i,j) = p(i,j)+alpha*(pne	83 do i = 1, LoopUpperBound,4
74 uold(i,j) = u(i,j)+alpha*(une	84 p(i,j) = pnw(i,j)
75 vold(i,j) = v(i,j)+alpha*(vne	85 u(i,j) = unew(i,j)
76 end do	86 v(i,j) = vnw(i,j)
77 end do	87 p((i+1),j) = pnw((i+1)
78 end if	88 u((i+1),j) = unew((i+1)
79	89 v((i+1),j) = vnw((i+1)
80 do j = jstart+1, jend+1	90 p((i+2),j) = pnw((i+2)
81 do i = 1, m	91 u((i+2),j) = unew((i+2)
82 p(i,j) = pnw(i,j)	92 v((i+2),j) = vnw((i+2)
83 u(i,j) = unew(i,j)	93 p((i+3),j) = pnw((i+3)
84 v(i,j) = vnw(i,j)	94 u((i+3),j) = unew((i+3)
85 end do	95 v((i+3),j) = vnw((i+3)
86 end do	96 end do
87 end subroutine	97 end do
88	98 end subroutine
	99
	00

Refactoring & Transformation – Exercises



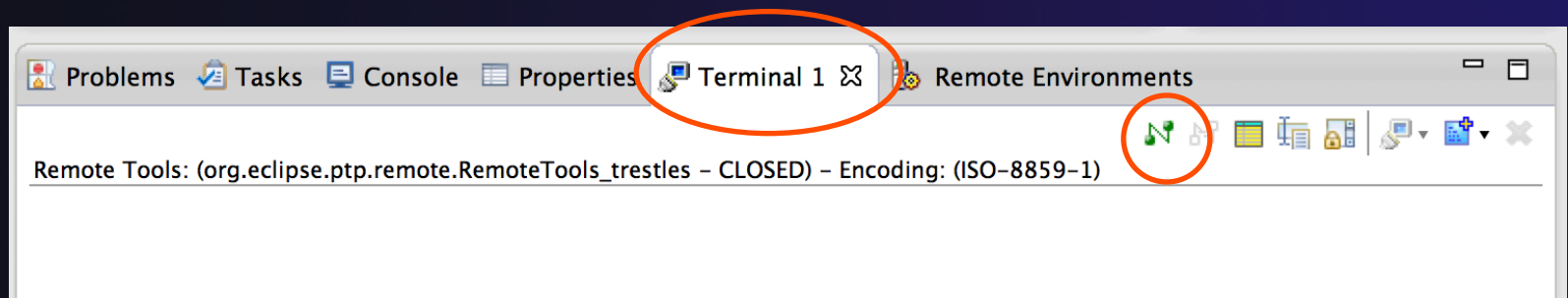
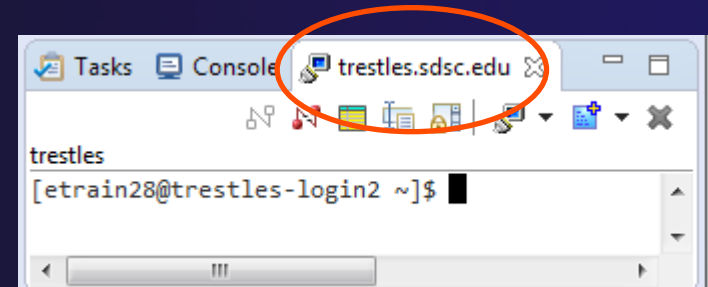
1. In `init.c`, extract the `printf` statements at the bottom of the file into a new function called `print_banner`
2. In `worker.c`, change the spellings of `neighbour_send` and `neighbour_receive` to American English
3. In `tstep.f90`, make the (Fortran) `tstep` subroutine `IMPLICIT NONE`

GUI Terminal

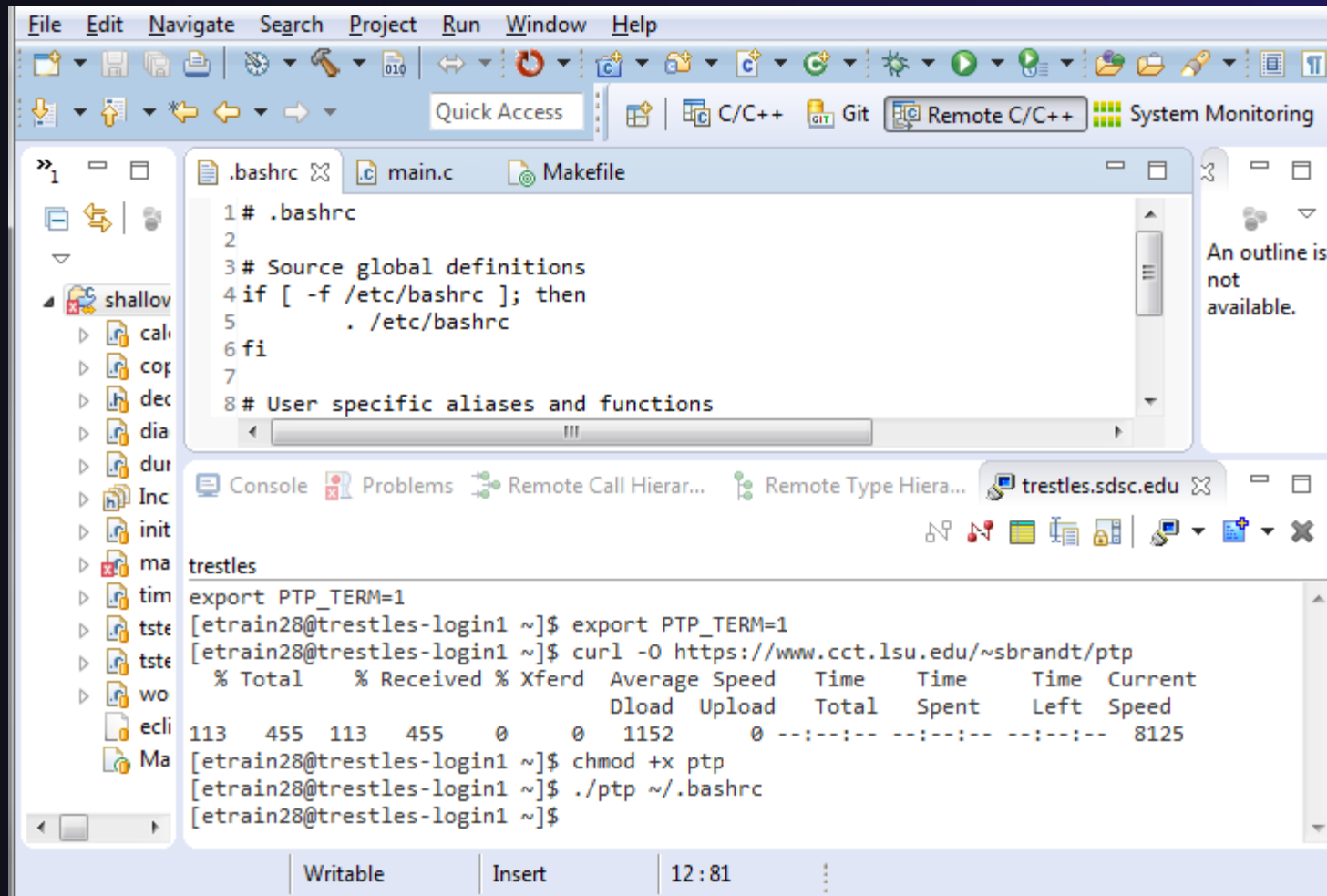
- ✦ Some people prefer to use the command line
- ✦ Some people prefer to use the GUI
- ✦ Which is better? Each has advantages
- ✦ Why do we have to choose?

GUI Terminal

- ✦ There is a remote terminal that can provide a shell from within Eclipse
- ✦ Right-Click on your synchronized project and select "Show Terminal"
 - Or
- ✦ If not in your workbench:
 - ✦ Select **Window>Show View>Other...**
 - ✦ Choose **Terminal** from the Terminal folder
- ✦ In the **Terminal** view, click on the **Connect** button
- ✦ It will use the previously configured connection from the dropdown, or create a new one



Opening the Eclipse Editor from the Command Line



The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. Below the menu is a toolbar with various icons. The main editor area displays the .bashrc file with the following content:

```
1 # .bashrc
2
3 # Source global definitions
4 if [ -f /etc/bashrc ]; then
5     . /etc/bashrc
6 fi
7
8 # User specific aliases and functions
```

The left sidebar shows a project named 'shallov' with a tree view of files and folders. The bottom panel contains a console window with the following output:

```
trestles
export PTP_TERM=1
[etrain28@trestles-login1 ~]$ export PTP_TERM=1
[etrain28@trestles-login1 ~]$ curl -O https://www.cct.lsu.edu/~sbrandt/ptp
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload  Total   Spent    Left   Speed
113   455  113   455    0     0    1152    0  --:--:--  --:--:--  --:--:--   8125
[etrain28@trestles-login1 ~]$ chmod +x ptp
[etrain28@trestles-login1 ~]$ ./ptp ~/.bashrc
[etrain28@trestles-login1 ~]$
```

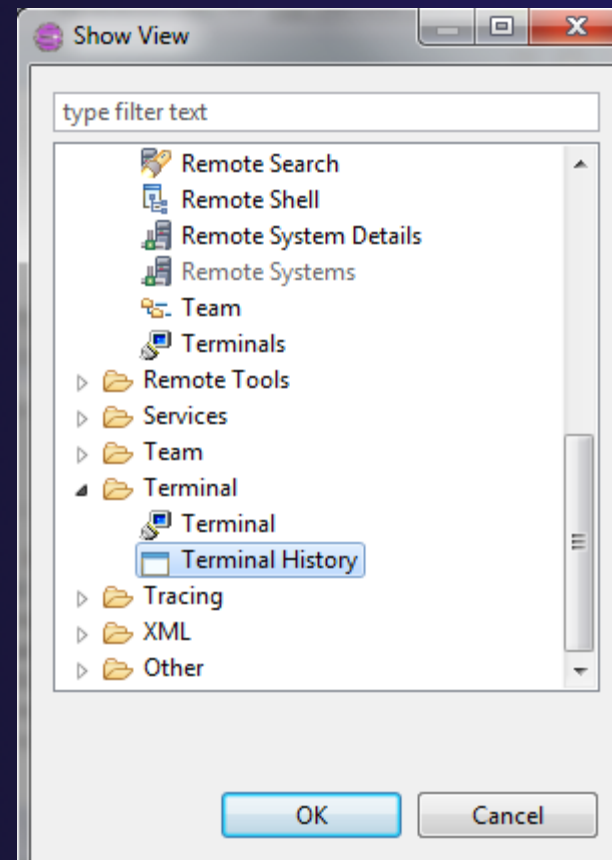
The status bar at the bottom indicates 'Writable', 'Insert', and '12:81'.

