parallel tools platform

http://eclipse.org/ptp

Eclipse and the Parallel Tools Platform

Jay Alameda, NCSA
alameda@illinois.edu

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

Beth Tibbitts, NCSA
beth@tibweb.com

April 10-11, 2014

Portions of this material are supported by or based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under its Agreement No. HR0011-07-9-0002, the United States Department of Energy under Contract No. DE-FG02-06ER25752 and 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 – Day 1

Time (Tentative!)	Module	Topics	Presenter
8:30-10:00	1. Installation & Overview 2. Eclipse Basics	✦ Installation & Overview of Eclipse and PTP, Survey student expectations / Adjust agenda ✦ Eclipse architecture & organization overview ✦ Creating a synchronized C project from CVS	Beth/Jay
10:00-10:15	BREAK		
10:15-12:15	3. Editor features, MPI	✦ Editor features; MPI Features	Beth
	4. Build/run	✦ Building w/Makefile ✦ Target configurations and launching a parallel app ✦ Including Modules (Build Environment Mgmt)	Jay
12:15 – 1:30	Lunch		
1:30-3:00	5. Fortran 6. Adv. Features 7. NCSA features	✦ Fortran PTP/Eclipse projects, editor, features ✦ Searching, Refactoring, NCSA features	Jay
	8. Parallel Debugging	✦ Debugging an MPI Program	Beth
3:00-3:15	BREAK		
3:15-4:45	9/10. Performance Tuning & Analysis Tools	✦ TAU, External Tools FrameWork	Wyatt
	12. gprof/gcov	✦ Linux tools: Gprof/Gcov	Jay/Wyatt

Tutorial Outline – Day 2

Time (Tentative!)	Module	Topics	Presenter
8:30-10:00	9/10. Perf Tools con't OTHER TOPICS	✦ Continue… ✦ Other topics, student requests, etc.	Wyatt/Jay/Beth
10:00-10:15	BREAK		
10:15-12:15	20. Other Tools/Wrapup	✦ PTP Other tools, website, mailing lists, etc ✦ Upcoming features	Beth
12:15 – 1:30	Lunch		

Final Slides, Installation Instructions

- Please go to http://wiki.eclipse.org/PTP/tutorials/SEA2014 for slides and installation instructions

Installation

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

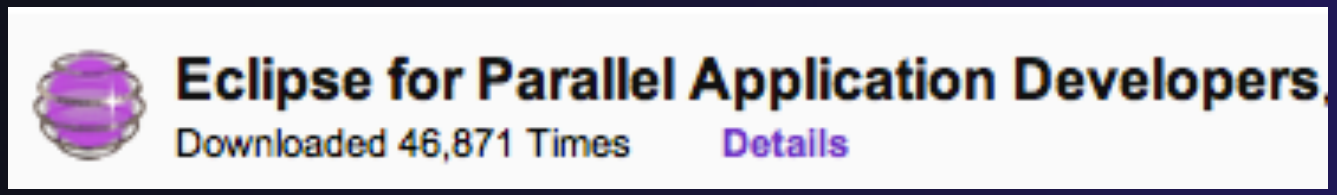
System Prerequisites

- Local system (running Eclipse)
  - Linux (just about any version)
  - MacOSX (10.5 Leopard or higher)
  - Windows (XP on)
- Java: Eclipse requires Sun or IBM Java
  - Only need Java runtime environment (JRE)
  - Java 1.6 or higher
    - Java 1.6 is the same as JRE 6.0
  - 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 (4.3) is also known as "Kepler"
- Eclipse is available in a number of different packages for different kinds of development
  - http://eclipse.org/downloads
- For PTP, we recommend the all-in-one download:
  - Eclipse for Parallel Application Developers

New! See next slide for update



We often call this the "Parallel Package"

New! Parallel Package updated

- The public Parallel Package on eclipse.org/downloads is only updated three times yearly
- We are now building updated all-in-one packages with new releases of PTP already installed.
  - 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

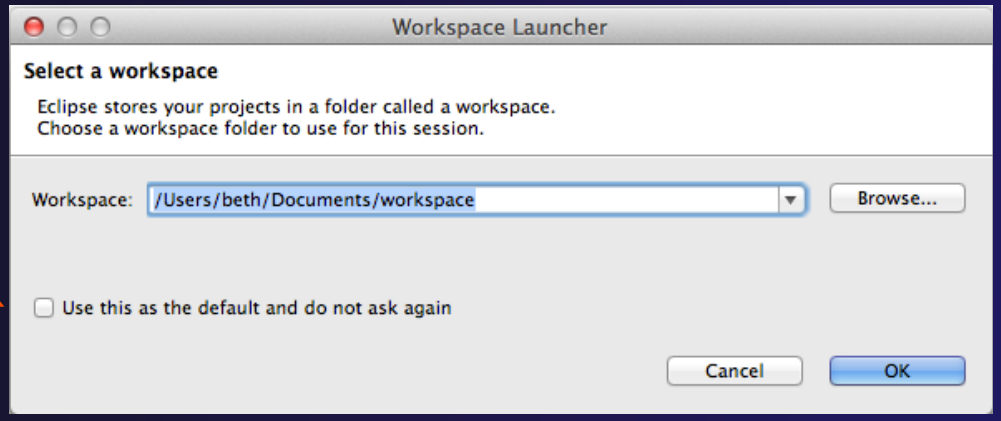
Starting Eclipse

- **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

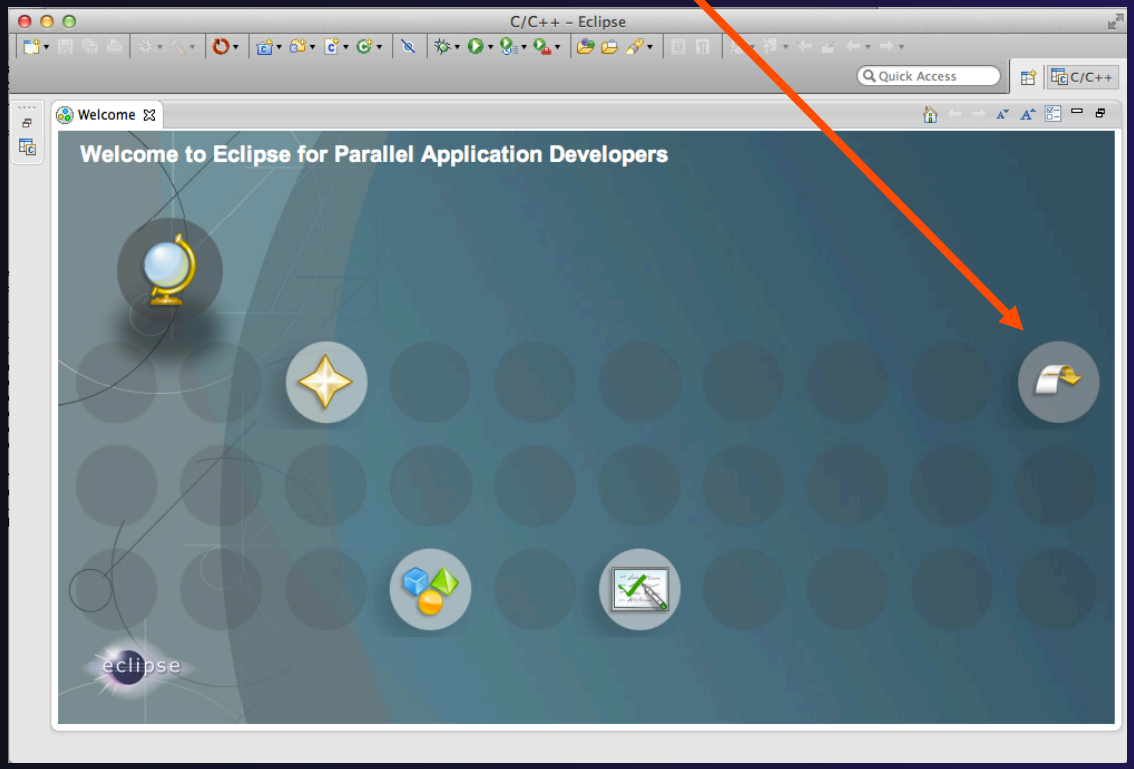
The prompt can be turned off



Eclipse Welcome Page

- Displayed when Eclipse is run for the first time

Select "Go to the workbench"

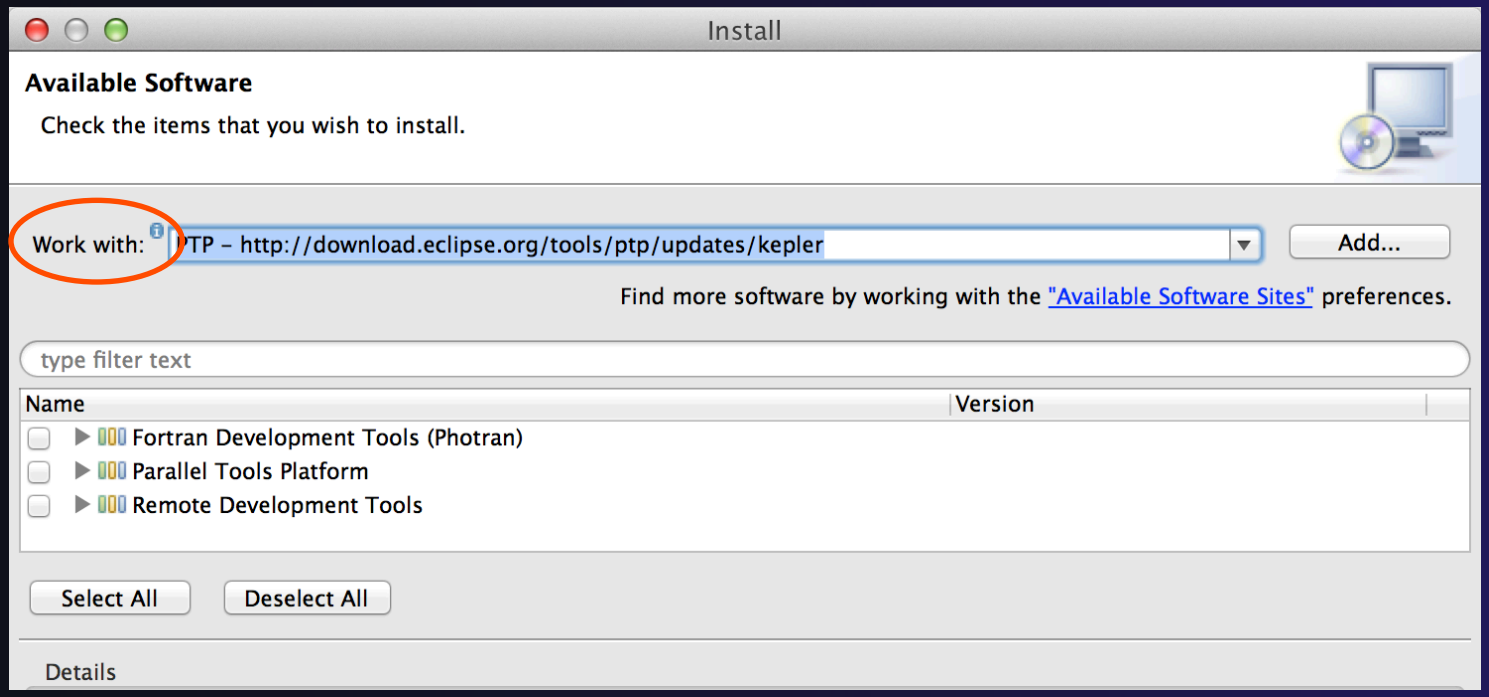


Checking for PTP Updates

- From time-to-time there may be newer PTP releases than the Kepler release
  - Kepler 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. Kepler), and what is within the parallel package
- You must enable (and install from) the PTP-specific update site before the updates will be found

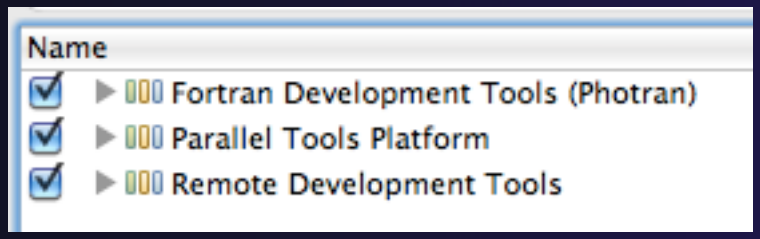
Updating PTP

- 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/kepler



Updating PTP (2)

- Easiest option is to check everything - which updates existing features and adds a few more

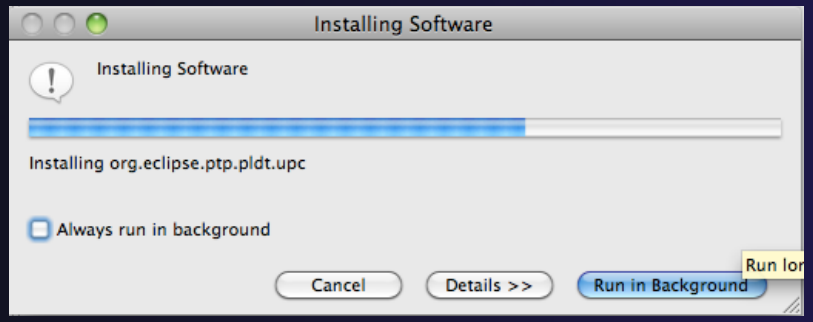


Note: for this tutorial, this installs extra features we'll refer to later anyway (GEM, TAU)

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

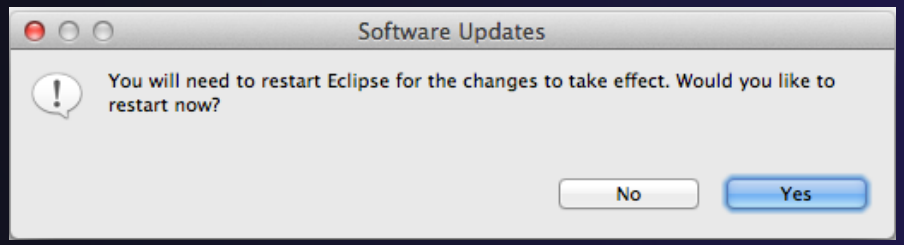
Updating PTP (3)

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



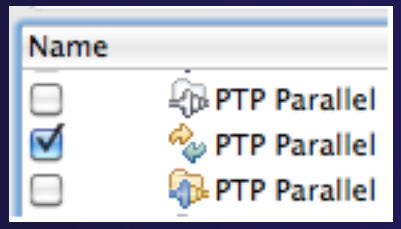
Updating PTP - restart

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

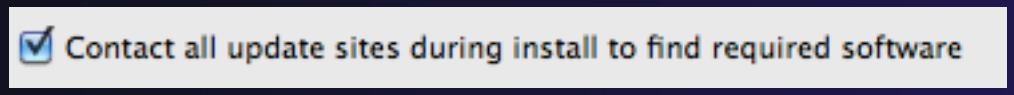


Updating Individual Features

- It's also possible (but a bit tedious) to update 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

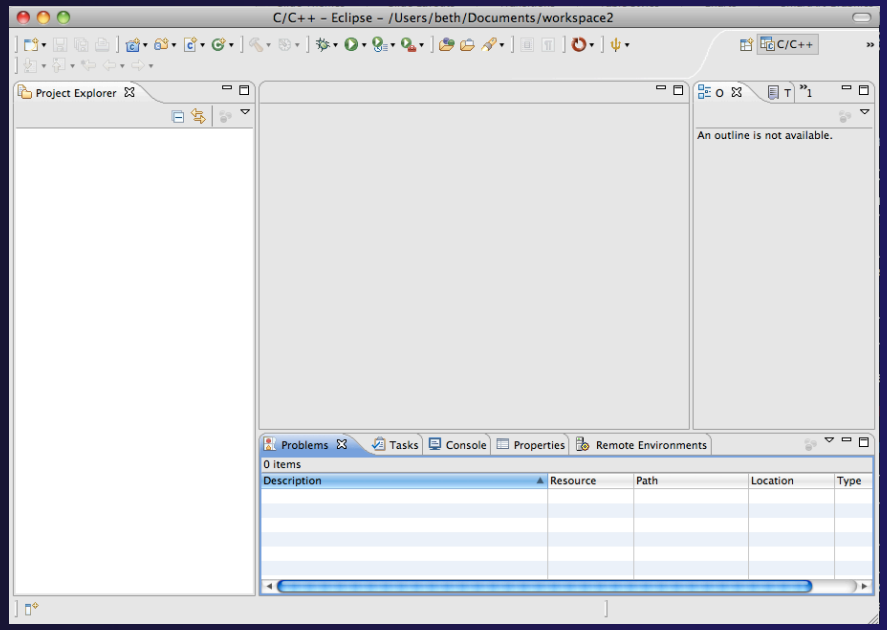


- Note: if network is slow, consider unchecking:



