



parallel tools platform

<http://eclipse.org/ptp>

Introduction to the Eclipse Parallel Tools Platform

Jay Alameda, NCSA
jalameda@ncsa.illinois.edu

Jeff Overbey, Auburn U.
jeffreyoverbey@acm.org

Slides by
Greg Watson, Beth Tibbitts,
Jay Alameda, Galen Arnold, Steve Brandt, Chris Navarro, Jeff Overbey, and Wyatt Spear

July 27, 2015

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



Tutorial Outline

Time (Tentative)	Module	Topics	Presenter
1:30-2:00	Eclipse Installation Intro/Overview	<ul style="list-style-type: none">✦ Installation of Eclipse and PTP✦ Eclipse overview	Jeff/Jay
2:00-2:45	Eclipse basics	<ul style="list-style-type: none">✦ Synchronized projects✦ Git support✦ Editor features	Jeff
2:45-3:15	BREAK		
3:15-3:30	(continue Basics)		Jeff
3:30-4:30	Build & Run (1:00)	<ul style="list-style-type: none">✦ GUI terminal✦ Building with Make✦ Target system configurations✦ Launching a parallel application✦ Modules/environment mgmt✦ Wrap-up	Jay

Installation instructions (and these slides) are available at
<http://wiki.eclipse.org/PTP/tutorials/XSEDE15>

Final Slides, Installation Instructions

- ★ Please go to <http://wiki.eclipse.org/PTP/tutorials/XSEDE15> for slides and installation instructions
- ★ Local copy of downloads: <http://dns.conference.xsede.org/>

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 (4.5) is also known as "Mars"
- ✦ 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



Eclipse for Parallel Application Developers
Downloaded 46,871 Times [Details](#)

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

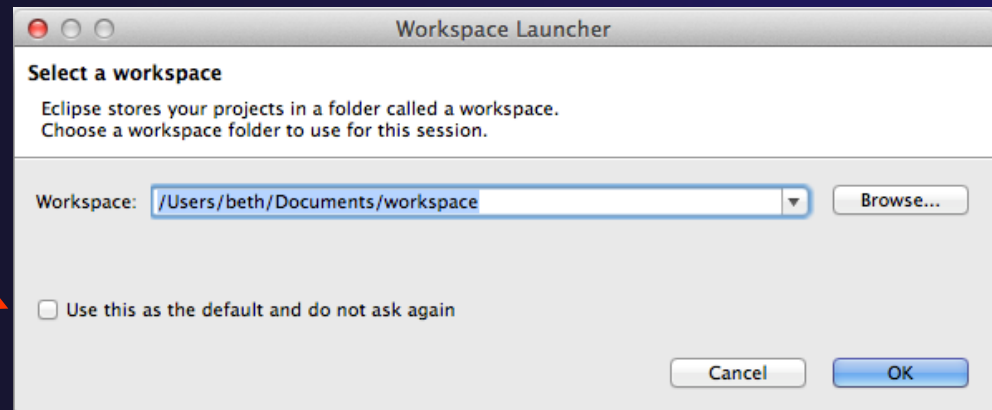
- ✦ Open the **eclipse** folder
- ✦ Double-click on the **eclipse** executable



Specifying A Workspace

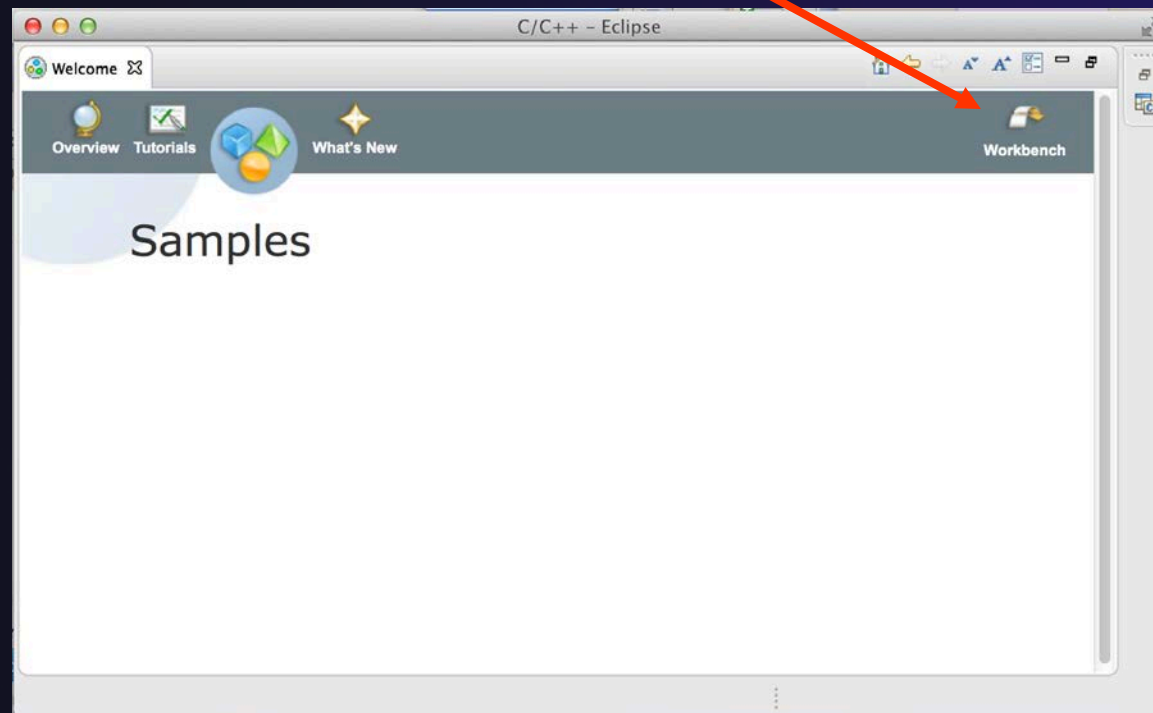
- ✦ Eclipse prompts for a workspace location at startup time
- ✦ The workspace contains all user-defined data
 - ✦ Projects and resources such as folders and files
 - ✦ The default workspace location is fine for this tutorial

The prompt can be turned off



Eclipse Welcome Page

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

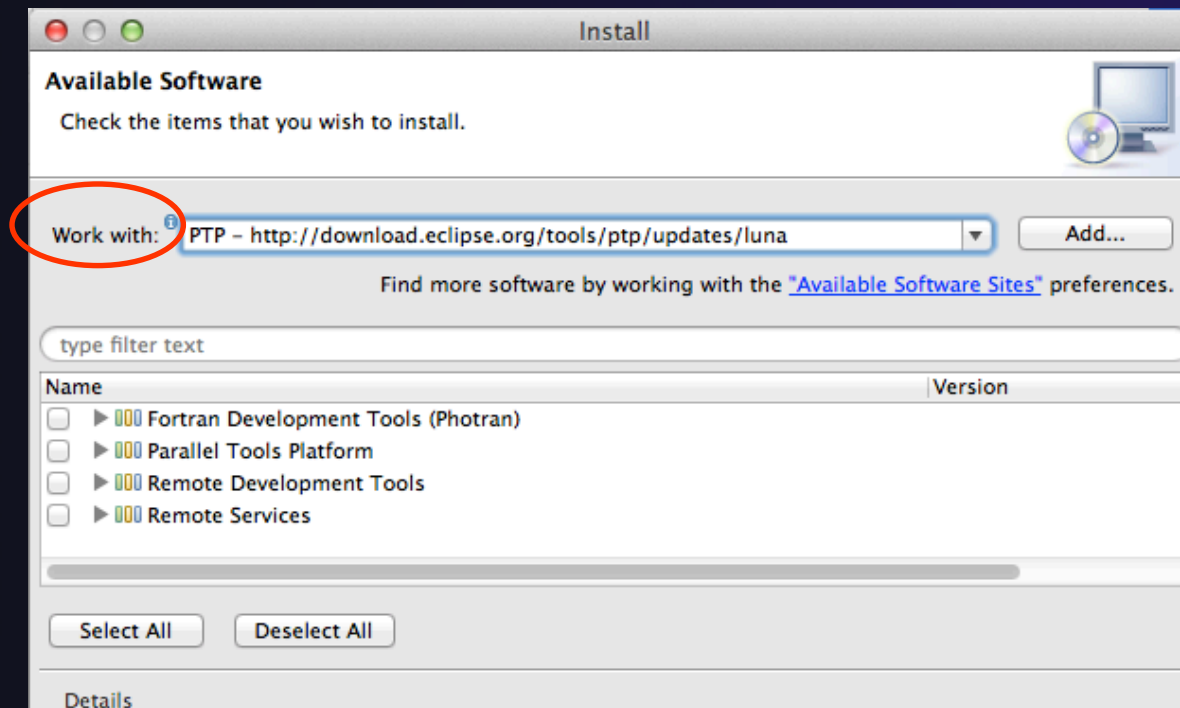


Checking for PTP Updates

- ✦ From time-to-time there may be newer PTP releases than the Mars release
 - ✦ Mars 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**

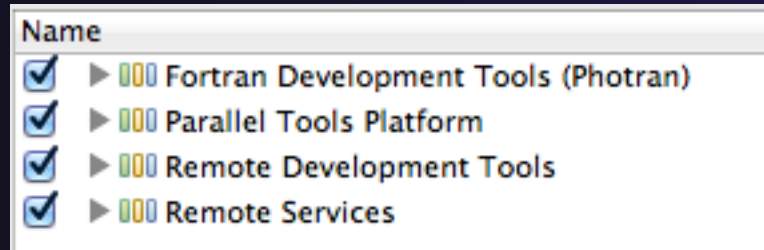
Updating PTP

- ★ 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/mars>



Updating PTP (2)

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

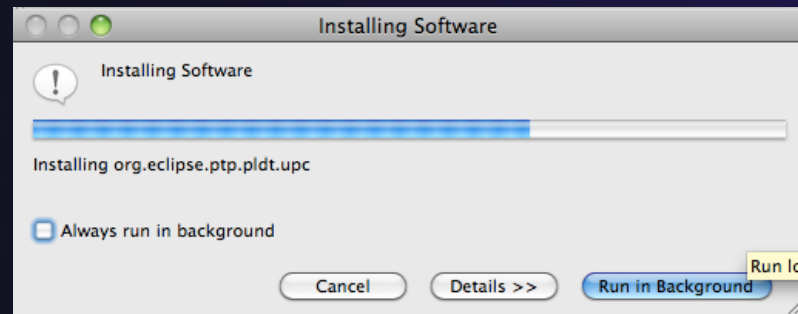


Note: for this tutorial, this installs extra features we'll refer to later anyway (TAU, PerfSuite)

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

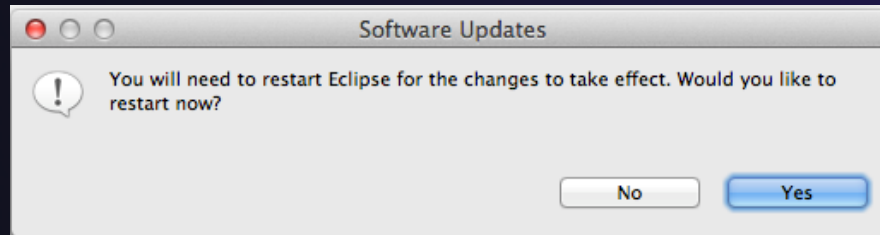
Updating PTP (3)

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



Updating PTP - restart

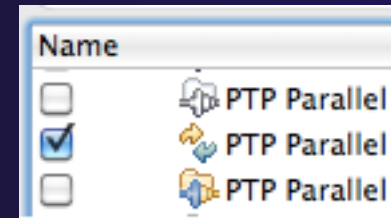
- ★ Select **Yes** when prompted to restart Eclipse



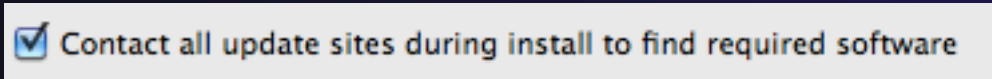
Updating Individual Features

- ★ 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

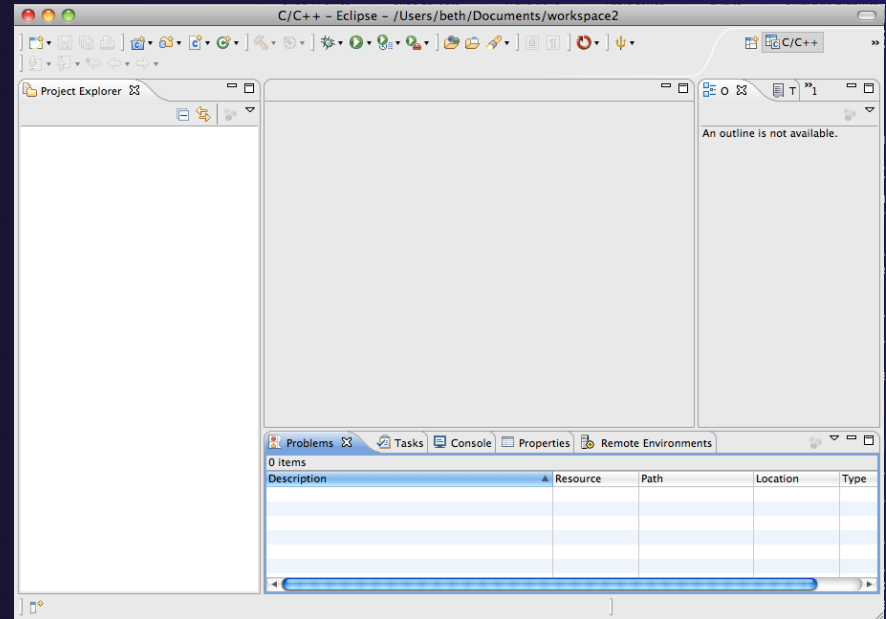


- ★ Note: if network is slow, consider unchecking:



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

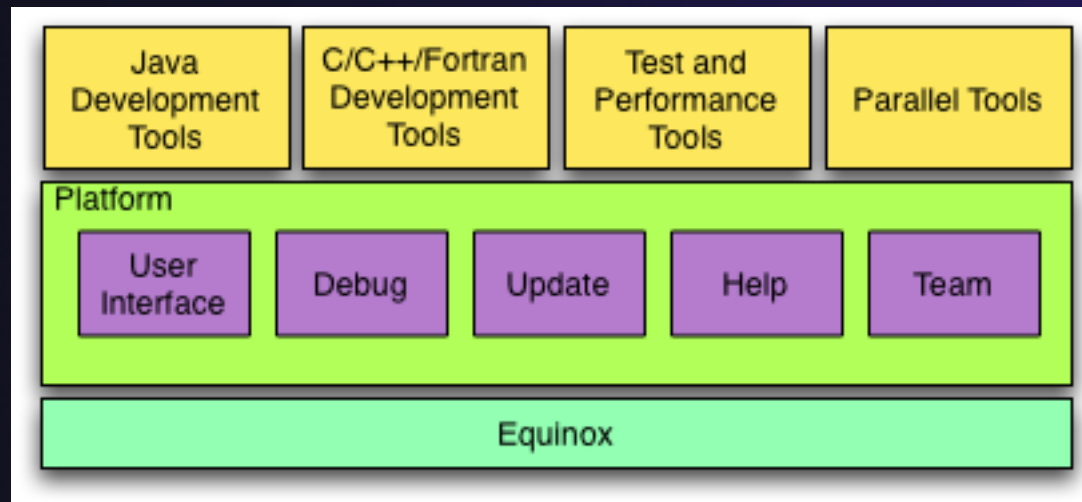
- ✦ To introduce the Eclipse platform and PTP

✦ 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

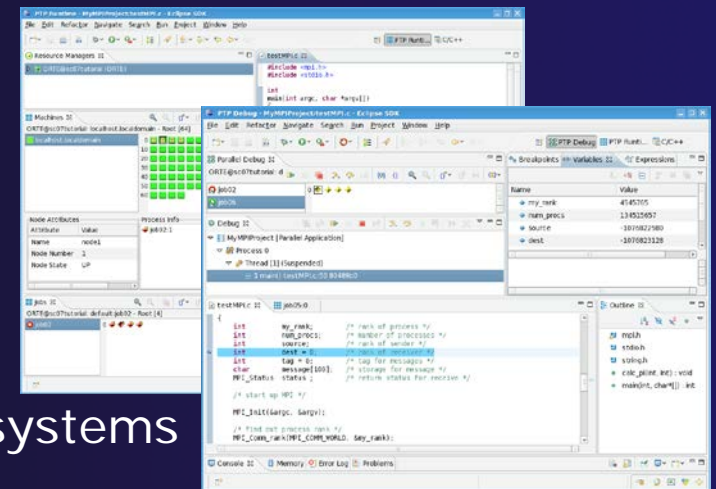


Eclipse Features

- ✦ 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

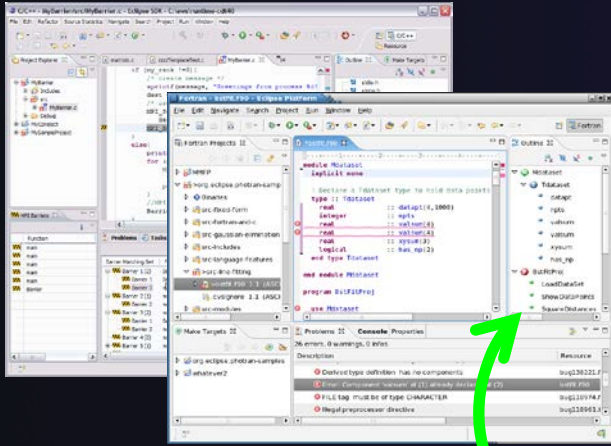
Parallel Tools Platform (PTP)

- ★ 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>

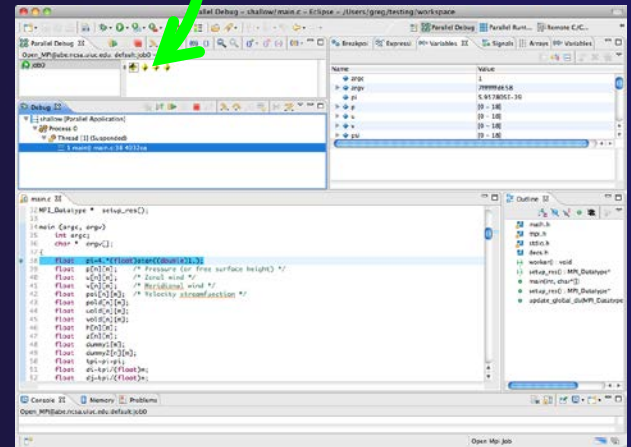
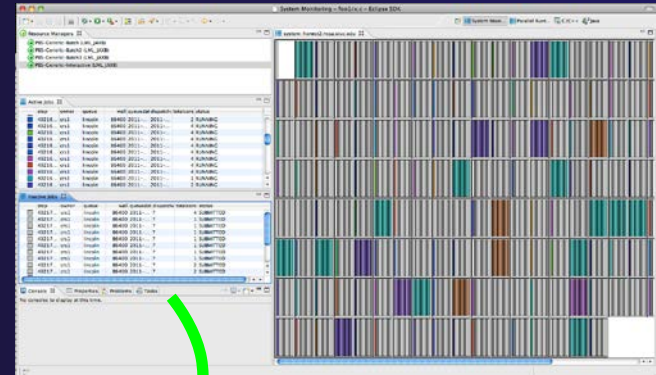


Eclipse PTP Family of Tools

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



Launching & Monitoring



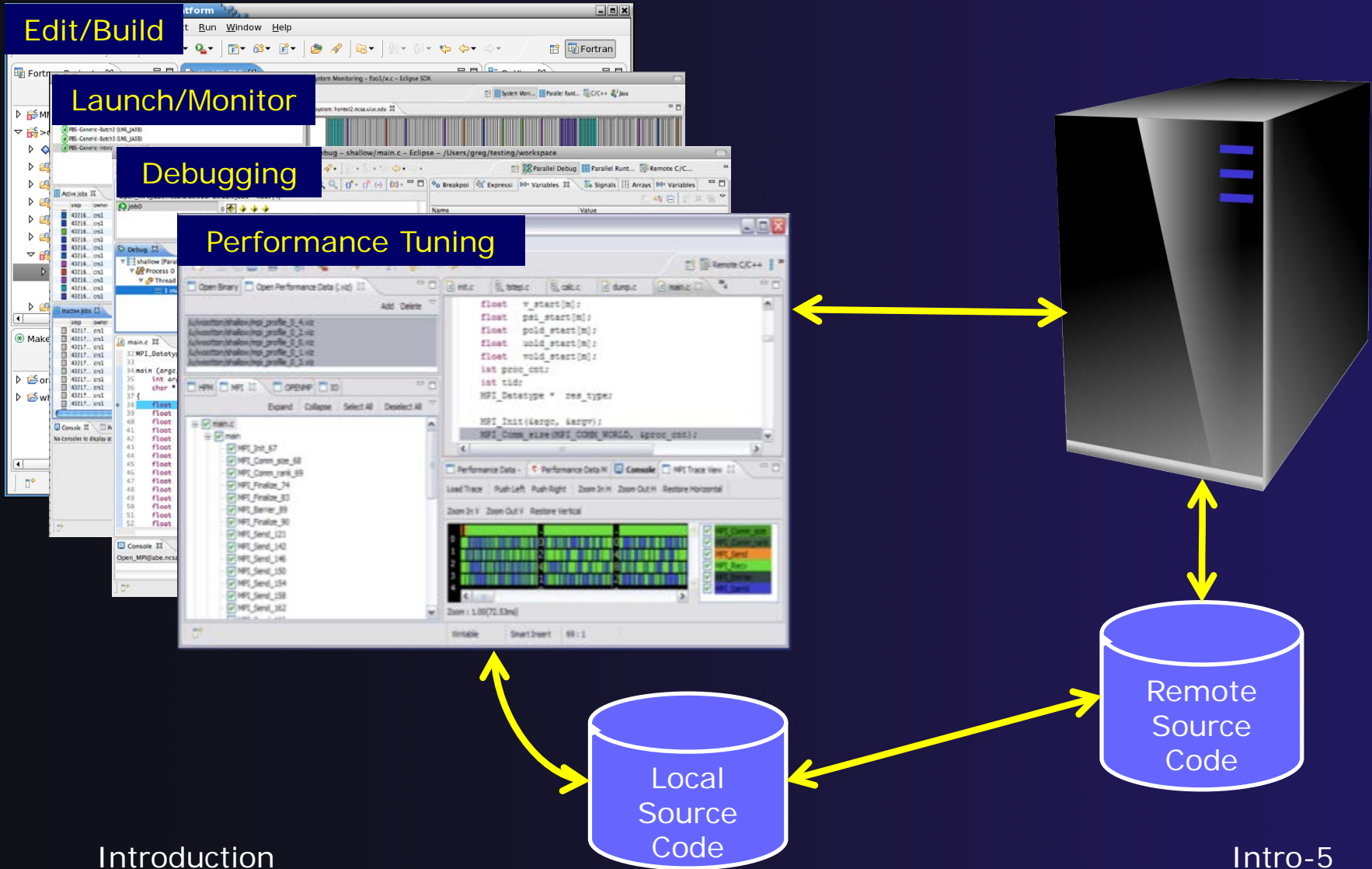
Performance Tuning
(TAU, PerfSuite, ...)

Parallel Debugging

Introduction

Intro-4

How Eclipse is Used



Eclipse Basics

✦ Objective

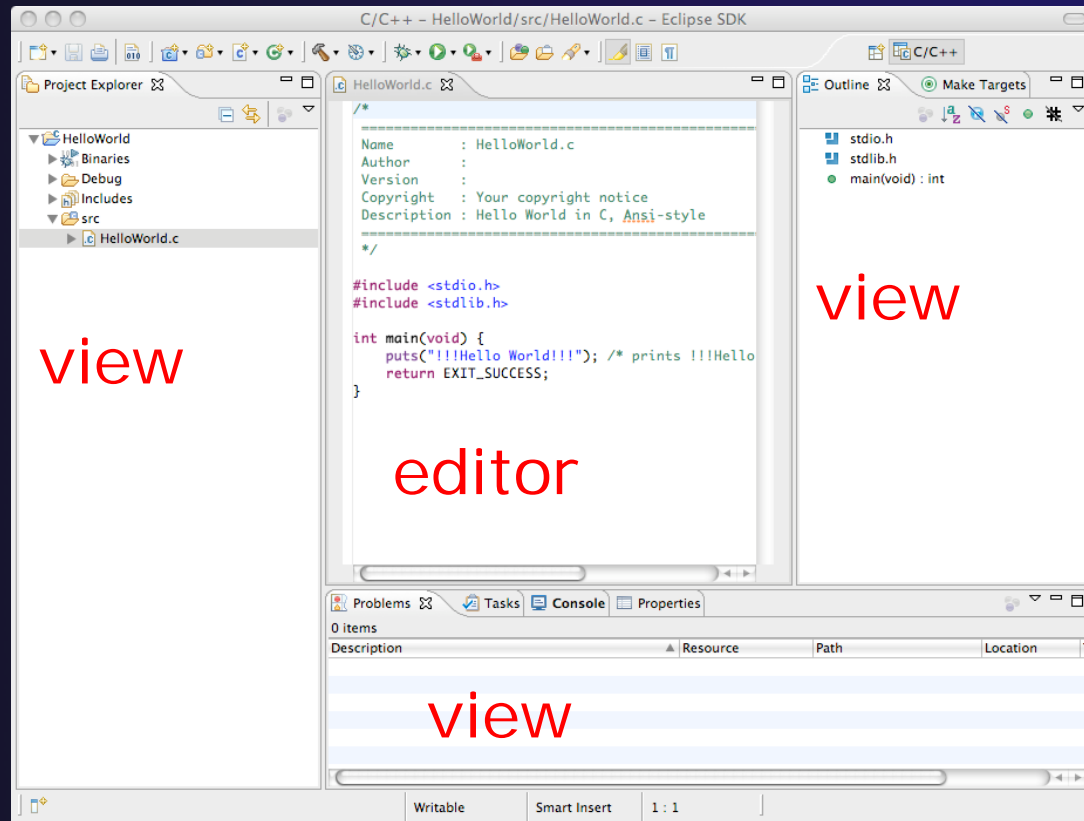
- ✦ Learn about basic Eclipse workbench concepts: projects,
- ✦ Learn about projects: local, synchronized, remote

✦ 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

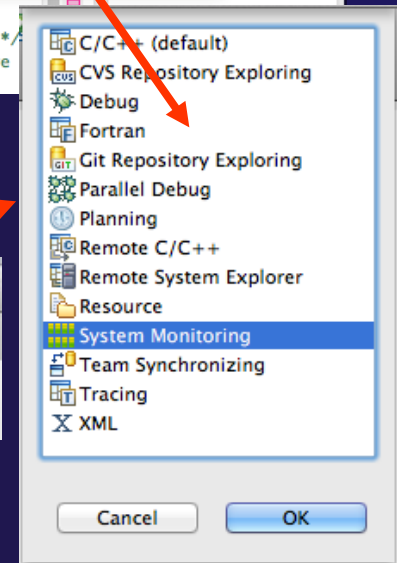
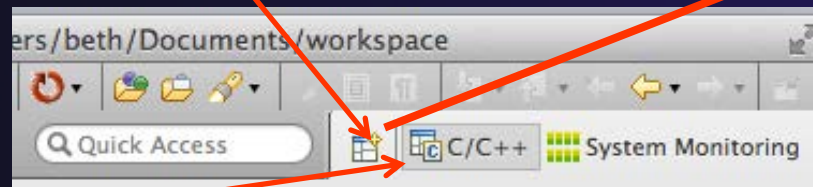
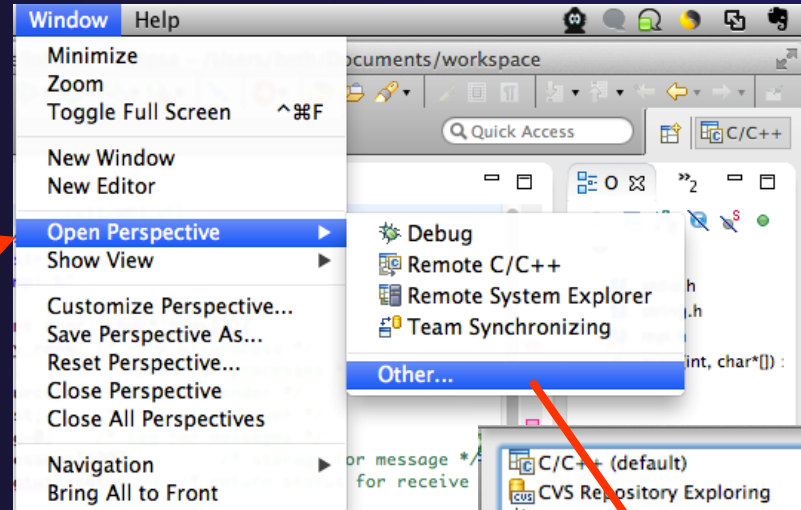
- ✦ Perspectives define the layout of views and editors in the workbench
- ✦ They are *task oriented*, i.e. they contain specific views for doing certain tasks:
 - ✦ **C/C++ Perspective** for manipulating compiled code
 - ✦ **Debug Perspective** for debugging applications
 - ✦ **System Monitoring Perspective**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



Switching Perspectives

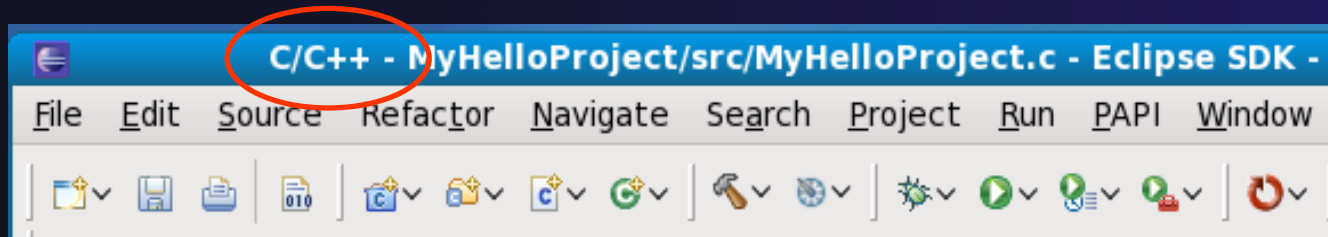
★ Three ways of changing perspectives

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



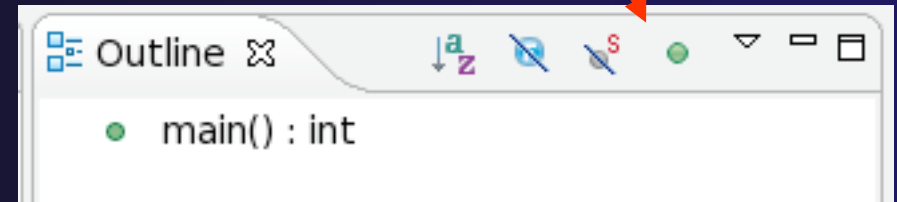
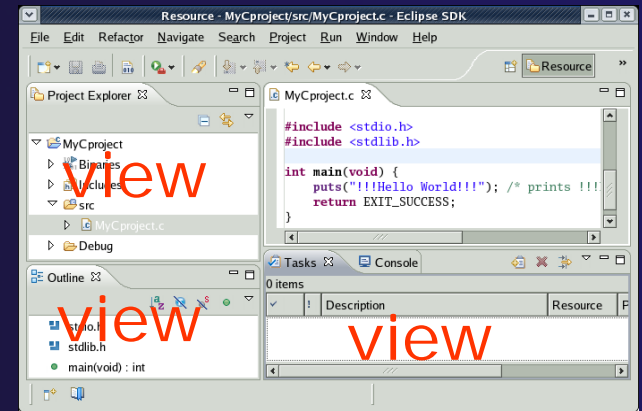
Which Perspective?