Editing Files

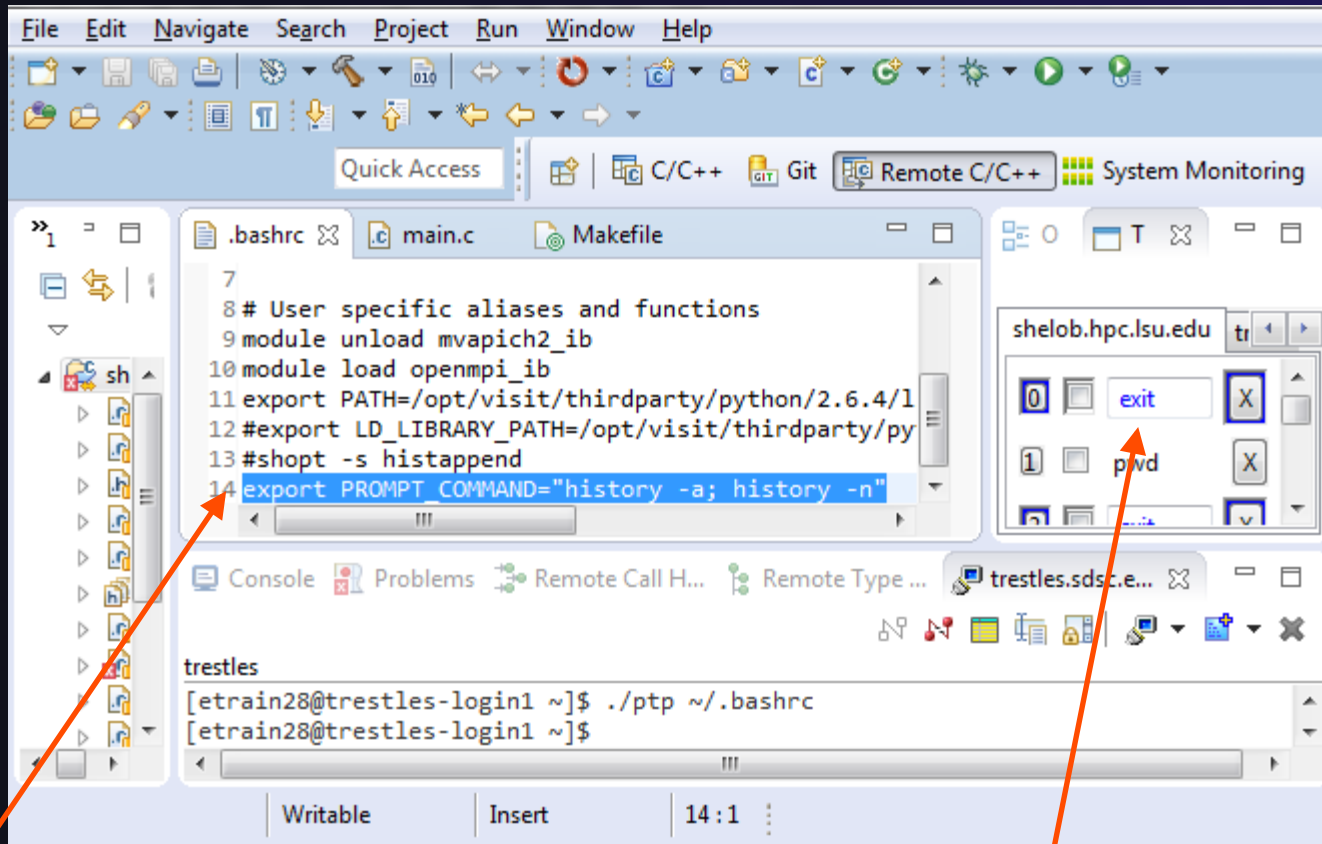
- ✦ Edits file locally within a project, and lets the sync mechanism update the remote
- ✦ Uses a remote protocol to edit files outside the project