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

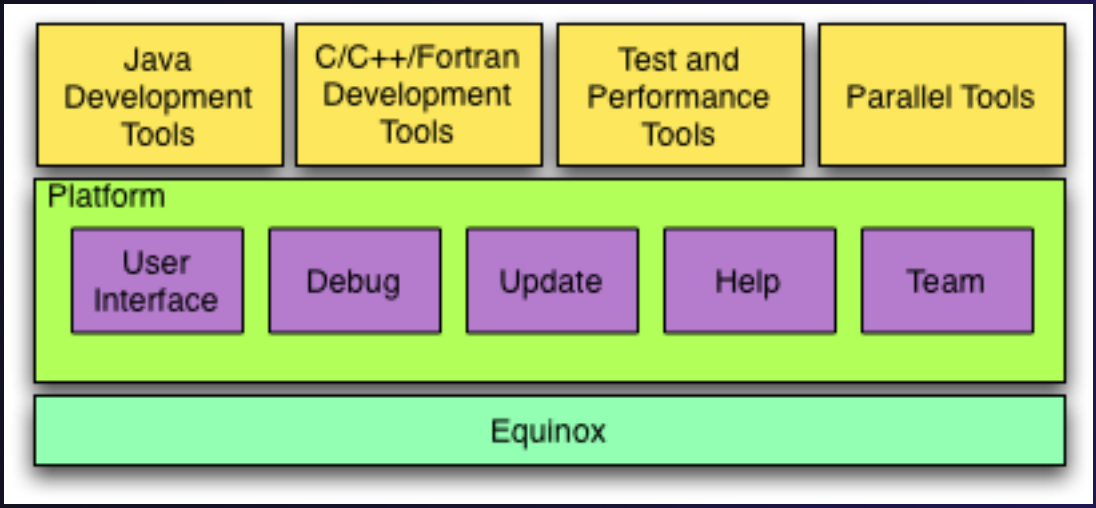
1. Launch Eclipse and select the default workspace
2. Configure Eclipse to check for PTP updates
3. Update all PTP features to the latest level
4. Install the optional features of PTP, including TAU and GEM
   - Selecting *all* features accomplishes 3. and 4.
5. Restart Eclipse once the installation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

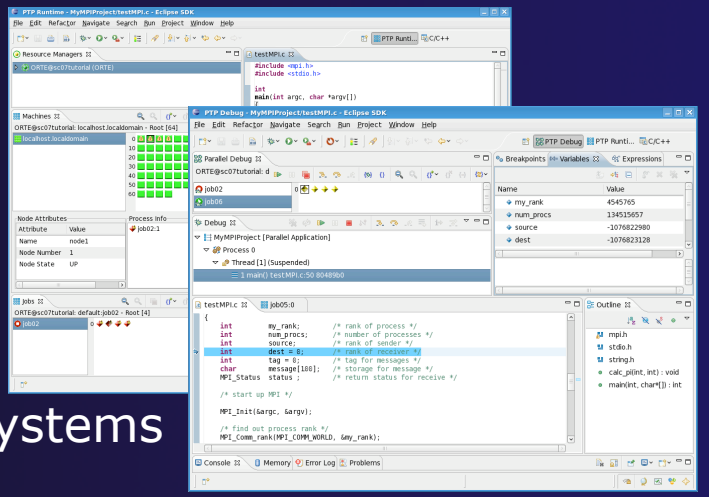


Eclipse Features

- 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

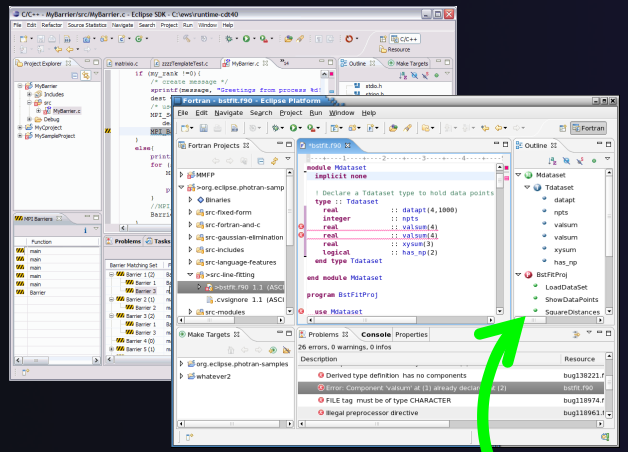
Parallel Tools Platform (PTP)

- 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

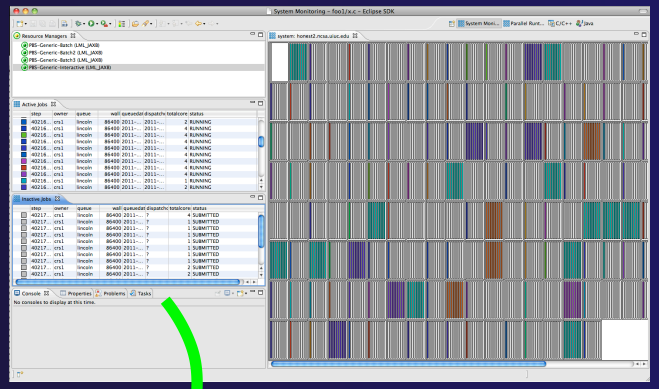


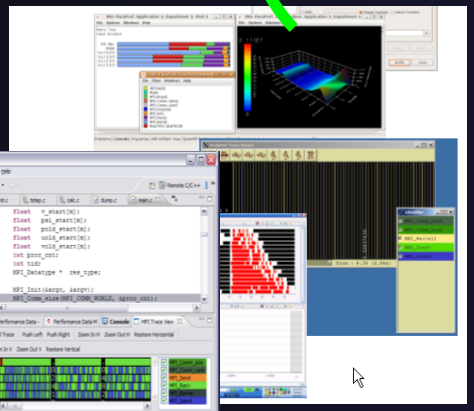
Eclipse PTP Family of Tools

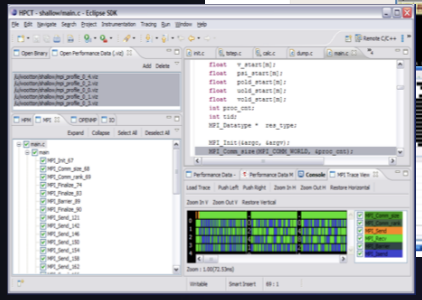
Coding & Analysis
(C, C++, Fortran)



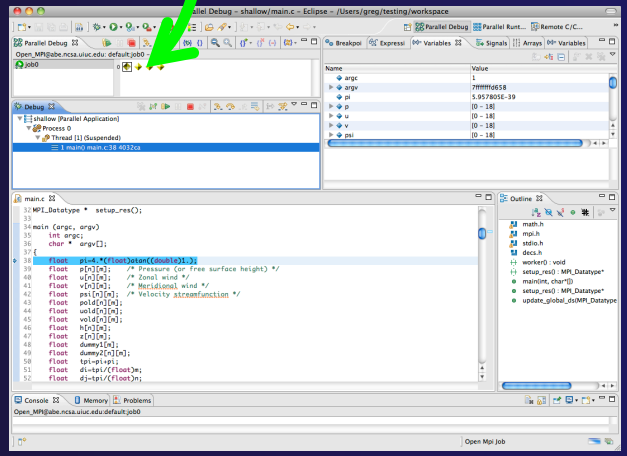
Launching & Monitoring





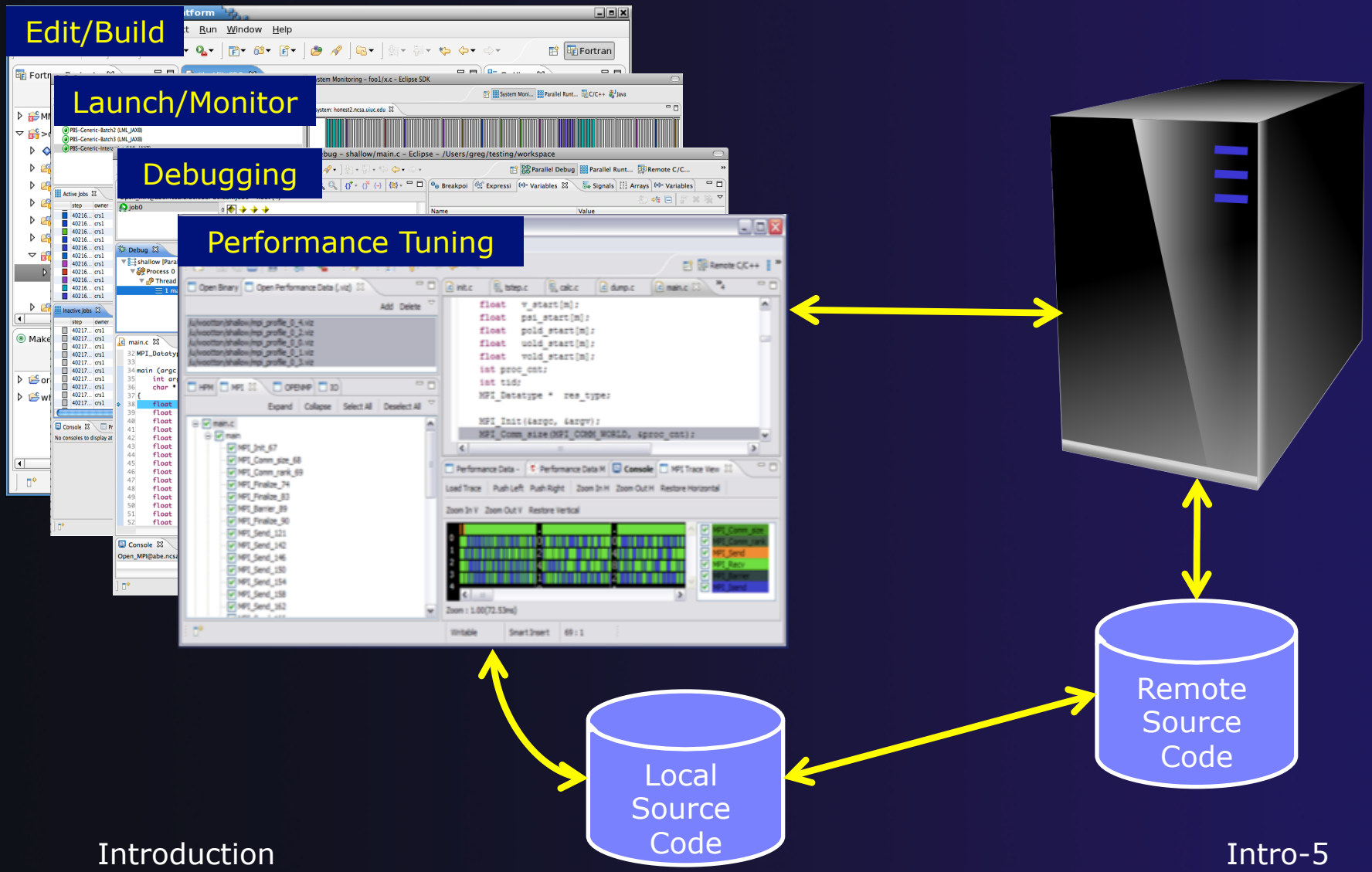


Performance Tuning
(TAU, PPW, …)



Parallel Debugging

How Eclipse is Used

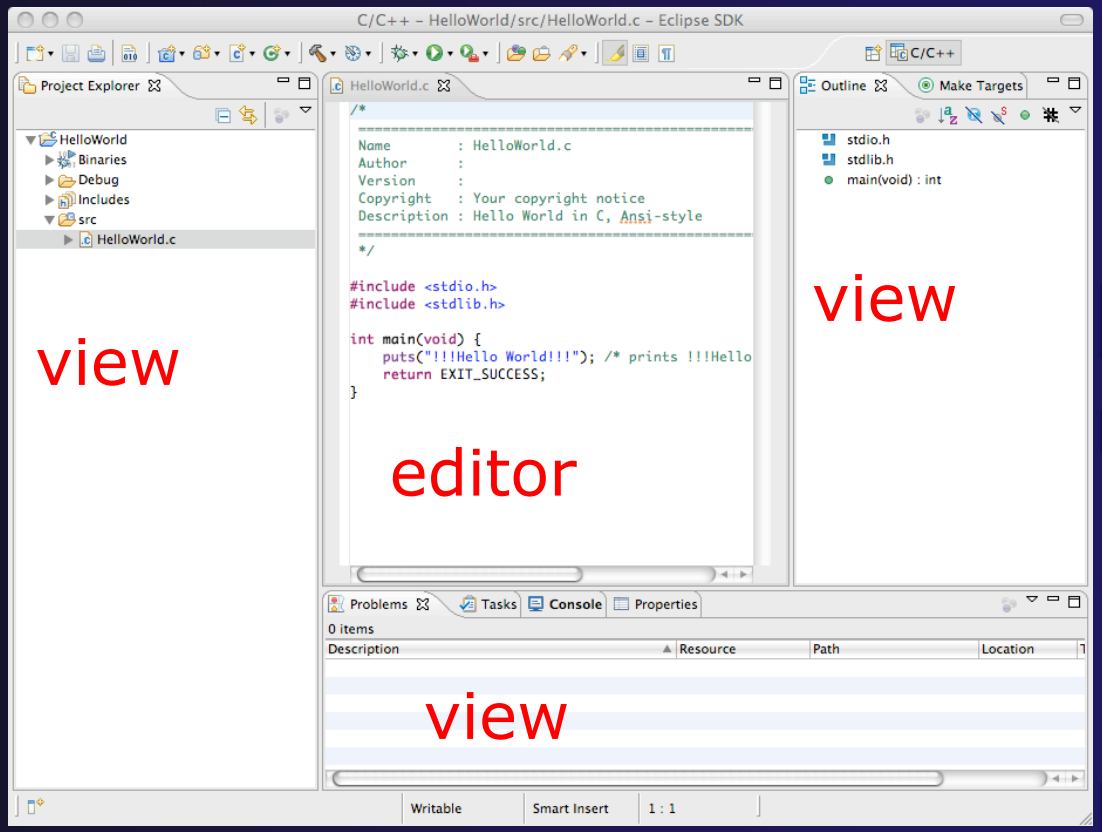


Eclipse Basics

- 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

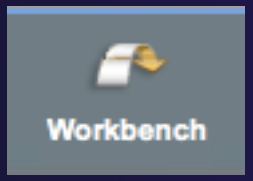
Eclipse Basics

- 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



Perspectives

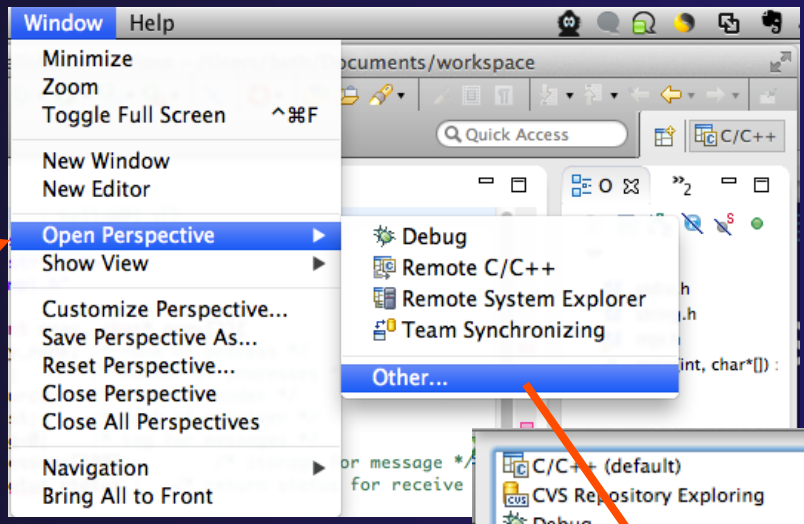
- Perspectives define the layout of views and editors in the workbench
- They are *task oriented*, i.e. they contain specific views for doing certain tasks:
  - **C/C++ Perspective** for manipulating compiled code
  - **Debug Perspective** for debugging applications
  - **System Monitoring Perspective**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