- ★ The current perspective is displayed in the title bar



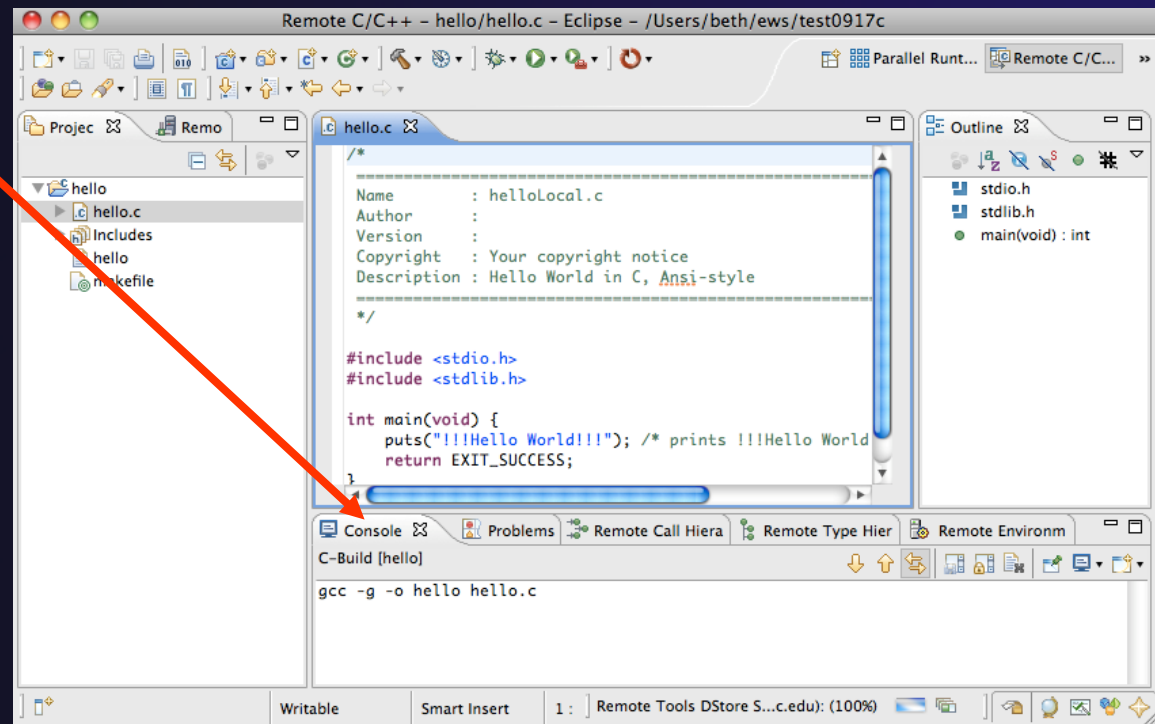
Views

- ★ 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



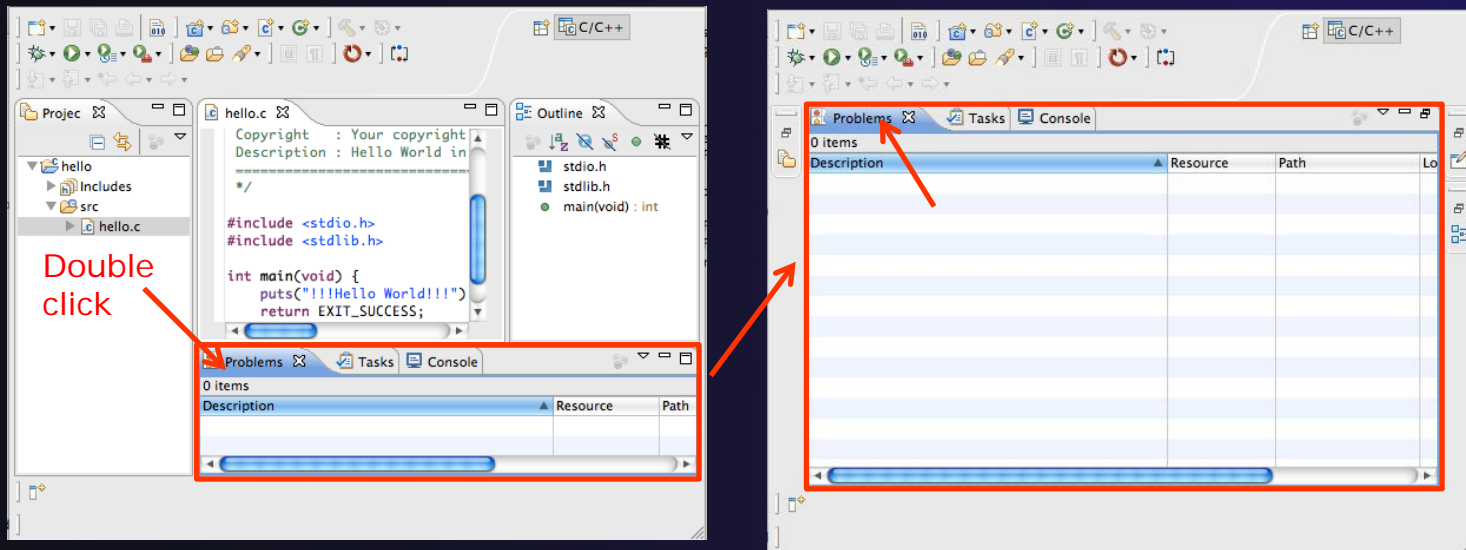
Stacked Views

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



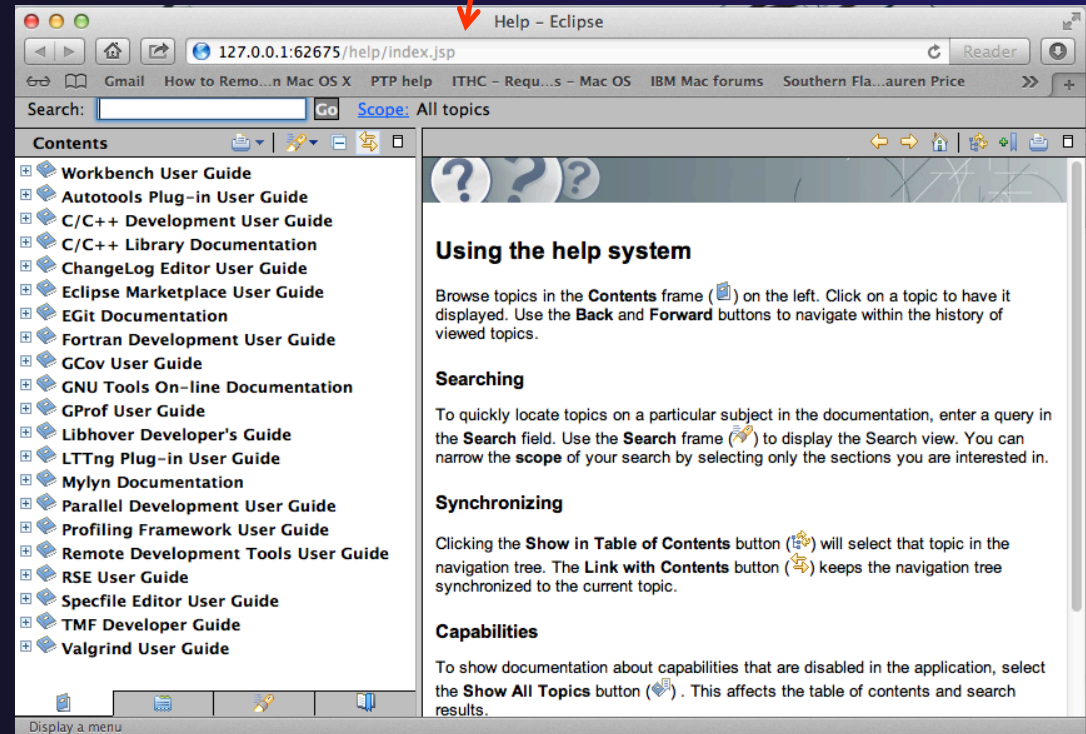
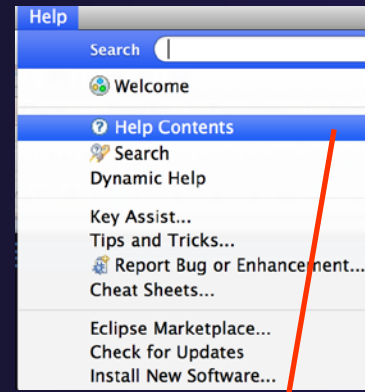
Expand a View

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



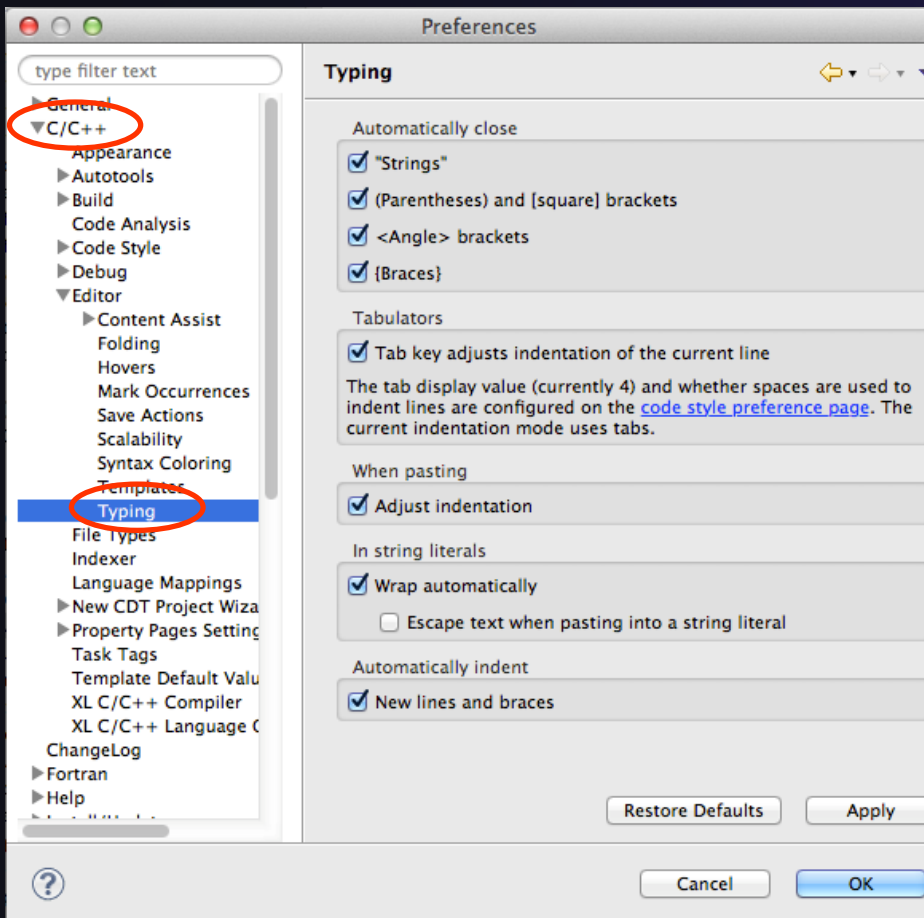
- ★ Window > Reset Perspective returns everything to original positions

Help



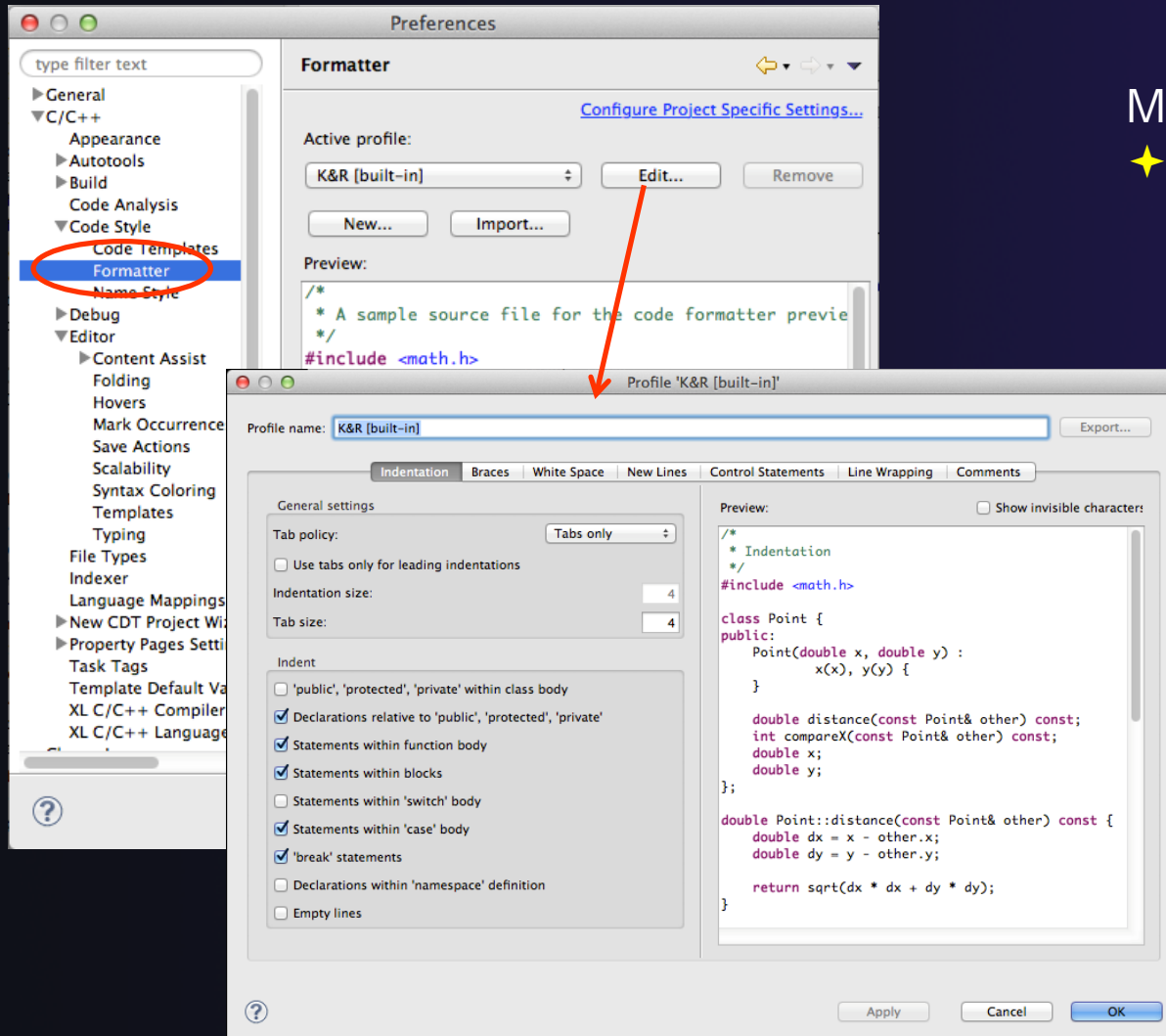
- ★ To access help
 - ★ **Help>Help Contents**
 - ★ **Help>Search**
 - ★ **Help>Dynamic Help**
- ★ **Help Contents** provides detailed help on different Eclipse features *in a browser*
- ★ **Search** allows you to search for help locally, or using Google or the Eclipse web site
- ★ **Dynamic Help** shows 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

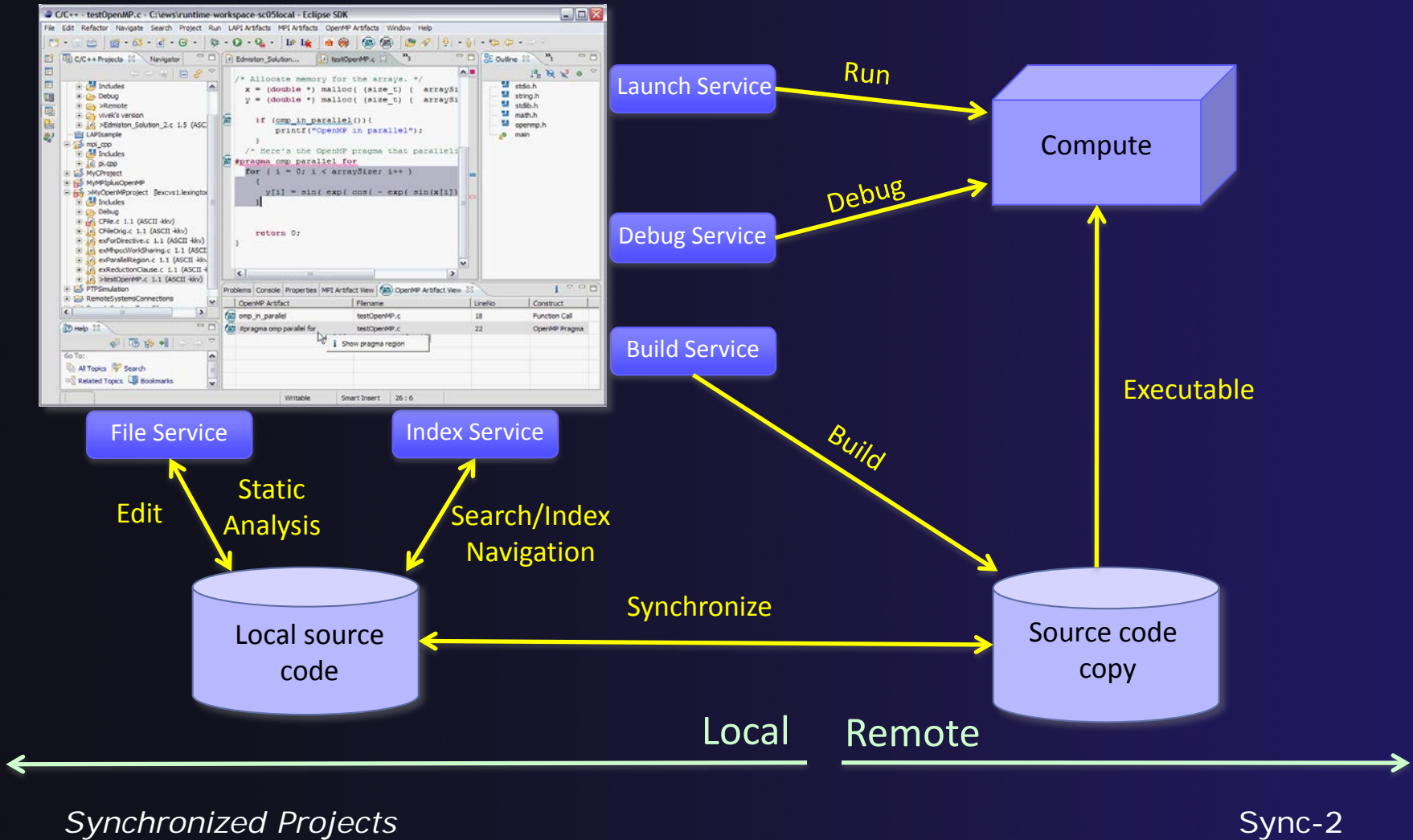
✦ Contents

- ✦ Eclipse project types
- ✦ Clone a git repository; create a synchronized project
- ✦ Using synchronize filters
- ✦ Remote Terminal view

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



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

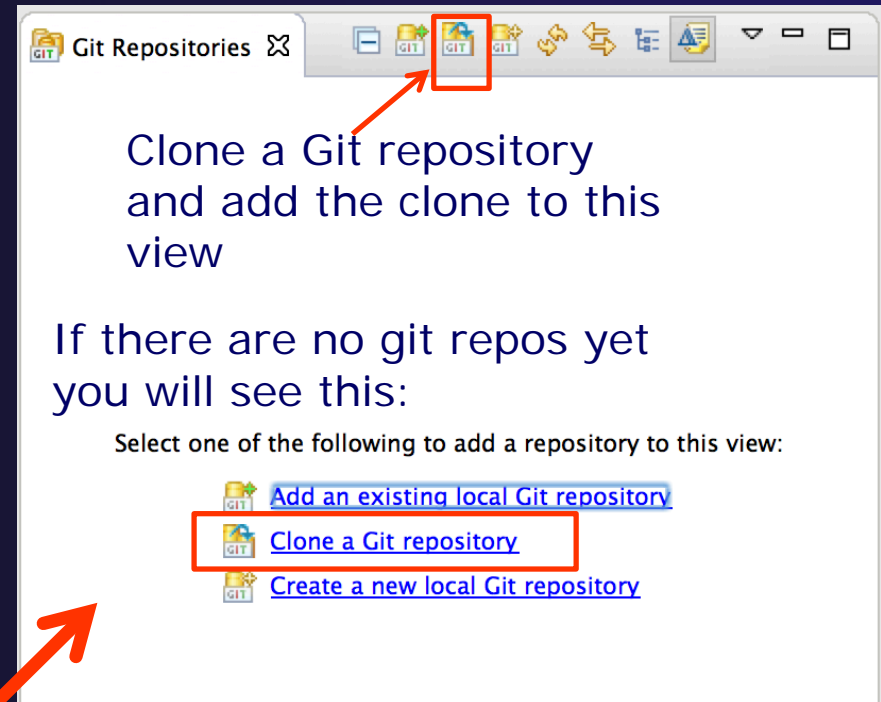
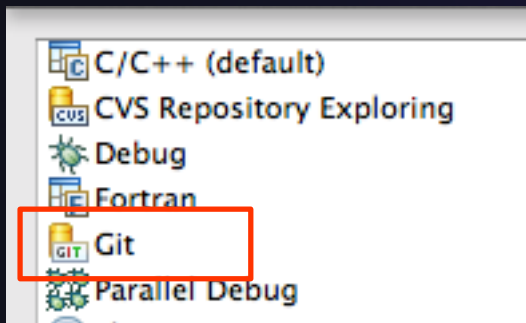
Create Synchronized project on the local machine at the same time.