Terminal History

Select the Terminal View
from Window >
Show View >
Others



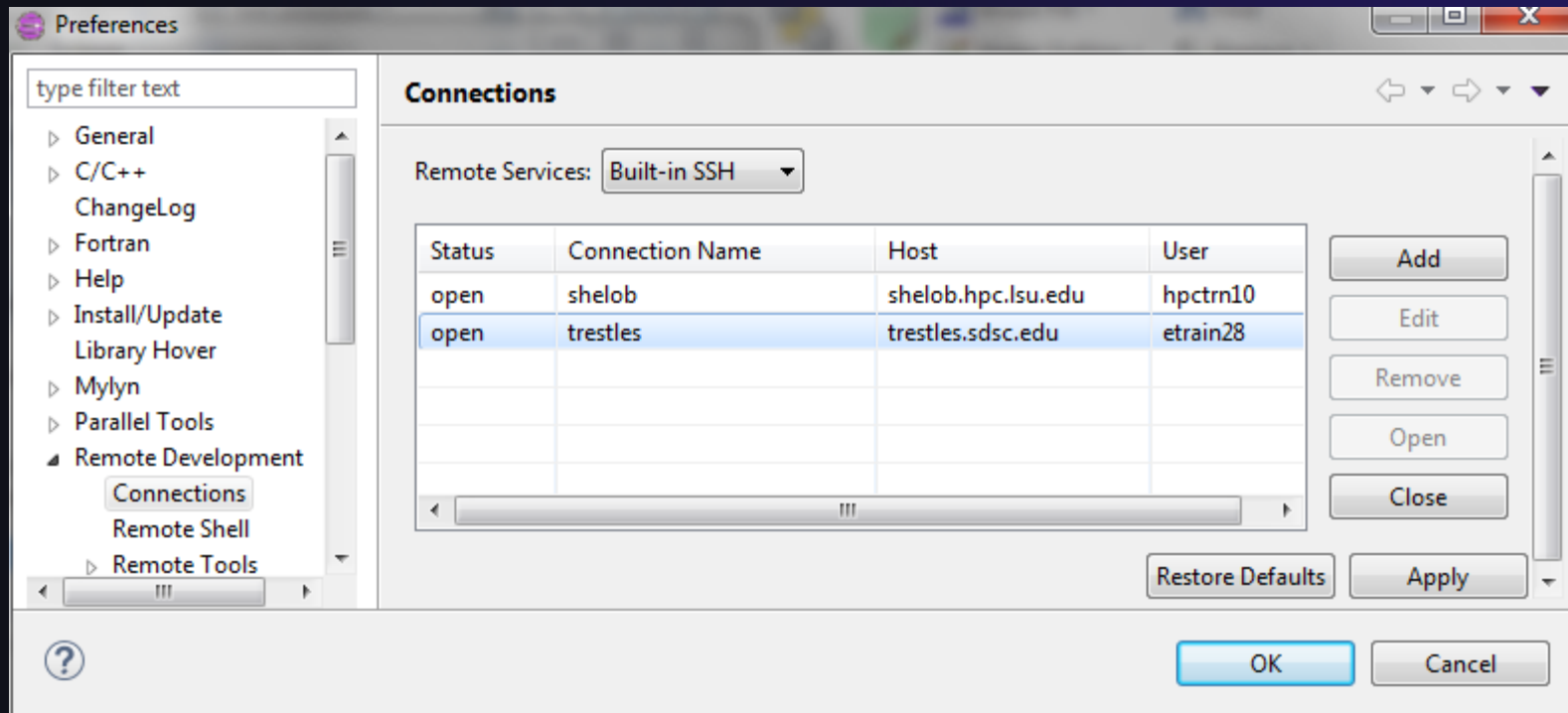
Terminal History



Add this line

History View

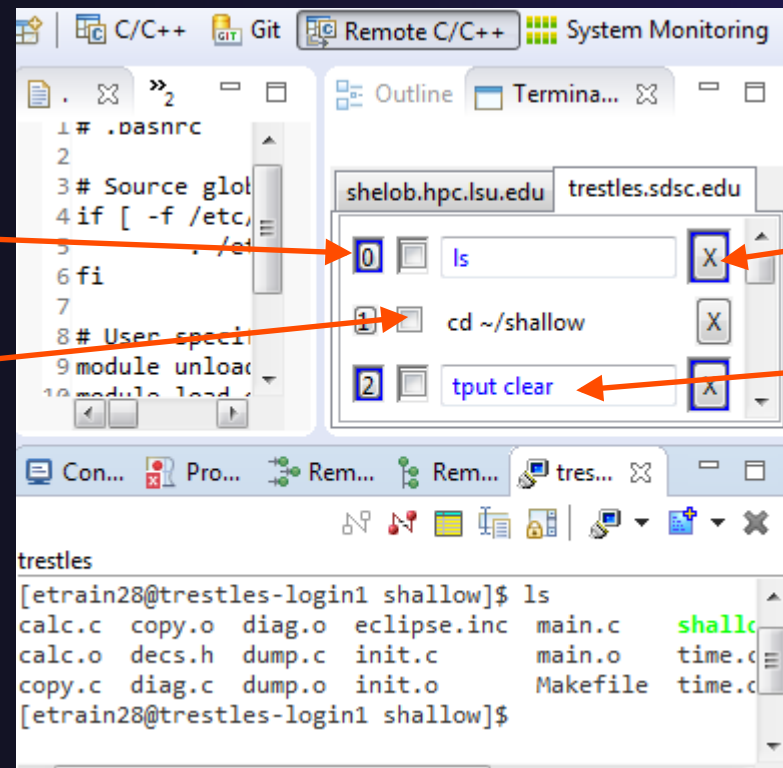
Terminal History



★ Close and open the connection, so that the bashrc will be re-run

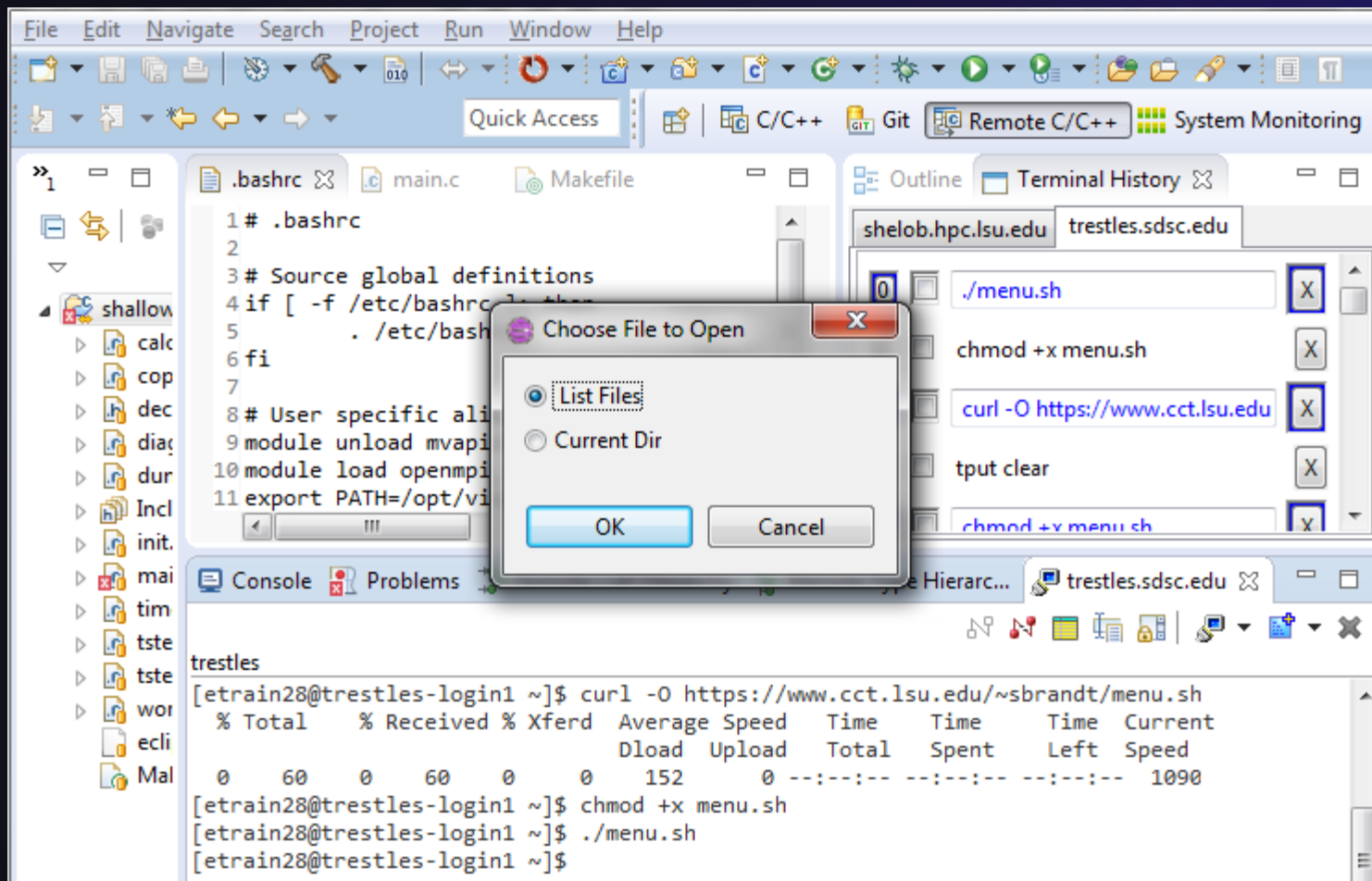
Terminal History

- ★ re-run the command
- ★ favorite the command



- ★ delete the command
- ★ edit the command

GUI Menus



NCSA/XSEDE Features

★ Objectives

- ★ Install NCSA's GSI auth and XSEDE support plug-ins
- ★ Become familiar with the System menu

★ Contents

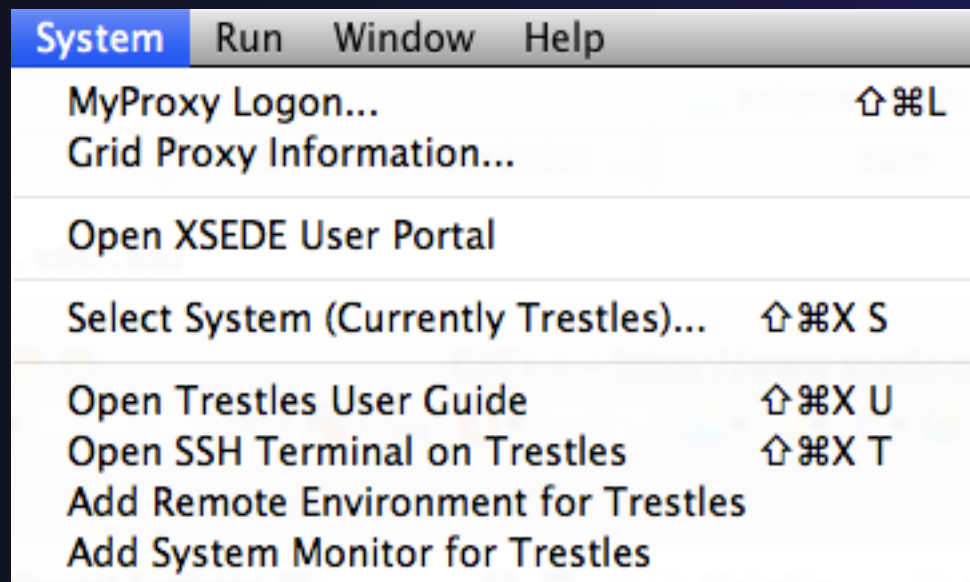
- ★ Capabilities
- ★ Installation

★ Prerequisites

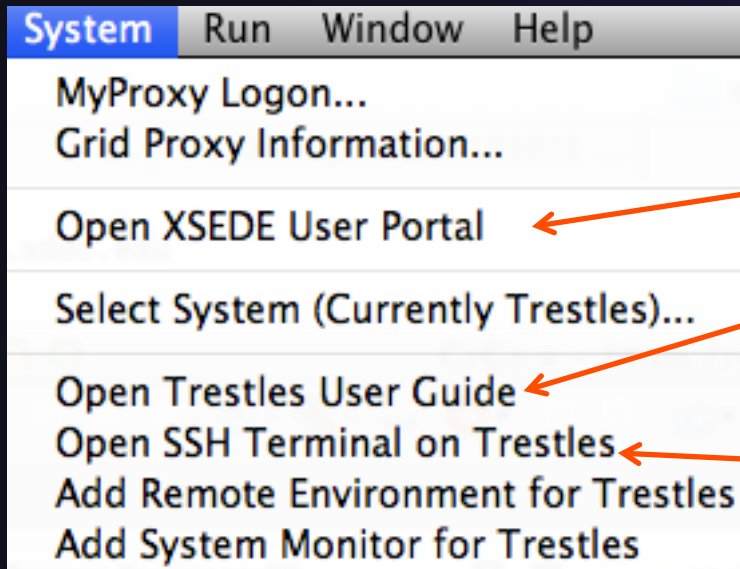
- ★ (none)

Additional Plug-ins from NCSA

- ★ NCSA publishes additional plug-ins that can be added to an existing PTP installation
- ★ Contribute a **System** menu to the menu bar with XSEDE- and NCSA-specific commands



System Menu



✦ Open Web content in Eclipse:

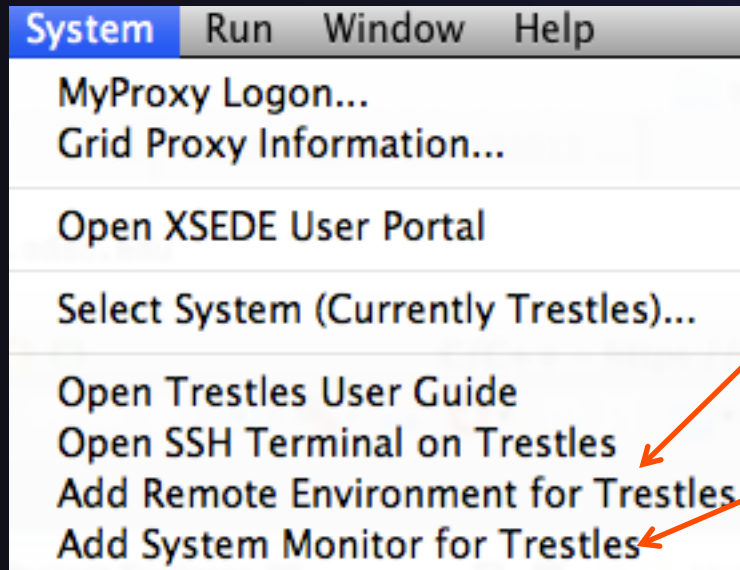
✦ **Open XSEDE User Portal**

✦ **Open User Guide** for a machine

✦ **Open an SSH terminal**
(as an Eclipse view)

Eclipse-integrated SSH terminals are provided by the Remote System Explorer (RSE), one of the features that is included in the Eclipse for Parallel Application Developers package.

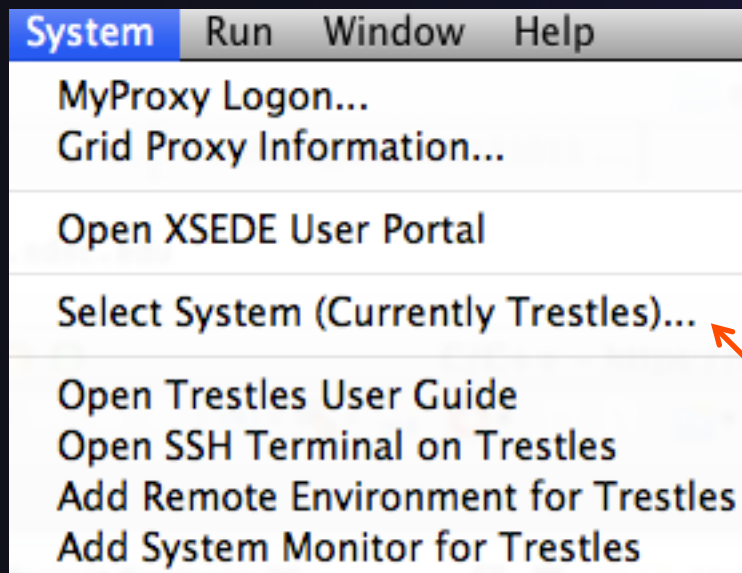
System Menu



★ Shortcuts for common PTP tasks:

- ★ **Add Remote Environment** adds a Remote Tools connection for a particular machine
- ★ **Add System Monitor** opens the System Monitoring perspective and begins monitoring a particular machine

System Menu



- ★ The plug-in is preconfigured with information about XSEDE and NCSA resources
- ★ The bottom four commands generally prompt for a system
- ★ **Select System** can be used to eliminate this prompt, so these commands always act on a particular system

MyProxy Logon



- ★ **MyProxy Logon** allows you to authenticate with a MyProxy server
 - ★ Often **myproxy.teragrid.org**
- ★ It stores a “credential,” which is usually valid for 12 hours
- ★ During these 12 hours, SSH connections to XSEDE resources will not require a password; they can use the stored credential
 - ★ However, you **must** enter the correct username for that machine!

Installation

1. Click **Help > Install New Software**
2. Click **Add** to open the Add Repository dialog
3. In the **Location** field, enter

`http://forecaster.ncsa.uiuc.edu/updates/luna`

and then click **OK** to close the Add dialog.