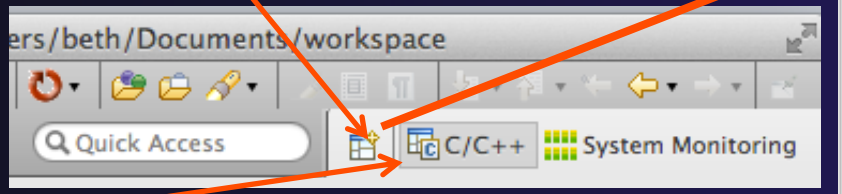


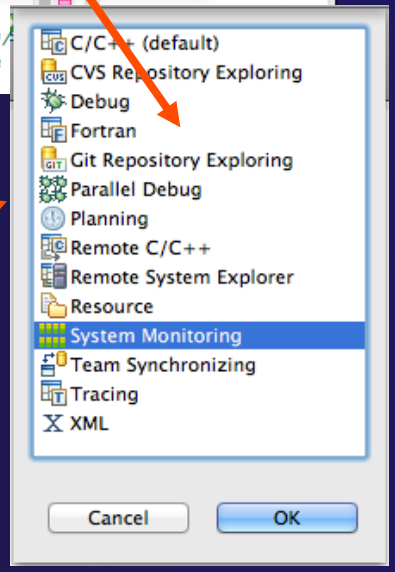
Switching Perspectives

- 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

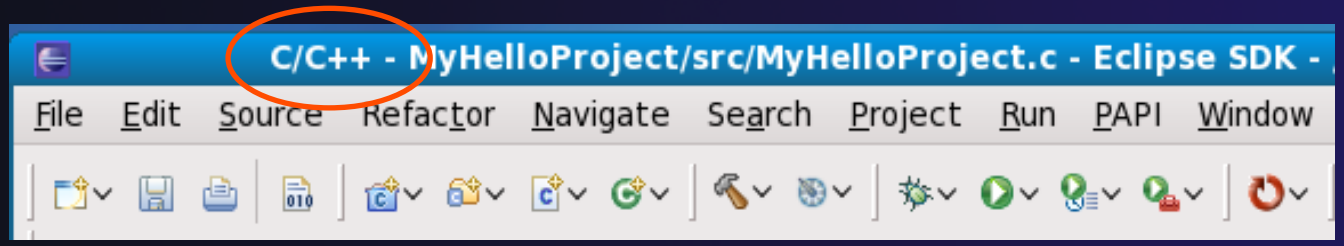




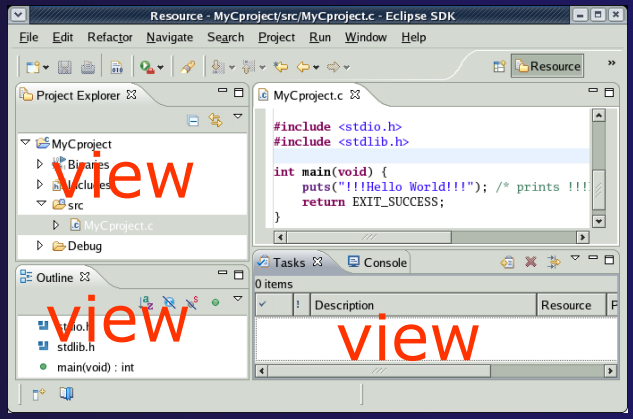


Which Perspective?

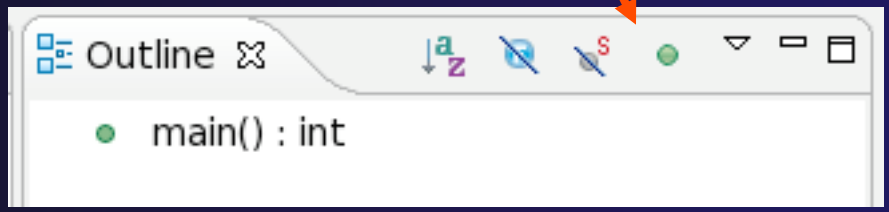
- The current perspective is displayed in the title bar



Views

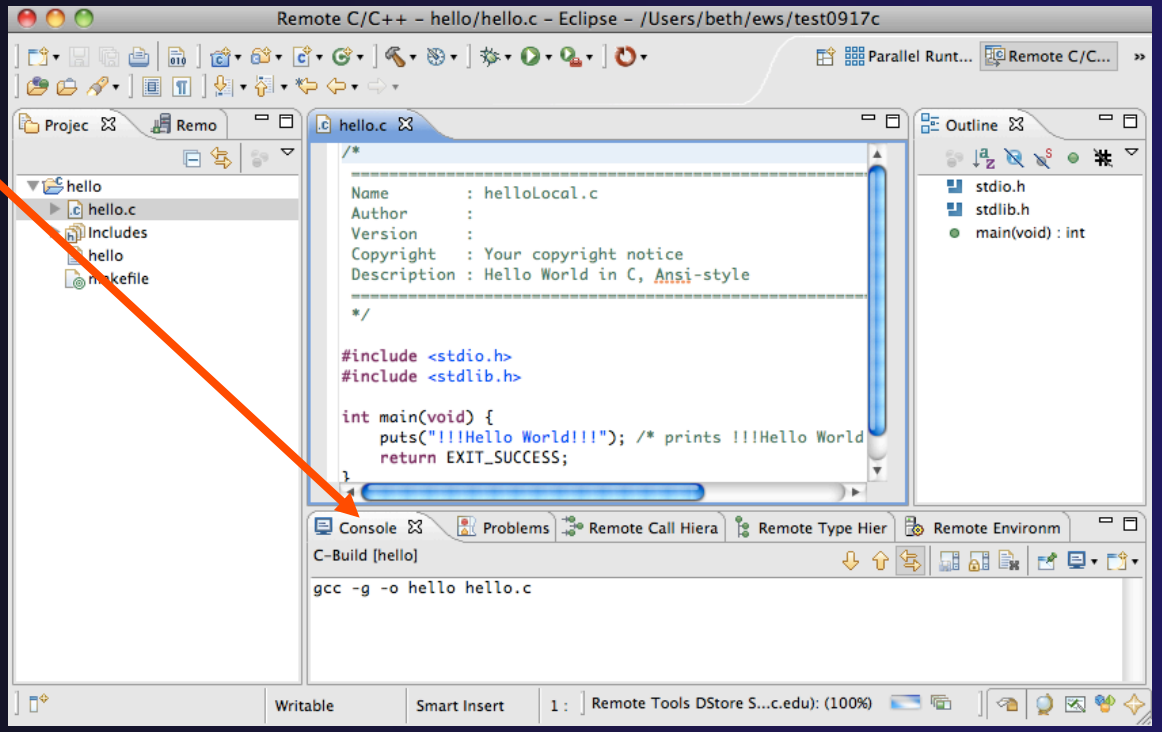


- The workbench window is divided up into Views
- The main purpose of a view is:
  - To provide alternative ways of presenting information
  - For navigation
  - For editing and modifying information
- Views can have their own menus and toolbars
  - Items available in menus and toolbars are available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



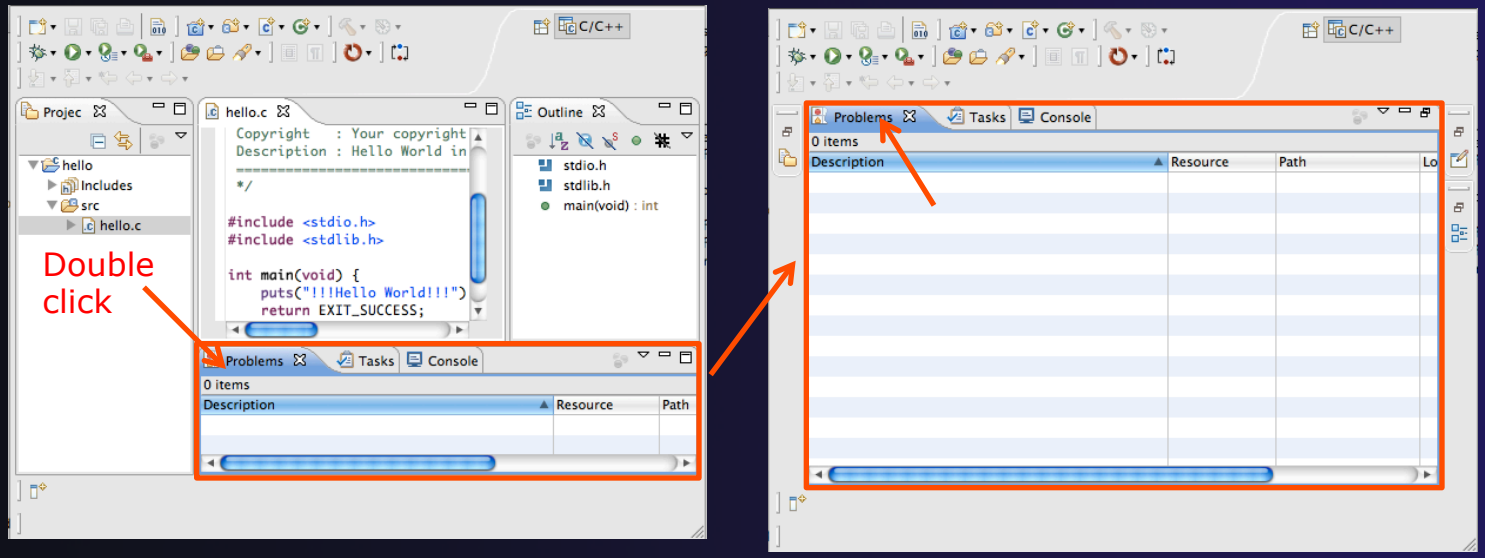
Stacked Views

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



Expand a View

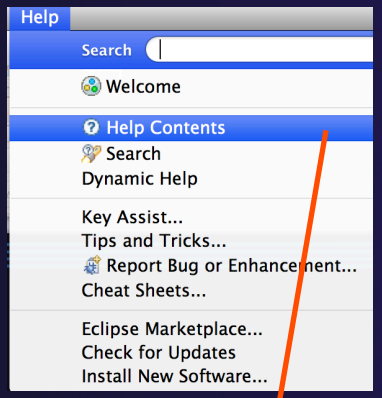
- Double-click on a view/editor's tab to fill the workbench with its content;
- Repeat to return to original size

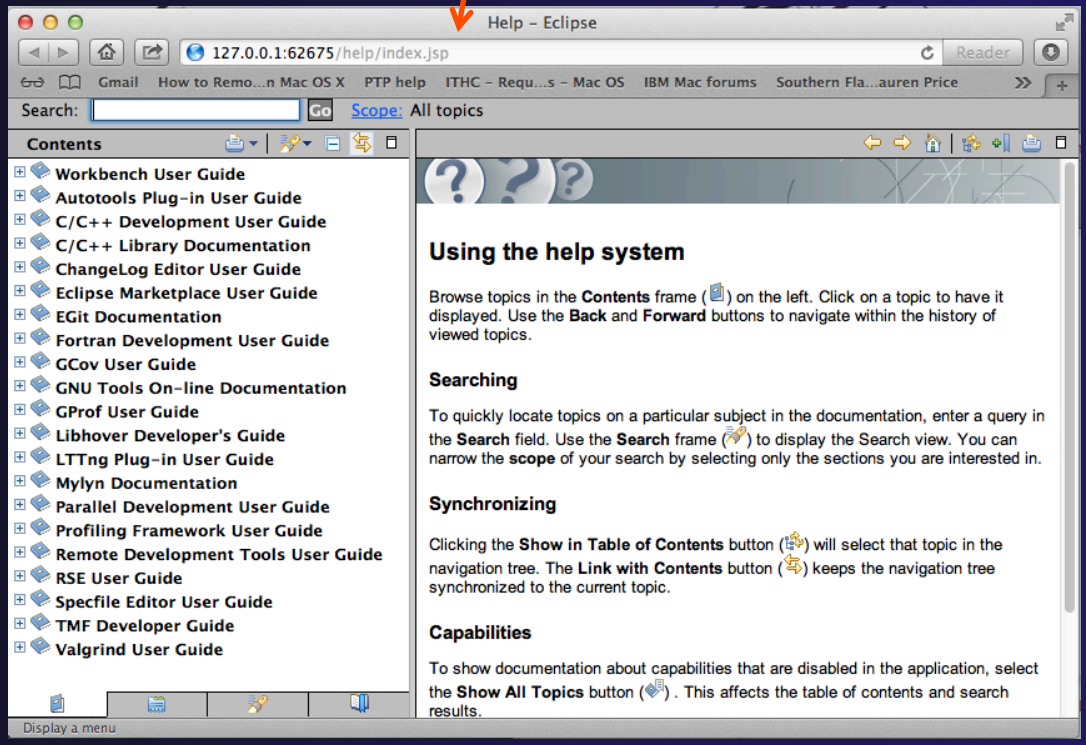


- Window > Reset Perspective returns everything to original positions

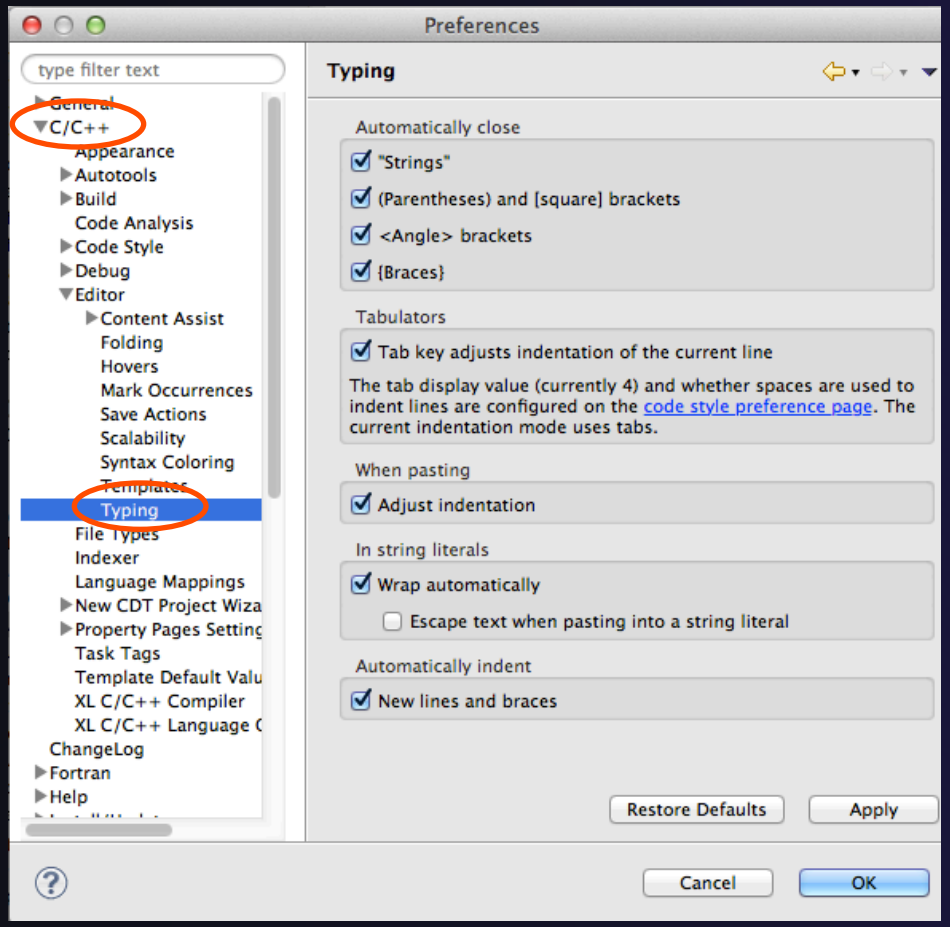
Help

- 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.)