Two steps:

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

Clone the git repo

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



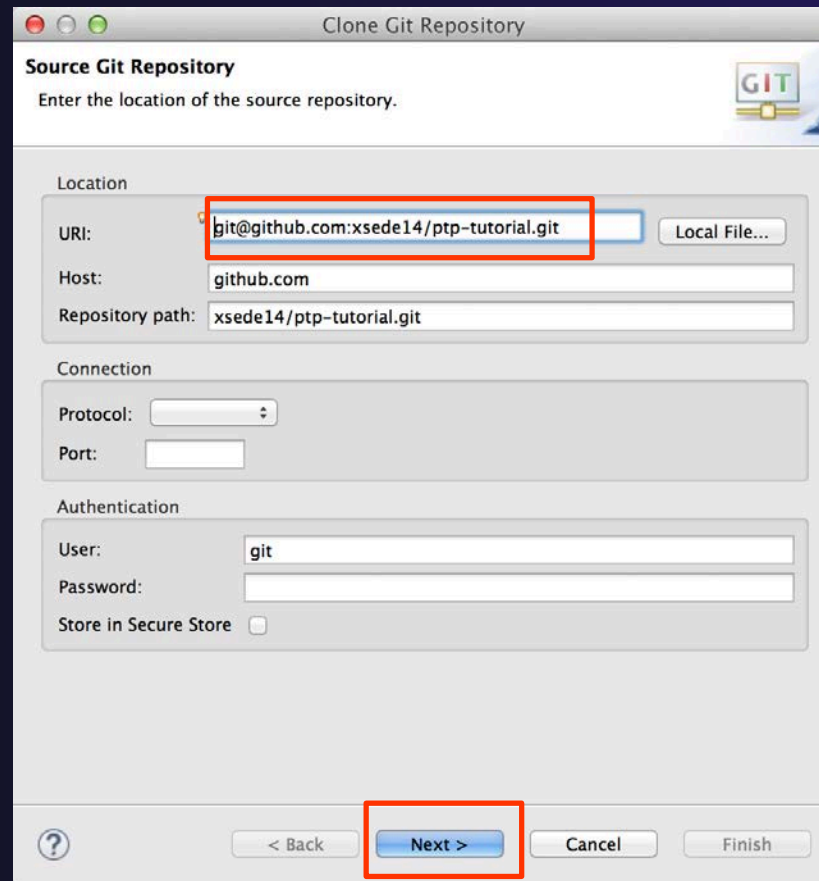
- ✦ In the view, select **Clone a Git repository** one of two ways

Specify remote git repo location

★ URI: <https://github.com/xsede14/ptp-tutorial.git>

★ Fill in URI and other fields fill themselves

★ Select **Next >**



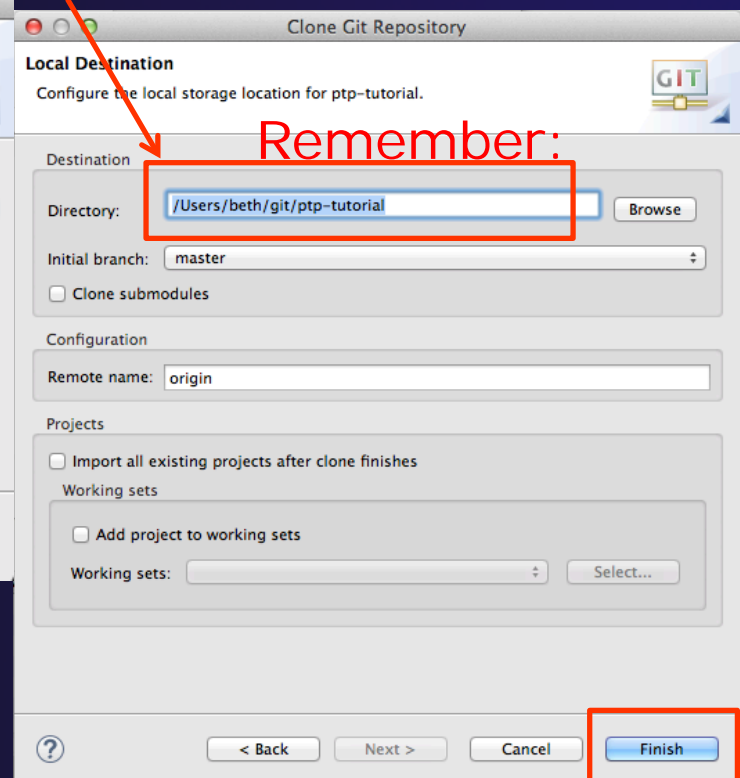
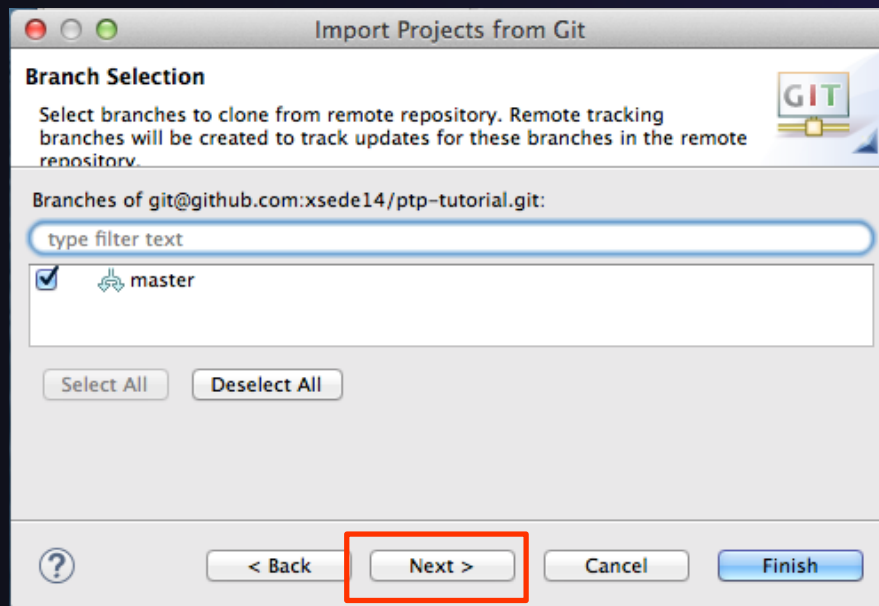
The screenshot shows a 'Clone Git Repository' dialog box with the following fields and values:

- 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: (empty dropdown)
 - Port: (empty text field)
- Authentication:**
 - User: `git`
 - Password: (empty text field)
 - Store in Secure Store:

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

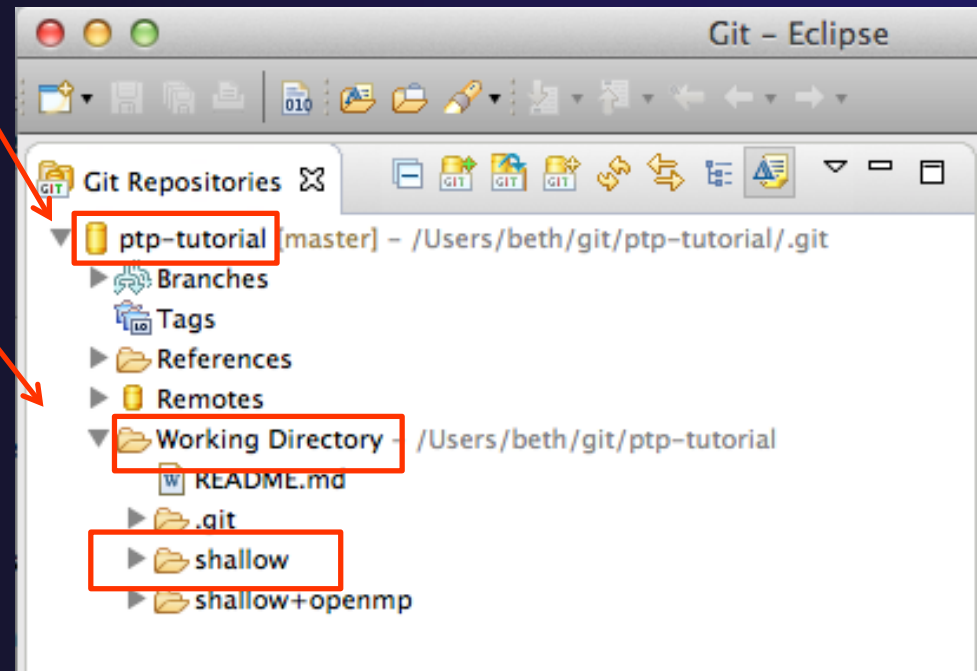
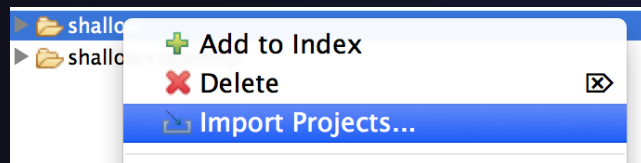
Finish git cloning

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



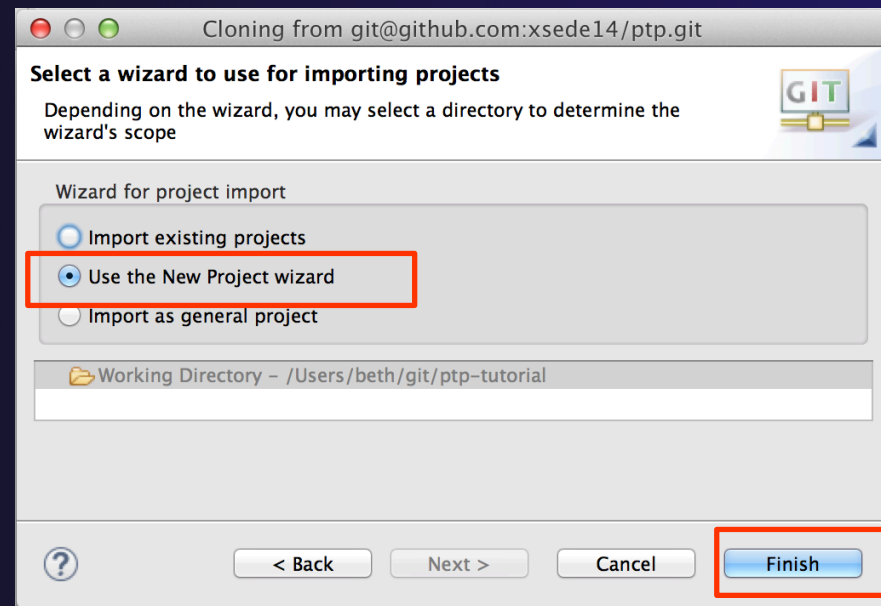
Import project from cloned repo

- ★ 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

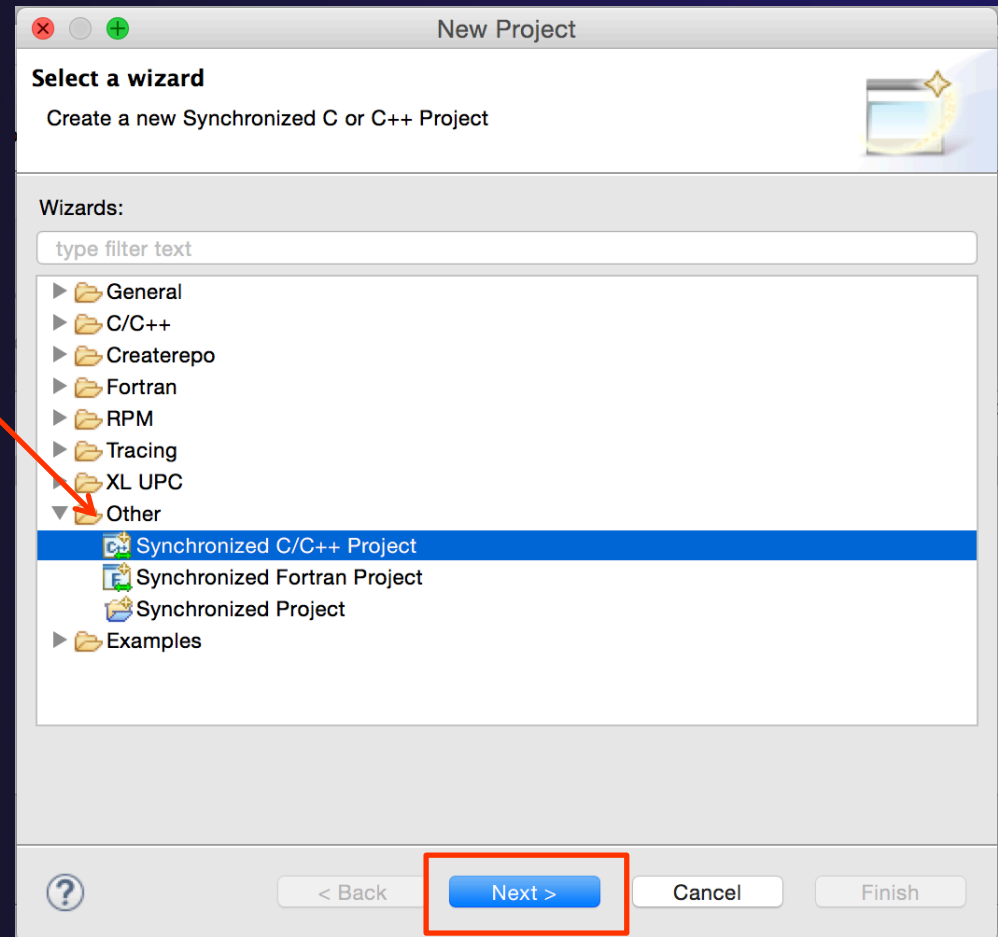
- ★ Select **Use the New Project Wizard** to be able to create the project as a Synchronized C/C++ project at creation
- ★ Select **Finish** to finish the git cloning, and you will be taken to Sync project info next.



New Project Wizard

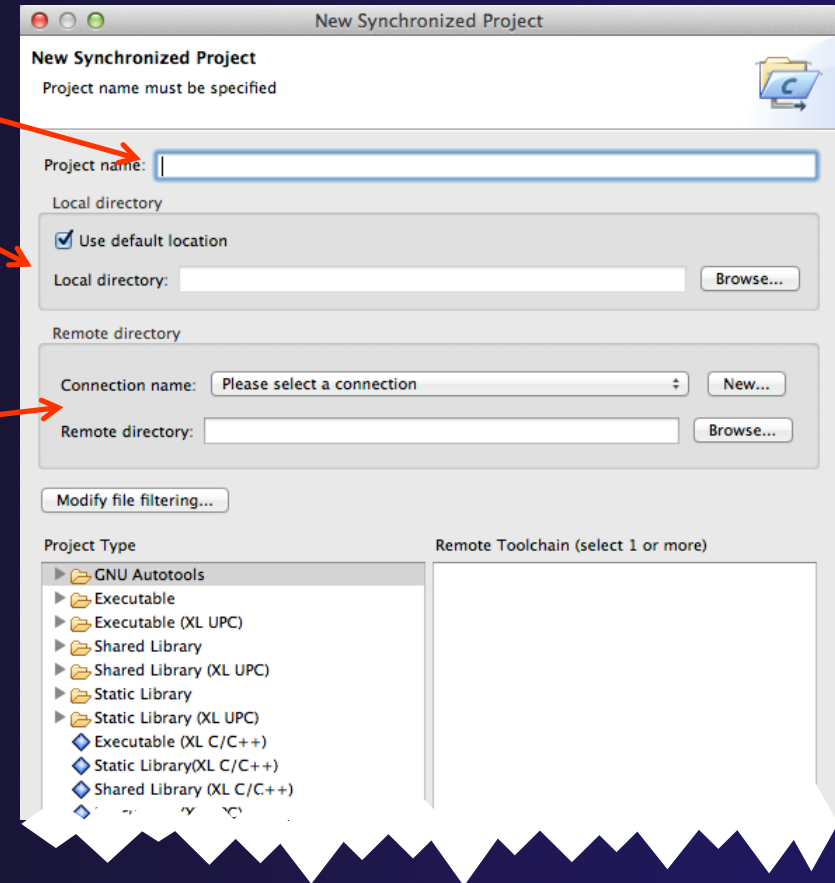
We are creating the project directly as a Synchronized C/C++ project

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



New Synchronized Project Wizard

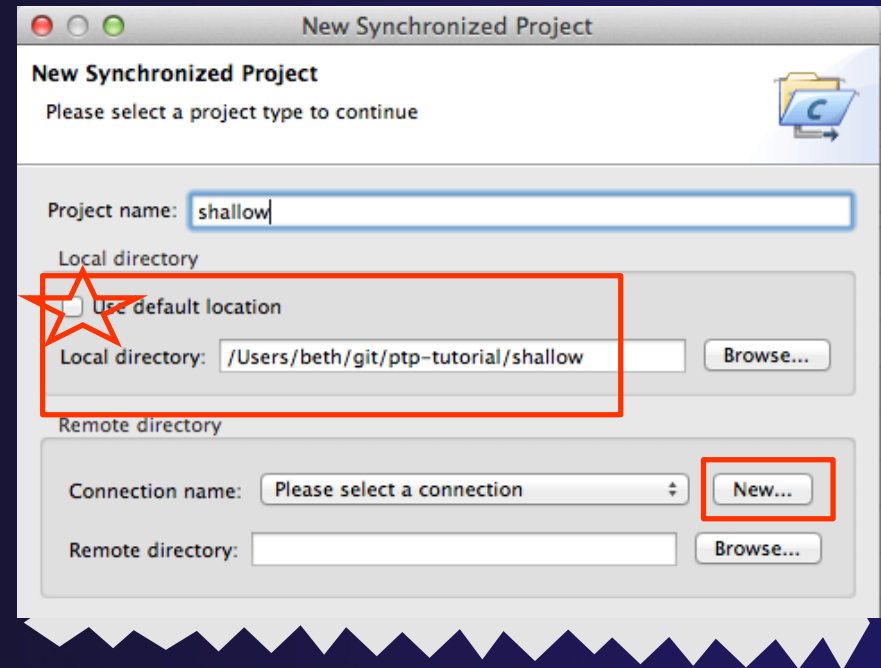
- ✦ Enter the **Project Name**
 - ✦ E.g. “shallow”
- ✦ Next we will specify the **Local Directory** where the local files are located (cloned from git)
 - ✦ Files are synchronized here, and we will edit them locally
- ✦ ...and the **Remote Directory** where the remote files are located
 - ✦ Our remote 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**
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...**



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
Specify properties of a new connection

Connection name:

Host information

Host:

User:

Public key based auth Keys are set at [Network Connections, SSH2](#)

Passphrase:

Password based authentication

Password:

▶ Advanced

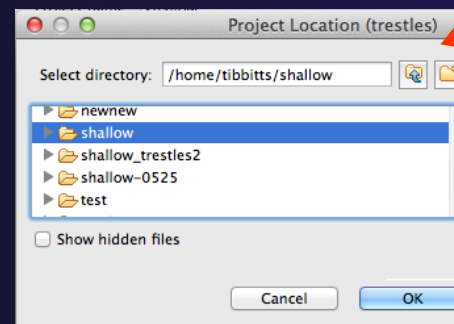
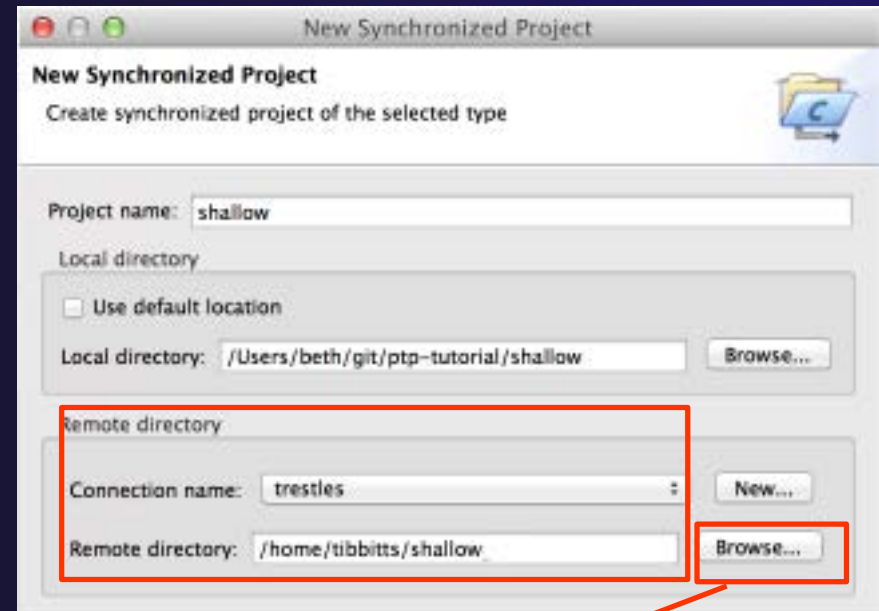
Specifying the remote directory

✦ 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.

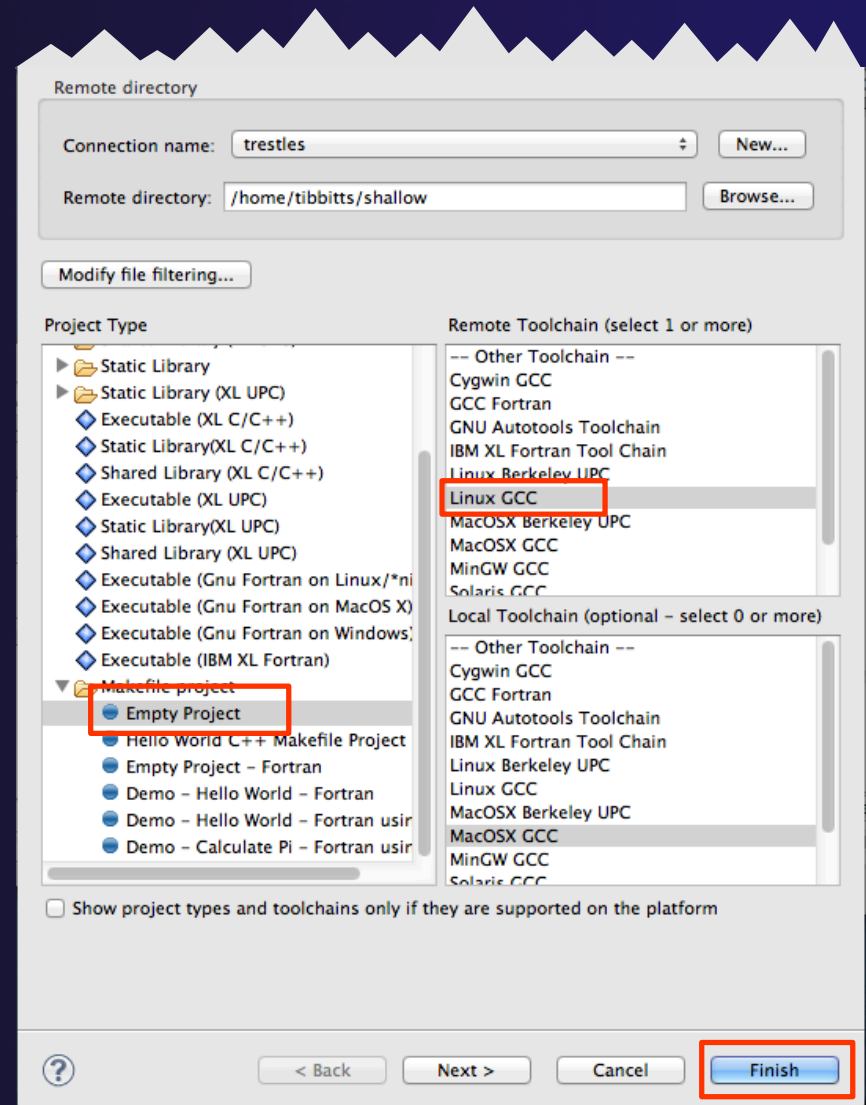
✦ If the remote dir exists, you can select it with the **Browse...** Note that this is the first time that the Connection information is utilized.

✦ *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



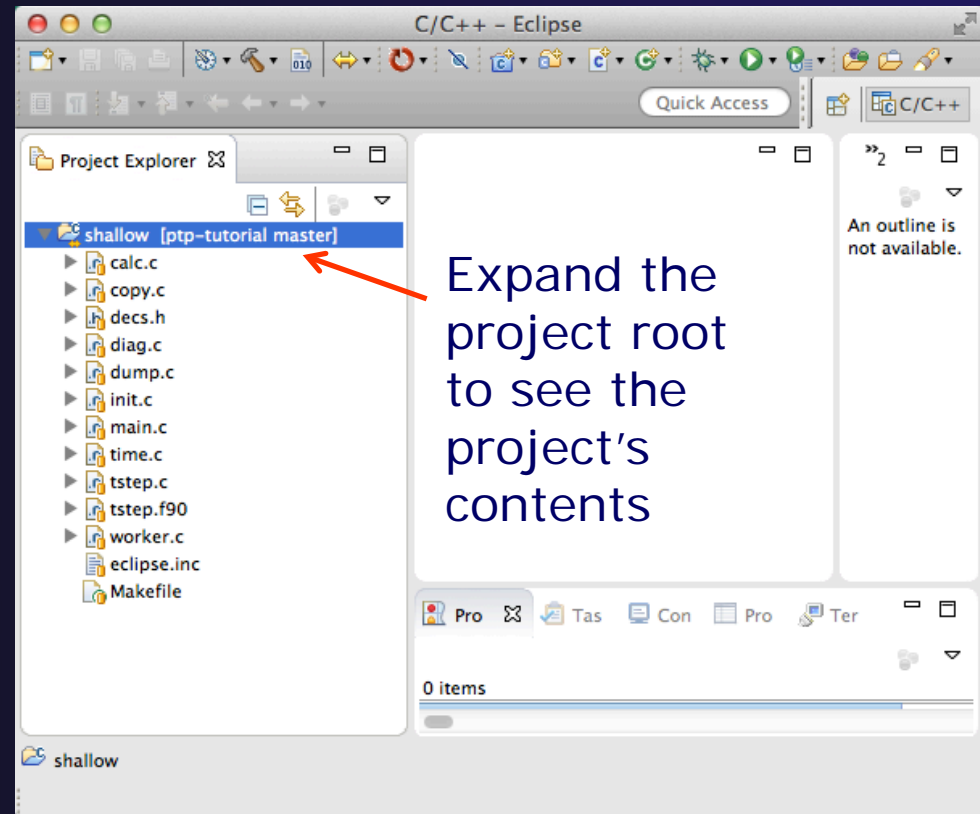


Project successfully created

- ✦ 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%)





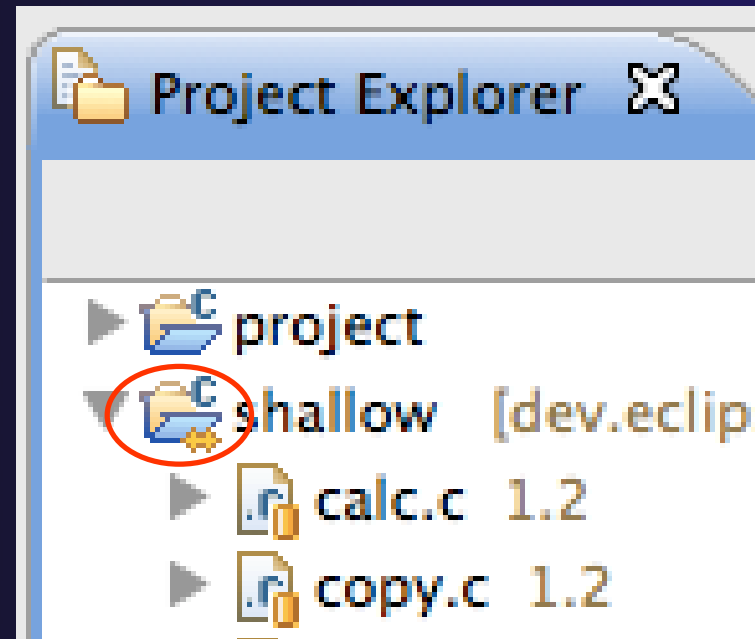
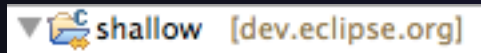
Synchronized Project

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

- ★ C Project w/o Sync



- ★ Synchronized Project

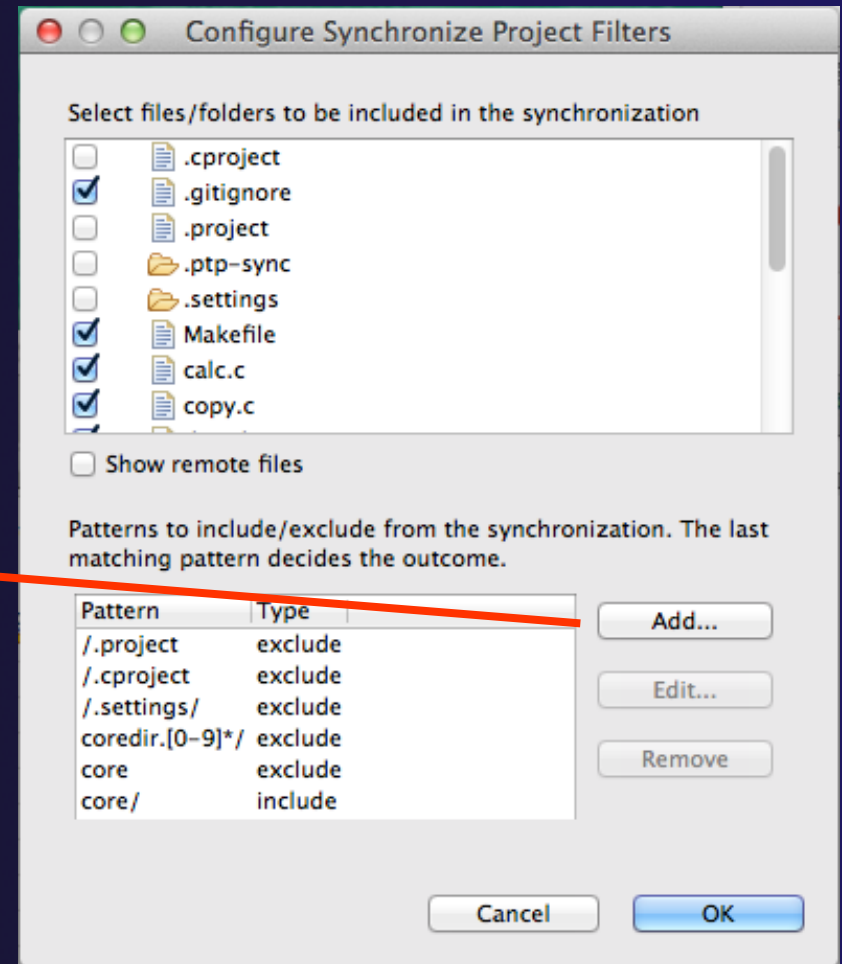
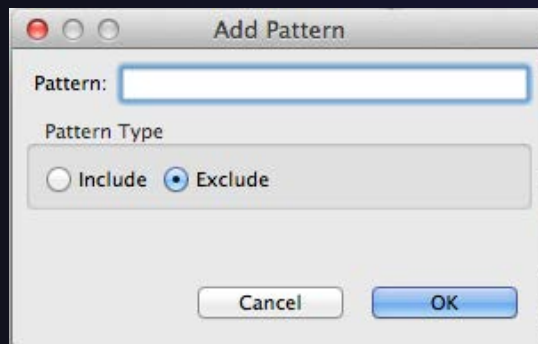


Synchronize Filters

- ✦ 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

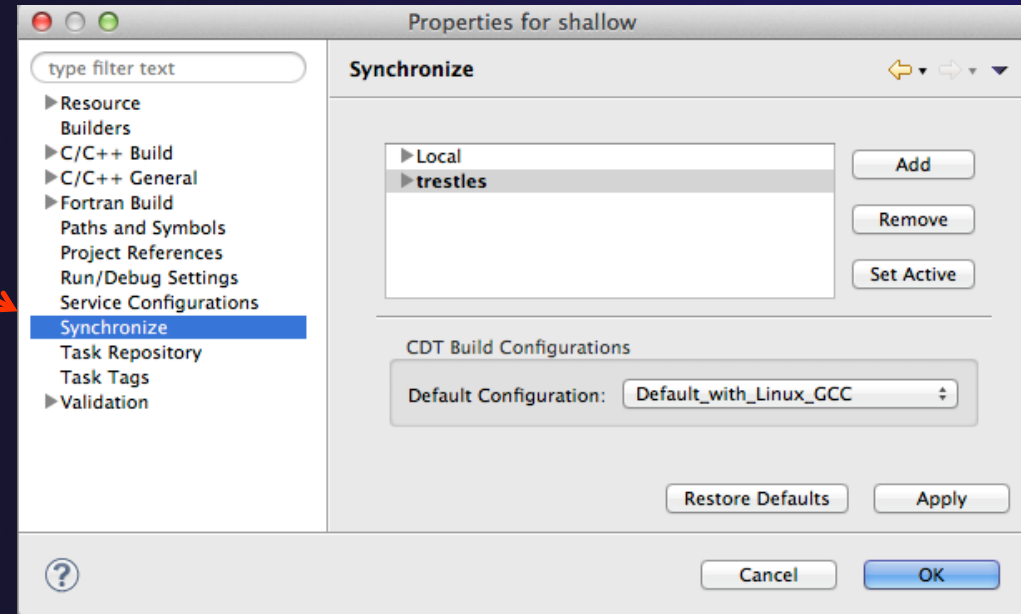
- Files can be filtered individually by selecting/unselecting them in the **File View** at the top
- Include or exclude files based on paths and expressions