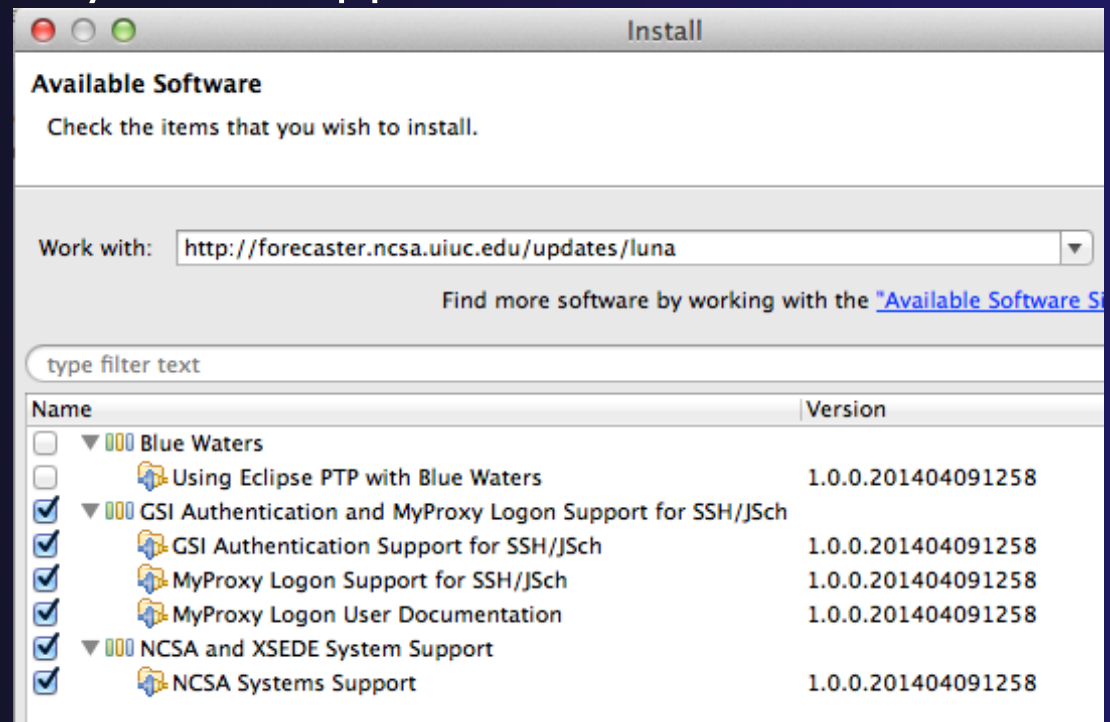
- Or, if you copied ncsa-update-site.zip from a USB drive, click **Archive**, select that file, and click **OK**.

See next slide...

Installation (2)

4. Select the following:
 - ✦ GSI Authentication and MyProxy Logon Support
 - ✦ NCSA and XSEDE System Support

5. Click **Next** and complete the installation



Performance Tuning and Analysis Tools

★ Objective

- ★ Become familiar with tools integrated with PTP, to help enhance performance of parallel applications

★ Contents

- ★ Overview of ETFw and Performance Tools

PTP/External Tools Framework

formerly “Performance Tools Framework”

Goal:

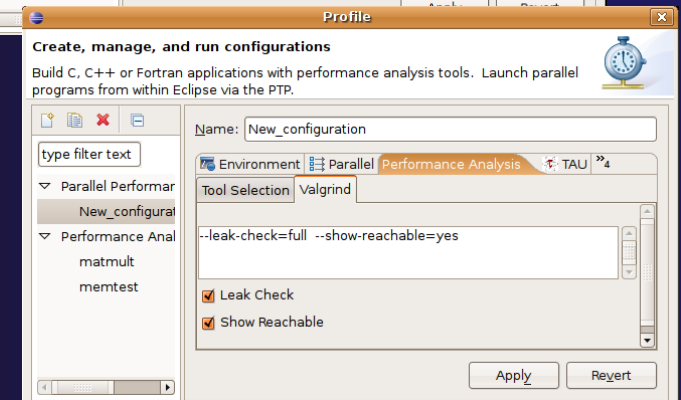
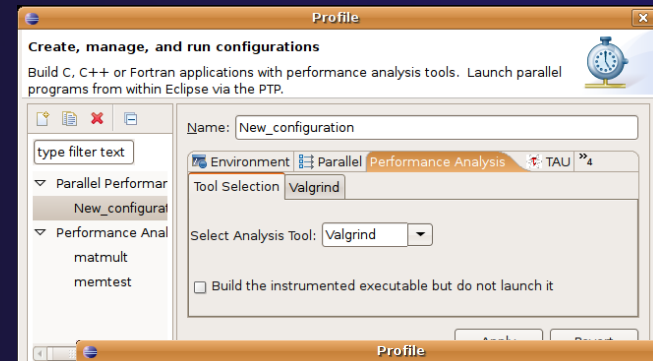
- ★ Reduce the “eclipse plumbing” necessary to integrate tools
- ★ Provide integration for instrumentation, measurement, and analysis for a variety of performance tools

- ★ Dynamic Tool Definitions: Workflows & UI
- ★ Tools and tool workflows are specified in an XML file
- ★ Tools are selected and configured in the launch configuration window
- ★ Output is generated, managed and analyzed as specified in the workflow
- ★ One-click ‘launch’ functionality
- ★ Support for development tools such as TAU, PPW and others.
- ★ Adding new tools is much easier than developing a full Eclipse plug-in

```

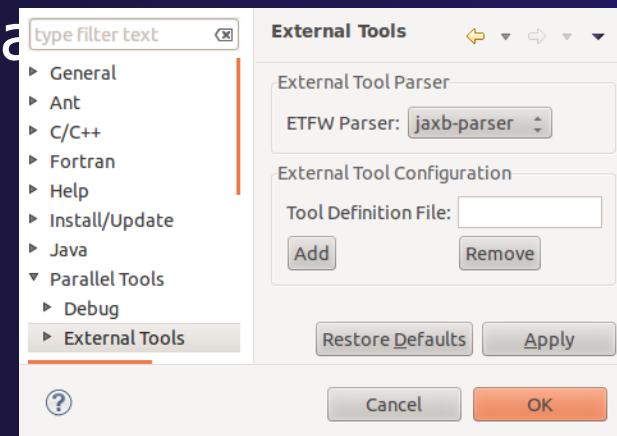
-<tool name="Valgrind">
-<execute>
  <utility command="bash" group="inbin"/>
  -<utility command="valgrind" group="valgrind">
    -<optionpane title="Valgrind" separatewith=" ">
      <togoption label="Leak Check" optname="--leak-check=full" tooltip="">
      <togoption label="Show Reachable" optname="--show-reachable=yes"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

```



SAX and JAXB Tool Definitions

- ✦ Prior implementations of ETFW used a simple SAX based schema to define tool workflows
- ✦ By default workflows now use the more powerful JAXB schema that defines PTP's resource manager
- ✦ Legacy workflows can still be loaded using the SAX parser in PTP options
 - ✦ Window->Preferences->Parallel Tools->External Tools



Performance Tuning and Analysis Tools - TAU

★ Objective

- ★ Become familiar with tools integrated with PTP, to help enhance performance of parallel applications

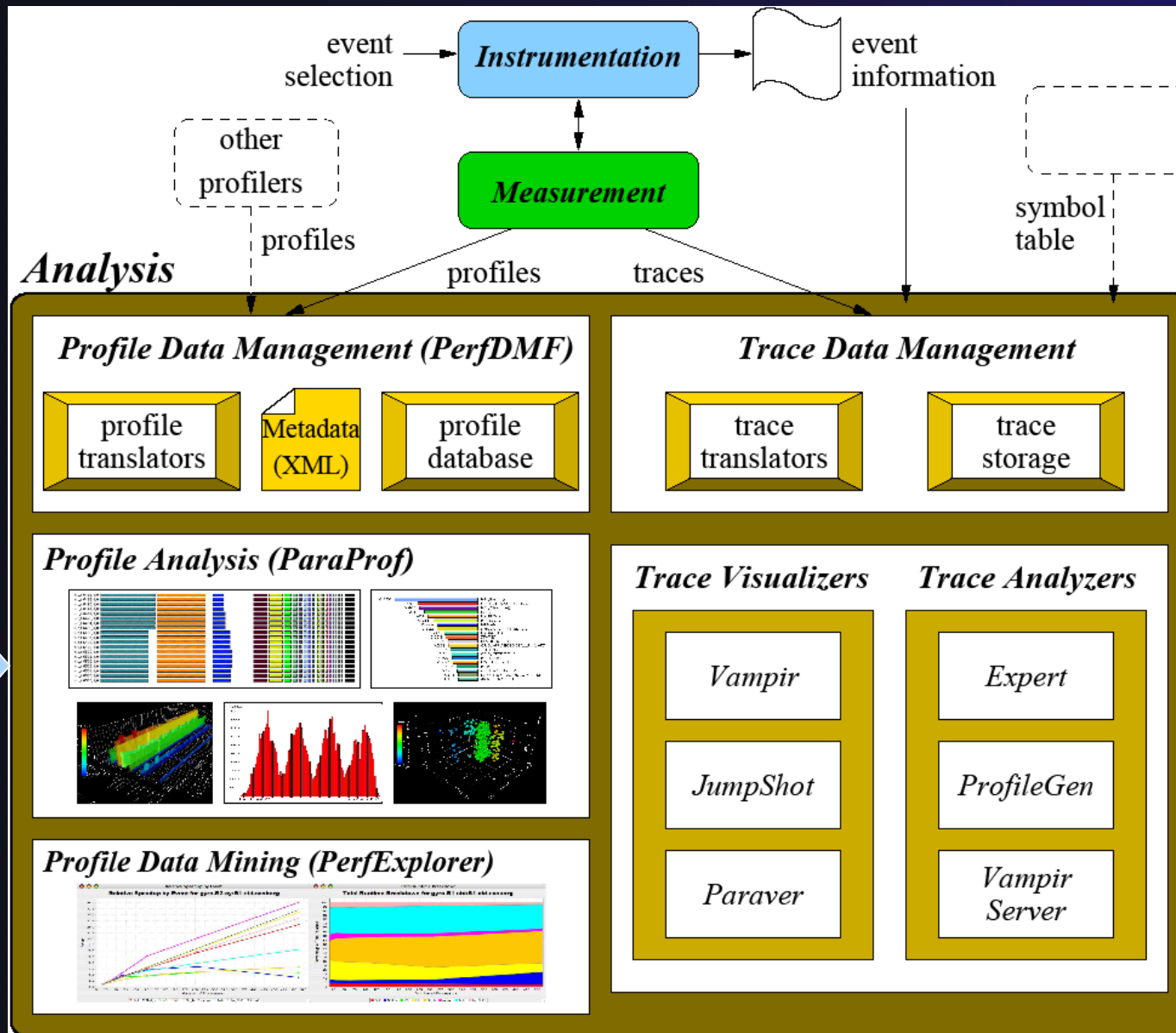
★ Contents

- ★ Performance Tuning and external tools:
 - ★ PTP External Tools Framework (ETFw), TAU
 - Hands-on exercise using TAU with PTP

TAU: Tuning and Analysis Utilities

- ★ TAU is a performance evaluation tool
- ★ It supports parallel profiling and tracing
 - ★ Profiling shows you how much (total) time was spent in each routine
 - ★ Tracing shows you *when* the events take place in each process along a timeline
- ★ TAU uses a package called PDT (Performance Database Toolkit) for automatic instrumentation of the source code
- ★ Profiling and tracing can measure time as well as hardware performance counters from your CPU (or GPU!)
- ★ TAU can automatically instrument your source code (routines, loops, I/O, memory, phases, etc.)
- ★ TAU runs on all HPC platforms and it is free (BSD style license)
- ★ TAU has instrumentation, measurement and analysis tools
 - ★ **paraprof** is TAU's 3D profile browser