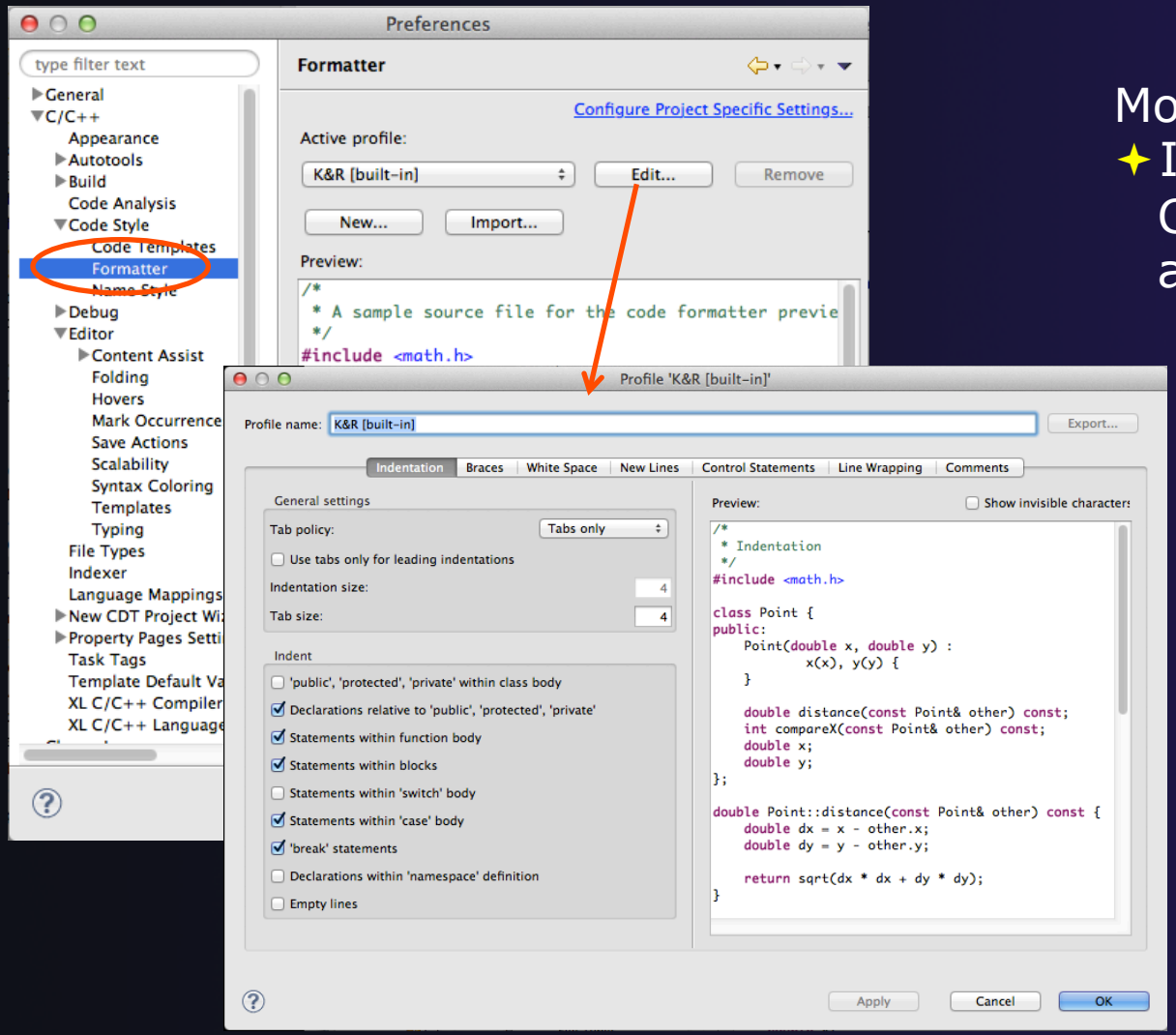


Eclipse Preferences



- 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

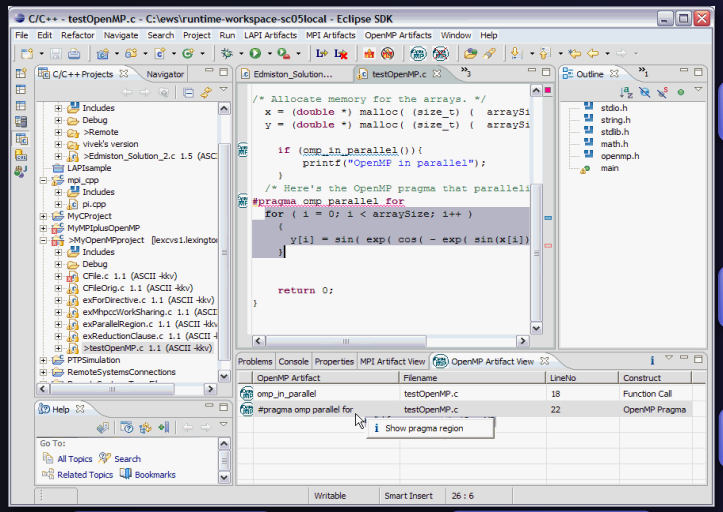
Creating a Synchronized Project

- Objective
  - Learn how to create and use synchronized projects
  - Learn how to create a sync project
    - From a source code repository in CVS, or
    - Import from source on a remote machine
- Contents
  - Eclipse project types
  - Creating a synchronized project from CVS or remote dir
  - Using synchronize filters
  - Remote Terminal view
  - Converting an existing project to synchronized

Project Location

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

Synchronized Projects



Launch Service
Run
Compute
Debug
Debug Service
Build Service
Executable
Build
File Service
Index Service
Edit
Search/Index
Navigation
Local source code
Synchronize
Source code copy
Local
Remote

Revision Control Systems (RCS)

- Eclipse supports a range of RCS, such as CVS, Git, and Subversion (and others)
- These are distinct from synchronized projects
- RCS 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

A slides

- 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

B slides

C, C++, and Fortran Projects

Build types

- Makefile-based
  - Project contains its own makefile (or makefiles) for building the application – or other build command
- Managed
  - Eclipse manages the build process, no makefile required

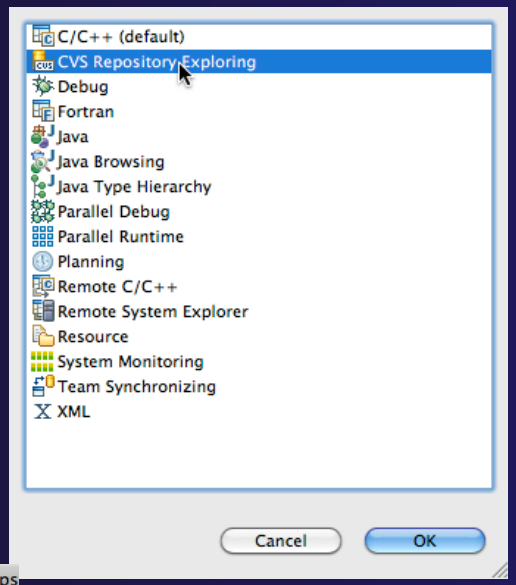
A

SCENARIO A: Check out source code from CVS repository

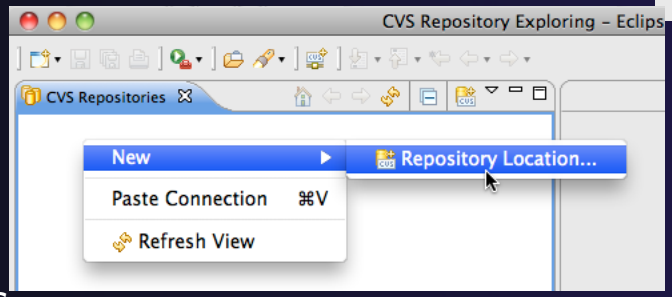
A

Importing a Project from CVS

- Switch to **CVS Repository Exploring** perspective
  - Window > Open Perspective > Other…
  - Select **CVS Repository Exploring**
  - Select **OK**



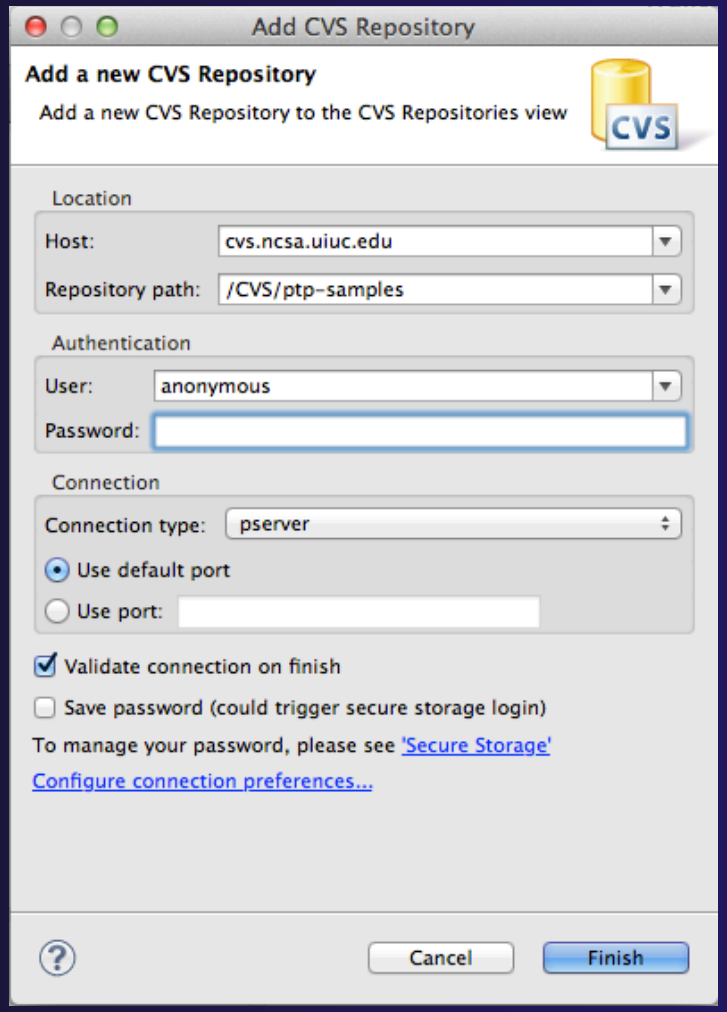
- Right click in **CVS Repositories** view and select **New>Repository Location…**



A

Add CVS Repository

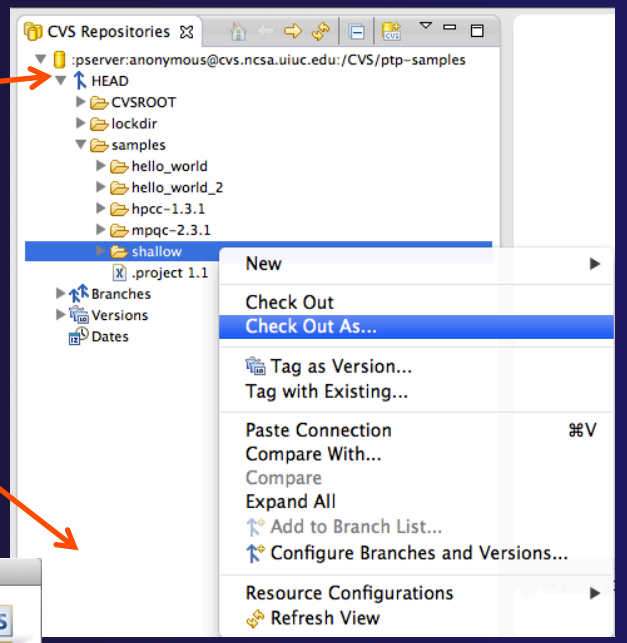
- Enter **Host:** cvs.ncsa.uiuc.edu
- **Repository path:** /CVS/ptp-samples
- For anonymous access:
  - **User**: anonymous
  - No password is required
  - **Connection type:** pserver (default)
- For authorized access:
  - **User**: your userid
  - **Password**: your password
  - **Connection type:** change to **extssh**
- Select **Finish**

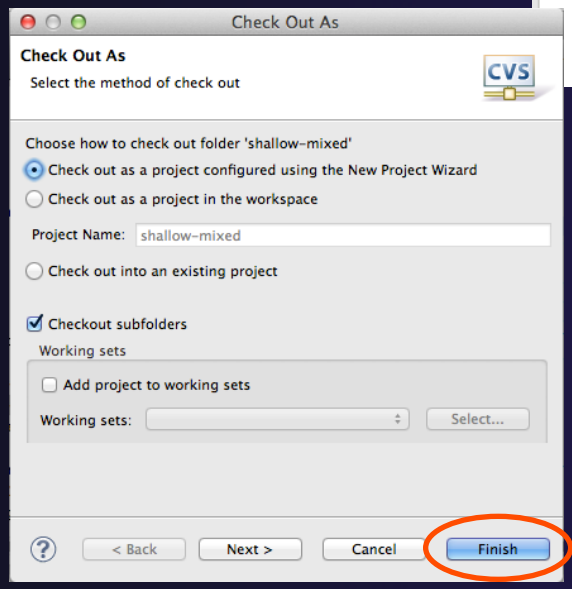


A

Checking out the Project

- Expand the repository location
- Expand **HEAD**
- Expand **samples**
- Right click on **shallow** and select **Check Out As...**
- On **Check Out As** dialog, select **Finish**





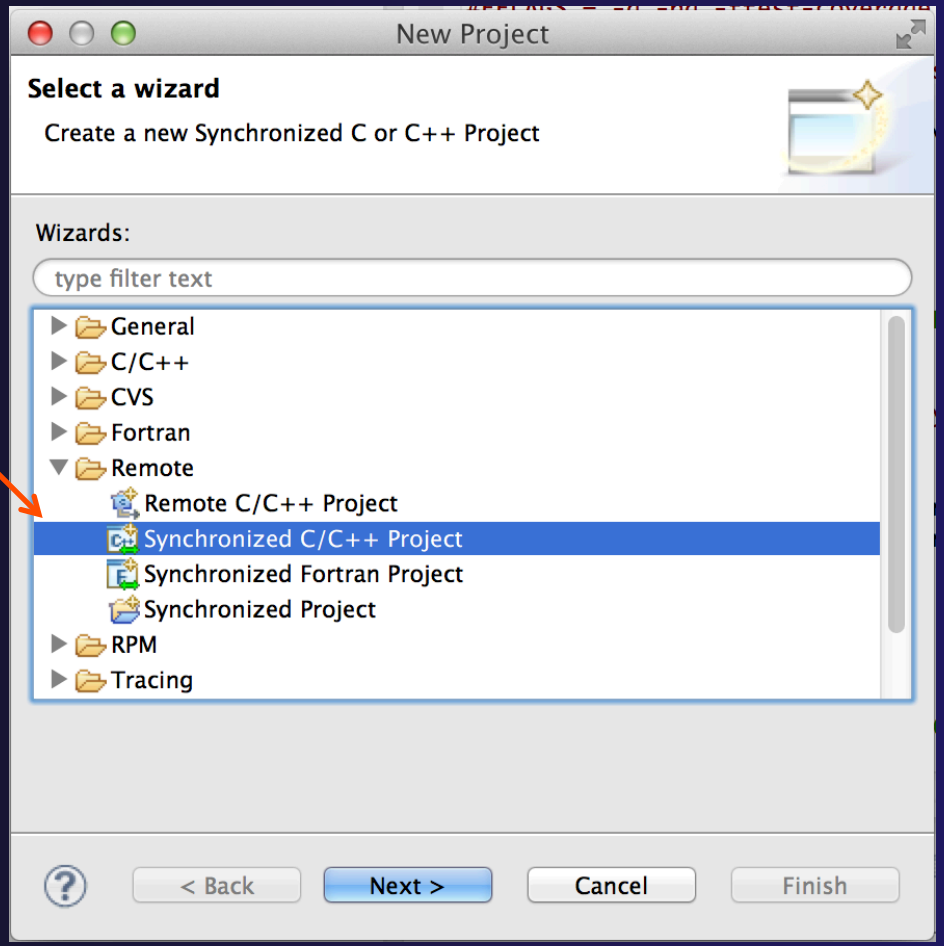
The default of "Check out as a project configured using the New Project Wizard" is what we want