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

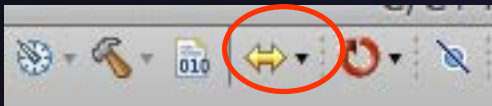
Synchronized Project Properties

- ✦ 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

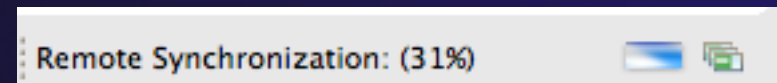
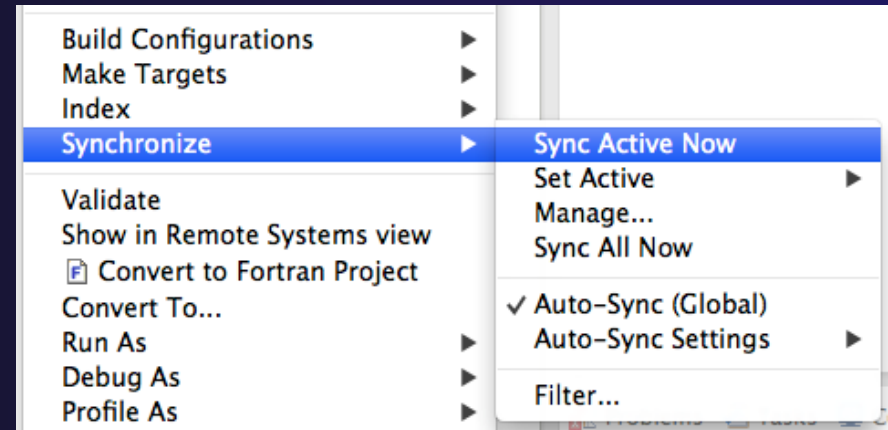


Forcing a Resync

- ✦ 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

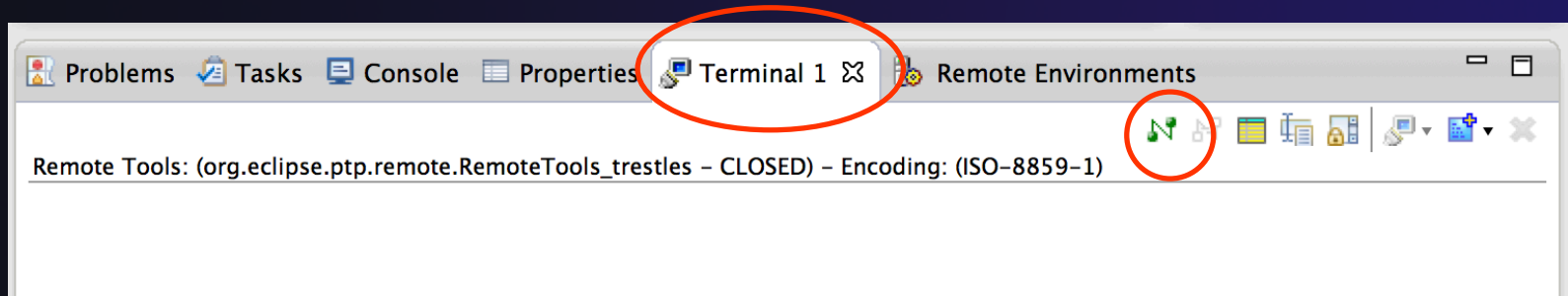
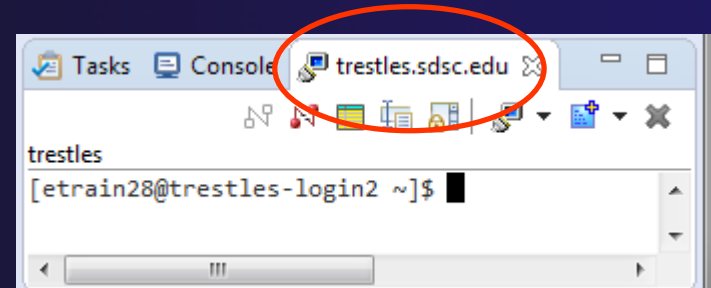


- ✦ Status area in lower right shows when Synchronization occurs



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...*



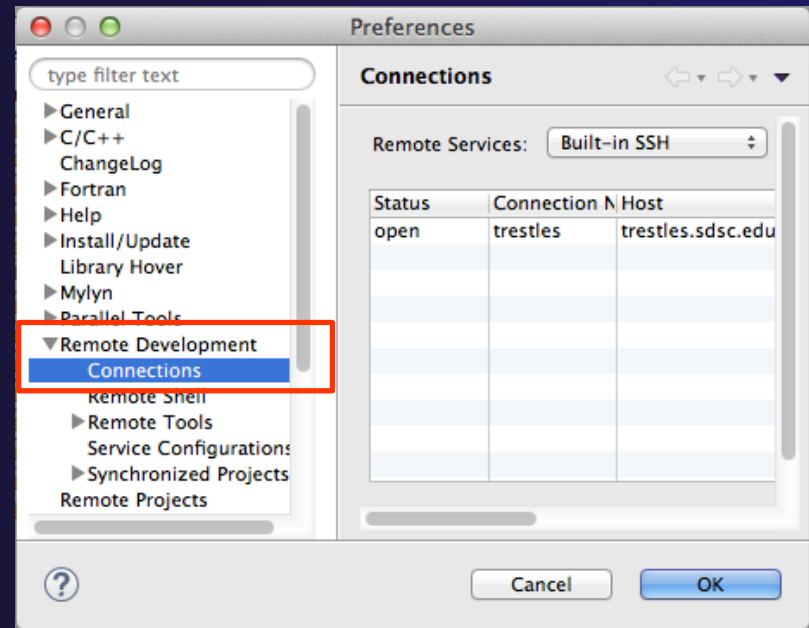
Changing Remote Connection Information

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

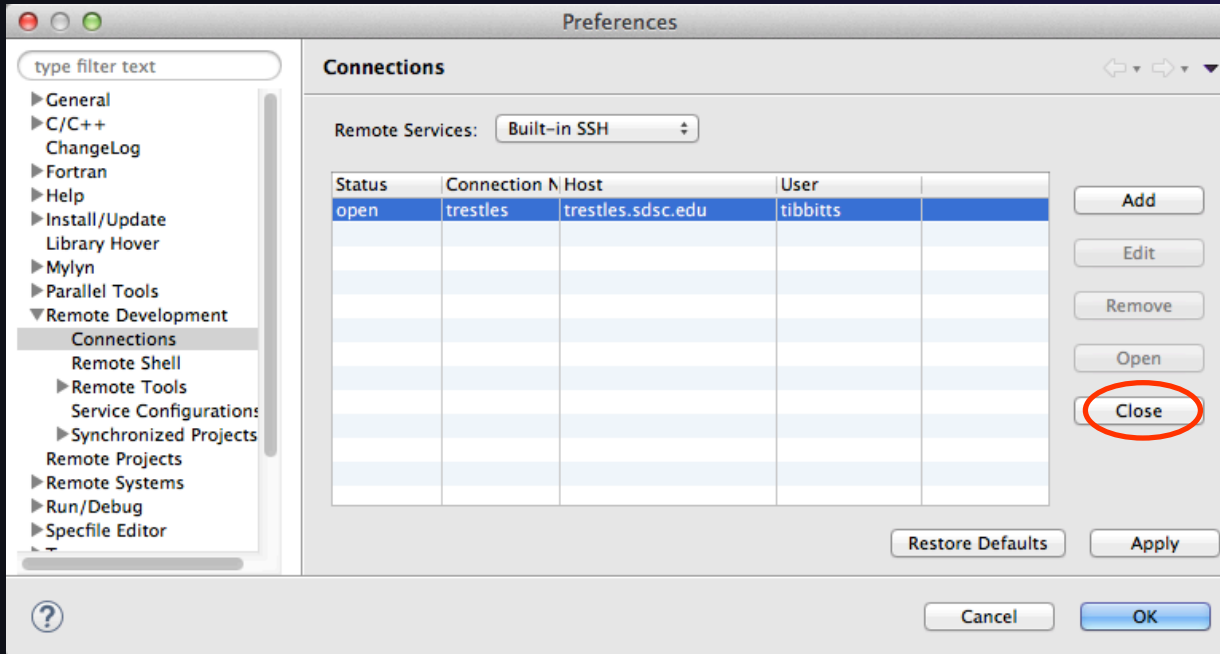
★ **Win/Linux: Window > Preferences**

★ **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

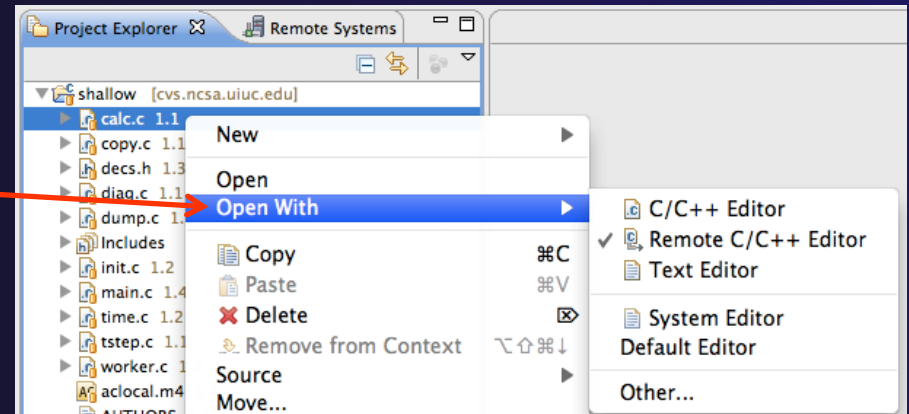
1. Modify Sync filters to not bring the *.o files and your executable back from the remote host
 - ★ Rebuild and confirm the files don't get copied

Editor Features

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

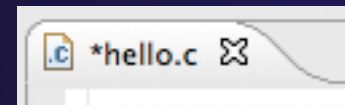
Editors

- ★ 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

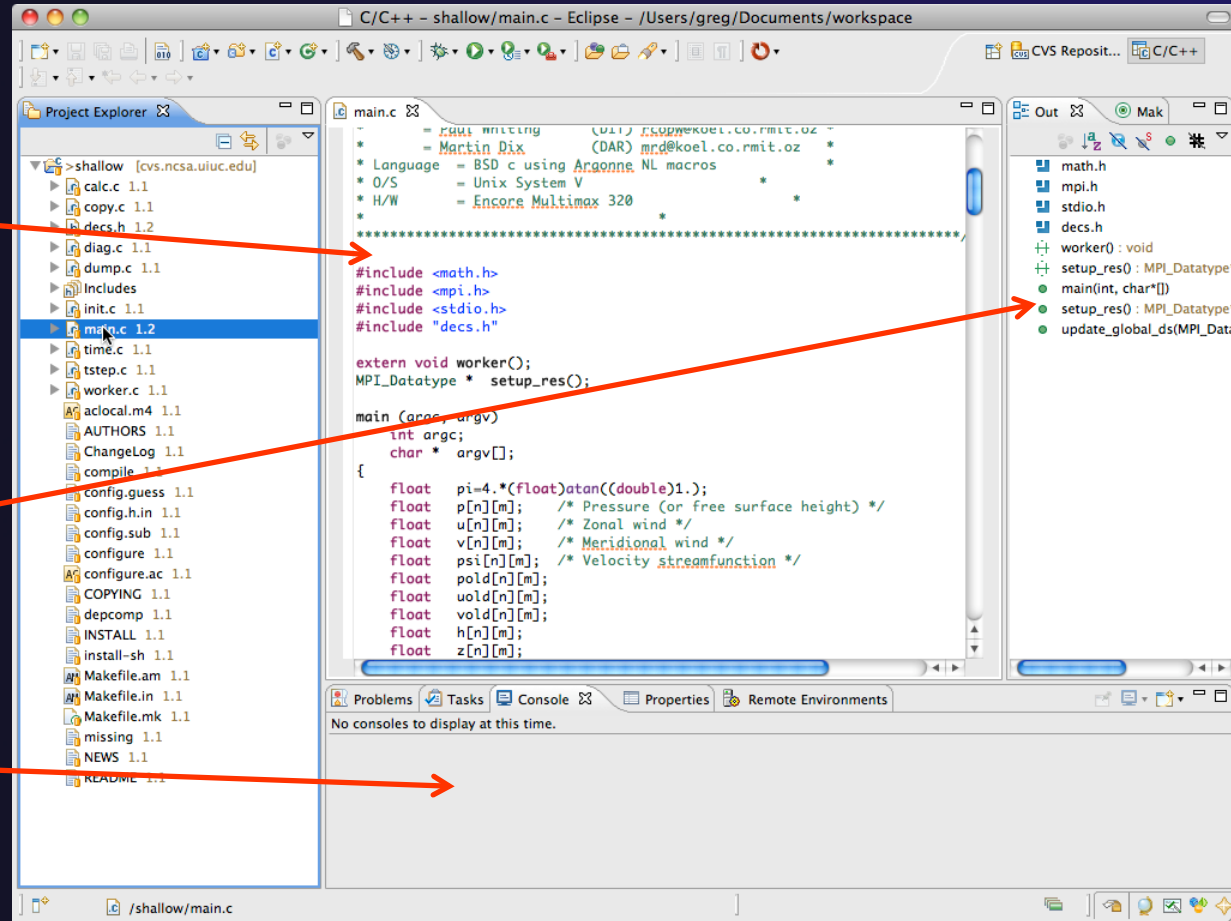
- ★ When you change a file in the editor, an asterisk on the editor's title bar indicates unsaved changes



- ★ 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.



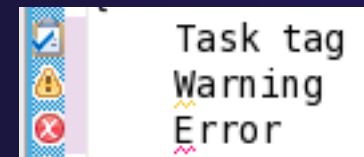
Source Code Editors & Markers

- ★ 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:



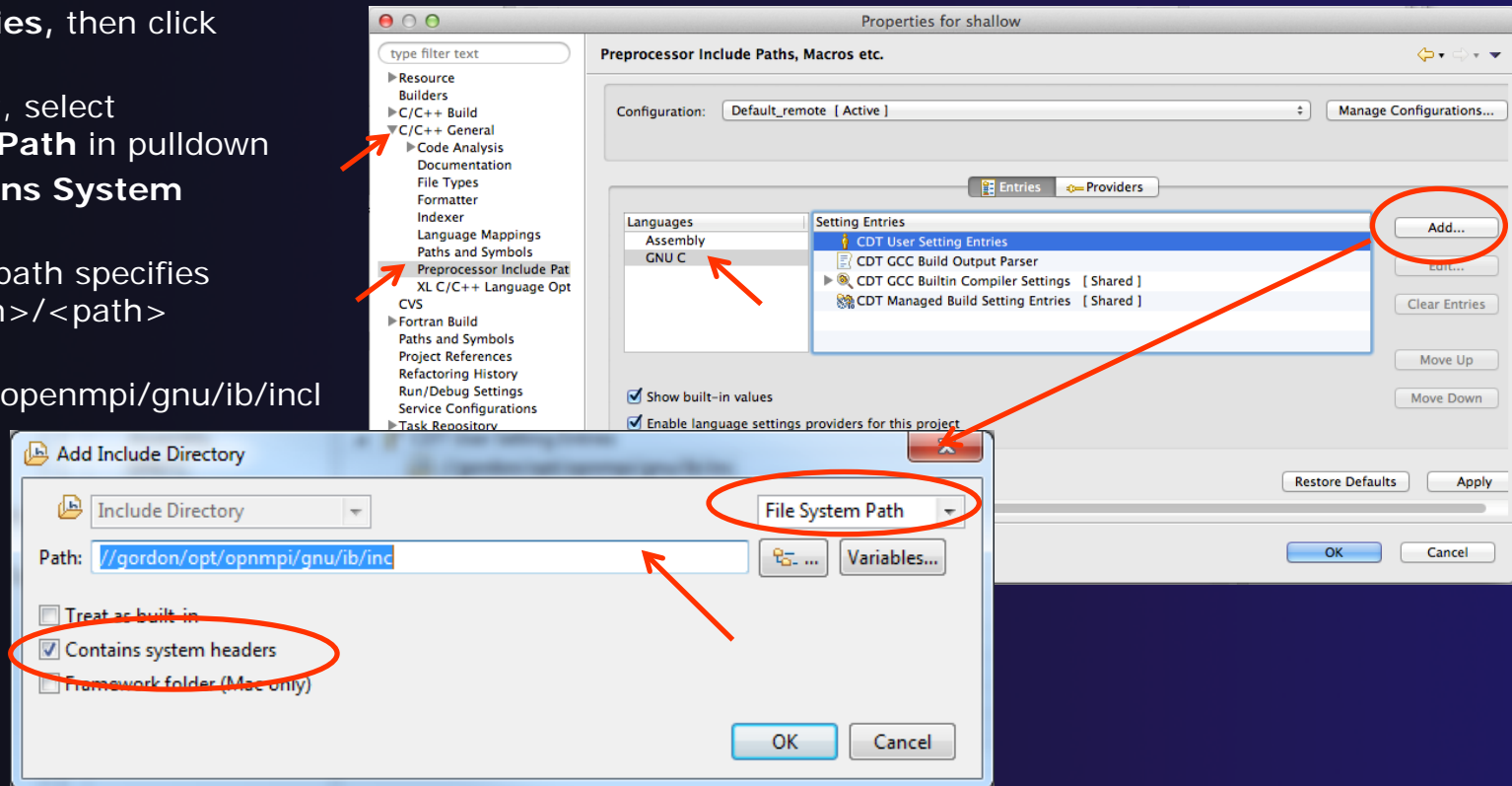
Remote Include Paths

- ✦ 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 `//gordon/opt/openmpi/gnu/ib/include`
- ✦ Select **OK**



A

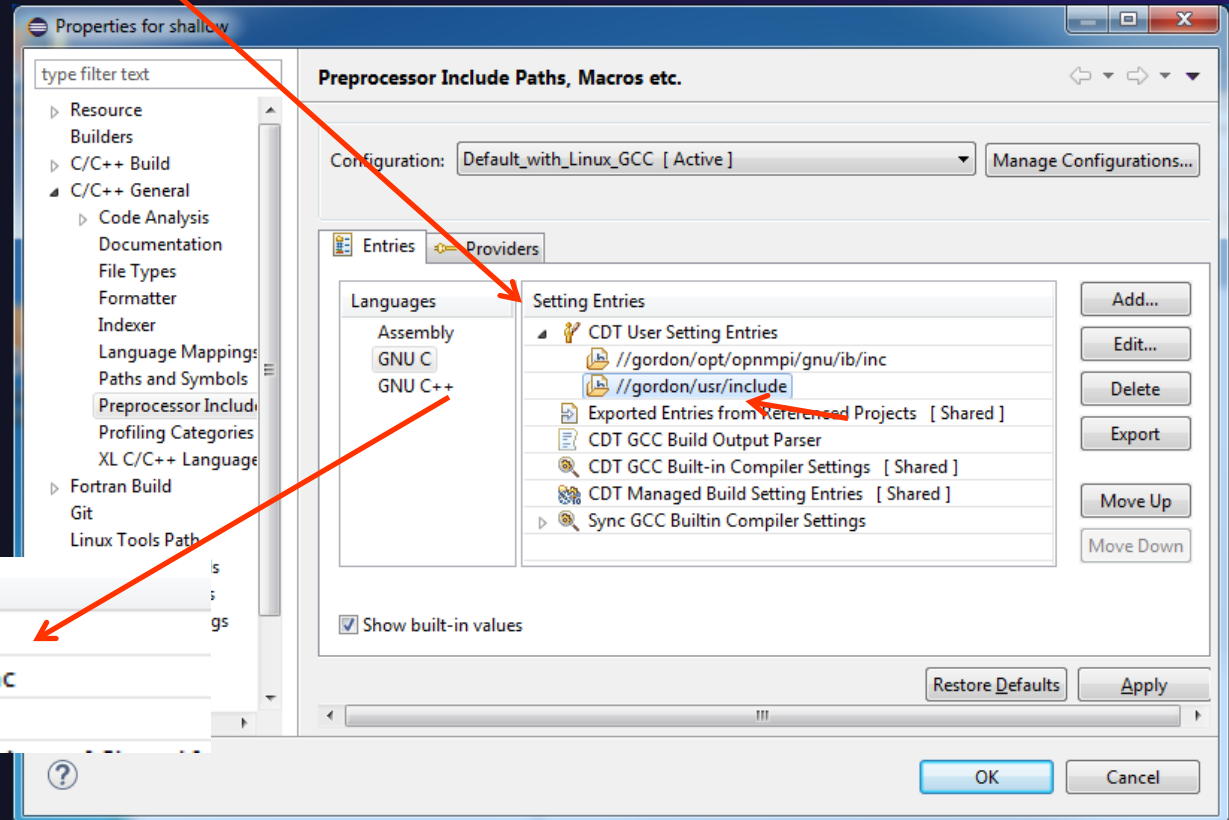
Include Paths con't

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

- ✦ Add second value:

//gordon/usr/include
... the same way

You should have
two entries:



Setting Entries

- CDT User Setting Entries
 - //gordon/opt/opnmpi/gnu/ib/inc
 - //gordon/usr/include

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

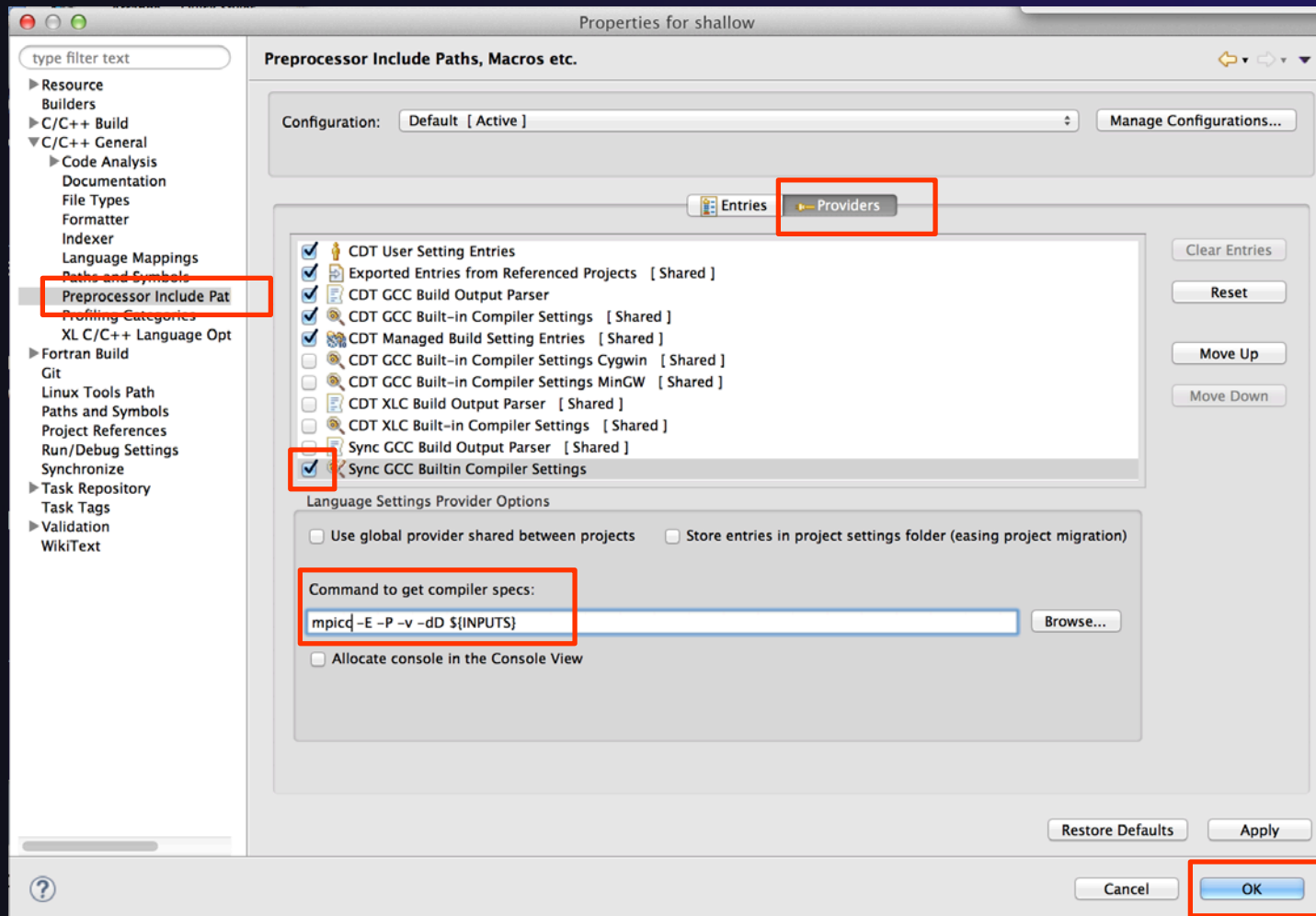
B

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
1. `mpi.h` and its symbols should now be resolved.