TAU Performance System Architecture



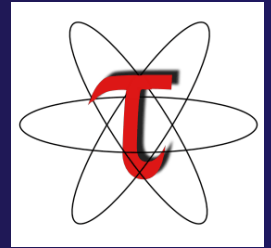
TAU Portal

TAU

TAU-3

PTP TAU plug-ins

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



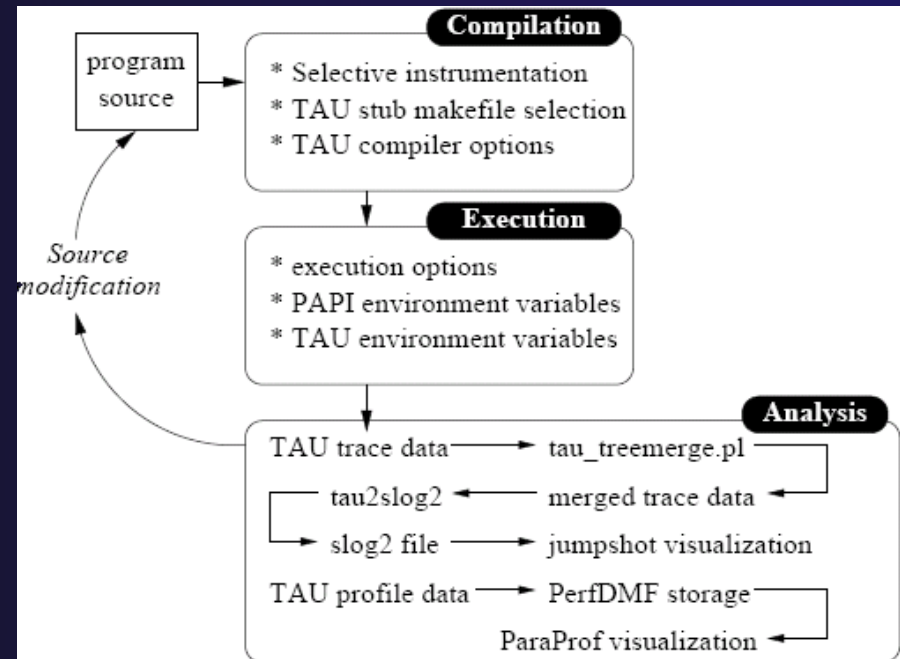
- ★ TAU (Tuning and Analysis Utilities)
- ★ First implementation of External Tools Framework (ETFw)
- ★ Eclipse plug-ins wrap TAU functions, make them available from Eclipse
- ★ Full GUI support for the TAU command line interface
- ★ Performance analysis integrated with development environment

TAU

TAU-4

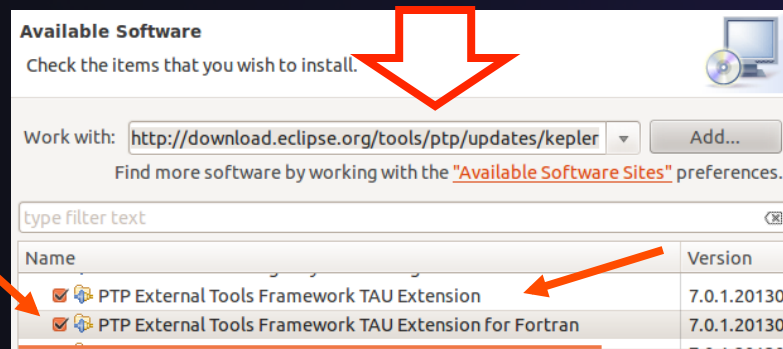
TAU Integration with PTP

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



TAU PTP Installation

- ✦ This tutorial assumes that the TAU extensions for PTP are installed – they are not included in the “Eclipse for Parallel Application Developers”
- ✦ The installation section (Module 1) shows how to install TAU and other features from the PTP update site – be sure TAU was selected



To confirm:

- ✦ Help>Install New Software...
- ✦ Select the link “What is already installed” at the bottom of the dialog
- ✦ You should see the TAU Extension

Installing TAU Analysis Tools

- ★ The TAU plugin can use ParaProf for visual analysis and TauDB for organization of profiles
- ★ To install these utilities on Mac or Linux platforms:
 - ★ Download (browser, curl or wget)
`tau.uoregon.edu/tautools-latest.tgz`
 - ★ `tar -zxf tautools-latest.tgz`
 - ★ `cd tautools-latest`
 - ★ `./configure`
 - ★ Set path as shown (launch eclipse from this environment)
 - ★ Run `taudb_configure` and follow the instructions
- ★ Java WebStart: `tau.uoregon.edu/paraprof`
- ★ TAU Installation, downloads and instructions: `tau.uoregon.edu`

Assumptions

- ★ Obtain and install TAU*
 - ★ Download at tau.uoregon.edu
 - ★ The website includes setup and user guides
- ★ Set up the \$PATH on the remote machine*
 - ★ For TAU you should be able to run 'which pprof' on a remote login and see a result from your TAU bin directory
 - ★ On trestles.sdsc.edu this is accomplished by including the TAU bin directory in the PATH variable in your bash configuration file
- ★ Include 'eclipse.inc' in the makefile*
 - ★ Create an empty eclipse.inc file in the same directory as the makefile
 - ★ Place 'include eclipse.inc' in the makefile after regular compiler definitions
 - ★ ETFw will modify eclipse.inc to set CC/CXX/FC variables

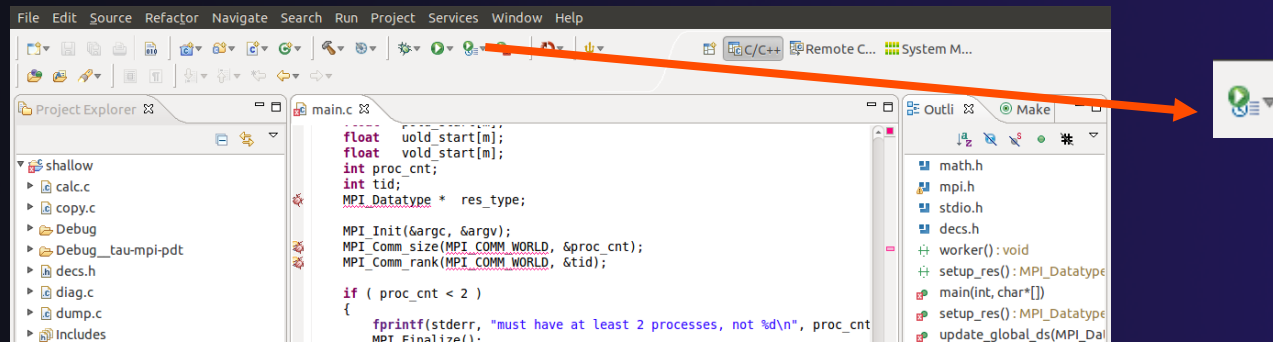
Selective Instrumentation

- ★ By default tau provides timing data for each subroutine of your application
- ★ Selective instrumentation allows you to include/exclude code from analysis and control additional analysis features
 - ★ Include/exclude source files or routines
 - ★ Add timers and phases around routines or arbitrary code
 - ★ Instrument loops
 - ★ Note that some instrumentation features require the PDT
- ★ Right click on `calc.c`, `init.c`, `diag.c` go to the Selective Instrumentation option and select Instrument Loops
- ★ Note the creation of `tau.selective` (refresh if needed)

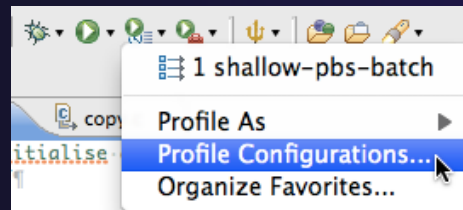


Begin Profile Configuration

- ✦ The ETFw uses the same run configurations and resource managers as debugging/launching
- ✦ Click on the 'Run' menu or the right side of the Profile button



- ✦ From the dropdown menu select 'Profile configurations...'

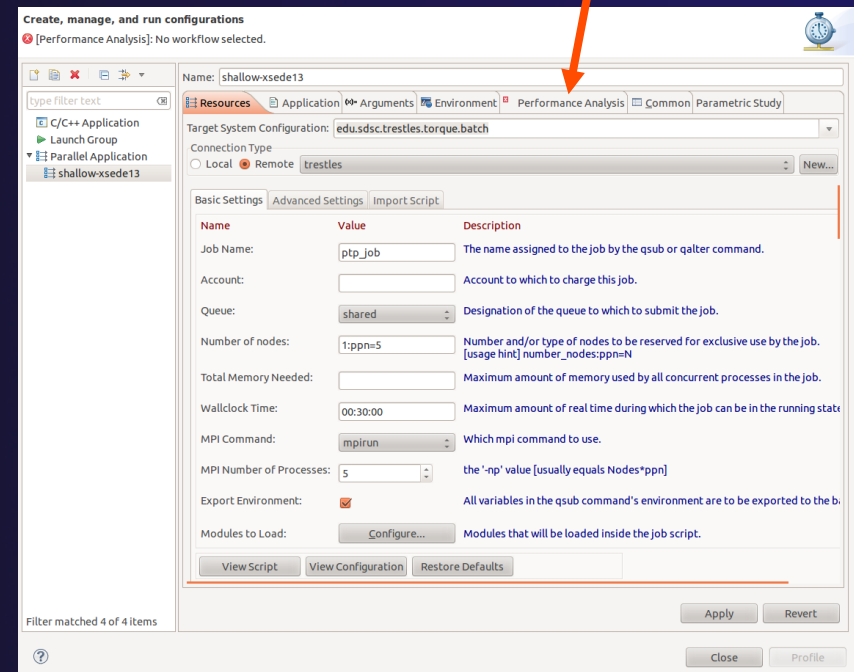




Select Configuration

- ★ Select the shallow configuration prepared earlier
- ★ The Resource and Application configuration tabs require little or no modification
 - ★ We are using the same resource manager and Torque settings
 - ★ Since we are using a makefile project the application will be rebuilt in and run from the previously selected location

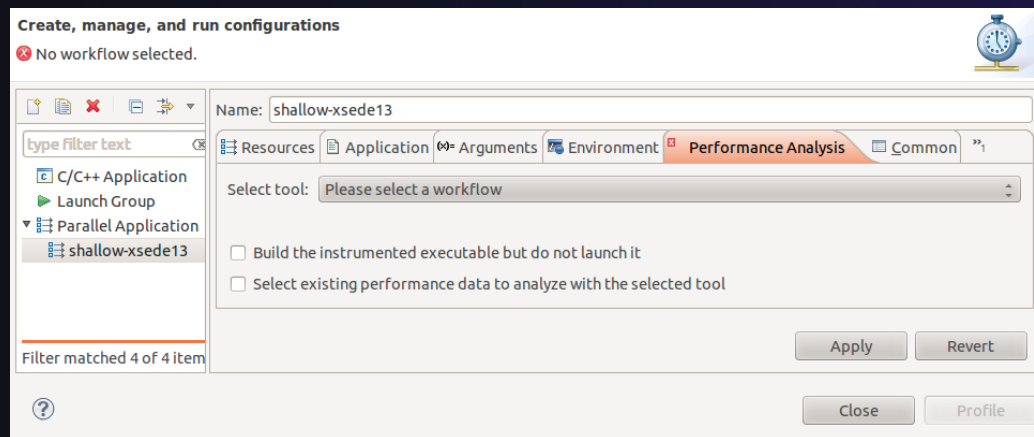
Performance Analysis tab is present in the **Profile Configurations** dialog





Select Tool/Workflow

- ★ Select the **Performance Analysis** tab and choose the TAU tool set in the 'Select Tool' dropdown box
 - ★ Other tools may be available, either installed as plug-ins or loaded from workflow definition XML files
 - ★ Configuration sub-panes appear depending on the selected tool