A

Next: New Project Wizard

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

- If asked to select a tag, select **HEAD** and hit **FINISH**



B

SCENARIO B: Source code on remote machine

B

Source Code for project

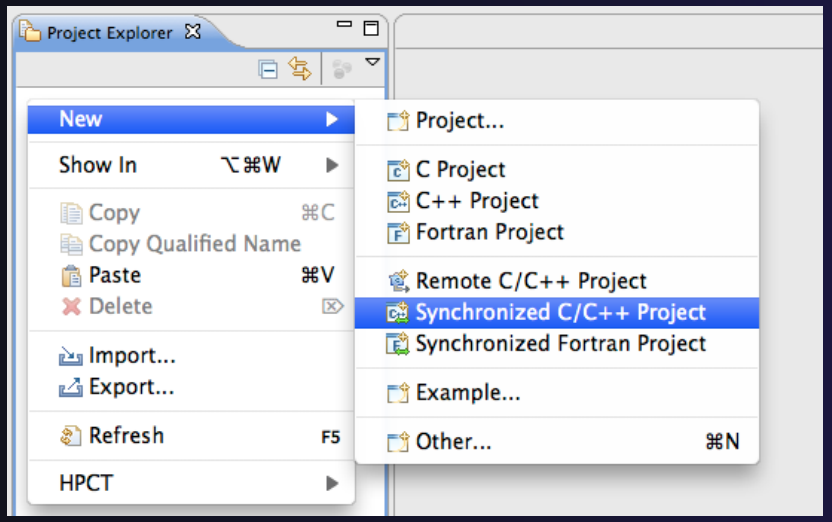
- Source code exists on remote target

```
$ pwd
/gpfs/ibmu/tibbitts/shallow
$ ls -la
total 2880
drwxr-xr-x 2 tibbitts users 32768 Mar 16 15:53 .
drwxr-xr-x 7 tibbitts users 32768 Mar 15 18:38 ..
-rw-r--r-- 1 tibbitts users  1741 Feb 11 16:25 calc.c
-rw-r--r-- 1 tibbitts users  2193 Feb 11 16:25 copy.c
-rw-r--r-- 1 tibbitts users  2873 Jan 25 08:52 decs.h
-rw-r--r-- 1 tibbitts users  2306 Feb 11 16:25 diag.c
-rw-r--r-- 1 tibbitts users  2380 Feb 11 16:25 dump.c
-rw-r--r-- 1 tibbitts users  2512 Feb 11 16:25 init.c
-rw-r--r-- 1 tibbitts users  6161 Mar 15 19:27 main.c
-rw-r--r-- 1 tibbitts users   718 Mar 15 18:34 Makefile
-rw-r--r-- 1 tibbitts users  1839 Feb 11 16:25 time.c
-rw-r--r-- 1 tibbitts users  2194 Feb 11 16:25 tstep.c
-rw-r--r-- 1 tibbitts users  8505 Feb 11 16:25 worker.c
$
```

B

Create Synchronized Project

- In the Project Explorer, right click then choose
  - **New>Synchronized C/C++ Project** if your project is C/C++ only

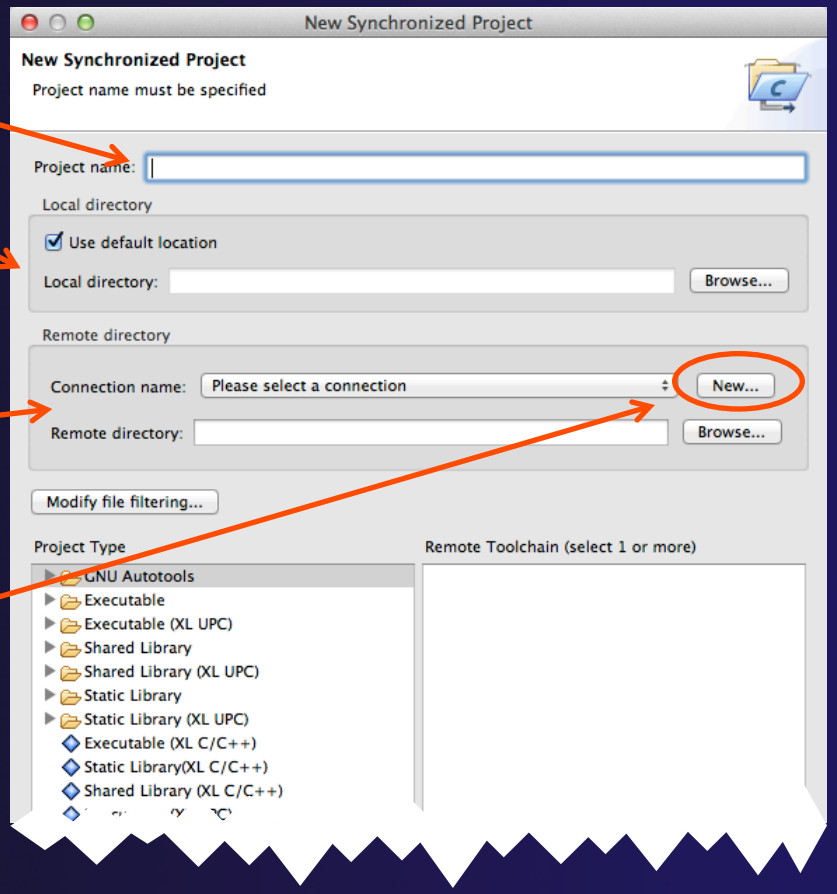


Or via menus:
File>New>Other…
And under **Remote,** choose **Synchronized C/C++ Project**

  - **New>Synchronized Fortran Project** if your project contains Fortran files
  - This adds a Fortran nature so you can access Fortran properties, etc.

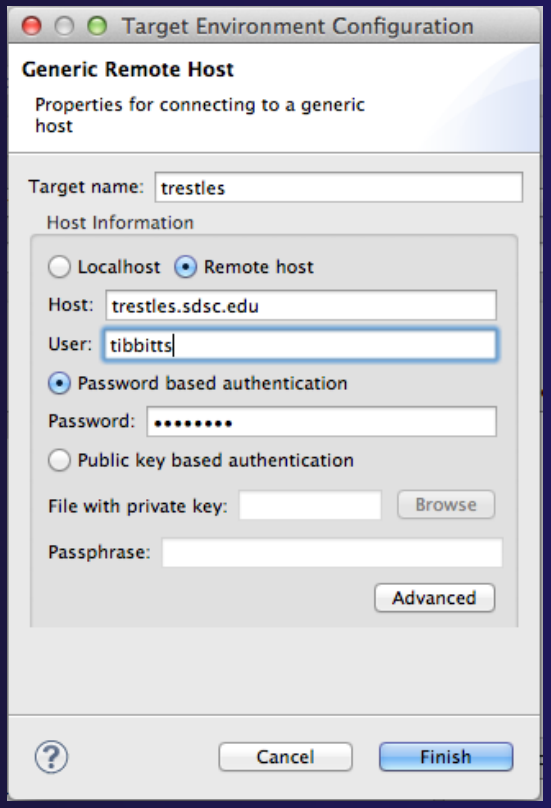
New Synchronized Project Wizard

- Enter the **Project Name**
  - E.g. “shallow”
- The **Local Directory** specifies where the local files are located
  - Leave as default
- The **Remote Directory** specifies where the remote files are located
  - Select a connection to the remote machine, or click on **New...** to create a new one
    (See next slide)
- Use **Modify File Filtering...** if required (see later slide)



Creating a Connection

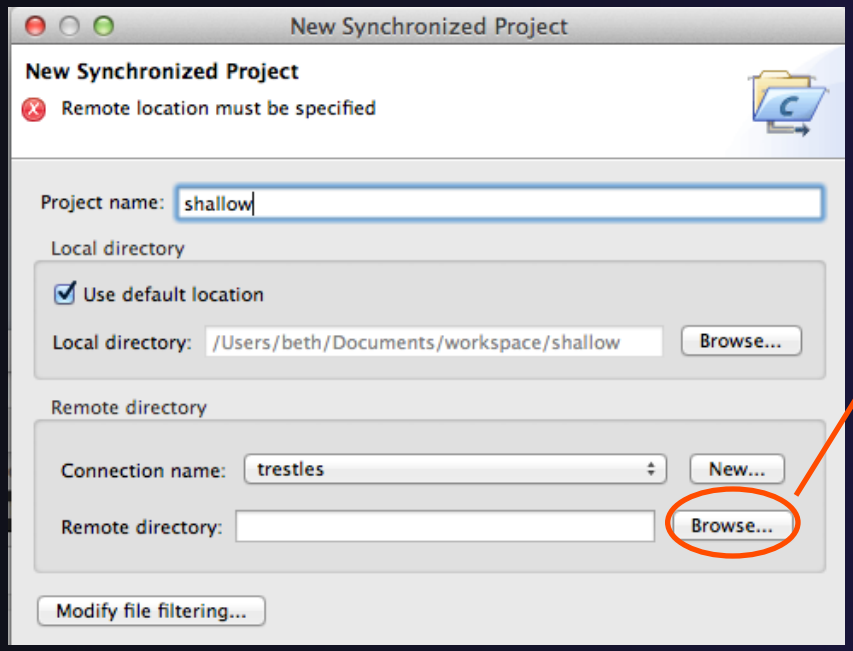
- In the **Target Environment Configuration** dialog
  - Enter a **Target name** for the remote host
  - Enter host name, user name, and user password or other credentials
  - Select **Finish**

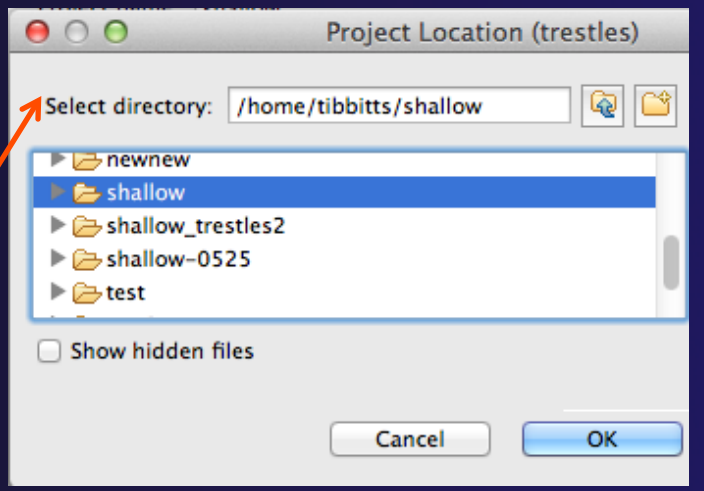


If your machine access requires ssh access through a frontend/ intermediate node, ***use localhost and port – see alternate instructions for ssh tunnel***

Remote Directory for Project

- After the connection has been specified,
  Browse for the directory on the remote machine

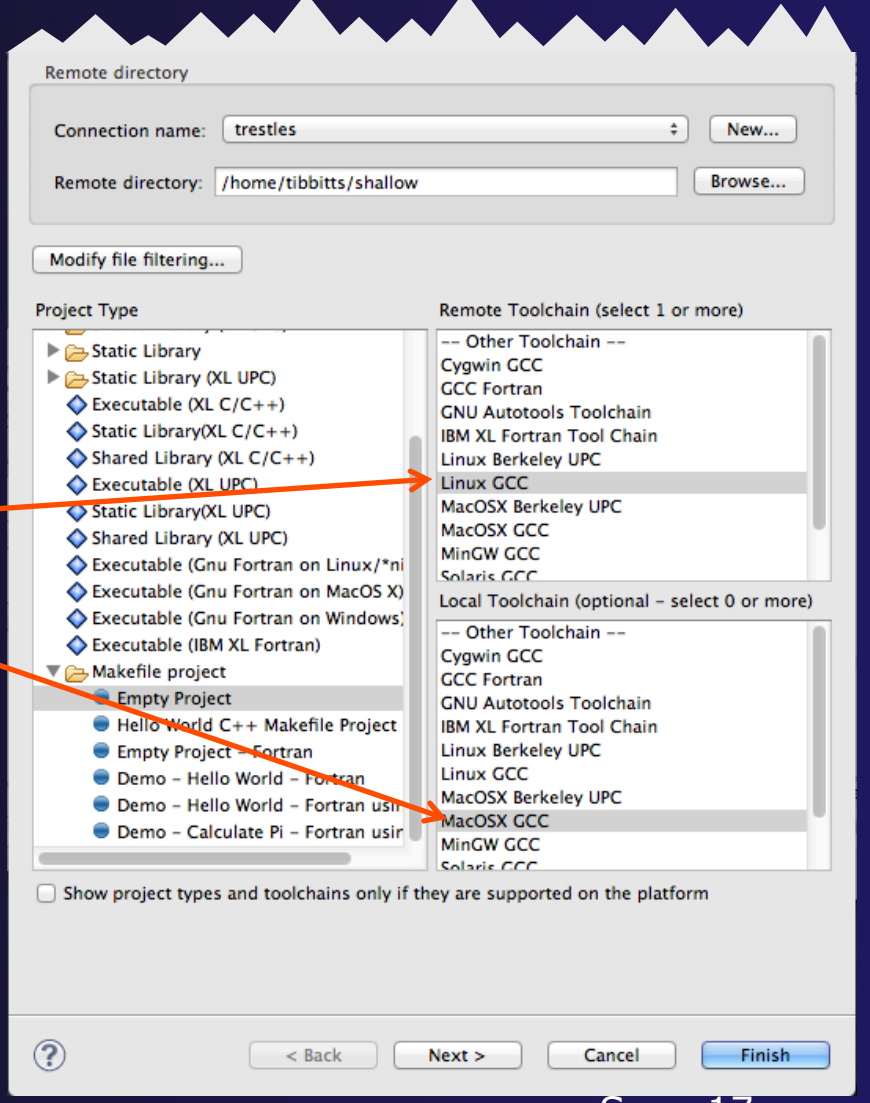




- Select directory if it exists, or enter a new directory name
- Hit **OK**

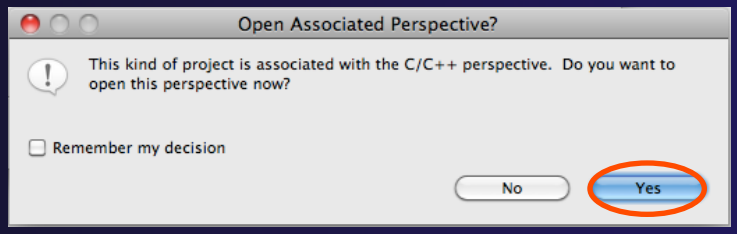
Project Type & Toolchain

- 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 the toolchain for the remote build
  - Use a toolchain that most closely matches the remote system
- Choose a toolchain for the local build
  - This is optional if you don't plan to build on the local machine
  - This is used for advanced editing/ searching
- Use **Modify File Filtering...** if required (see later slide)
- Click **Finish** to create the project

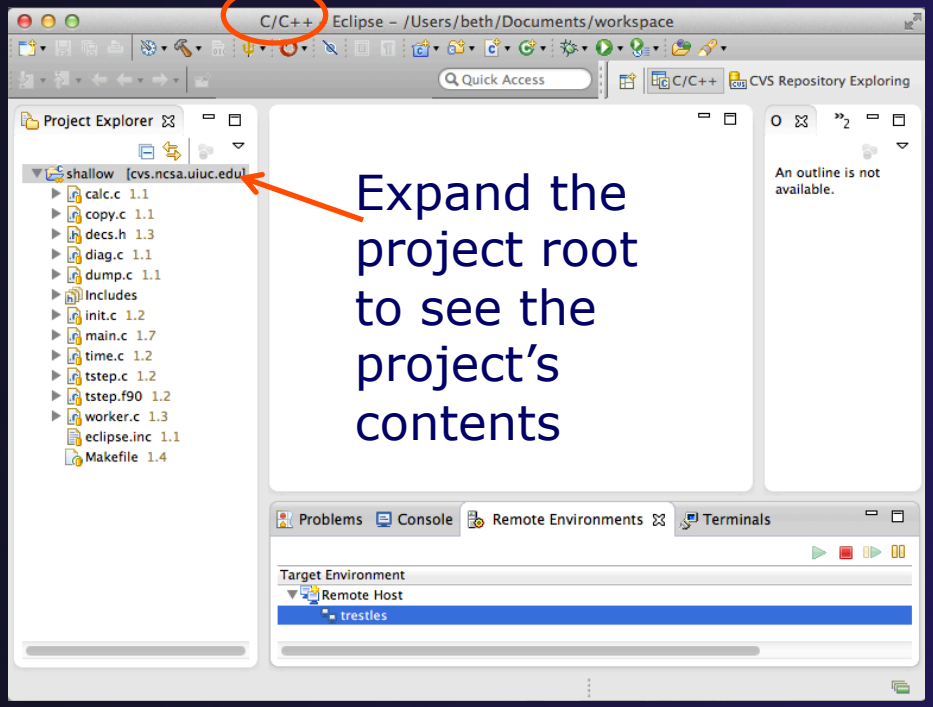


Project successfully created

- If prompted, switch to the C/C++ Perspective after creating the files

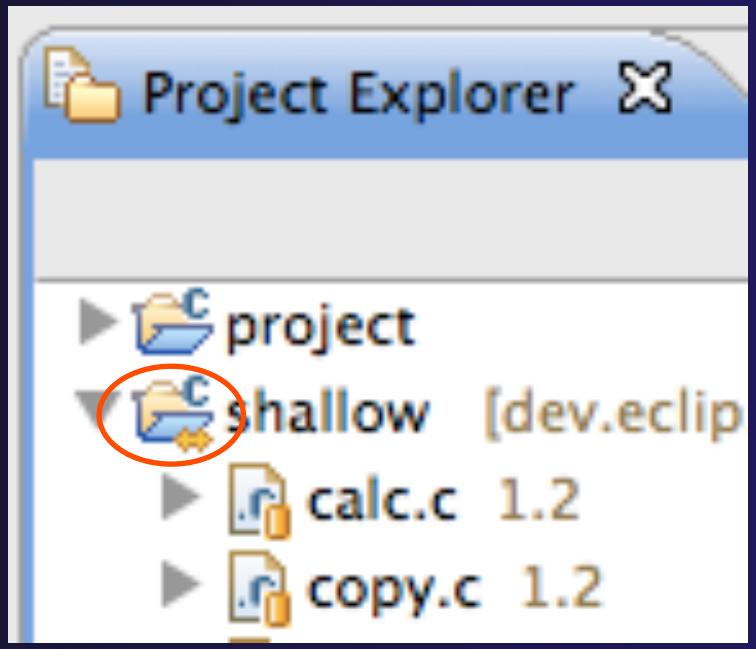


- You should now see the “shallow” project in your workspace
- Project is synchronized with remote host