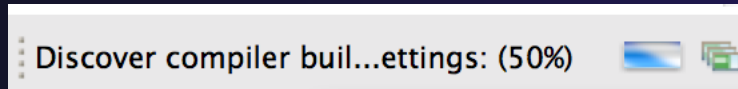
B

Set include paths automatically (con't)

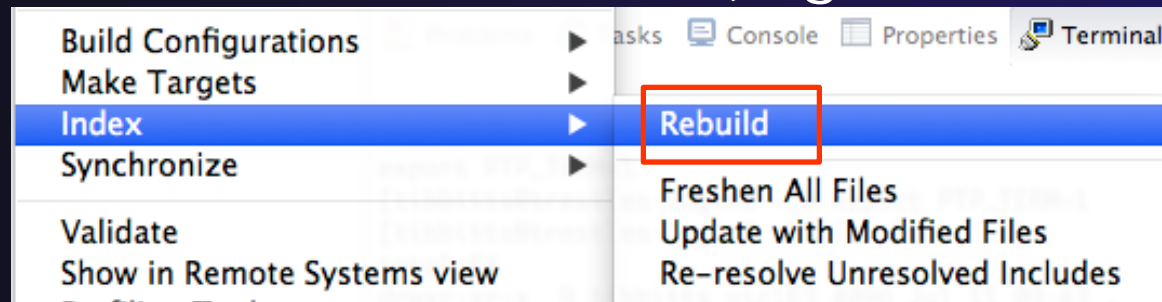


B Set include paths automatically (con't)

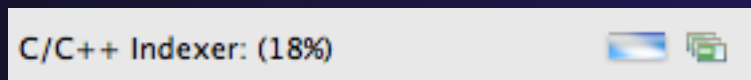
- ★ You may see in lower right:



- ★ 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

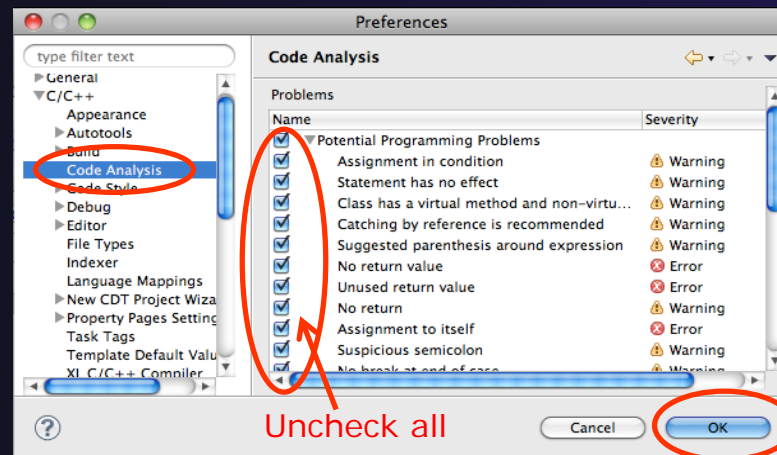
✦ To turn them off, use Preferences

Window > Preferences or Mac: Eclipse > Preferences

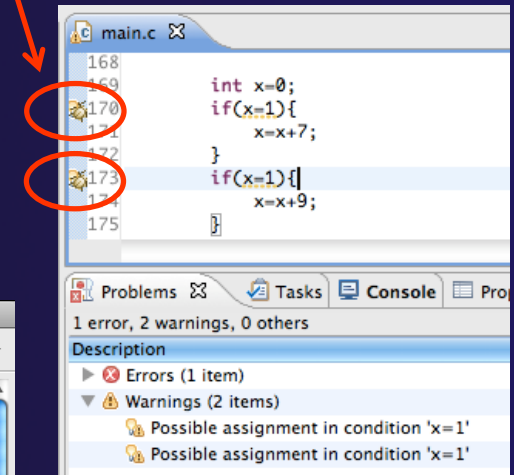
C/C++ > Code Analysis

and uncheck
all problems

✦ Select OK to
close
Preferences



Uncheck all



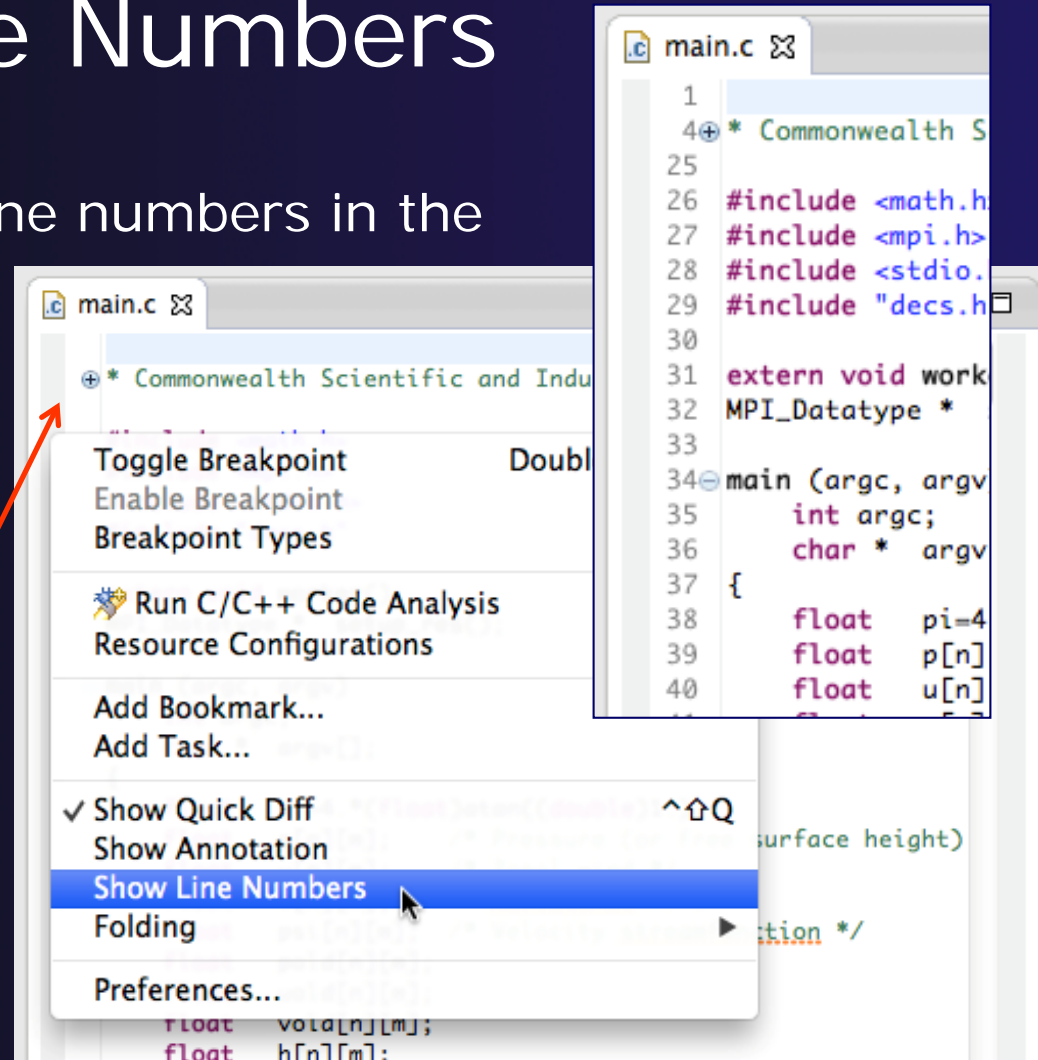
✦ If icons don't disappear:
Right mouse on Project >
Run C/C++ Code Analysis

✦ You can also enable/disable
this per project in Project
Properties

Editor-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++) {

```

- ★ 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**

```

26 #include <math.h>
27 #include "decs.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];

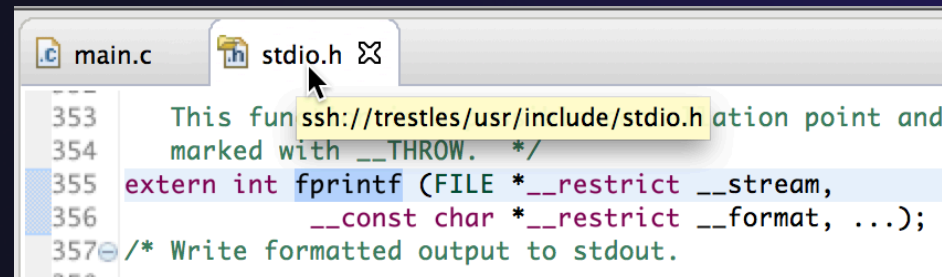
```

Option	Shortcut
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 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.  
358
```

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 completion list for the word "get". The list includes functions like `getchar_unlocked`, `getdelim`, `getenv`, `getline`, and `getloadavg`. A red arrow points to the `getenv` option. The editor background shows a C program with a `main` function that prints "Hello World".

```

13
14 int main(void) {
15     puts("!!!Hello World!!!"); /* prints !!!Hello World!!! */
16     get
17     • getchar_unlocked(void) : int
18     • getdelim(char ** __lineptr,* __n,int __delimit
19     • getenv(const char * __name) : char *
20     • getline(char ** __lineptr,* __n,FILE * __stream
21     • getloadavg(double * __loadavg,int __nelem)
    
```

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 a completion list for the word "for". The list includes templates like "for - for loop" and "for - for loop with temporary variable". The editor background shows a C program with a `main` function that prints "Hello World".

```

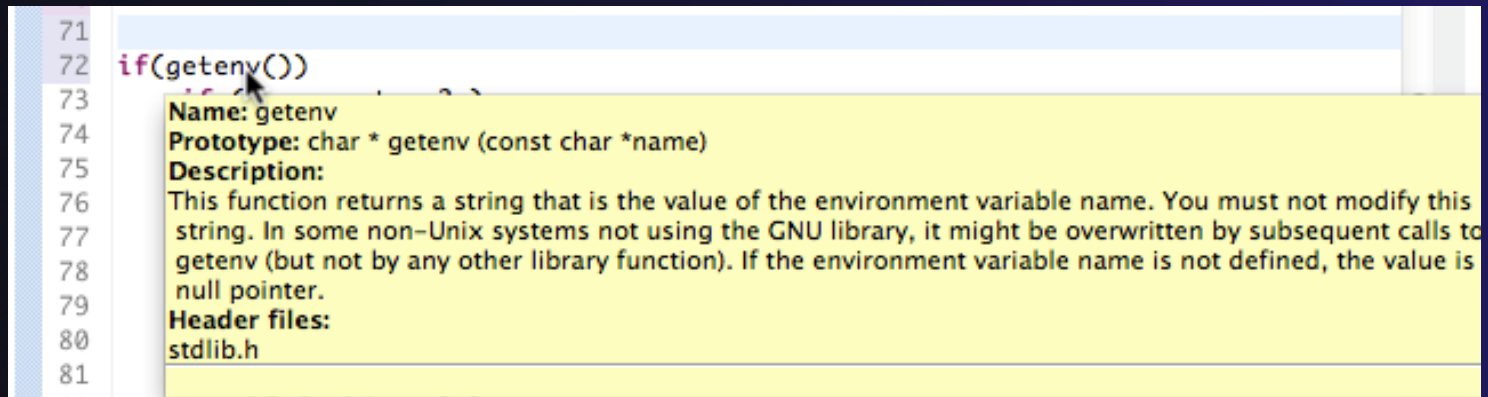
17 for
18   for - for loop
19   for - for loop with temporary variable
20 }
21
    
```

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 with a yellow tooltip displayed over the function call `if(getenv())` on line 72. 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

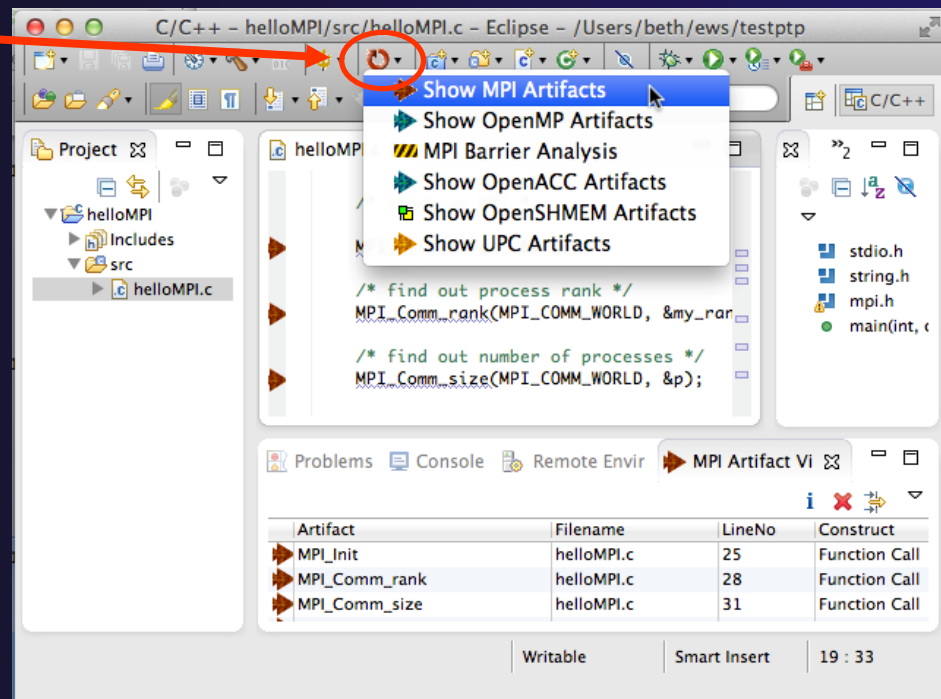
- ✦ Using Editor features for MPI
- ✦ MPI Help features
- ✦ Finding MPI Artifacts
- ✦ MPI New Project Wizards
- ✦ MPI Barrier Analysis

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

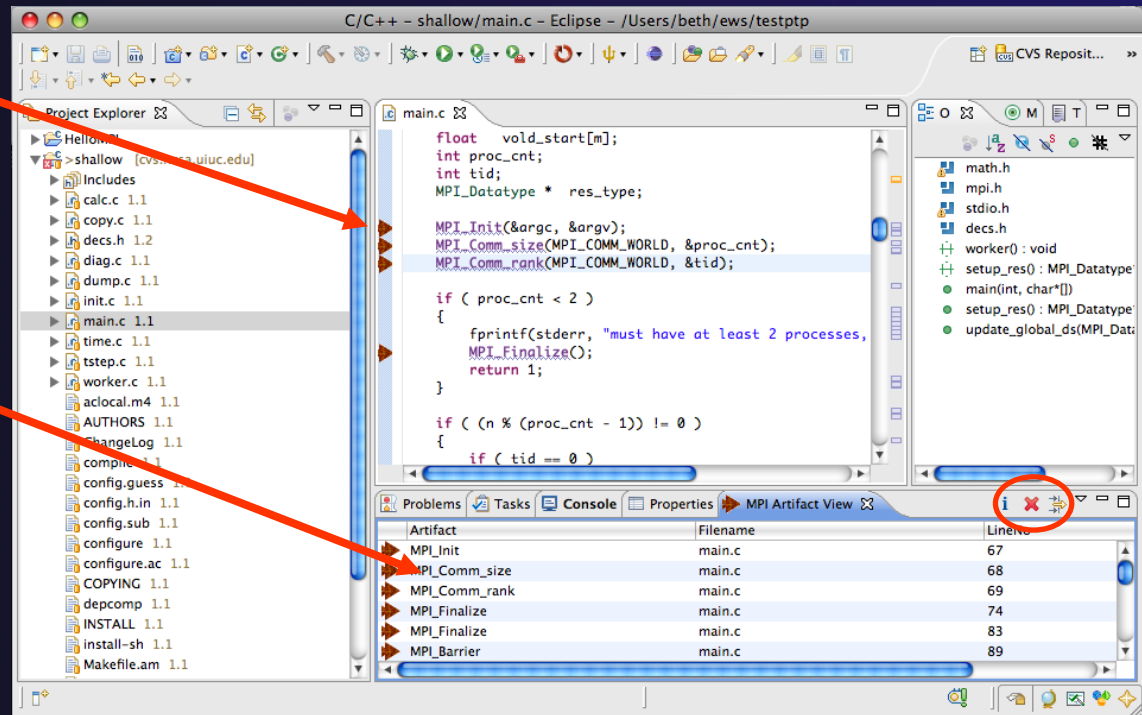
Show MPI Artifacts

- ✦ 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**

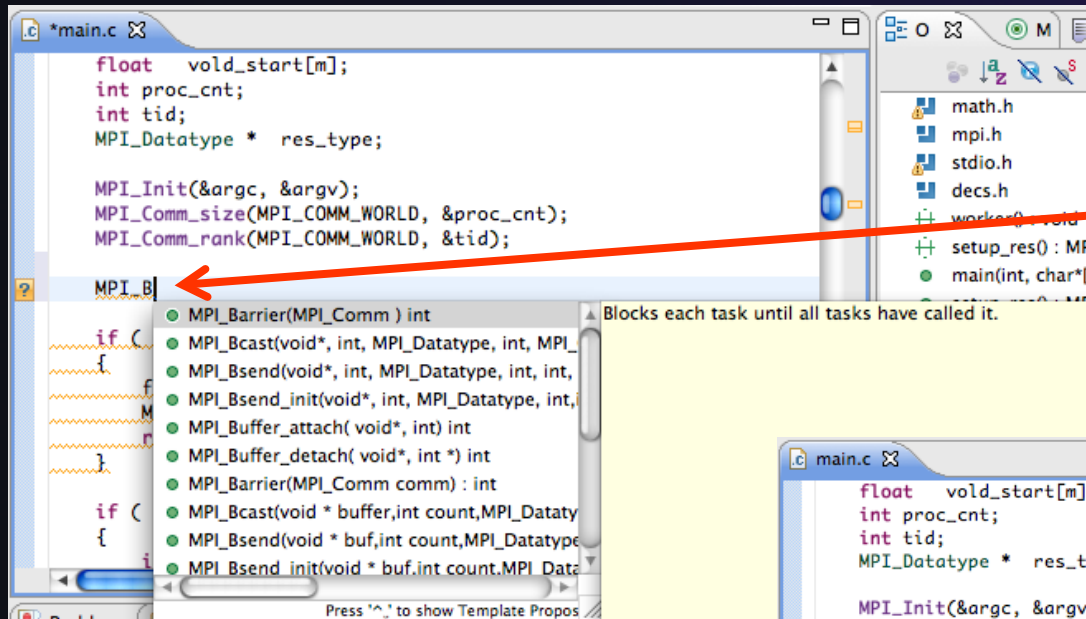


MPI Artifact View

- ✦ 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 

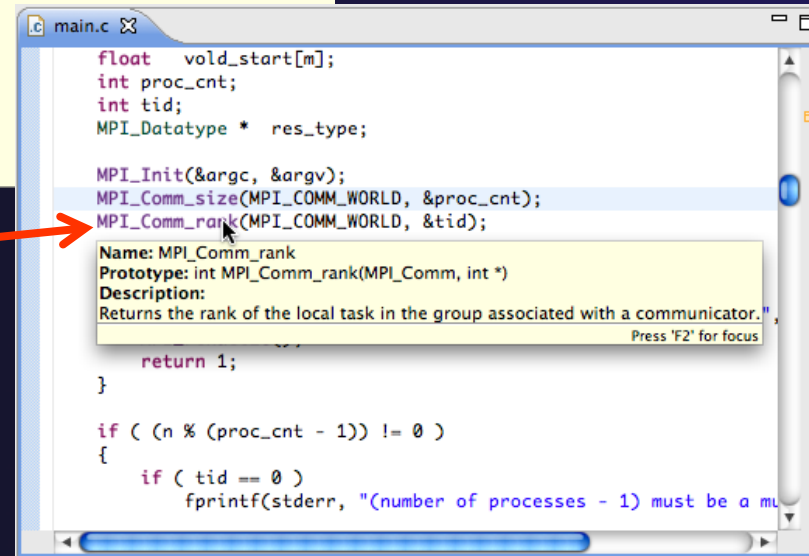


MPI Editor Features



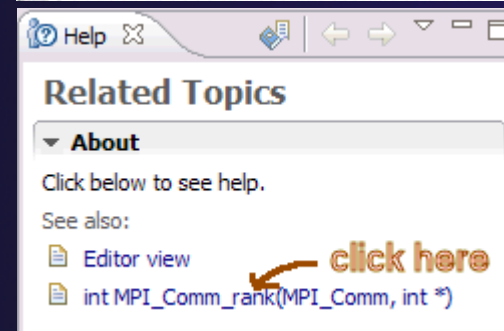
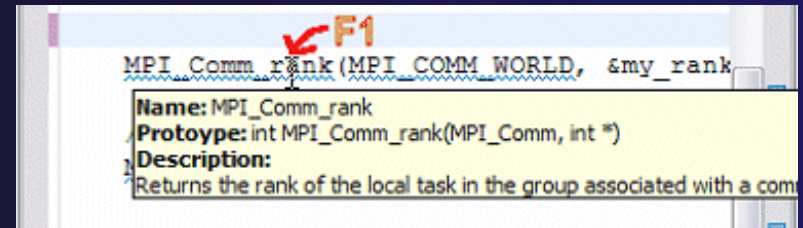
- ★ Code completion will show all the possible MPI keyword completions
- ★ Enter the start of a keyword then press <ctrl-space>

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

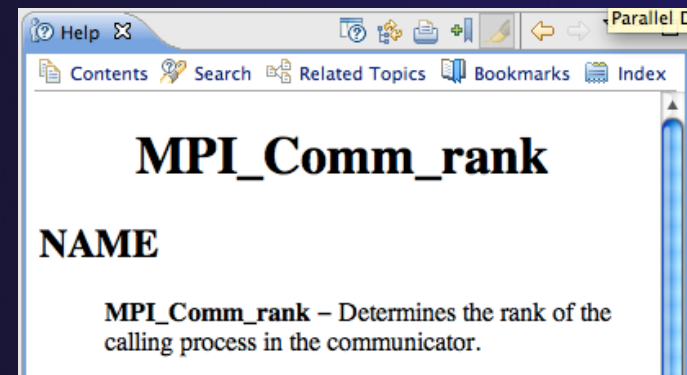


Context Sensitive Help

- ★ 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

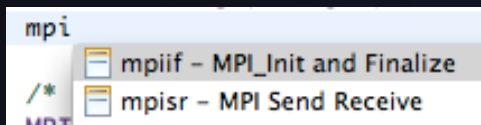
✦ Enter:

`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' view at the bottom left shows a table of barrier matching sets, and the 'Barrier Errors' view at the bottom right shows a tree structure 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

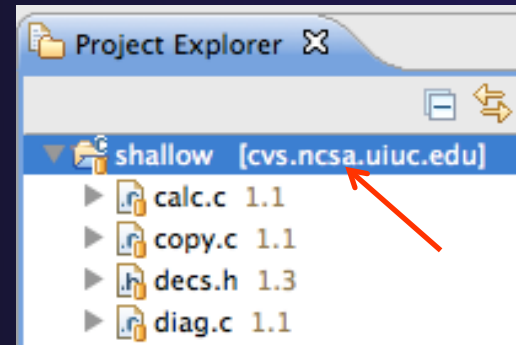
- ★ Verify barrier synchronization in C/MPI programs
- ★ For verified programs, lists barrier statements that synchronize together (match)
- ★ For synchronization errors, reports counter example that illustrates and explains the error

Local files only

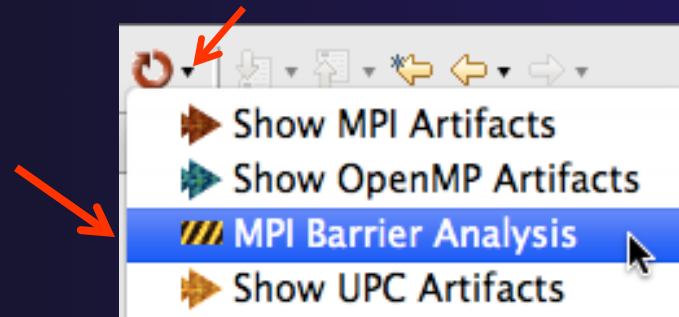
MPI Barrier Analysis (2)

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



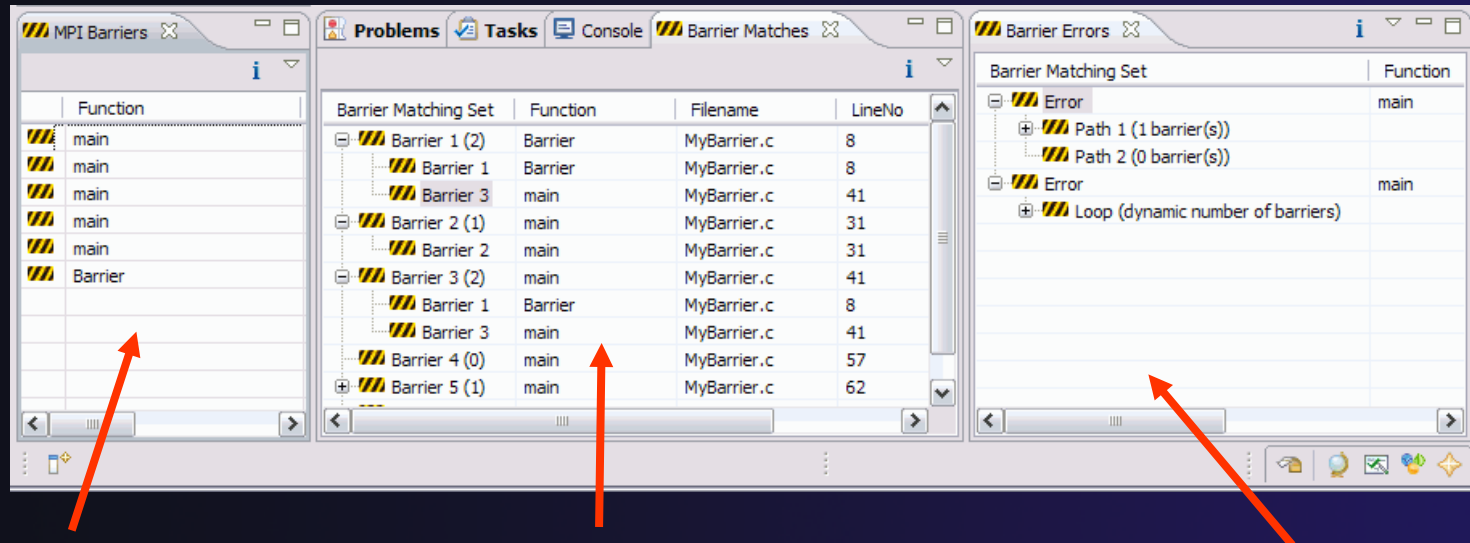
MPI Barrier Analysis (3)

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

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 89 highlighted: `MPI_Barrier(MPI_COMM_WORLD);`. Below the code editor, the 'MPI Barrier Matches' tab is active, displaying a table with the following data:

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 Blocks each task until
128  MPI_Bcast(void*, int, MPI_Datatype, int, MPI_
129  MPI_Bsend(void*, int, MPI_Datatype, int, int,
130  MPI_Bsend_init(void*, int, MPI_Datatype, int,
131  MPI_Buffer_attach(void*, int) int
132  MPI_Buffer_detach(void*, int*) int
  
```