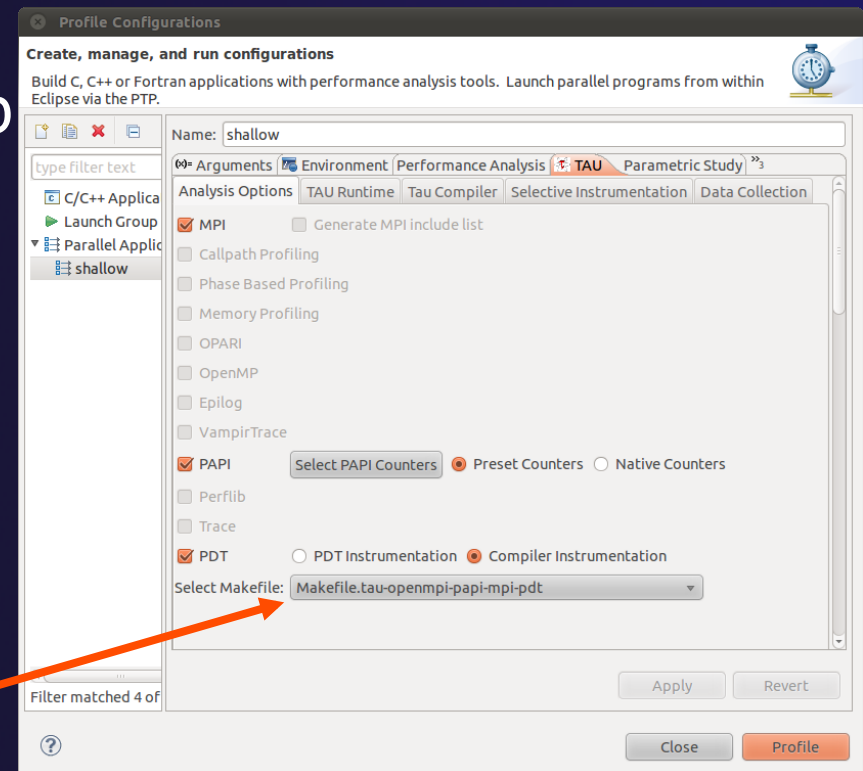


← Tabs may be hidden if the window is too small



Select TAU Configuration

- ★ Choose the TAU Makefile tab
 - ★ All TAU configurations in remote installation are available
 - ★ Check MPI and PDT checkboxes to filter listed makefiles
 - ★ Make your selection in the **Select Makefile:** dropdown box
 - ★ Select Makefile.tau-mpi-pdt



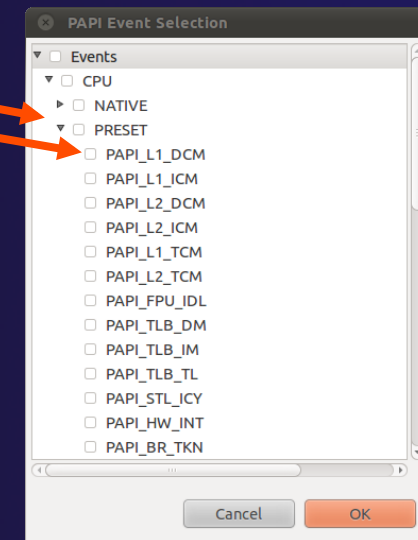


Choose PAPI Hardware Counters

- ★ When a PAPI-enabled TAU configuration is selected the PAPI Counter tool becomes available



- ★ Select the 'Select PAPI Counters' button to open the tool
- ★ Open the PRESET subtree
- ★ Select PAPI_L1_DCM (Data cache misses)
- ★ Scroll down to select PAPI_FP_INS (Floating point instructions)
- ★ Invalid selections are automatically excluded
- ★ Select **OK**

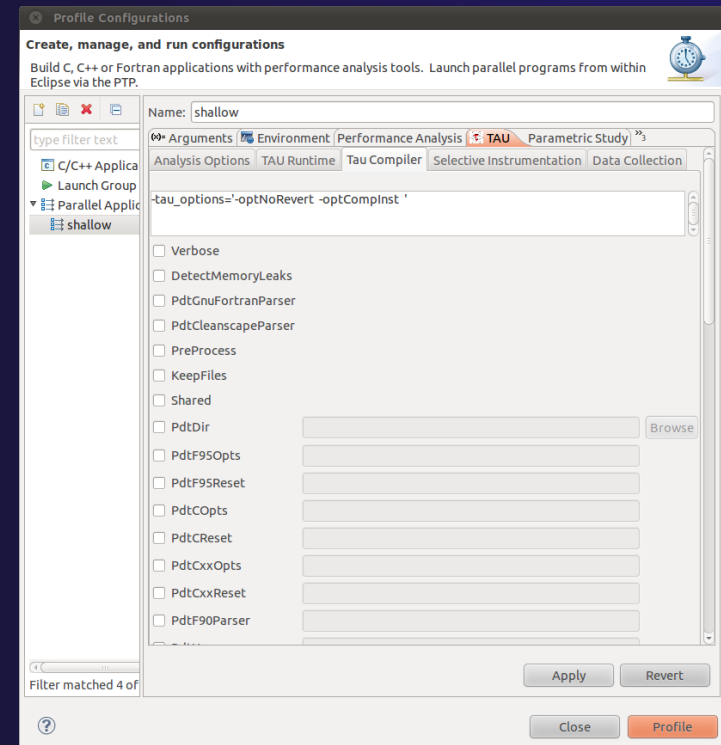


- ★ **Not available on trestles.sdsc.edu**



Compiler Options

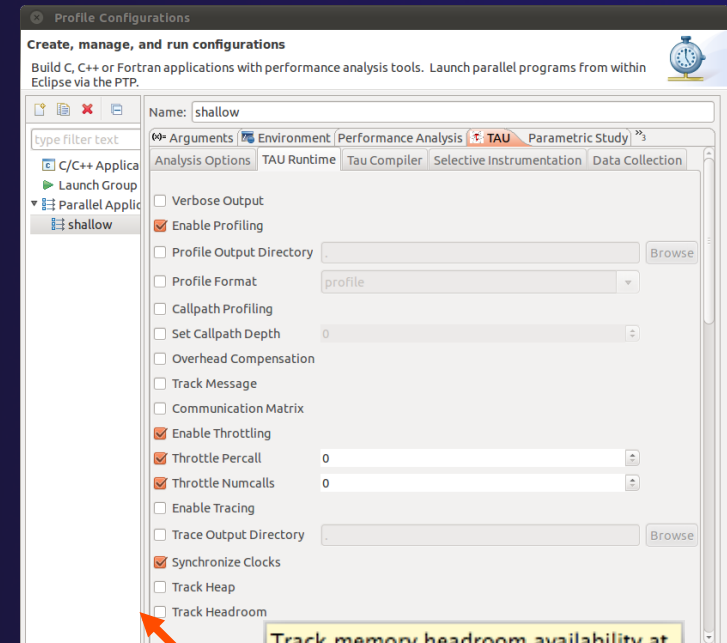
- ✦ TAU Compiler Options
 - ✦ Set arguments to TAU compiler scripts
 - ✦ Control instrumentation and compilation behavior
 - ✦ Verbose shows activity of compiler wrapper
 - ✦ KeepFiles retains instrumented source
 - ✦ PreProcess handles C type ifdefs in fortran
- ✦ In the Selective Instrumentation tab select Internal then hit Apply
- ✦ Scroll to bottom of the Tau Compiler tab and activate TauSelectFile to use tau.selective





Runtime Options

- ✦ TAU Runtime options
 - ✦ Set environment variables used by TAU
 - ✦ Control data collection behavior
 - ✦ Verbose provides debugging info
 - ✦ Callpath shows call stack placement of events
 - ✦ Throttling reduces overhead
 - ✦ Tracing generates execution timelines
- ✦ Set Profile Format to merged



Hover help



Working with Profiles

- ★ Profiles are uploaded to selected database
- ★ A text summary may be printed to the console
- ★ Profiles may be uploaded to the TAU Portal for viewing online
 - ★ tau.nic.uoregon.edu
- ★ Profiles may be copied to your workspace and loaded in ParaProf from the command line. Select Keep Profiles

Analysis Options | TAU Runtime | Tau Compiler | Selective Instrumentation | Data Collection

Select Database:

Keep profiles

Print Profile Summary

Upload profile data to TAU Portal

Console | Properties | Problems | Tasks

TAU Profile Output

MULTI_GET_TIME_OF_DAYReading Profile files in profile.*

FUNCTION SUMMARY (total):

%Time	Exclusive msec	Inclusive total msec	#Call	#Subrs	Inclusive Name usec/call
100.0	196	16,797	9	9	1866428 .TAU application
98.8	56	16,601	9	3662	1844640 main
68.7	11,538	11,538	9	0	1282002 MPI_Init()
26.0	204	4,360	8	59684	545009 worker
9.5	803	1,602	80000	160000	20 neighbour_receive
8.3	789	1,402	80000	160000	18 neighbour_send
8.3	234	1,398	8000	64000	175 time_load
7.1	1,188	1,188	81968	0	14 MPI_Recv()
6.5	182	1,085	8000	48000	136 calc_load
...

TAU Portal URL:

TAU Portal Username:

TAU Portal Password:

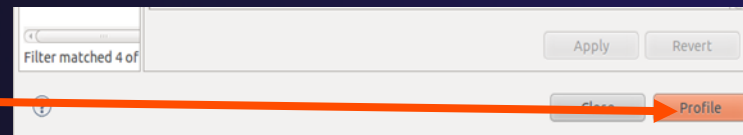
Select a workspace

Select Workspace:

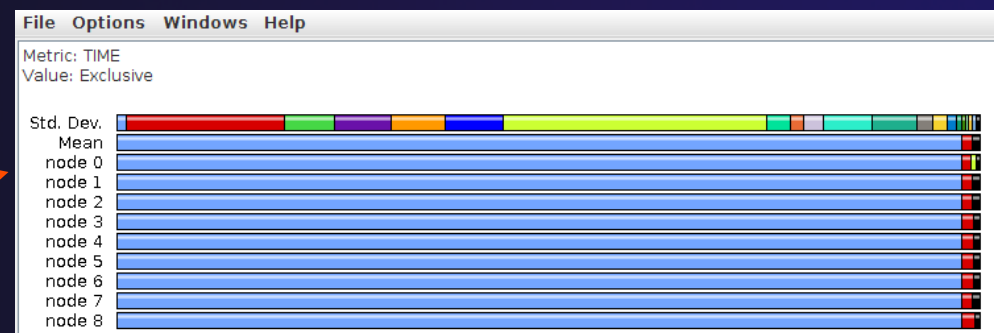


Launch TAU Analysis

- ★ Once your TAU launch is configured select 'Profile'
- ★ Notice that the project rebuilds with TAU compiler commands
- ★ The project will execute normally but TAU profiles will be generated
- ★ TAU profiles will be processed as specified in the launch configuration.
- ★ If you have a local profile database the run will show up in the Performance Data Management view
 - ★ Double click the new entry to view in ParaProf
 - ★ Right click on a function bar and select **Show Source Code** for source callback to Eclipse