Synchronized Project

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

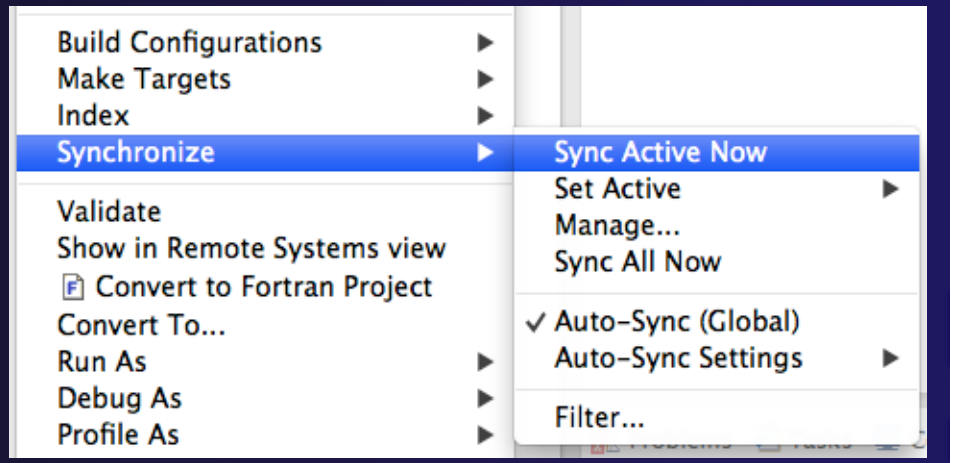


- C Project w/o Sync

- Synchronized Project

Synchronized Project Menu

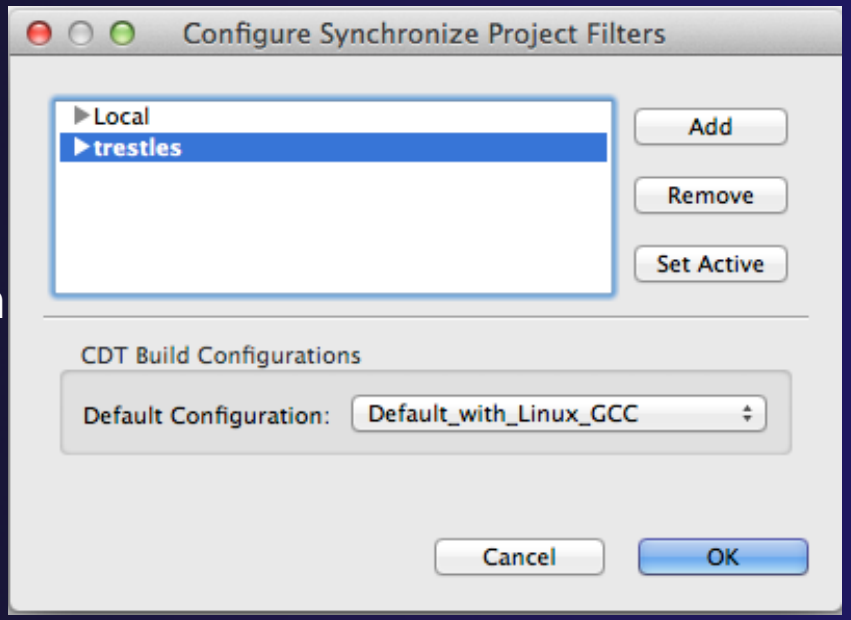
- Synchronized projects are indicated with a “synchronized” icon
- Right click on project to access **Synchronize** menu
  - **Sync Active Now** will manually synchronize the active configuration
  - **Set Active** can be used to select the active configuration
  - **Manage...** is used to create new configurations to synchronize to different target systems
  - **Sync All Now** will manually synchronizes all configurations



- **Auto-Sync (Global)** will enable or disable automatic synchronization
- **Auto-Sync Settings** can be used to select which configurations will be synchronized
- **Filter...** is used to change the filter settings for the project

Manage Configurations

- Used to manage synchronize configurations
- Use **Set Active** to change the active configuration (shown in **bold** in the list of configurations)
- Use **Add** to add a new configuration in order to synchronize to a different target system
- Other configuration information, such as the default build configuration, can also be changed



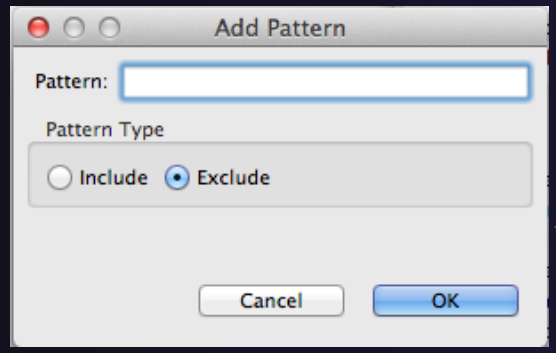
*By default, there will be a configuration for the target system (active) and a **Local** configuration. The Local configuration can be used to build a local copy of the project if desired.*

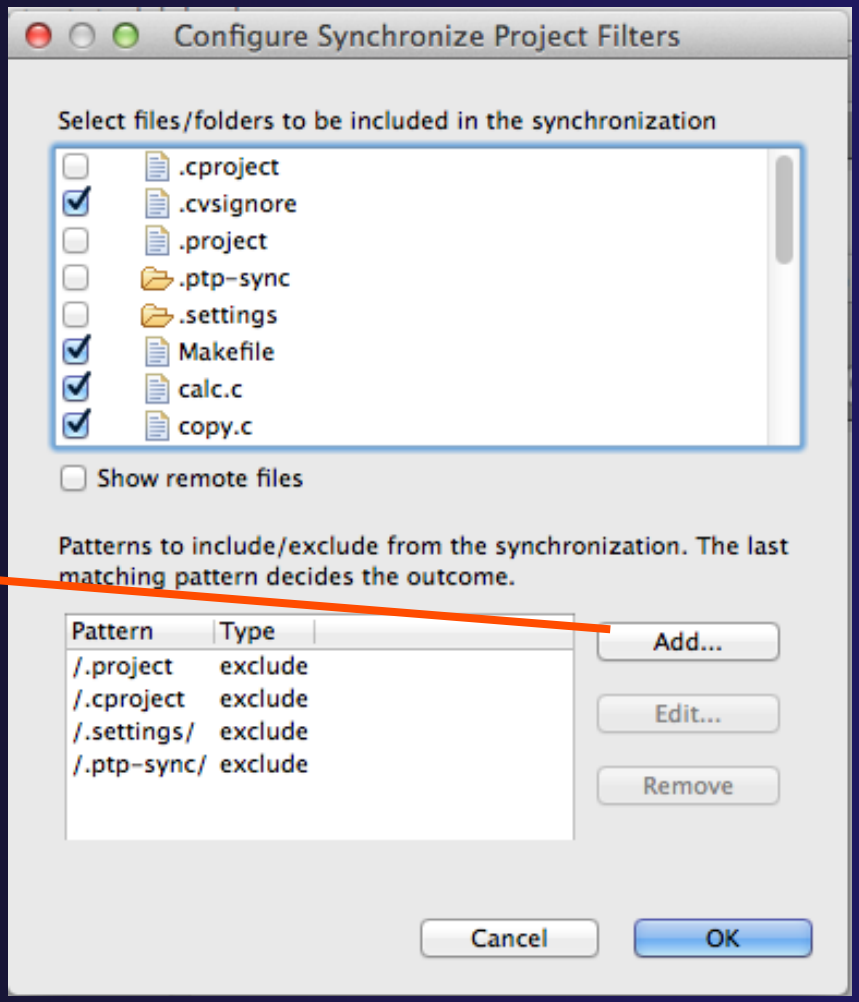
Synchronize Filters

- 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 **Synchronization>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

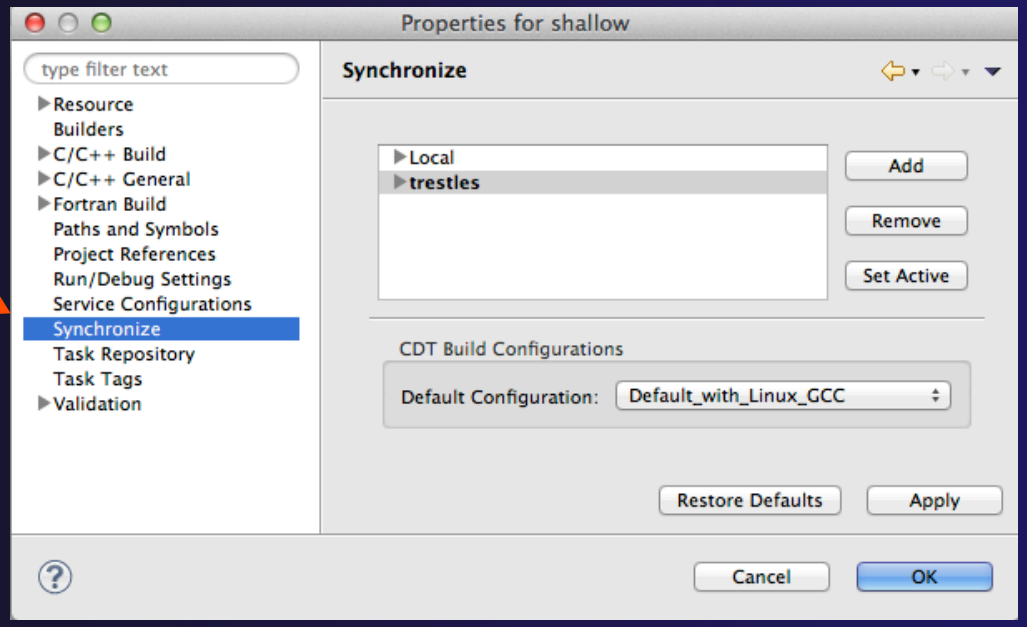




- Suggestion: add filter for 'shallow' so the executable, built on remote machine, doesn't get synced back

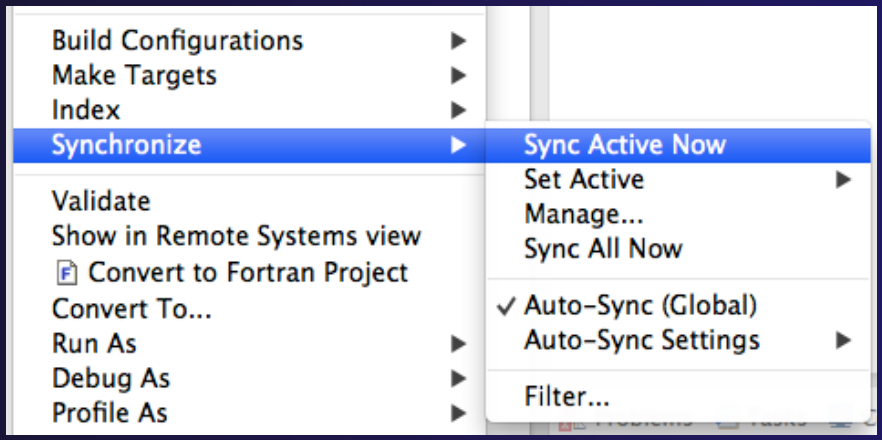
Synchronized Project Properties

- Synchronized configurations can also 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



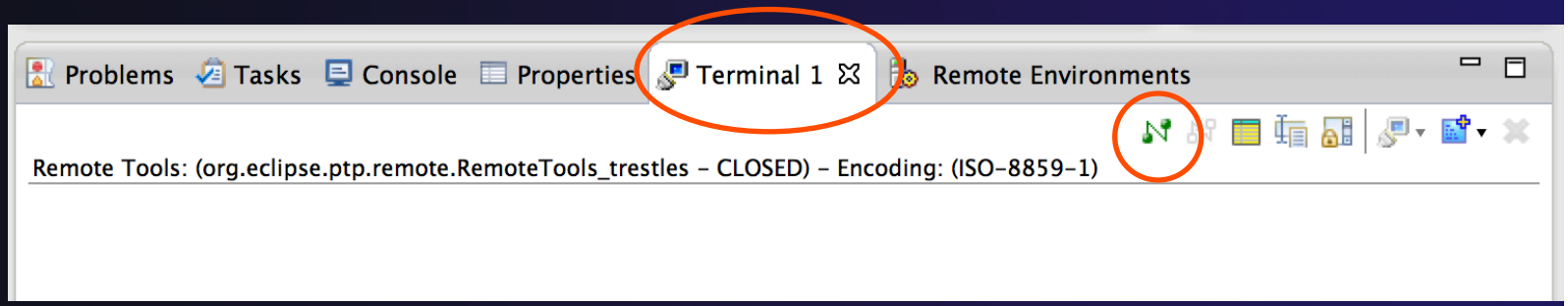
Forcing a Resync

- If Auto-sync is set, the project should automatically resync with remote system when things change
- Sometimes you may need to do it explicitly
- Right click on project and select **Synchronization>Sync Active Now**
- Status area in lower right shows when Synchronization occurs



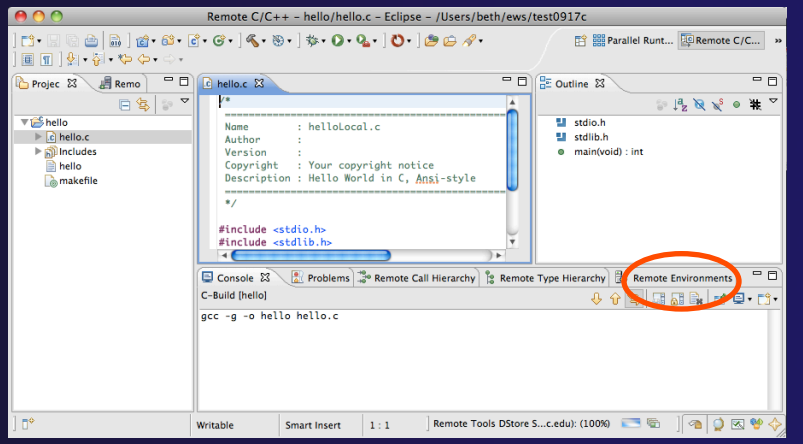
Remote Terminal

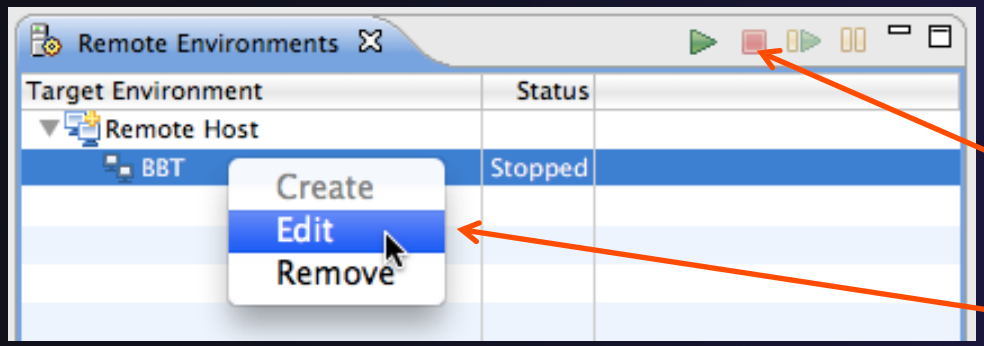
- There is a remote terminal that can be used to provide a shell from within Eclipse
- If not in your workbench:
  - Select **Window>Show View>Other...**
  - Choose **Terminal** from the Terminal folder
- In the **Terminal** view, click on the **Connect** button
- It will use the previously configured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



Changing Remote Connection Information

- If you need to change remote connection information (such as username or password), use the **Remote Environments** view