```

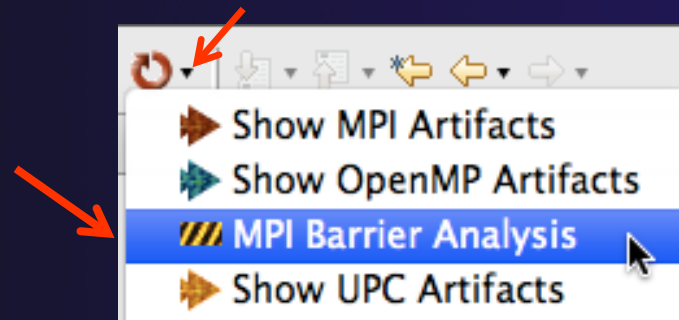
124
125  MPI_Barrier(MPI_COMM_WORLD);
126
  
```

Barrier Errors (2)

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

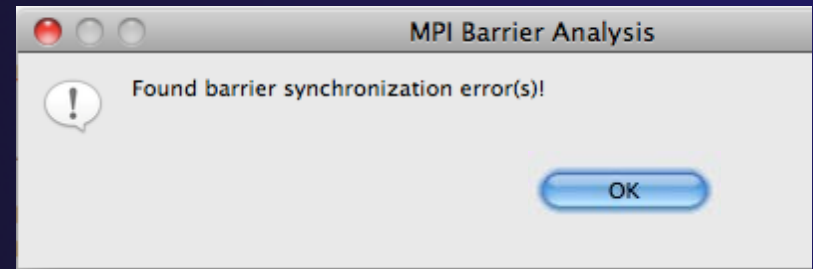


- ★ Run barrier analysis on shallow project again
 - ★ Select shallow project in Project Explorer first



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

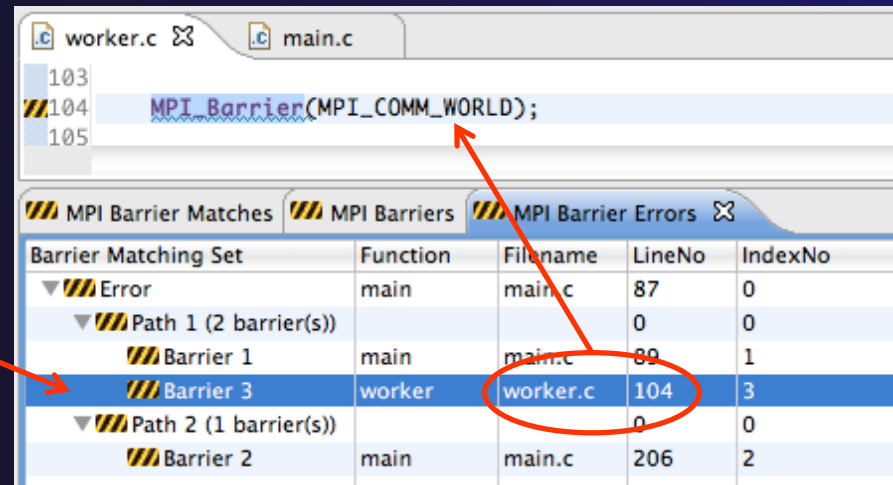


Barrier Matching Set	Function	Filename	LineNo	IndexNo
▼ Error	main	main.c	87	0
▼ Path 1 (2 barrier(s))			0	0
Barrier 1	main	main.c	89	1
Barrier 3	worker	worker.c	125	3
▼ Path 2 (1 barrier(s))			0	0
Barrier 2	main	main.c	206	2

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

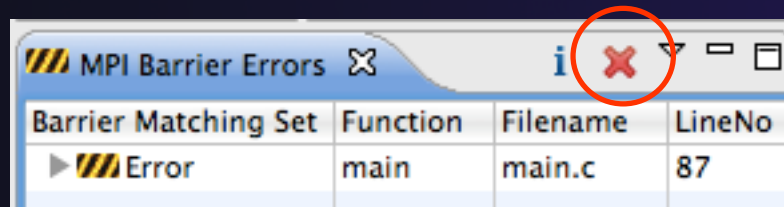
Fix Barrier Error

- ★ 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



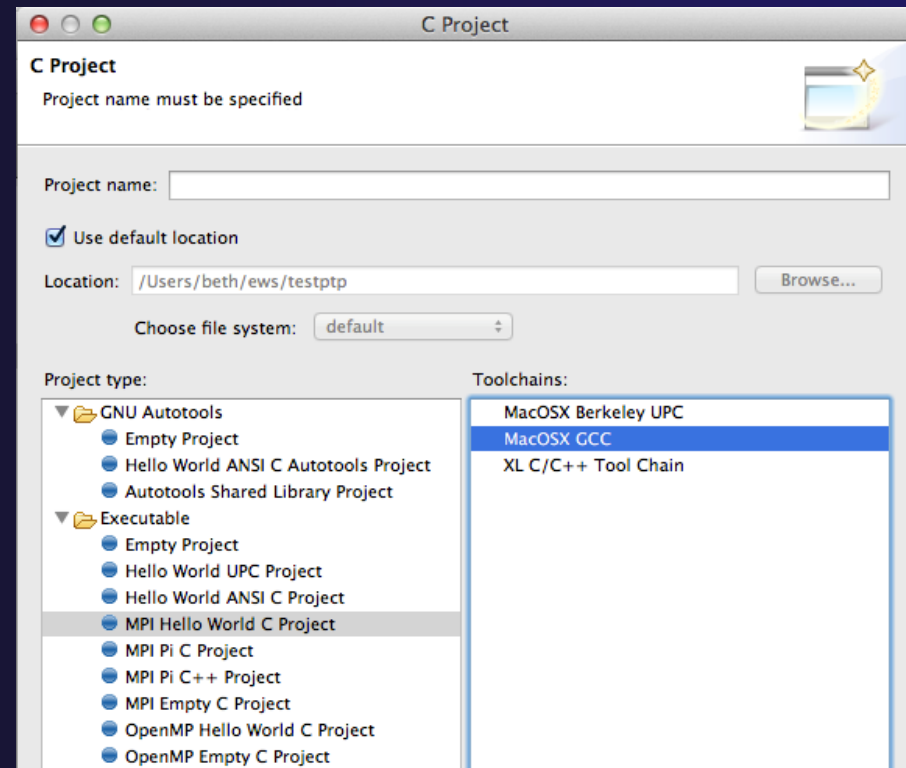
Remove Barrier Markers

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



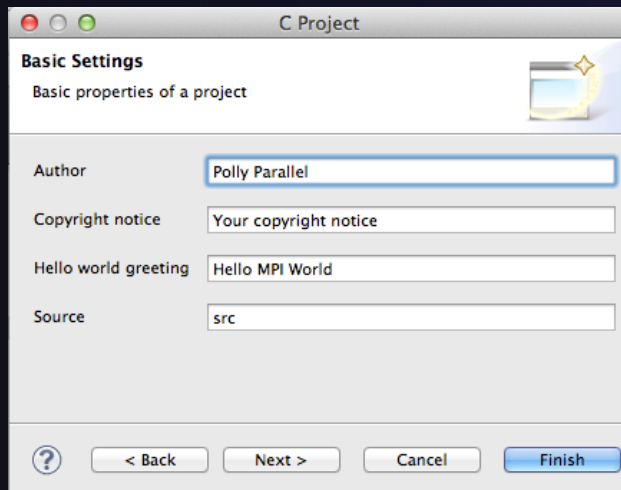
MPI New Project Wizards

- ★ 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)

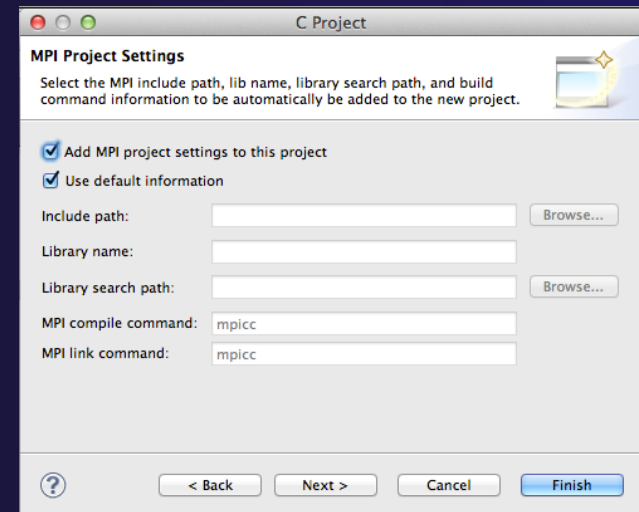
✦ Next > and fill in (optional) Basic Settings



The screenshot shows the 'Basic Settings' dialog box for a 'C Project'. The title bar reads 'C Project'. The dialog has a subtitle 'Basic Settings' and a description 'Basic properties of a project'. It contains four text input fields: 'Author' with the value 'Polly Parallel', 'Copyright notice' with the value 'Your copyright notice', 'Hello world greeting' with the value 'Hello MPI World', and 'Source' with the value 'src'. At the bottom, there are four buttons: a help button (question mark), '< Back', 'Next >', and 'Finish'.

✦ Next > and fill in MPI Project Settings

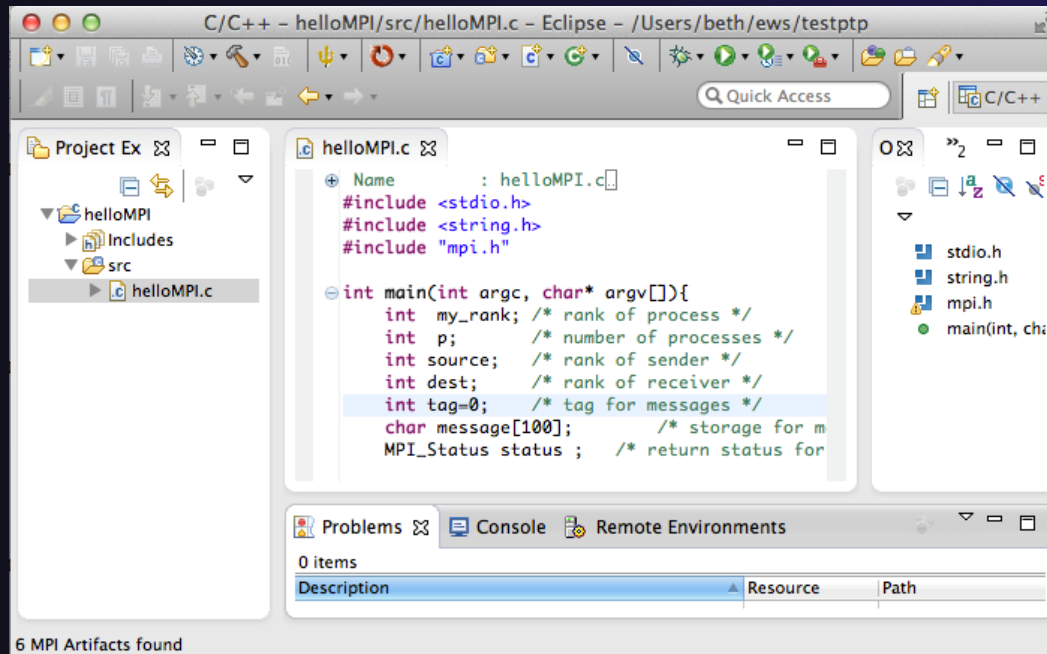
✦ Include path set in MPI Preferences can be added to project



The screenshot shows the 'MPI Project Settings' dialog box for a 'C Project'. The title bar reads 'C Project'. The dialog has a subtitle 'MPI Project Settings' and a description 'Select the MPI include path, lib name, library search path, and build command information to be automatically be added to the new project.' It contains two checked checkboxes: 'Add MPI project settings to this project' and 'Use default information'. Below these are four text input fields: 'Include path:' with a 'Browse...' button, 'Library name:', 'Library search path:' with a 'Browse...' button, 'MPI compile command:' with the value 'mpicc', and 'MPI link command:' with the value 'mpicc'. At the bottom, there are four buttons: a help button (question mark), '< Back', 'Next >', and 'Finish'.

MPI New Project Wizards (3)

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



The screenshot shows the Eclipse IDE interface for a C/C++ project named 'helloMPI'. The main editor displays the source file 'helloMPI.c' with the following code:

```
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
```

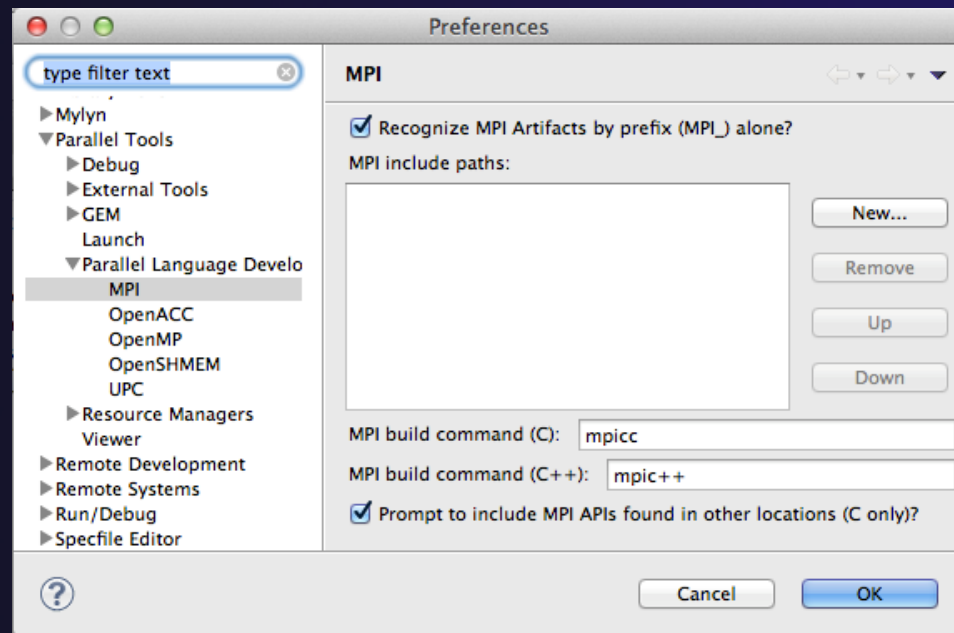
The IDE also shows a Project Explorer on the left with the project structure:

- Project Ex
- helloMPI
 - Includes
 - src
 - helloMPI.c

The bottom of the IDE shows the Problems, Console, and Remote Environments tabs. The Problems tab is active and shows 0 items. The status bar at the bottom indicates '6 MPI Artifacts found'.

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

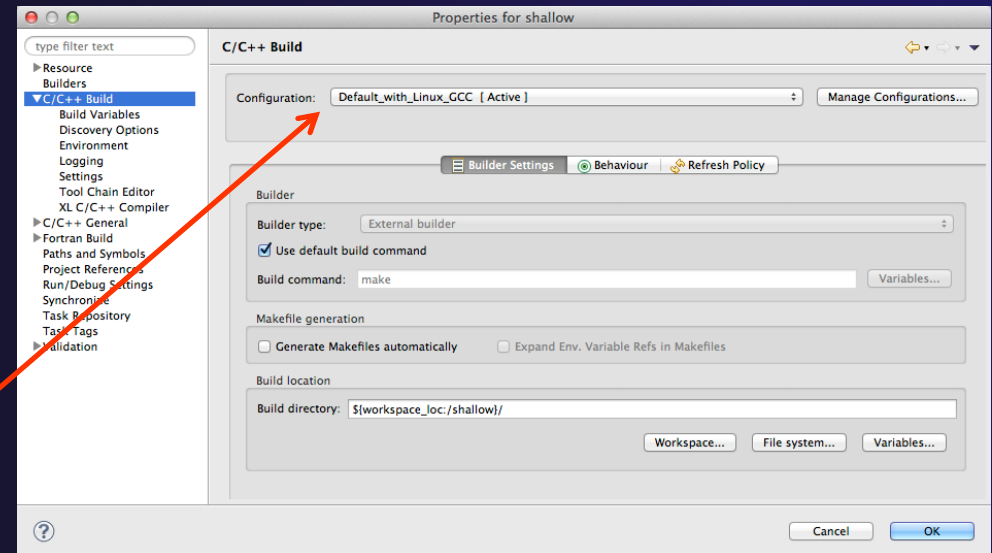
- ✦ Learn how to build an MPI program on a remote system

✦ 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

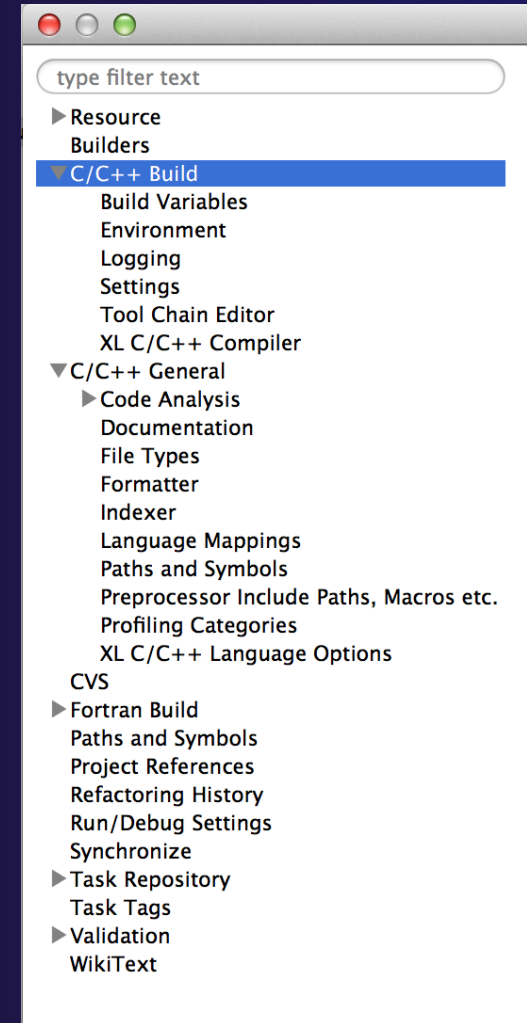
- ✦ A build configuration provides the necessary information to build the project
- ✦ The build configuration information is specified in the project properties
- ✦ Projects can have multiple build configurations, each configuration specifies a different set of options for a build
- ✦ Open the properties 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

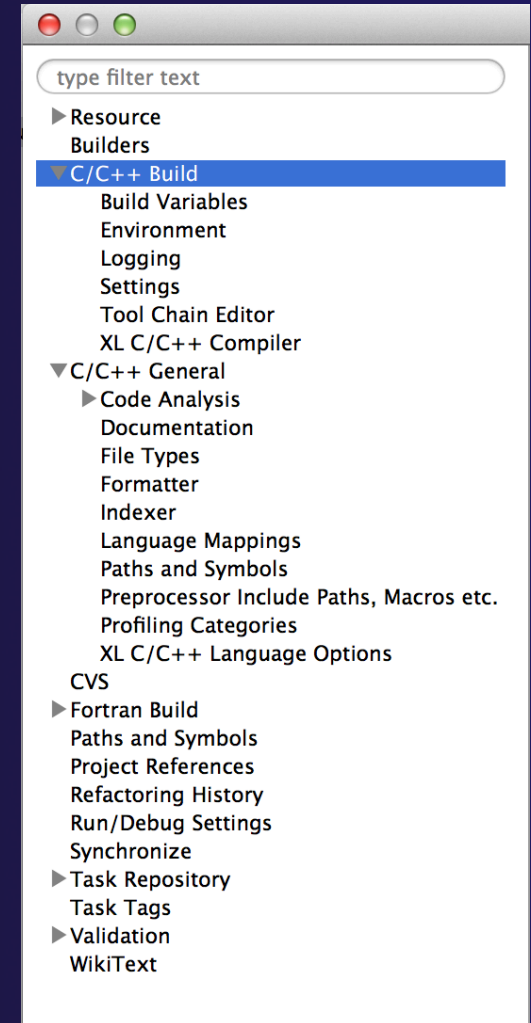
- ★ Allows the tools in a particular tool chain to be modified

★ XL C/C++ Compiler


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

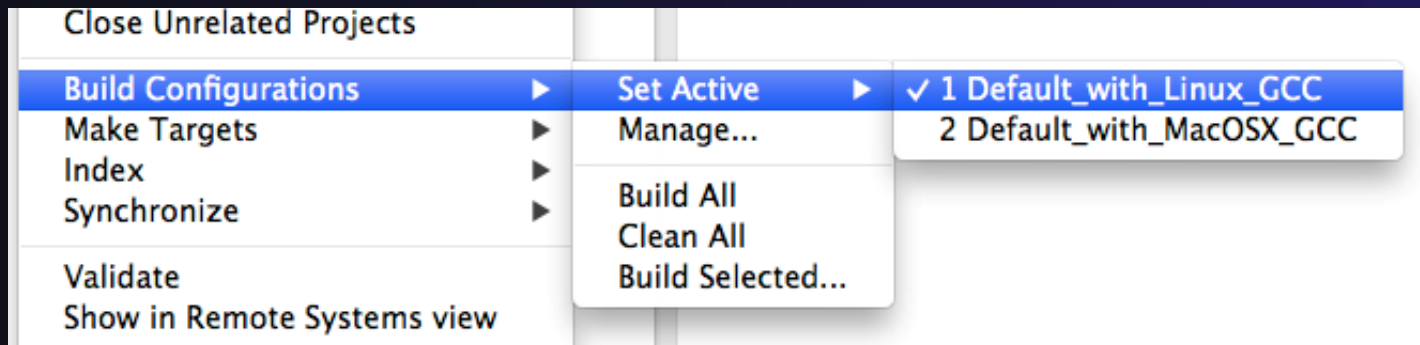
★ C/C++ General/Preprocessor Include Paths...

- ★ Set include paths here



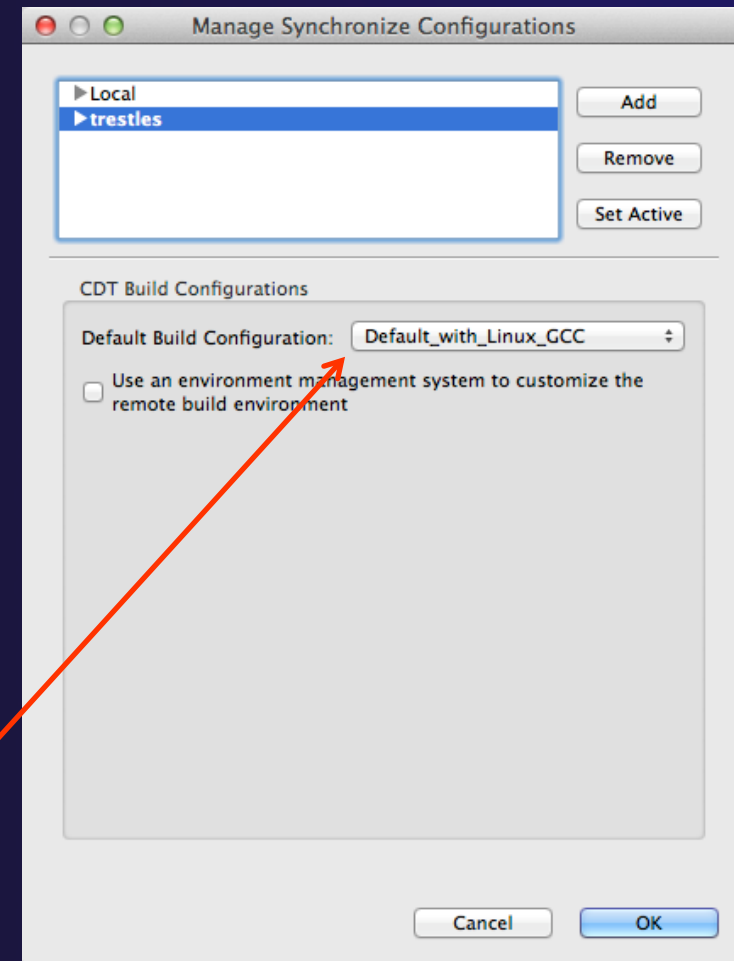
Selecting Build Configuration

- ★ 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



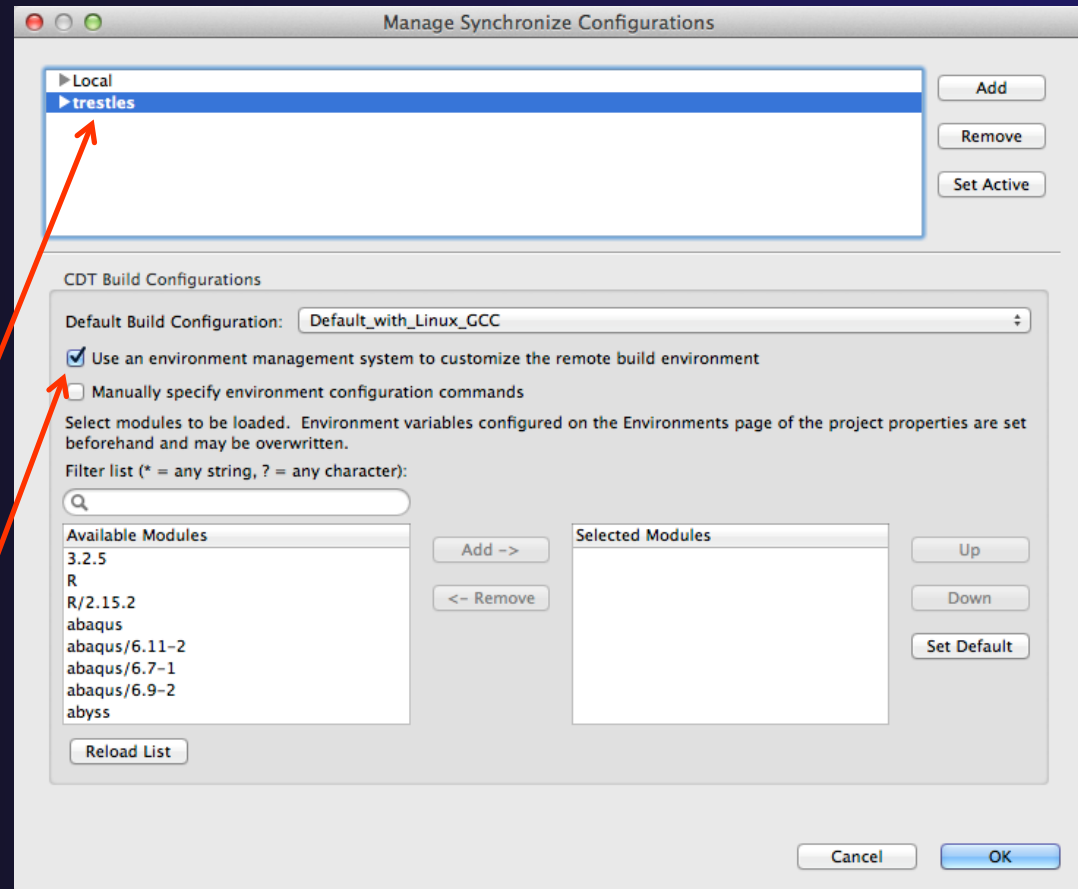
Building Synchronized Projects

- ★ 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



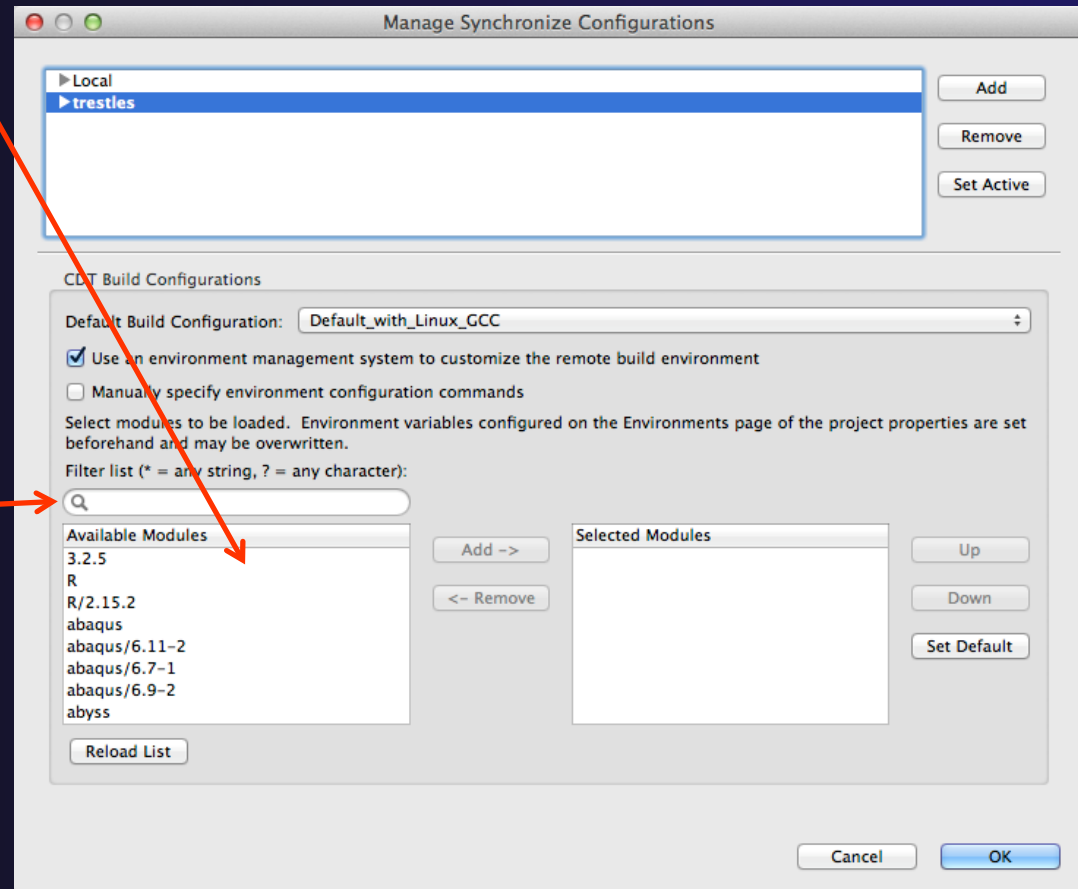
Configuring the Build Environment

- ✦ 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*

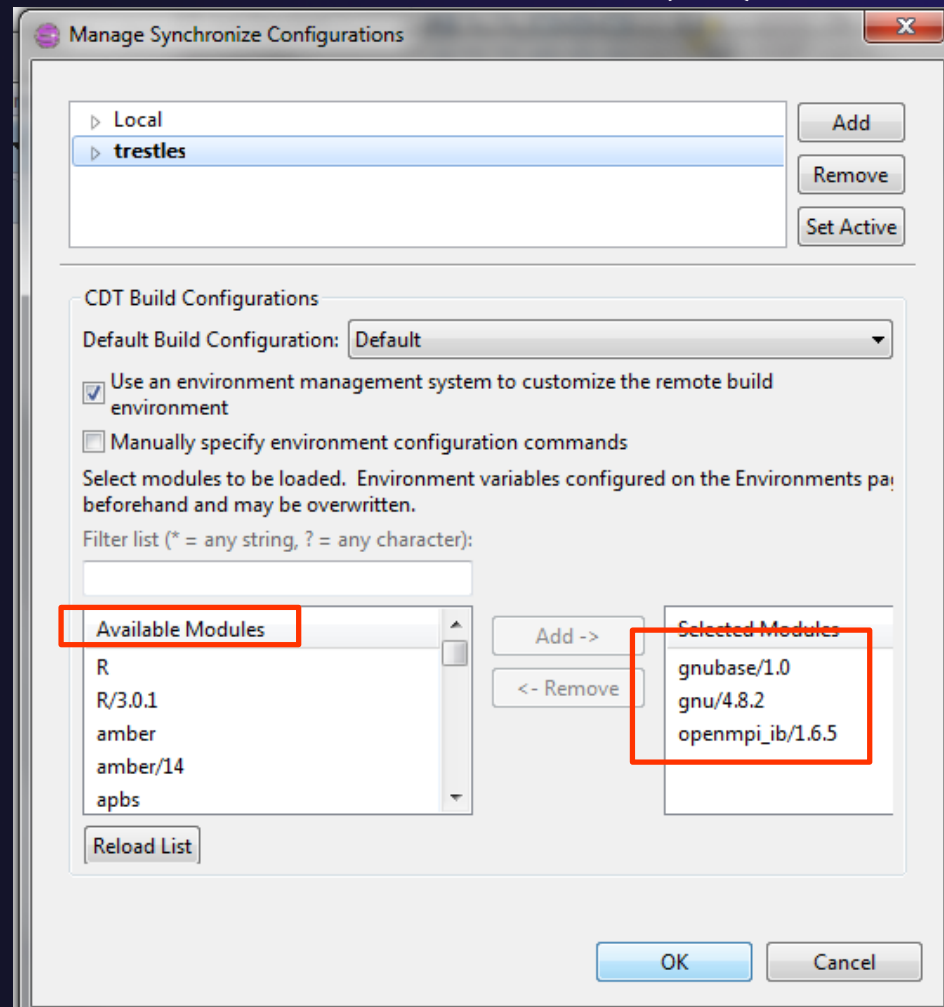


```
Console [X]
CDT Build Console [shallow]
17:53:20 *** Build of configuration Default_remote for project shallow ***
make all

**** 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 cblas_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

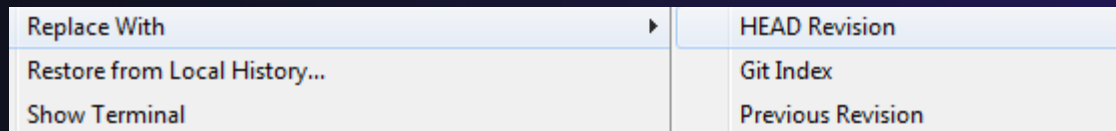
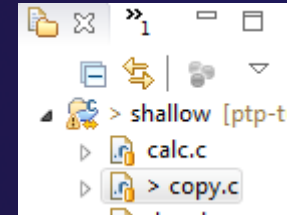


Start with original 'shallow'

★ Start with original 'shallow' code:

★ Project checked out from git:

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



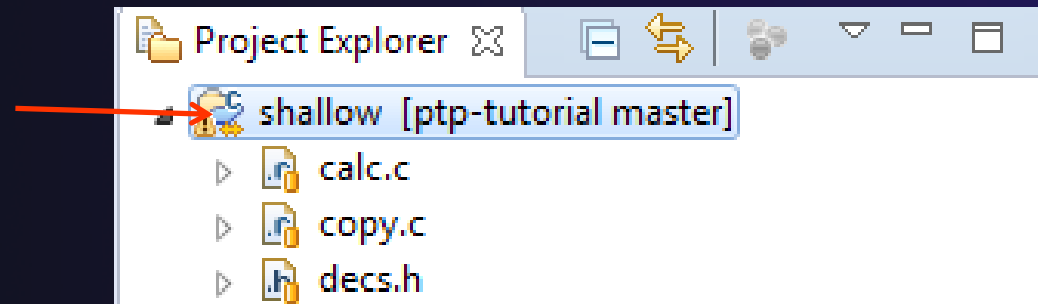
Also see Compare With ...