TAU

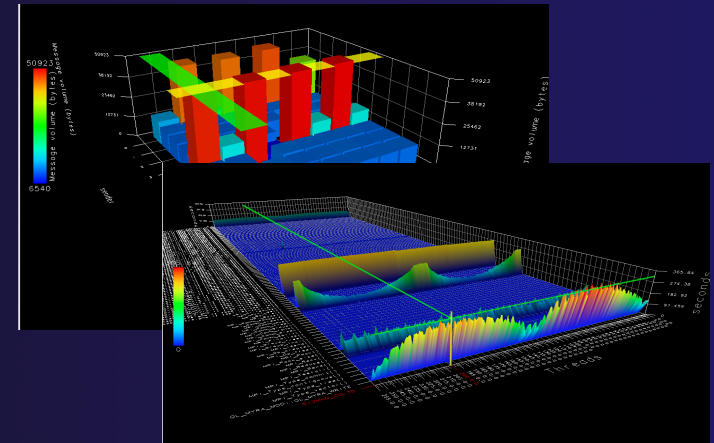


TAU-18



Paraprof

- ★ Use ParaProf for profile visualization to identify performance hotspots
 - ★ Inefficient sequential computation
 - ★ Communication overhead
 - ★ IO/Memory bottlenecks
 - ★ Load imbalance
 - ★ Suboptimal cache performance
- ★ Compare multiple trials in PerfExplorer to identify performance regressions and scaling issues
- ★ To use ParaProf, install TAU from tau.uoregon.edu or use Java webstart from tau.uoregon.edu/paraprof





Exercise

- ★ Multi-Trial profile comparison
 1. Edit the shallow Makefile, adding -O3 to CFLAGS and FFLAGS
 2. Rerun the analysis (Run->Profile Configurations. Hit Profile)
 3. A second trial, distinguished by a new timestamp, will be generated
 - ★ It will appear in your Performance Data Manager view if a profile database is available
 - ★ Also present in the Profile subdirectory of your project directory
 - ★ If you do not see a Profile directory right click on your project and go to Synchronization->'Sync All Now'
 4. Load the two trials in paraprof (on the command line: paraprof / path/to/tauprofile.xml)
 5. Open Windows->ParaProf Manager
 6. Expand your database down to reveal all trials
 7. Right click on each trial and click 'Add Mean to Comparison Window' to visualize the two trials side by side

PerfSuite

✦ Objectives

- ✦ Become familiar with PerfSuite
- ✦ Install Perfsuite Feedback Feature

✦ Contents

- ✦ Running a PerfSuite analysis within Eclipse/PTP
- ✦ Using the PerfSuite Feedback view to view the results generated by PerfSuite

✦ Prerequisites

- ✦ Compile and Install PerfSuite command line tools
- ✦ Install supporting libraries used by PerfSuite (e.g. PAPI)



PerfSuite

- ★ PerfSuite is a collection of tools and libraries for software performance analysis on Linux-based systems.
- ★ It supports counting and profiling using
 - ★ CPU and other hardware performance counters
 - ★ Network, thermal
 - ★ Interval timer profiling
- ★ PerfSuite can be used from the command line or within Eclipse/PTP
- ★ PerfSuite results can be viewed via its Feedback View.



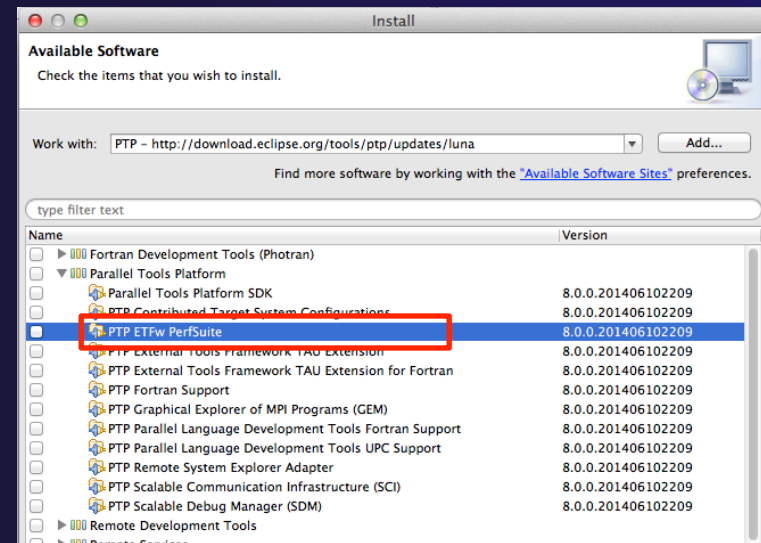
PerfSuite Demo

- ✦ Demonstration using a local project
 - ✦ Setup and run **psrun** to gather performance information
 - ✦ Setup and run **psprocess** to post process results for feedback view
- ✦ Should work on an synchronized project – not tested



Installing PerfSuite plug-ins

- ★ PerfSuite Eclipse plug-ins are not in the Parallel Package.
- ★ Update PTP per the slides on installation – and select PerfSuite as follows:
- ★ Help>Install New Software...
 - ★ <http://download.eclipse.org/tools/ptp/updates/luna>
 - ★ Select “PTP ETFw PerfSuite”
 - ★ Next, Next, Accept, Finish



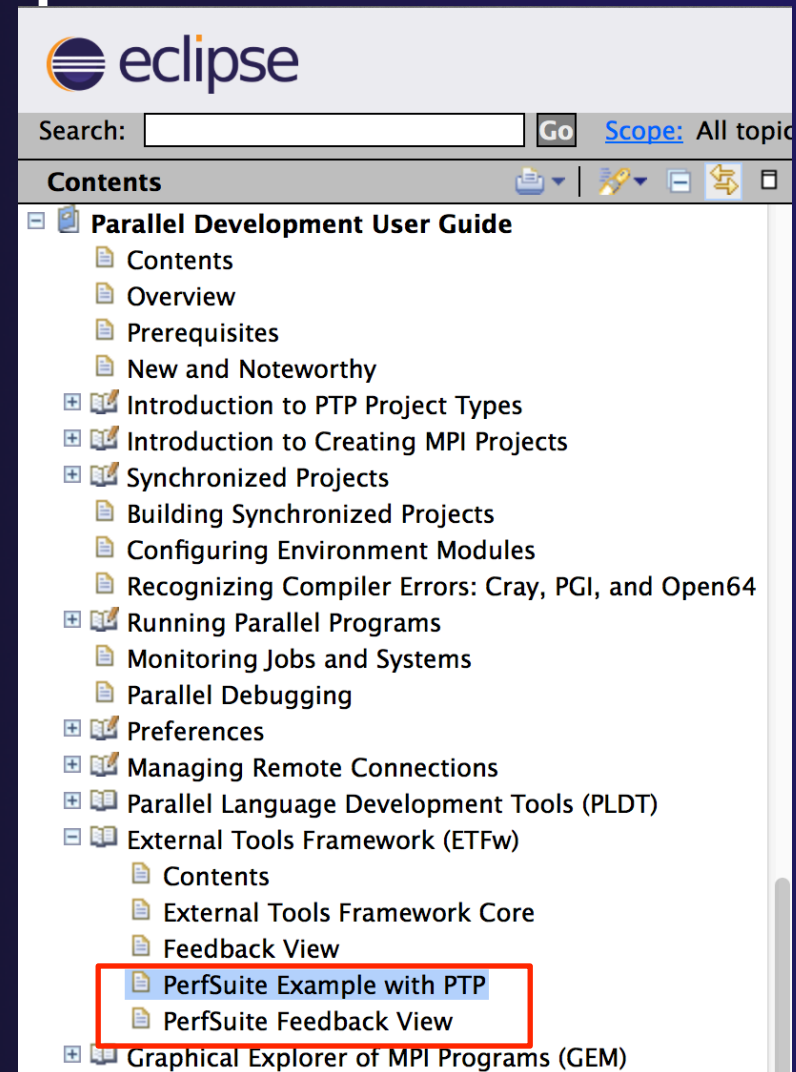


Prerequisites

- ★ To analyze your application using Perfsuite, **psrun** and **psprocess** must be available in your system path.
 - ★ If they aren't, then you will need to compile and install them from the source code found at <http://perfsuite.ncsa.illinois.edu>
- ★ You may also need to install additional software that is required by PerfSuite
 - ★ See <http://perfsuite.ncsa.illinois.edu>

PerfSuite Help

<http://help.eclipse.org> or
help within the PTP
Eclipse workbench



Launch profile configuration



★ Run > Profile Configurations...

★ Fill out Resources tab

Fill out Application tab



★ Application tab

The screenshot shows the 'Profile Configurations' dialog box in Eclipse. The title bar reads 'Profile Configurations'. Below the title bar, it says 'Create, manage, and run configurations' and 'Build C, C++ or Fortran applications with performance analysis tools. Launch parallel programs from within Eclipse via the PTP.' The dialog has a tree view on the left with a search filter 'type filter text'. The tree view shows a hierarchy: C/C++ Application, Launch Group, Parallel Application, and shallow-local. The 'shallow-local' configuration is selected. The main area has tabs: Resources, Application (highlighted with a red box), Arguments, Environment, Performance Analysis, Common, and Parametric Study. The 'Application' tab is active, showing fields for Name (shallow-local), Project (shallow-local), Application program (/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow), Path to local executable, and C/C++/Fortran Build Configuration (Default). There are 'Apply', 'Revert', 'Close', and 'Profile' buttons at the bottom.

Performance Analysis - PSRun



★ PSRun Settings

The screenshot shows the Eclipse IDE's "Profile Configurations" dialog box. The title bar reads "Profile Configurations" and the subtitle is "Create, manage, and run configurations". Below the subtitle, it says "Build C, C++ or Fortran applications with performance analysis tools. Launch parallel programs from within Eclipse via the PTP." The dialog is divided into several tabs: Resources, Application, Arguments, Environment, Performance Analysis (highlighted with a red box), Common, and Parametric Study. The "Performance Analysis" tab is active, showing a "Select tool:" dropdown set to "PerfSuite". Below this, there are two sub-tabs: "PSRun" (highlighted with a red box) and "PSProcess". The "PSRun" sub-tab is selected, displaying various configuration options: "File Format" is set to "xml"; "Configuration" is checked and set to "/usr/local/share/perfsuite/xml/pshwpc/papi_profile_cycles.xml"; "MD5 checksum" is unchecked; "Fork Support" is checked; "Domain" is set to "user"; "POSIX" is unchecked; "Resource Usage" is set to "1"; "Resource Usage Only" is set to "1"; "Write Signal Numbers" is unchecked; "Overflow Threshold" is empty; "Output Prefix" is checked and set to "psprocess". At the bottom of the "PSRun" sub-tab is a "Restore Defaults" button. Below the sub-tabs, there are two unchecked checkboxes: "Build the instrumented executable but do not launch it" and "Select existing performance data to analyze with the selected tool". At the bottom right of the dialog are "Apply", "Revert", "Close", and "Profile" buttons. The left sidebar shows a tree view with "shallow-local" selected, and a filter bar at the top left.