- Stop the remote connection first
- Right-click and select **Edit**

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

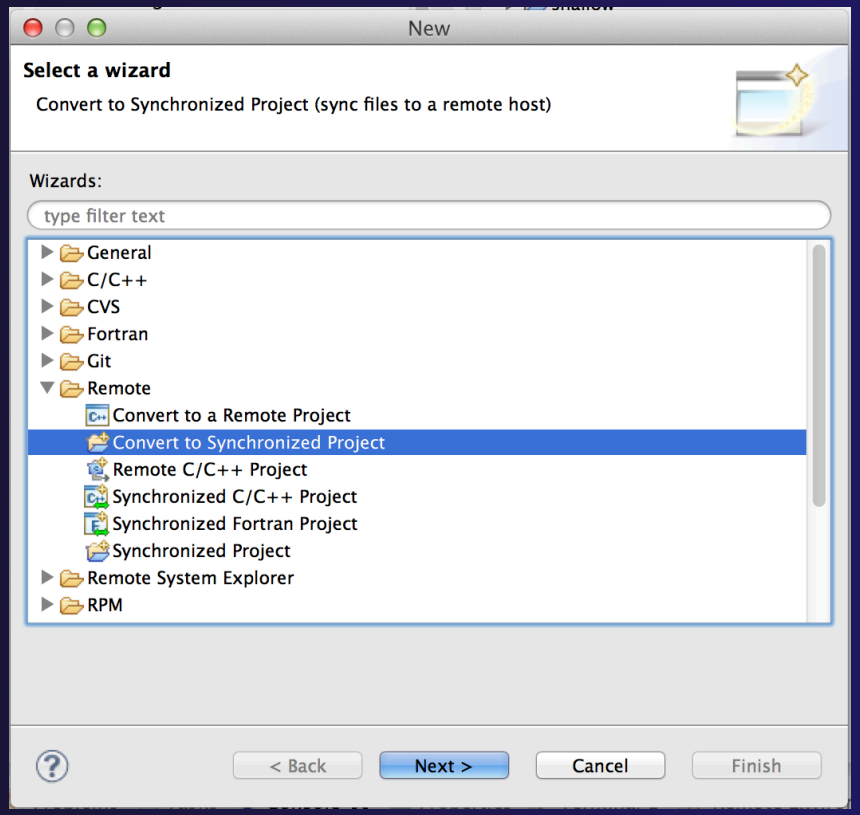
Converting a Local C/C++/Fortran Project to a Synchronized Project

The following slides are for reference.
Our project is already a Synchronized Project.

Converting To Synchronized

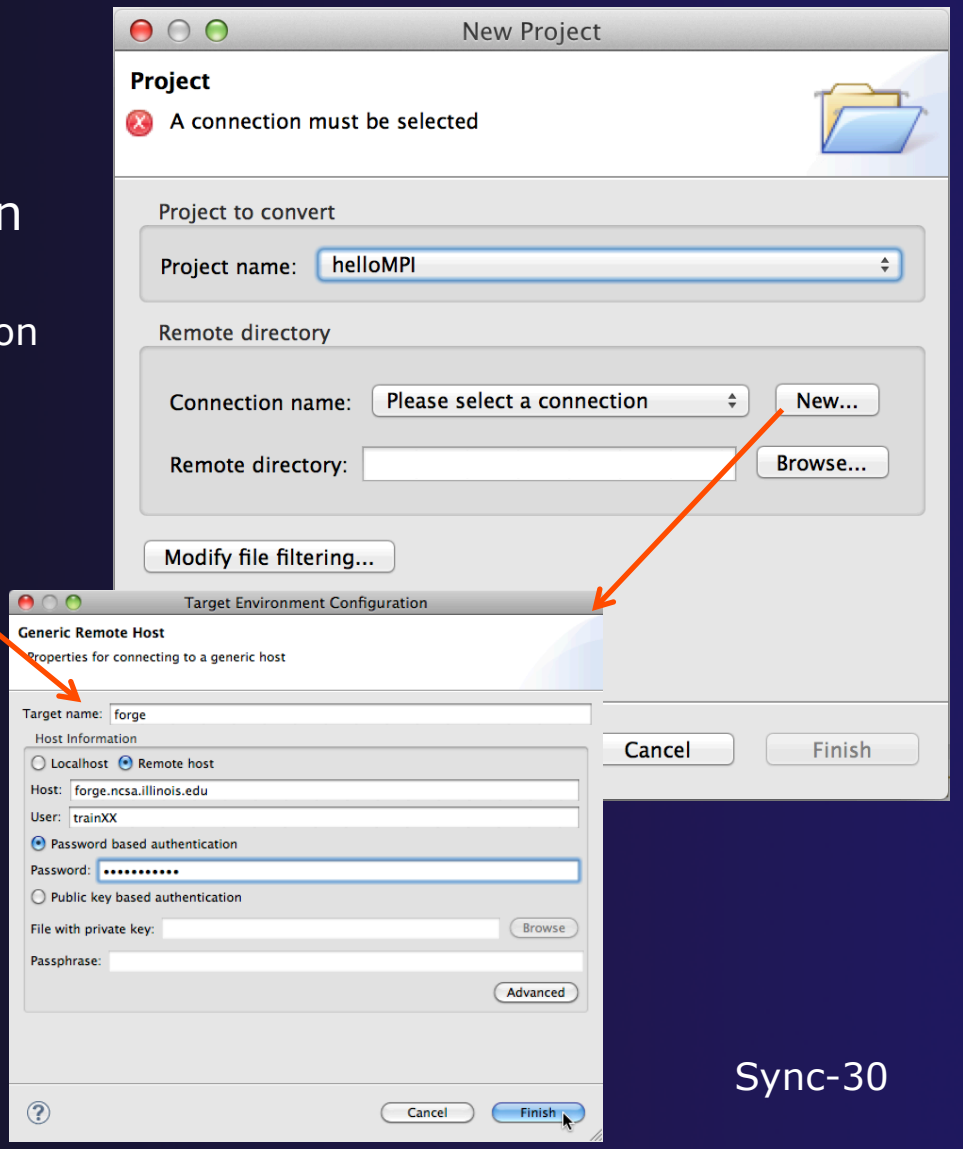
If source files exist on the local machine and you wish to convert it to a Synchronized Project on a remote system…

- Select **File>New>Other…**
- Open the Remote folder
- Select **Convert to Synchronized Project**
- Click **Next>**



Convert Projects Wizard

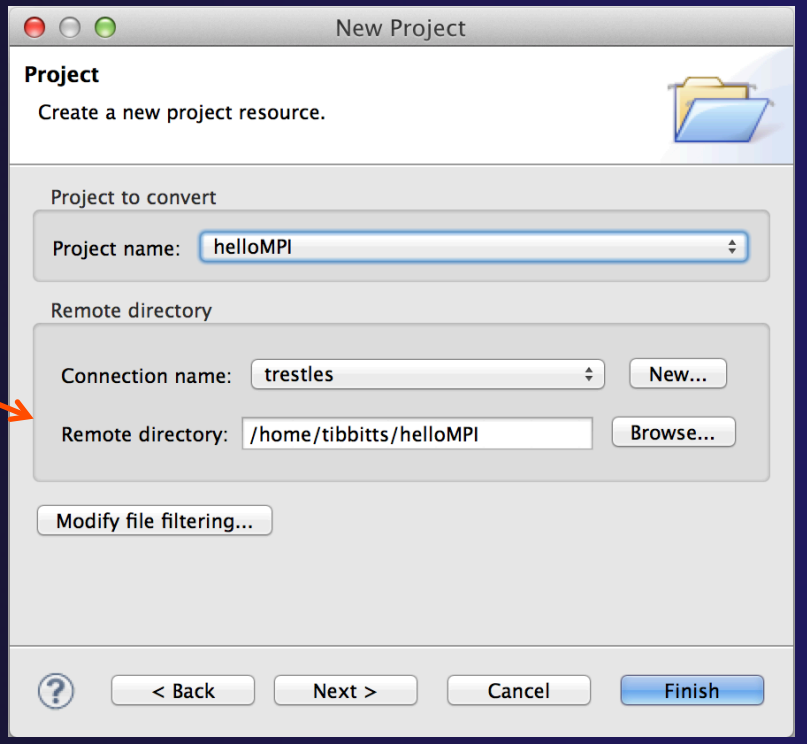
- For **Project to convert**
  - Choose your local project
- For **Connection name:** click on **New...**
  - Unless you already have a connection
- Enter information for:
  - **Target name**
  - **Host** name of remote system
  - **User** ID
  - **Password**
- Click **Finish** to close it
- The connection name will appear in the **Connection** list

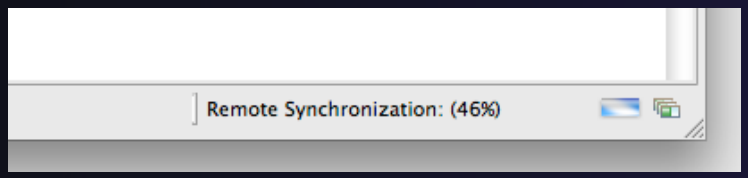


Convert Projects Wizard (2)

Back in the conversion wizard dialog, we specify where the remote files will be stored

- Enter a directory name in the **Remote Directory** field: select **Browse...**
  - Sample: /u/ac/trainXX/shallow
    Typing a new directory name creates it
  - This should normally be an empty directory – since local files will be copied there
  - Project files will be copied under this directory
- Click **Finish**
- The project should synchronize automatically





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

Eclipse CVS – "Team" Features

- Objective
  - Learn how to use Eclipse source code repository features on your project
- Contents
  - How the files look in the **Project Explorer**
  - Handling changes
  - Comparing files (diffs)
- This module assumes project was created in previous module

“Team” Features

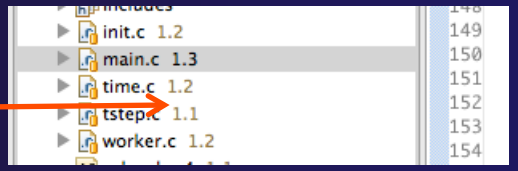
- Eclipse supports integration with multiple version control systems (VCS)
  - CVS, SVN, Git, and others
  - Collectively known as “Team” services
- Many features are common across VCS
  - Compare/merge
  - History
  - Check-in/check-out
- Some differences
  - Version numbers
  - Branching

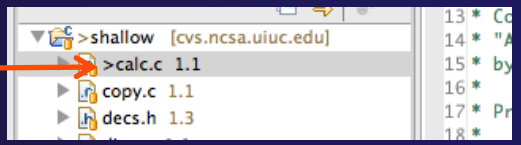
Two meanings for 'Synchronize'

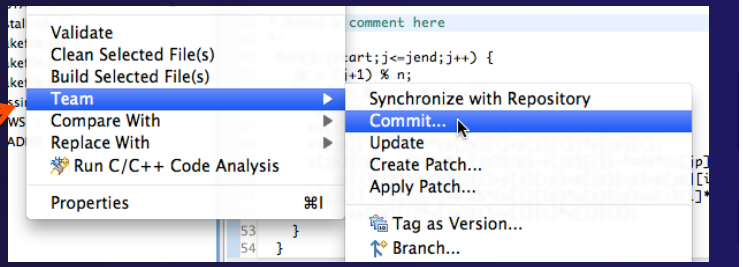
- PTP's *synchronize*
  - *Copy files in synchronized projects between local and remote to mirror them*
- Team *synchronize*
  - *Show differences between local project and source code repository versions*

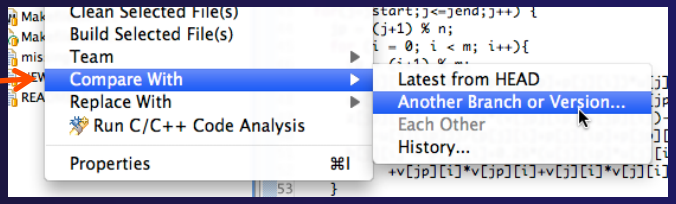
CVS Features

- Shows version numbers next to each resource
- Marks resources that have changed
  - Can also change color (preference option)
- Context menu for Team operations
- Compare to latest, another branch, or history
- Synchronize* whole project (or any selected resources)







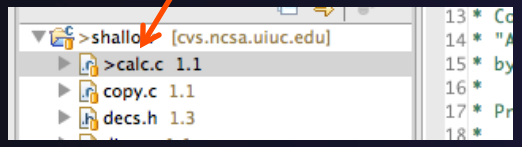


* Team synchronize

How to tell that you've changed something

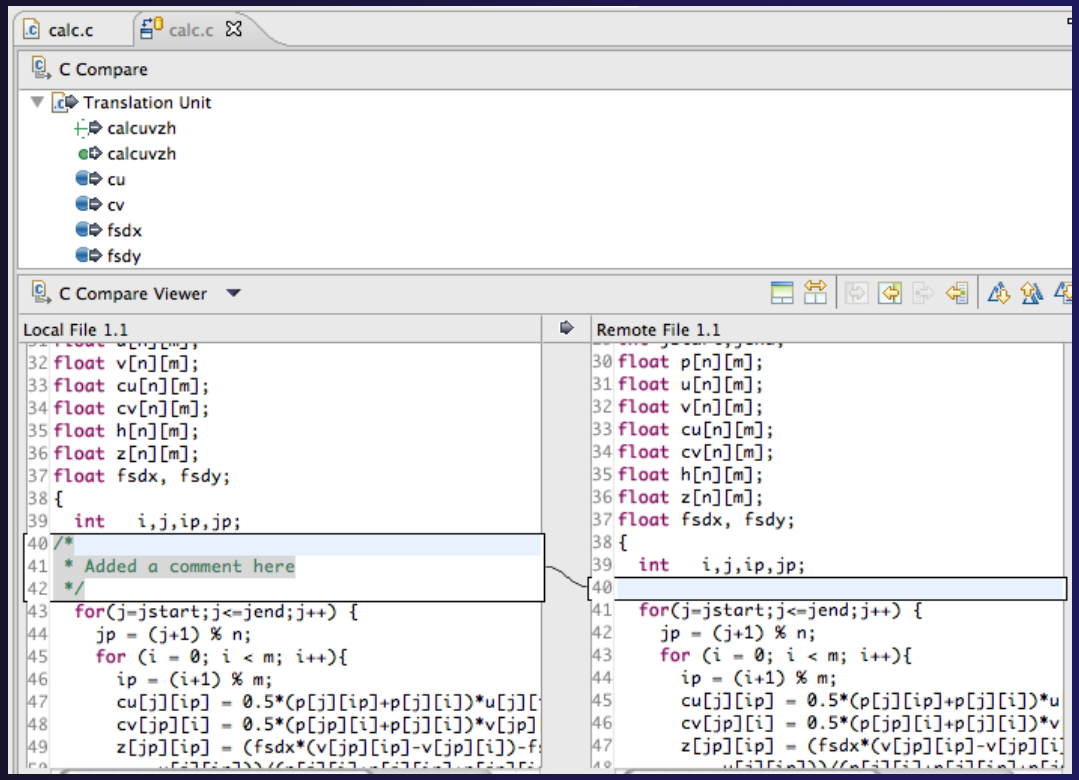
- Open "calc.c"
- Add comment at line 40
- Save file
- File will be marked ">" to indicate that it has been modified

```
27
28 void calcuvzh(jstart,jend,p,u,v,cu,cv,h,z,fsdx,fsdy)
29 int jstart,jend;
30 float p[n][m];
31 float u[n][m];
32 float v[n][m];
33 float cu[n][m];
34 float cv[n][m];
35 float h[n][m];
36 float z[n][m];
37 float fsdx, fsdy;
38 {
39   int   i,j,ip,jp;
40 /*
41  * Added a comment here
42  */
43   for(j=jstart;j<=jend;j++) {
44     jp = (j+1) % n;
45     for (i = 0; i < m; i++){
46       ip = (i+1) % m;
47       cu[j][ip] = 0.5*(p[j][ip]+p[j][i])*u[j][ip];
48       cv[jp][i] = 0.5*(p[jp][i]+p[j][i])*v[jp][i];
49       z[jp][ip] = (fsdx*(v[jp][ip]-v[jp][i])-fsdy*(u[jp][ip]
50          -u[j][ip]))/(p[j][i]+p[j][ip]+p[jp][ip]+p[jp][i]);
51       h[j][i] = p[j][i]+0.25*(u[j][ip]*u[j][ip]+u[j][i]*u[j][i]
52            +v[jp][i]*v[jp][i]+v[j][i]*v[j][i]);
53     }
```