★ Other project:

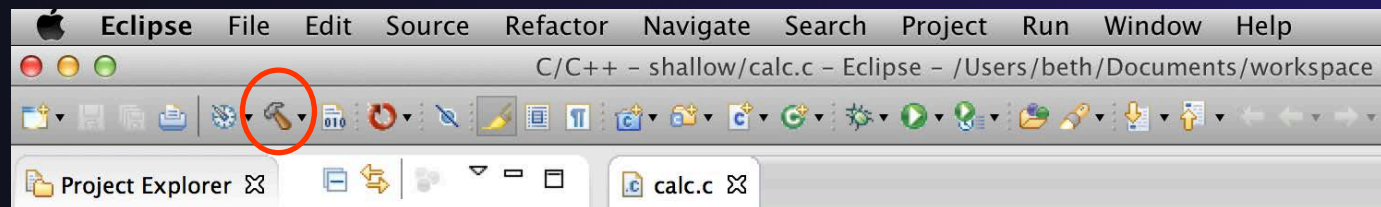
- ★ Right mouse on project,
Restore from local history – finds deleted files
- ★ Right mouse on file, **Compare With**
or **Replace With**

Starting the Build

- ★ Select the project in Project Explorer



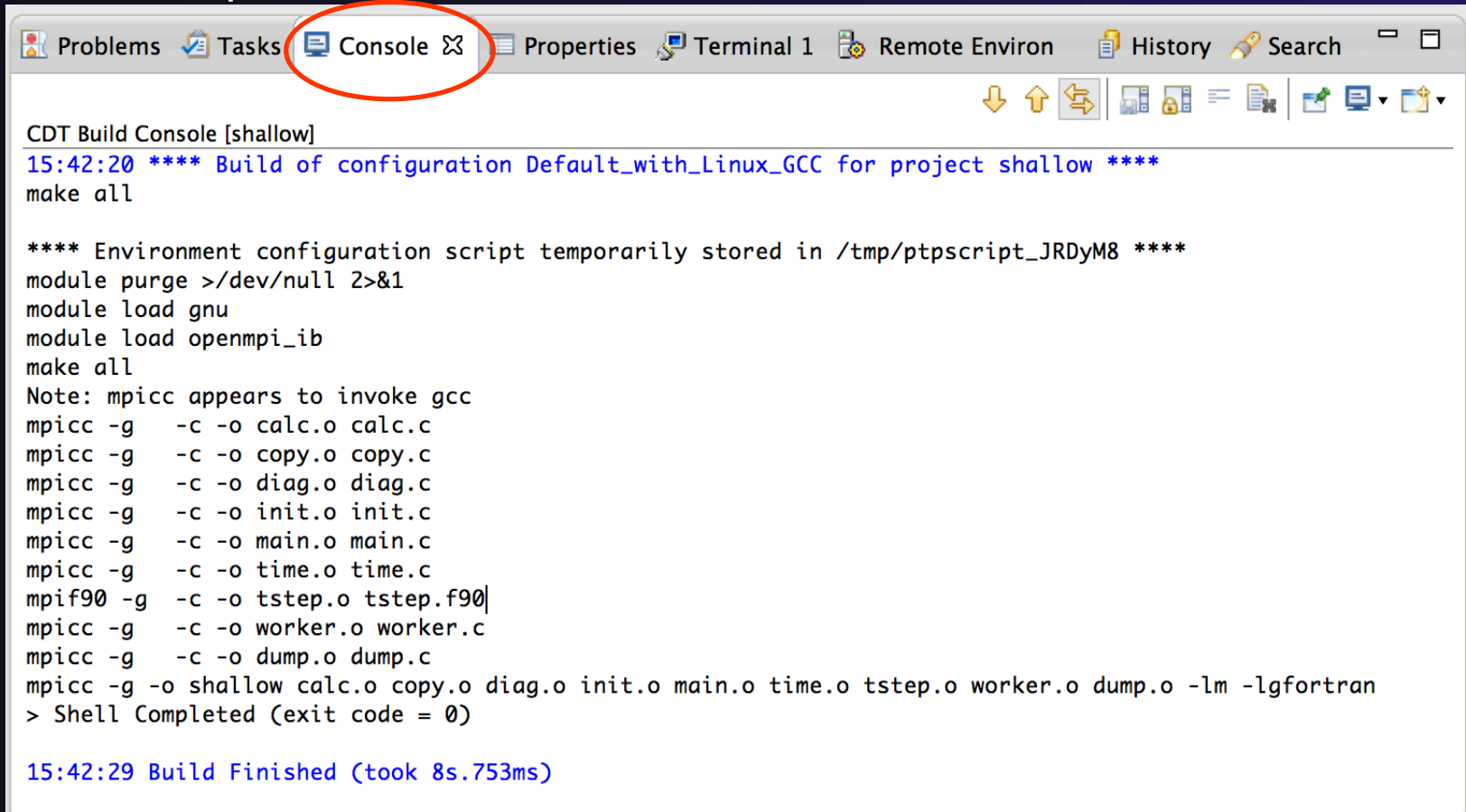
- ★ Click on the  hammer button in toolbar to run a build using the active build configuration



- ★ 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

A screenshot of an IDE's console window. The window title is "CDT Build Console [shallow]". The "Console" tab is selected and circled in red. The output text shows a successful build process for a project named "shallow". It includes a timestamp "15:42:20", a status message "**** Build of configuration Default_with_Linux_GCC for project shallow ****", and a list of compilation commands using "mpicc" and "mpif90". The build finishes at "15:42:29" with a duration of "8s.753ms".

```
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 shows a project named 'shallow' with various source files. The main editor window shows the file 'main.c' with a syntax error highlighted in blue. The error is a 'syntax error before '' token' on line 97. The Problems view at the bottom shows a list of errors, with the first three items being 'syntax error before '' token' on line 97. Red arrows point from the text on the left to the corresponding elements in the screenshot: a marker on the file 'main.c' in the Project Explorer, a marker on line 97 in the editor, the highlighted line in the editor, a marker on the overview ruler, the error listed in the Problems view, and a double-click on the error in the Problems view.

```

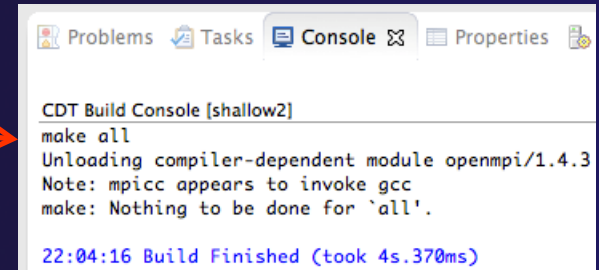
87  if (tid != 0) {
88      worker();
89      MPI_Barrier(MPI_COMM_WORLD);
90      MPI_Finalize();
91  } else {
92
93      /* master process */
94
95      chunk_size = n / (proc_cnt - 1);
96
97      for (i = 1; i < proc_cnt; i++) {
98          /* calculate each worker's boundary */
99          master_packet[JSTART] = (i - 1) * chunk_size;
100
101          if (i == proc_cnt - 1)
102              master_packet[JEND] = n - 1;
103          else
104              master_packet[JEND] = i * chunk_size - 1;
105
106          if (i == 1)
107              prv = proc_cnt-1;
108          else
109              prv = i-1;
110
111          master_packet[PREV] = prv;
112
113          if (i == proc_cnt - 1)
114              nxt = 1;
115          else
116              nxt = i+1;
117
118          master_packet[NEXT] = nxt;

```

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

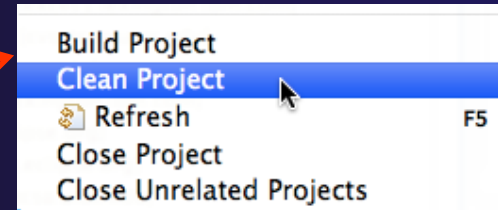
✦ If no changes have been made, `make` doesn't think a build is needed e.g. if you only change the Makefile



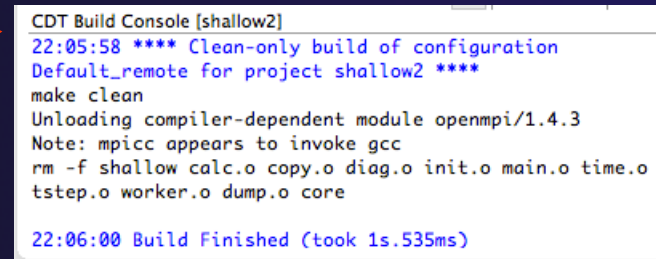
```
CDT Build Console [shallow2]
make all
Unloading compiler-dependent module openmpi/1.4.3
Note: mpicc appears to invoke gcc
make: Nothing to be done for `all'.

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

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



✦ Build console will display results



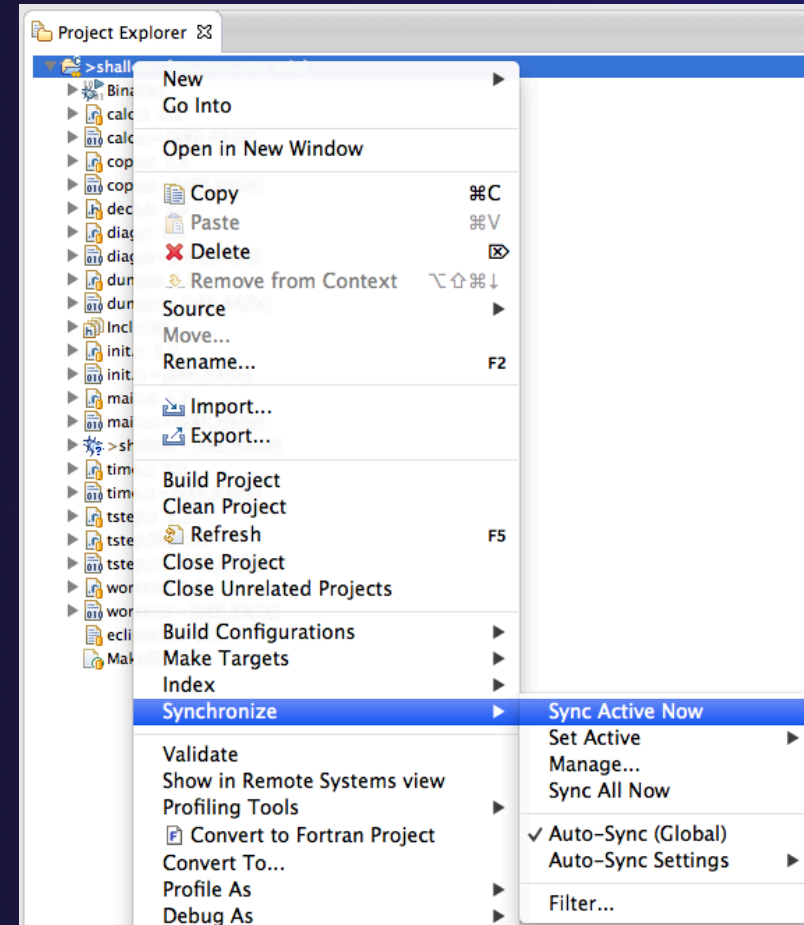
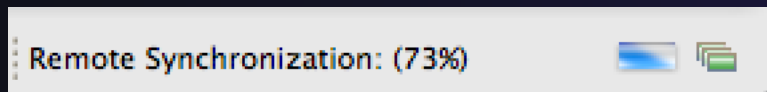
```
CDT Build Console [shallow2]
22:05:58 **** Clean-only build of configuration
Default_remote for project shallow2 ****
make clean
Unloading compiler-dependent module openmpi/1.4.3
Note: mpicc appears to invoke gcc
rm -f shallow calc.o copy.o diag.o init.o main.o time.o
tstep.o worker.o dump.o core

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

✦ Rebuild project by clicking on build button again 

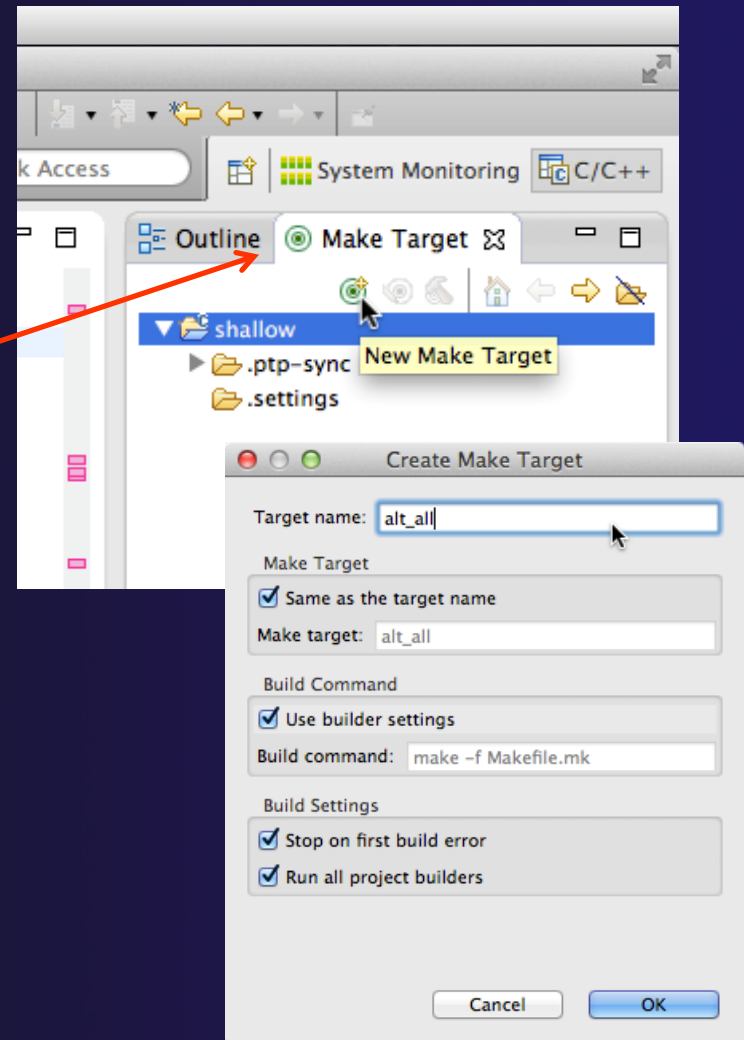
Forcing a Resync

- ✦ 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

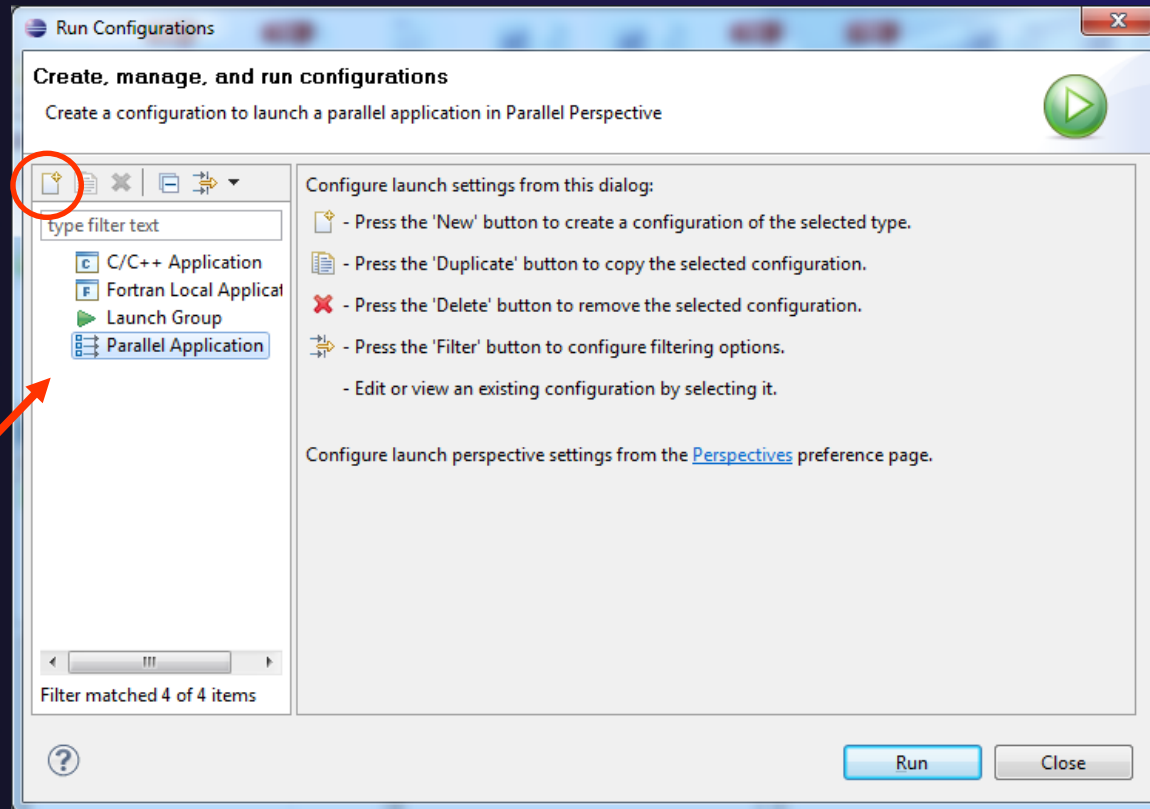
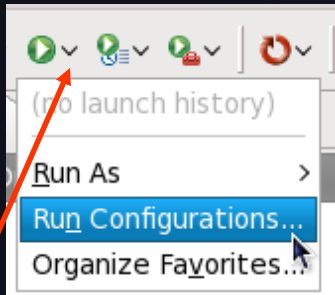
★ Objective


- ★ 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



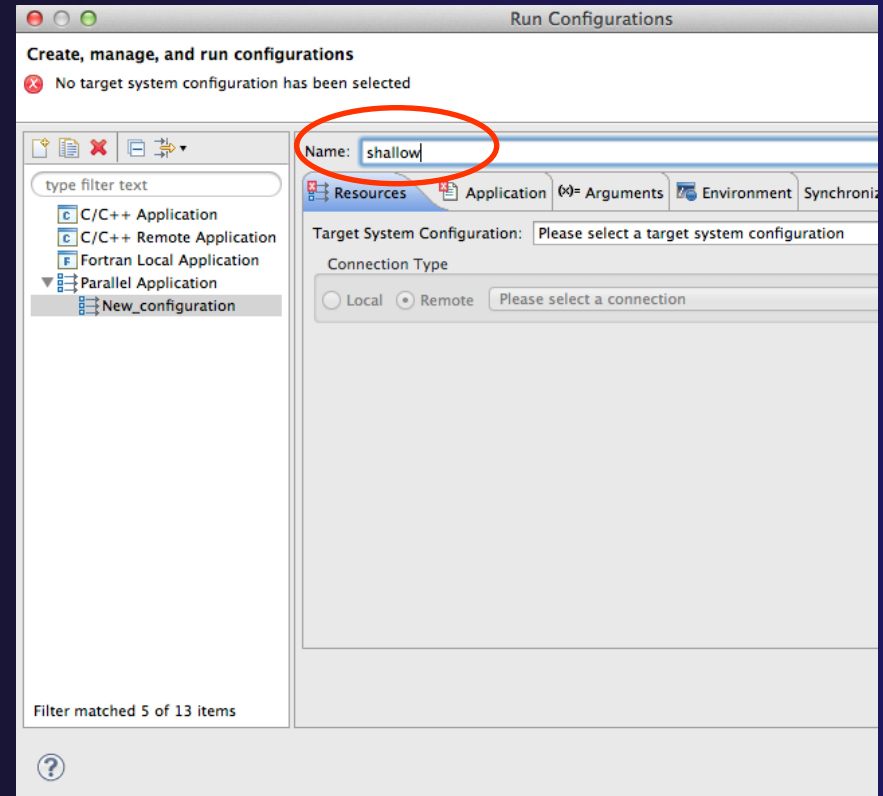
- ★ Open the run configuration dialog **Run>Run Configurations...**
- ★ Select **Parallel Application**
- ★ Select the **New** button 

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.

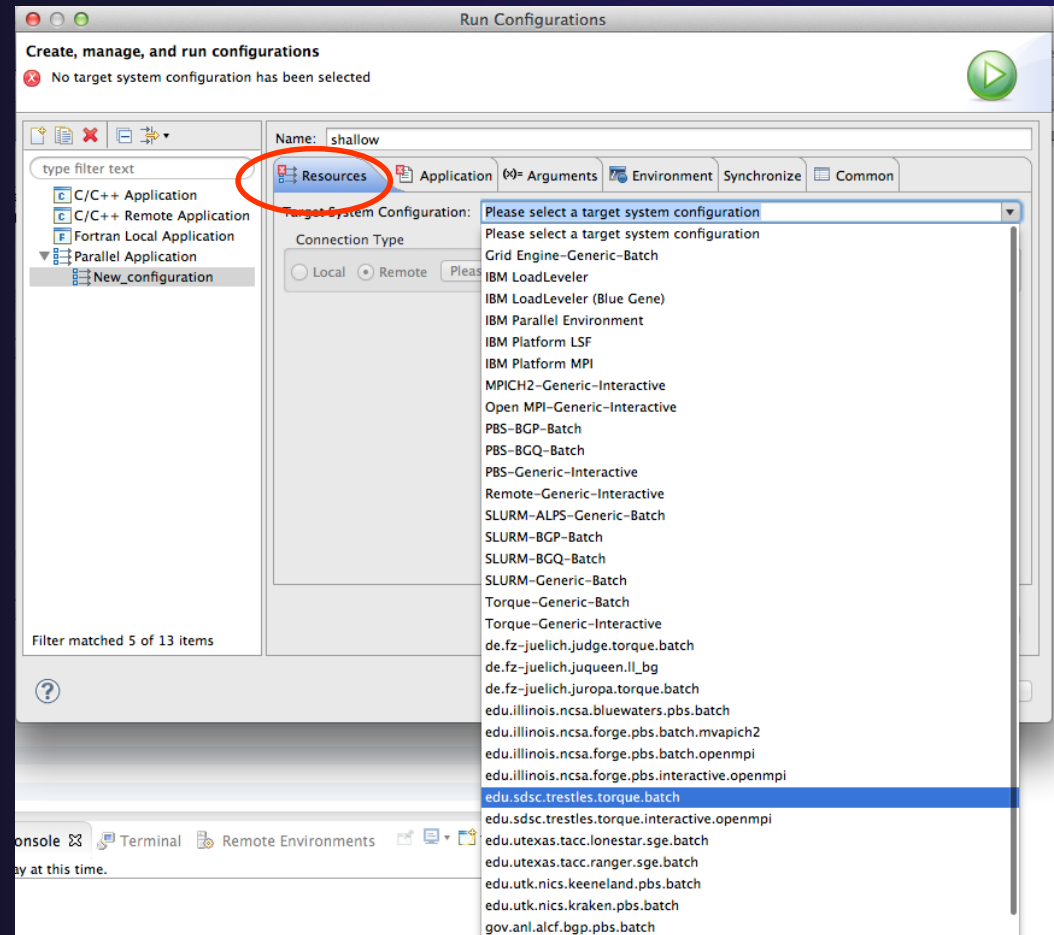
Set Run Configuration Name

- ✦ 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



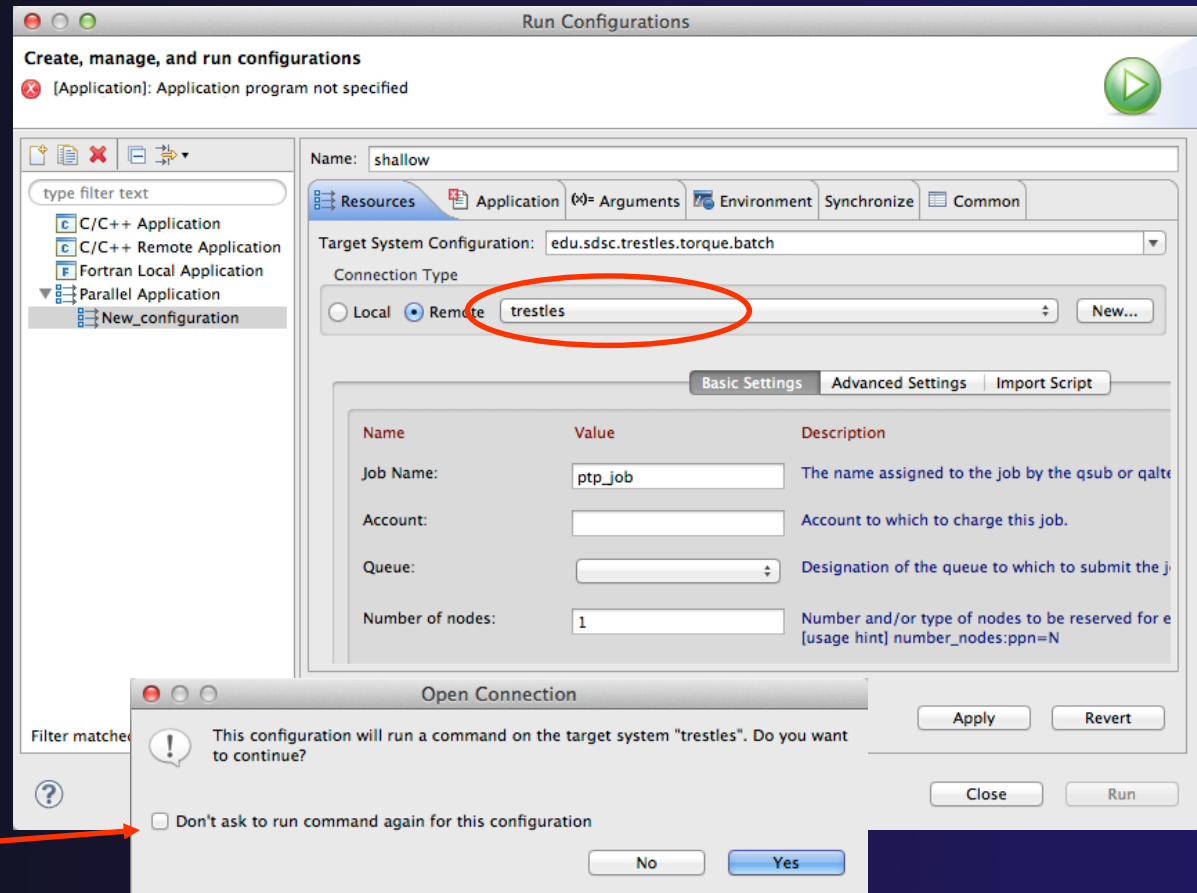
Configuring the Target System

- ★ In **Resources** tab, select a **Target System Configuration** that corresponds to your target system
 - ★ Use **Generic 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 *gordon.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
Queue:		Designation of the
Number of nodes:	1	Number and/or type of nodes [usage hint] number of nodes
Total Memory Needed:		Maximum amount of memory
Wallclock Time:	00:30:00	Maximum amount of 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.

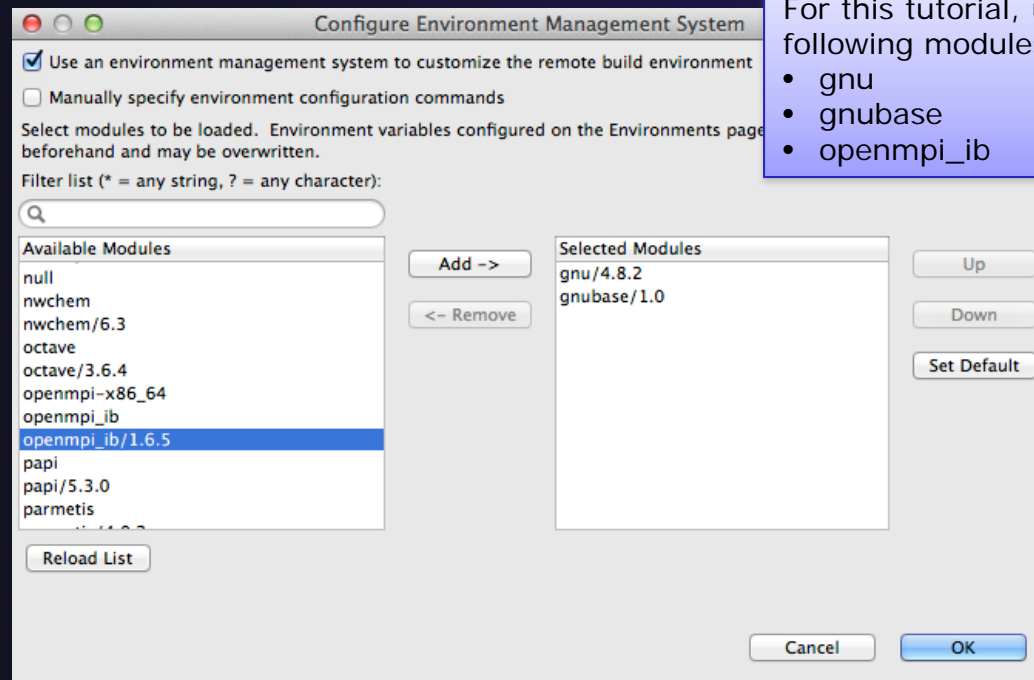
View Script View Configuration Restore Defaults

For this tutorial:

- Queue: **normal**
- Number of nodes: **1:ppn=5**
- MPI Command: **mpirun**
- MPI Number of Processes: **5**
- 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



Viewing the Job Script

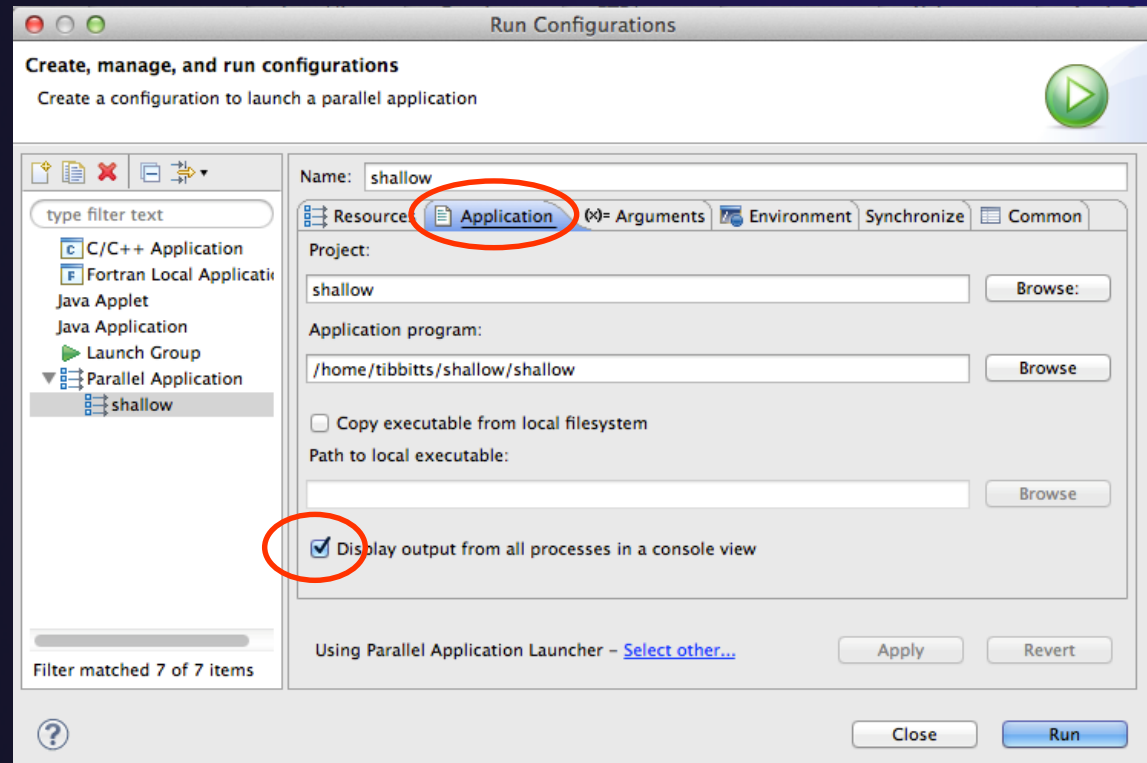
- Some target configurations will provide a **View Script** button
- Click on this to view the job script that will be submitted to the job scheduler
- Batch scheduler configurations should also provide a means of importing a batch script

The screenshot shows a web-based job configuration interface. The main form includes fields for Account, Queue (set to 'shared'), Number of nodes (set to '1:ppn=5'), Total Memory Needed, Wallclock Time (set to '00:30:00'), MPI Command (set to 'mpirun'), MPI Number of Processes (set to '5'), and Export Environment (checked). At the bottom, there is a 'View Script' button. An orange arrow points from the 'View Script' button in the list to this button in the interface. A secondary 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}
```