Performance Analysis PSProcess



★ PSProcess settings

The screenshot shows the Eclipse IDE's Profile Configurations dialog for a configuration named 'shallow-local'. The 'Performance Analysis' tab is selected and highlighted with a red box. Within this tab, the 'PSProcess' tool is selected from the 'Select tool:' dropdown, also highlighted with a red box. The configuration settings are as follows:

- XML Output
- Limit Output
- Reporting Threshold: [empty field]
- Custom Metrics: [empty field] Browse...
- Output Directory: Browse...
- Input file pattern:
- Output:
-

At the bottom of the dialog, there are two unchecked options:

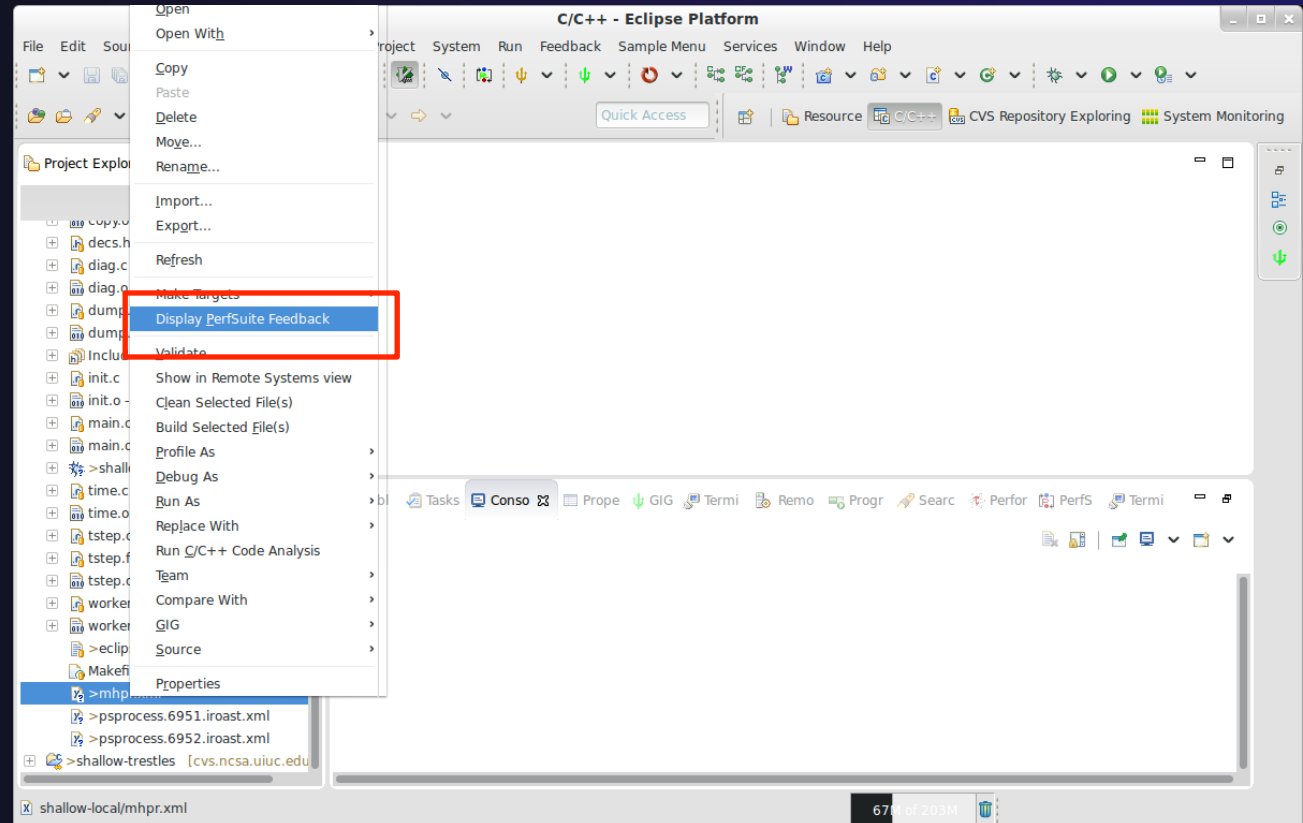
- Build the instrumented executable but do not launch it
- Select existing performance data to analyze with the selected tool

Buttons at the bottom right include 'Apply', 'Revert', 'Close', and 'Profile'. The status bar at the bottom left indicates 'Filter matched 4 of 4 items'.

Launching feedback view



- ★ Right click on mhpr.xml
- ★ Display PerfSuite Feedback





PerfSuite Feedback view opens

★ The Perfsuite Feedback view opens

The screenshot shows a window titled "PerfSuite Feedback Items" with a search icon. The window contains a table with the following columns: Module, File, Function, LineNo, and Number of Samples. The data is filtered to show results for the module "tstep.f90".

Module	File	Function	LineNo	Number of Samples
troast-PID_6952-thread_0				
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	tstep.f90	tstep	65	9
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	tstep.f90	tstep	74	7
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	tstep.f90	tstep	75	6
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	tstep.f90	tstep	64	2
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	tstep.f90	tstep	73	2
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	tstep.f90	tstep	82	2
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	tstep.f90	tstep	63	1
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	tstep.f90	tstep	83	1
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	tstep.f90	tstep	84	1
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	time.c	timetend	47	6
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	time.c	timetend	52	5
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	time.c	timetend	49	4
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	time.c	timetend	50	4
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	time.c	timetend	53	3
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	time.c	timetend	44	2
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	time.c	timetend	48	1
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	time.c	timetend	51	1
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	calc.c	calcuvzh	47	4
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	calc.c	calcuvzh	45	3
/home/cnavarro/workspaces/runtime-Eclipse-Luna/shallow-local/shallow	calc.c	calcuvzh	46	3

Navigate from feedback view



- ★ Double click on a line in the feedback view to go to the corresponding line in the source code

The screenshot shows the Eclipse IDE with a C++ source file open. The source code is as follows:

```

53 call c_f_pointer(cv, v, shape=[m, n])
54 call c_f_pointer(cpnew, pnew, shape=[m, n])
55 call c_f_pointer(cunew, unew, shape=[m, n])
56 call c_f_pointer(cvnnew, vnnew, shape=[m, n])
57 call c_f_pointer(cdpt, dpdt, shape=[m, n])
58 call c_f_pointer(cdvt, dvdt, shape=[m, n])
59 call c_f_pointer(cdvt, dvdt, shape=[m, n])
60
61 do j = jstart+1, jend+1
62 do i = 1, m
63   pnew(i,j) = pold(i,j) + tdt*dpdt(i,j)
64   unew(i,j) = uold(i,j) + tdt*dudt(i,j)
65   vnnew(i,j) = vold(i,j) + tdt*dvdt(i,j)
66 end do
67 end do
68
69 ! Don't apply time filter on first step
70 if ( firststep == 0 ) then
71 do j = jstart+1, jend+1
72 do i = 1, m
73   pold(i,j) = p(i,j)+alpha*(pnew(i,j)-2.0_c_float*p(i,j)+pold(i,j))
74   uold(i,j) = u(i,j)+alpha*(unew(i,j)-2.0_c_float*u(i,j)+uold(i,j))
75   vold(i,j) = v(i,j)+alpha*(vnnew(i,j)-2.0_c_float*v(i,j)+vold(i,j))
76 end do
77 end do
78 end if
79
80 do j = jstart+1, jend+1
81 do i = 1, m
82   p(i,j) = pnew(i,j)
83   u(i,j) = unew(i,j)
84   v(i,j) = vnnew(i,j)
85 end do
86 end do
87 end subroutine
88

```

The PerfSuite Feedback Items view is open at the bottom, showing the following data:

Module	File	Function	LineNo	Number of Samples	
iroast-PID_27850-thread_0	/home/cnavarro/workspaces/xsede14-tuto	tstep.f90	tstep	73	9
	/home/cnavarro/workspaces/xsede14-tuto	tstep.f90	tstep	75	7
	/home/cnavarro/workspaces/xsede14-tuto	tstep.f90	tstep	74	5
	/home/cnavarro/workspaces/xsede14-tuto	tstep.f90	tstep	65	4
	/home/cnavarro/workspaces/xsede14-tuto	tstep.f90	tstep	63	3
	/home/cnavarro/workspaces/xsede14-tuto	tstep.f90	tstep	64	3
	/home/cnavarro/workspaces/xsede14-tuto	tstep.f90	tstep	84	3

Tutorial Wrap-up

✦ Objective

- ✦ How to find more information on PTP
- ✦ Learn about other tools related to PTP
- ✦ See PTP upcoming features

✦ Contents

- ✦ Links to other tools, including performance tools
- ✦ Planned features for new versions of PTP
- ✦ Additional documentation
- ✦ How to get involved

Useful Eclipse Tools

- ✦ Linux Tools (autotools, valgrind, Oprofile, Gprof)
 - ✦ <http://eclipse.org/linuxtools> (part of Parallel package)
- ✦ Python
 - ✦ <http://pydev.org>
- ✦ Ruby
 - ✦ <http://www.apptana.com/products/radrails>
- ✦ Perl
 - ✦ <http://www.epic-ide.org>
- ✦ VI bindings
 - ✦ Wrapper (open source) - <http://vrwrapper.sourceforge.net>
 - ✦ viPlugin (commercial) - <http://www.viplugin.com>

Online Information

- ★ Information about PTP
 - ★ PTP online help
 - ★ <http://help.eclipse.org>
 - ★ Main web site for downloads, documentation, etc.
 - ★ <http://eclipse.org/ptp>
 - ★ Wiki for designs, planning, meetings, etc.
 - ★ <http://wiki.eclipse.org/PTP>

- ★ Information about Photran
 - ★ Main web site for downloads, documentation, etc.
 - ★ <http://eclipse.org/photran>

Mailing Lists

- ★ User Mailing Lists
 - ★ PTP
 - ★ <http://dev.eclipse.org/mailman/listinfo/ptp-user>
 - ★ Photran
 - ★ <http://dev.eclipse.org/mailman/listinfo/photran>
 - ★ Major announcements (new releases, etc.) - low volume
 - ★ <http://dev.eclipse.org/mailman/listinfo/ptp-announce>

- ★ Developer Mailing Lists
 - ★ Developer discussions - higher volume
 - ★ <http://dev.eclipse.org/mailman/listinfo/ptp-dev>

Getting Involved

- ★ See <http://eclipse.org/ptp>
- ★ Read the developer documentation on the wiki
 - ★ <http://wiki.eclipse.org/PTP>
- ★ Join the mailing lists
- ★ Attend the monthly developer meetings
 - ★ Conf Call Monthly: Second Tuesday, 1:00 pm ET
 - ★ Details on the PTP wiki
- ★ Attend the monthly user meetings
 - ★ Teleconf Monthly: 4th Wednesday, 1:00 pm ET
 - ★ Details on the PTP wiki

XSEDE BOF

- ✦ PTP BOF is Tuesday morning 7:00 am
- ✦ PTP Hackathon
 - ✦ At LSU Aug 12-13, 2014
 - ✦ Become familiar with PTP development
 - ✦ Get help contributing to PTP
 - ✦ Eat Cajun!
 - ✦ \$\$ Travel assistance may be available – please ask!

PTP Tutorial Wrap-Up

★ Your feedback is valuable!

Thanks for attending
We hope you found it useful