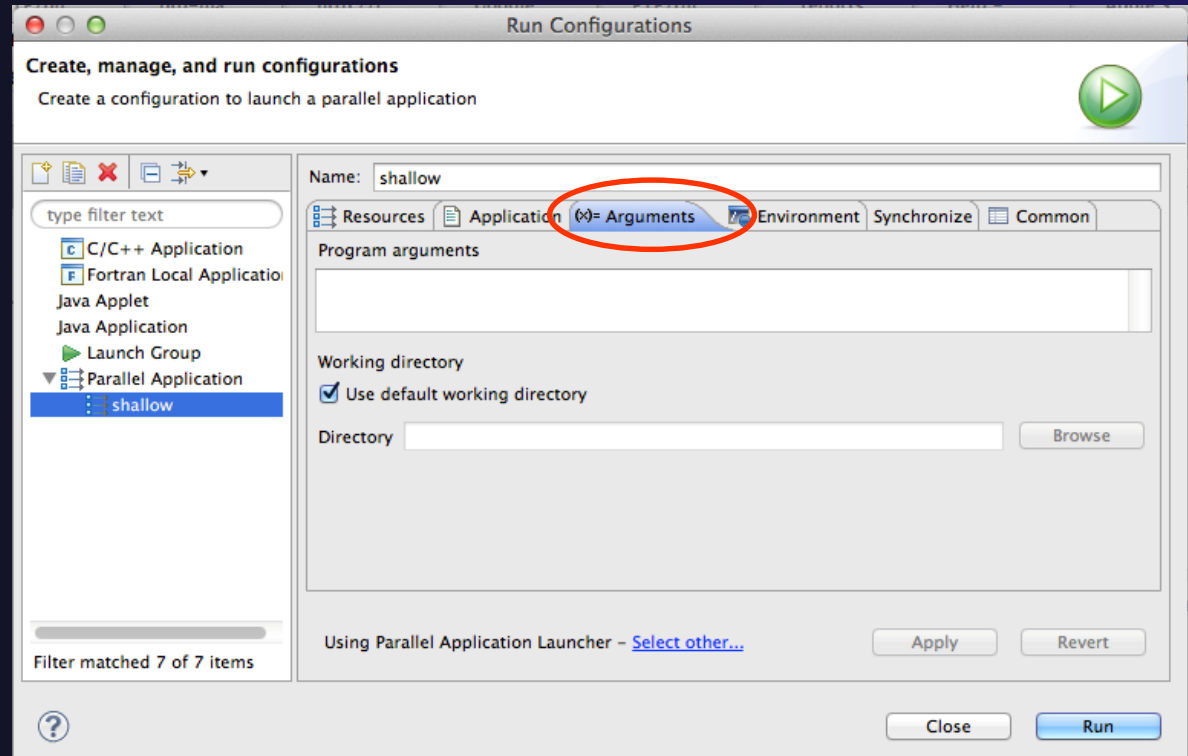
Application Tab

- ✦ 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**



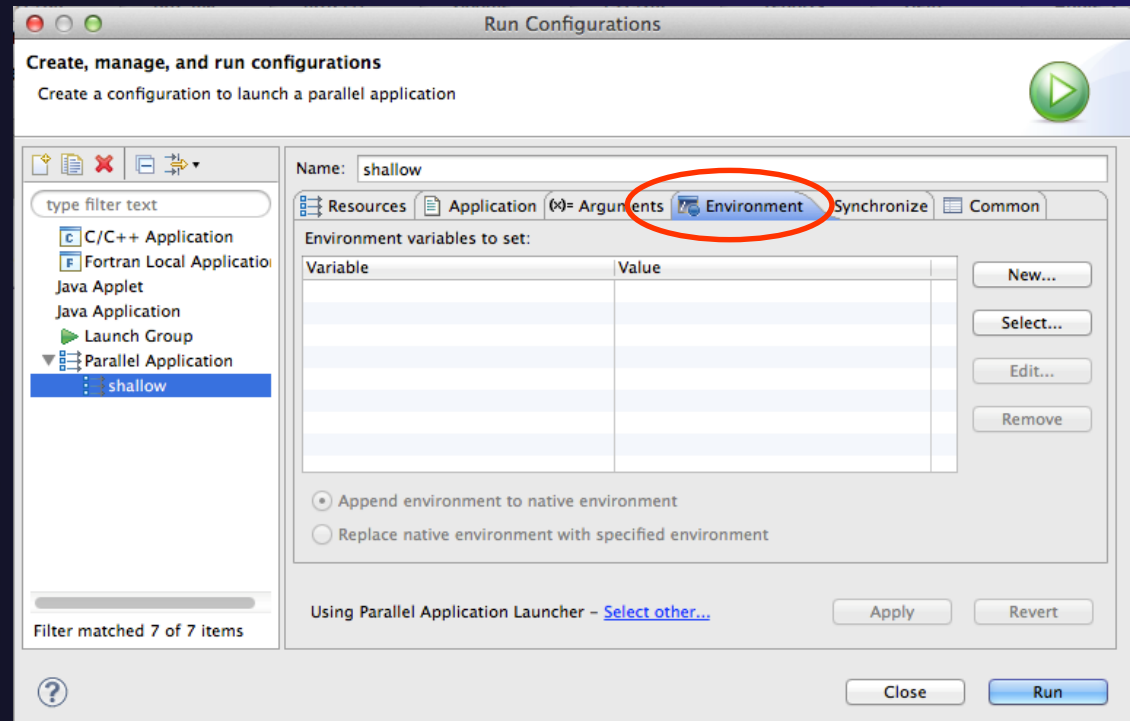
Arguments Tab (Optional)

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



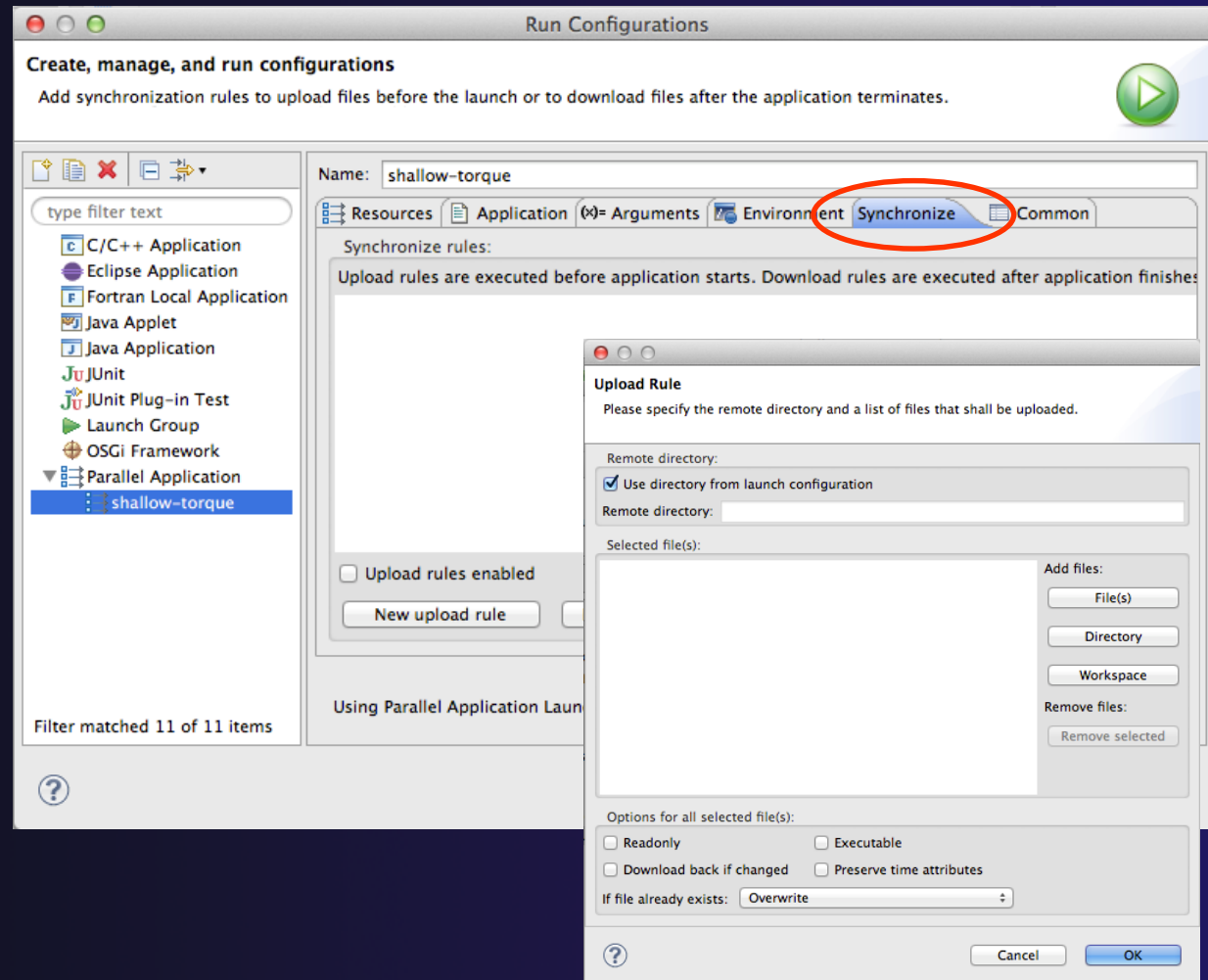
Environment Tab (Optional)

- ★ 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



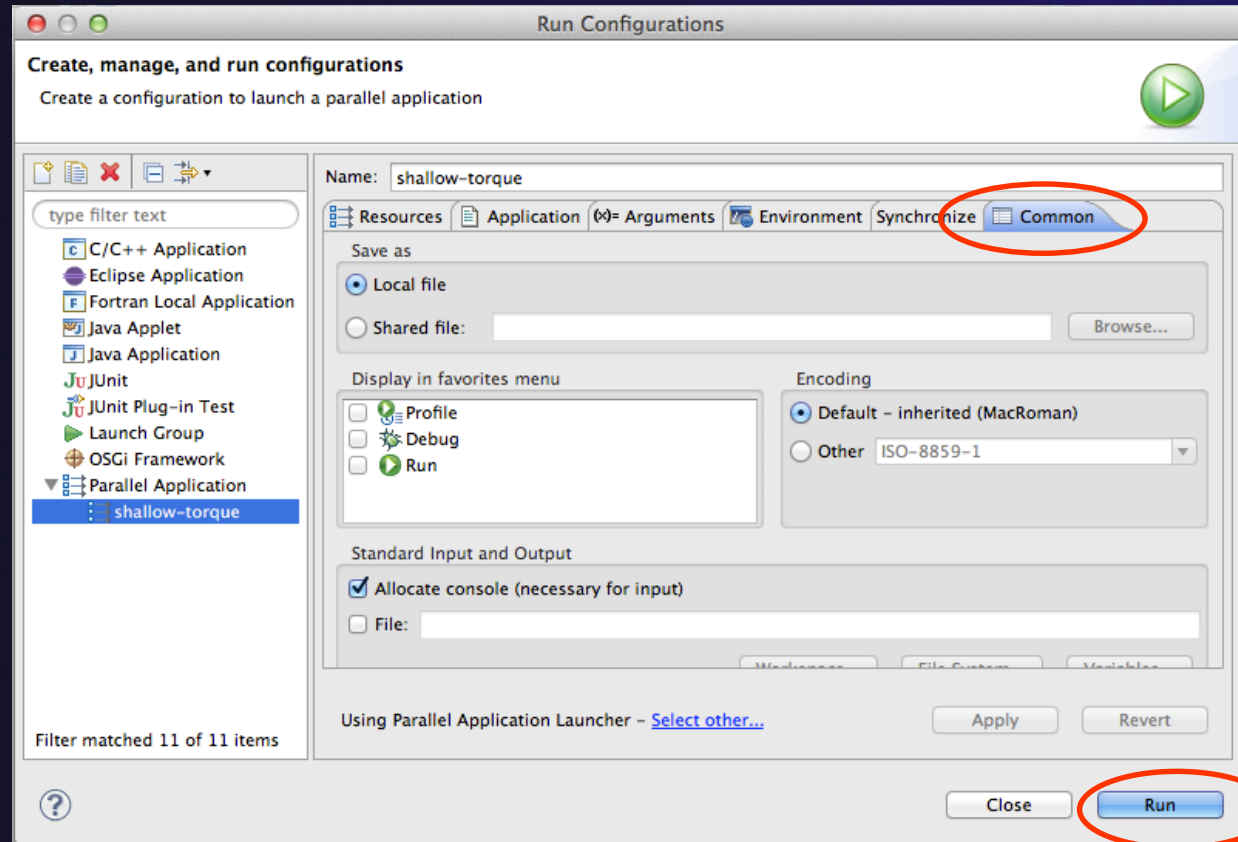
Synchronize Tab (Optional)

- ★ 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



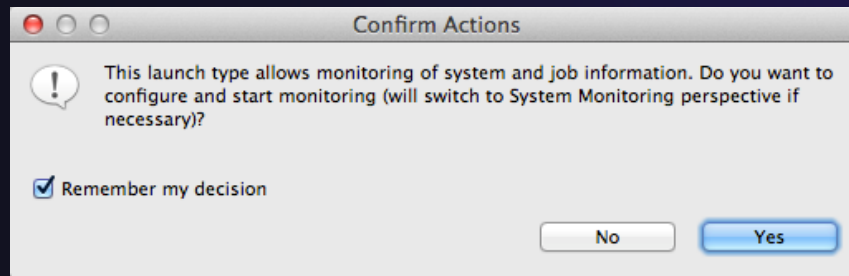
Common Tab (Optional)

- ★ 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. The top toolbar includes icons for System Monitoring and Parallel Debug. The main area is divided into several panes:

- Monitors:** Shows the connection name 'trestles.sdsc.edu' and system type 'TORQUE Resource Manager'.
- Active Jobs:** A table listing jobs with columns for step, owner, queue, wall, queued, dispatched, and total.

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
- Inactive Jobs:** A table listing jobs with columns for step, owner, queue, wall, queued, dispatched, and total.

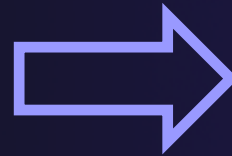
step	owner	queue	wall	queued	dispatch	total
1056...	tibbitts	shared	1800	201...	201...	5
1056...	tibbitts	shared	?	?	?	?
1056...	tibbitts	shared	?	?	?	?
- Messages:** Shows a message: '<terminated> shallow [Parallel Application] Runtime process 1'.
- Console:** Shows the same message as the Messages pane.
- Resource Grid:** A large grid of colored squares representing resource usage across multiple nodes. Red arrows point from the 'Active Jobs' table to specific colored squares in the 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 shows the System Monitoring interface with the following components:

- Monitors:** Shows connection name (z25c2s2, forge.ncsa.illinois.edu) and system type (IBM LoadLeveler, Open MPI, TORQUE Resource Manager).
- Active Jobs:** A table listing jobs with columns: step, owner, queue, wall, queuec, dispatc, totalco, status.

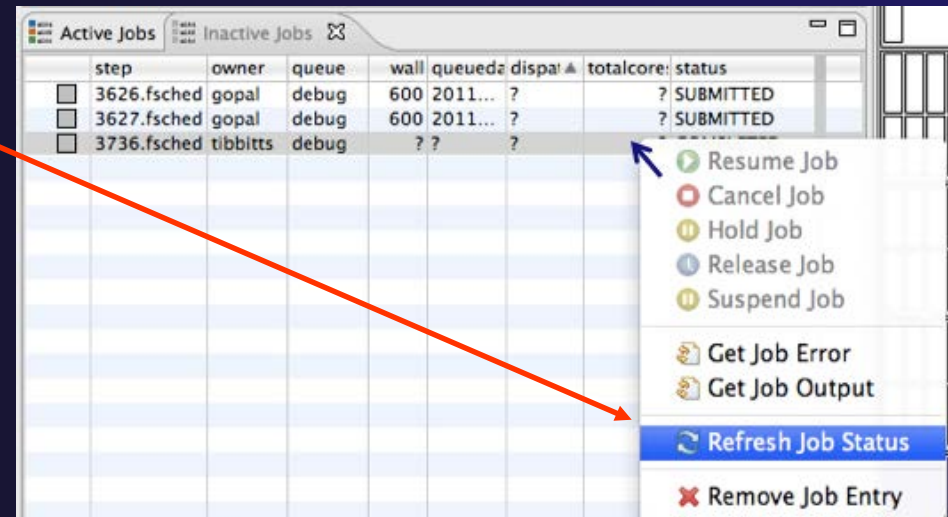
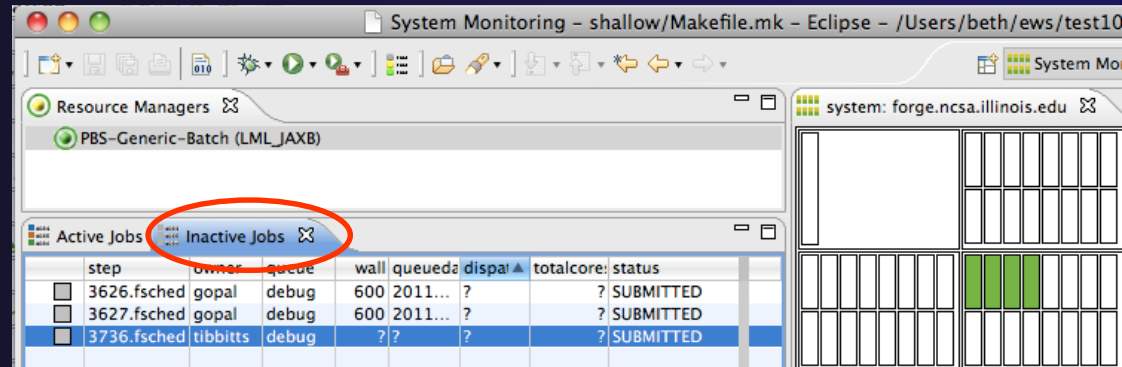
step	owner	queue	wall	queuec	dispatc	totalco	status
495...	rarijit	normal	172...	201...	201...	12	RUNNING
500...	rarijit	eight	50400	201...	201...	6	RUNNING
500...	rarijit	eight	43200	201...	201...	6	RUNNING
500...	rarijit	eight	43200	201...	201...	6	RUNNING
500...	rarijit	eight	43200	201...	201...	6	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
- Inactive Jobs:** A table listing jobs with columns: step, owner, queue, wall, queuec, dispatc, totalco, status.

step	owner	queue	wall	queuec	dispatc	totalco	status
390...	sgot...	normal	3300	201...	?	?	SUBMITTED
500...	rarijit	eight	86400	201...	?	6	SUBMITTED
500...	rarijit	eight	82800	201...	?	6	SUBMITTED
501...	alberto	eight	172...	201...	?	16	SUBMITTED
501...	rarijit	eight	79200	201...	?	6	SUBMITTED
501...	rarijit	eight	79200	201...	?	6	SUBMITTED
501...	rarijit	eight	79200	201...	?	6	SUBMITTED
501...	rarijit	eight	43200	201...	?	6	SUBMITTED
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
501...	rarijit	eight	79200	201...	?	6	SUBMITTED
501...	rarijit	eight	79200	201...	?	6	SUBMITTED
- System View:** A grid of nodes with colored bars representing running jobs. A red box highlights a node with 16 cores.

One node with
16 cores

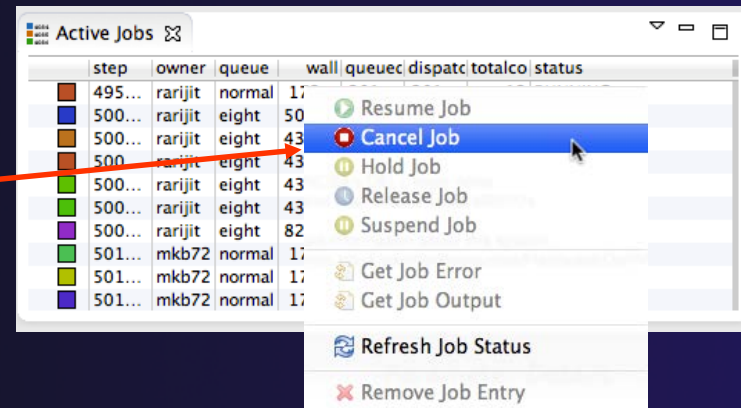
Job Monitoring

- ✦ 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

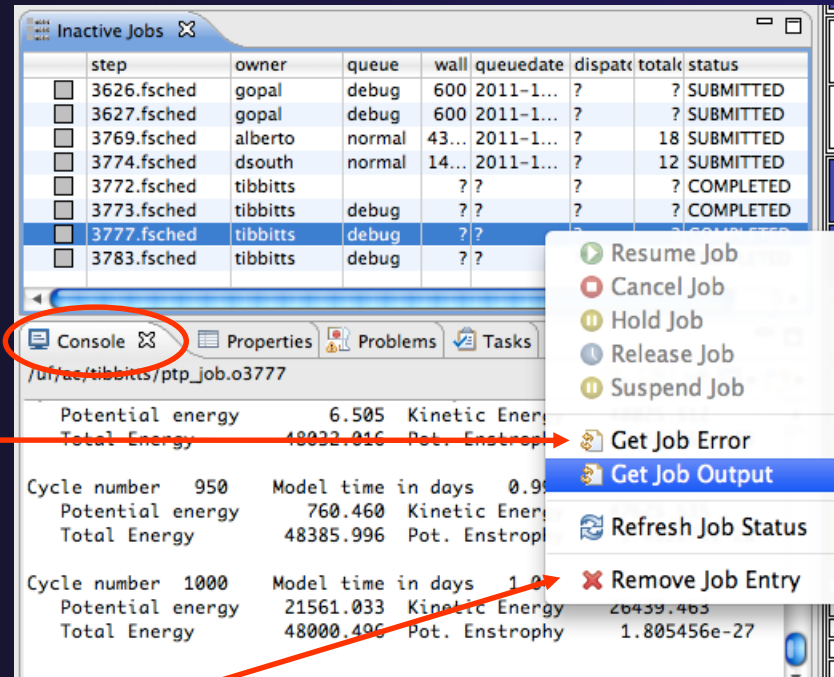


The screenshot shows the 'Inactive Jobs' window with a table of jobs. The last row shows a job with status 'COMPLETED'.

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

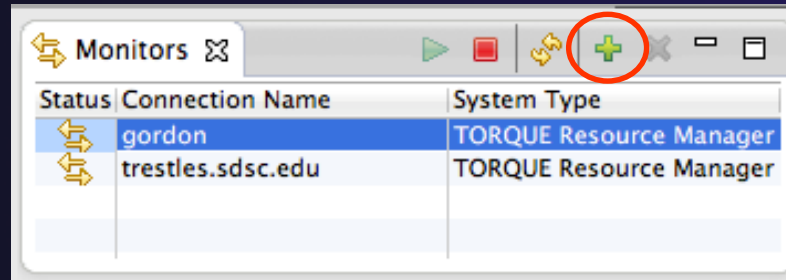
Obtaining Job Output

- ★ 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**

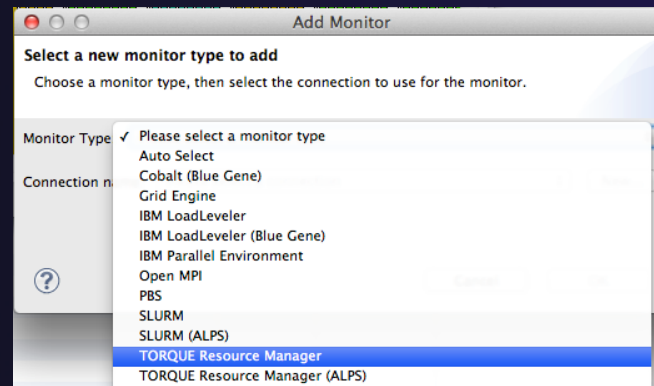


Add a Monitor

- ★ 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

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.apтана.com/products/radrails>
- ✦ Perl
 - ✦ <http://www.epic-ide.org>
- ✦ VI bindings
 - ✦ Wrapper (open source) - <http://wrapper.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

PTP Tutorial Wrap-Up

★ Your feedback is valuable!

Thanks for attending
We hope you found it useful