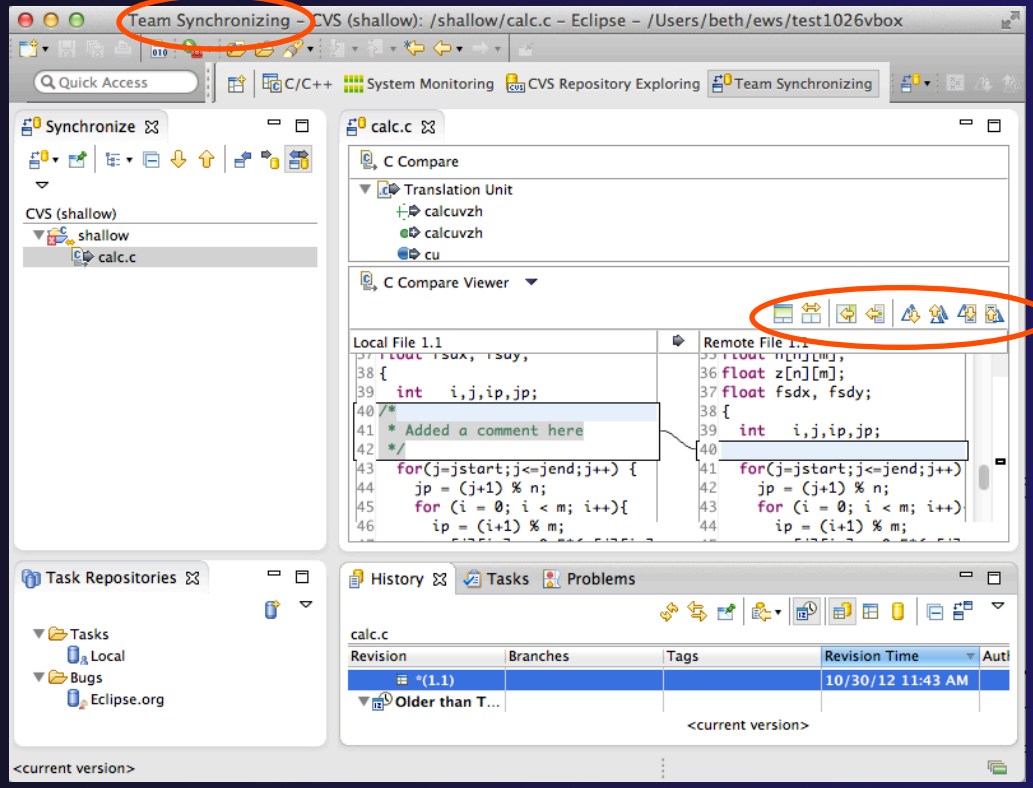
Comparing *single file* with what's in the repository

- Right-click on "calc.c" and select **Compare With>Latest from HEAD**
  - Even if you didn't create project from CVS, you can try **Compare With>Local History…**
- Compare editor will open showing differences between local (changed) file and the original
- Buttons allow changes to be merged from right to left
- Can also navigate between changes using buttons



Comparing *your project* with what's in the repository

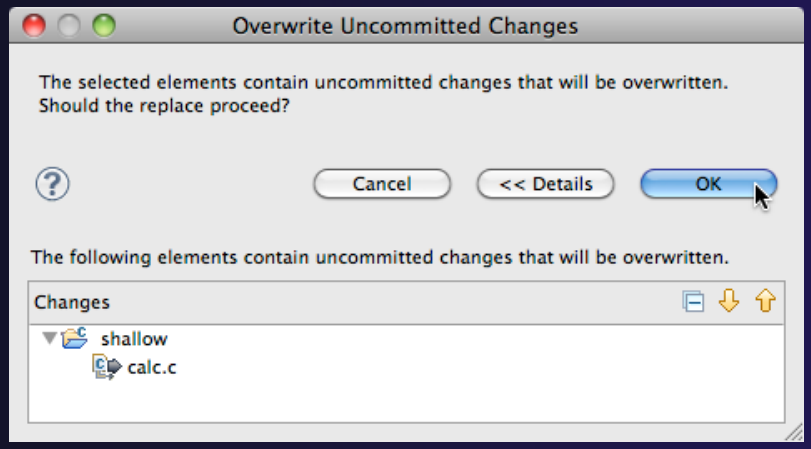
- Right-click on project name (or any subset) and select **Team>Synchronize with Repository**
- **Team Synchronizing** perspective will open
- List of changed files appears
- Double-click on a file to see the diff viewer
- Buttons allow changes to be merged from right to left
- Can also navigate between changes using buttons



Revert To The Latest Version

To replace your project contents to the current contents of the project in the src code repo,

- Right-click on the “shallow” project
  … and select **Replace With>Latest from HEAD**
- Review the resources that will be replaced, then click **OK**



Exercise

- Check out the *shallow* project from CVS as a synchronized project - as described in this module

Optional Exercise

1. Name every person who modified the Makefile
2. Identify which parts of the Makefile changed since revision 1.3

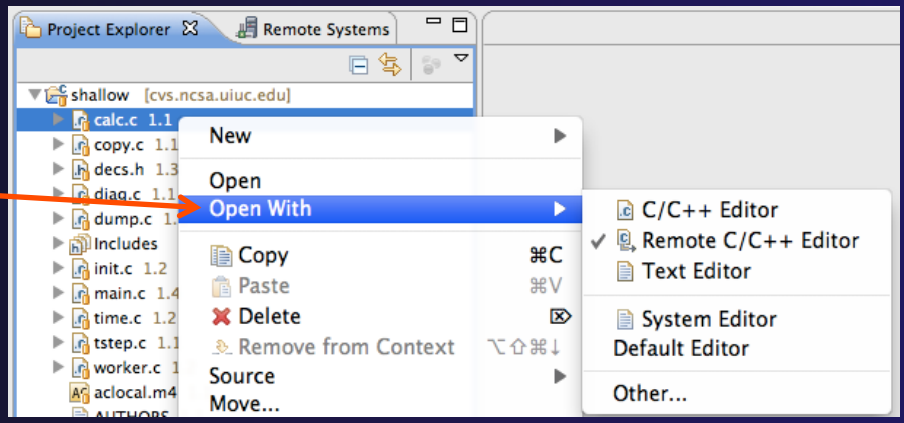
Hint: *Right-click the Makefile and select* ***Team > Show History****. Both of these can be done from the History view.*

Editor Features

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

Editors

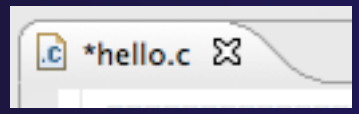
- An editor for a resource (e.g. a file) opens when you double-click on a resource
- The type of editor depends on the type of the resource
  - .c files are opened with the C/C++ editor by default
  - You can use **Open With** to use another editor
  - In this case the default editor is fine (double-click)



  - 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

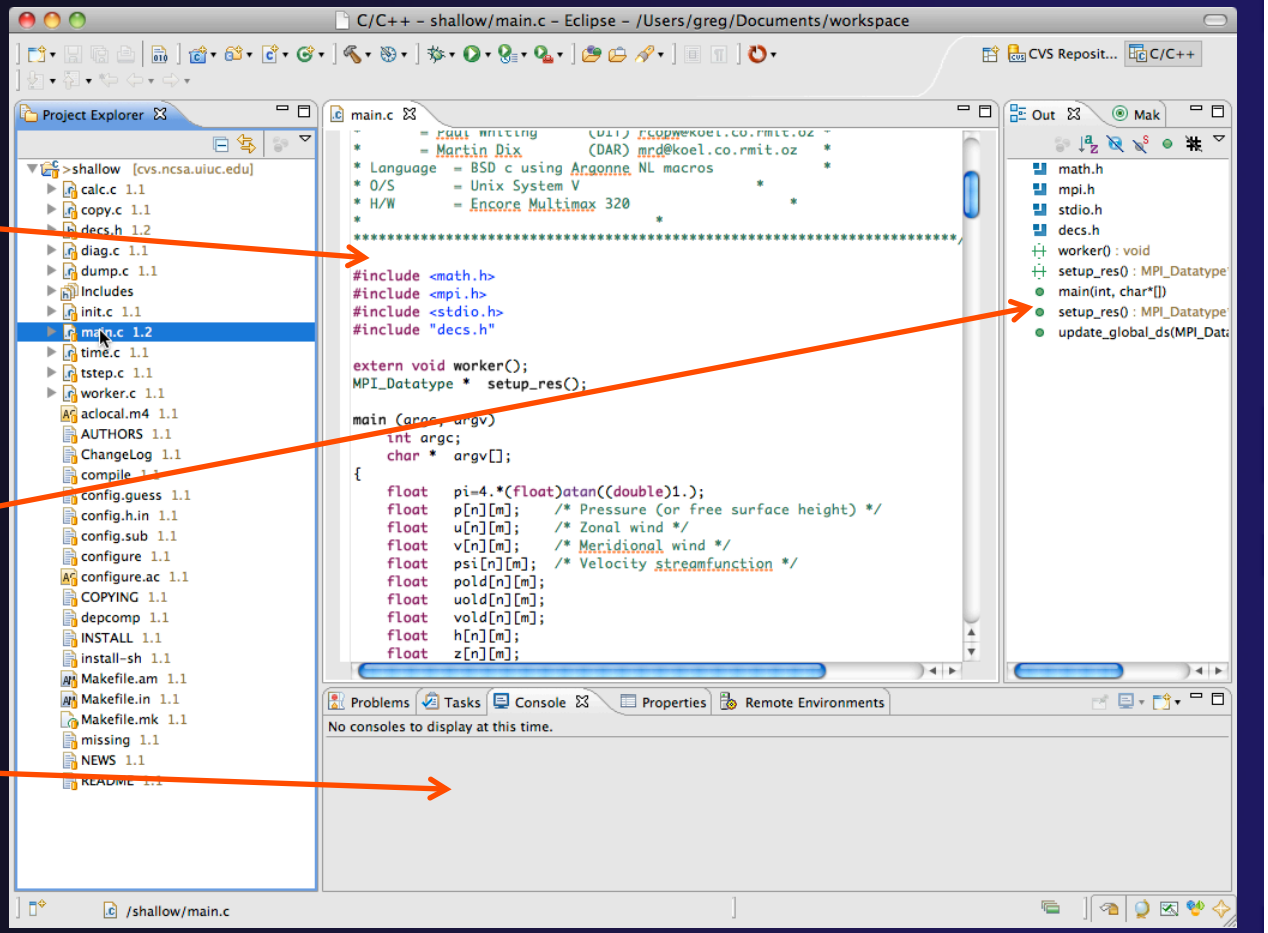
- When you change a file in the editor, an asterisk on the editor's title bar indicates unsaved changes



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

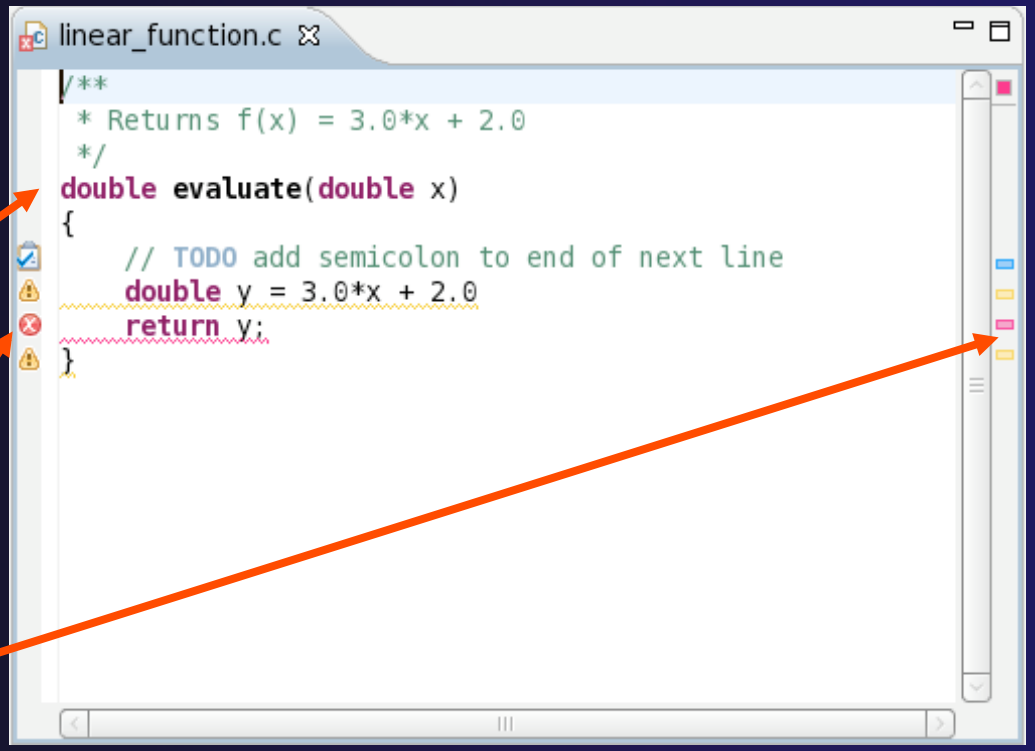
Editor and Outline View

- 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

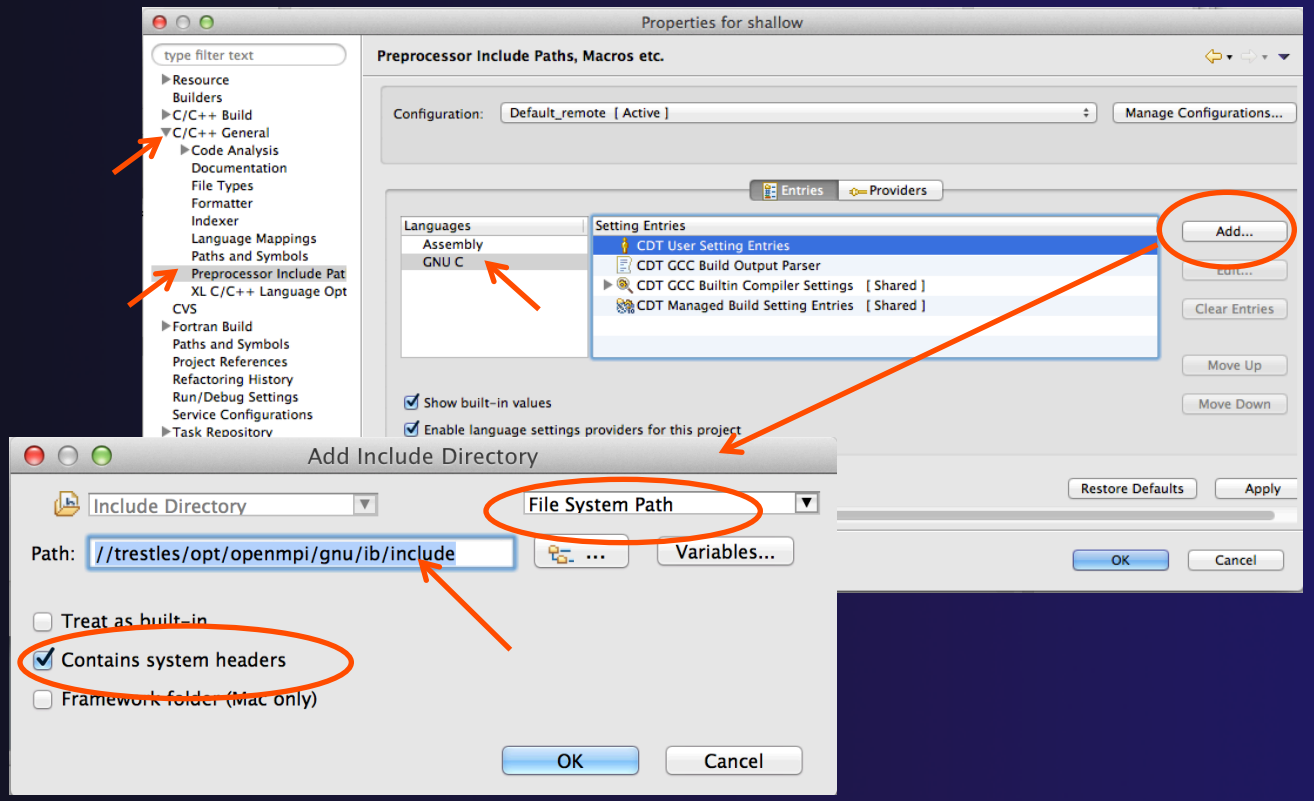


Icons:

Include Paths (1)

- 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., but we must tell Eclipse so that indexing, search, completion, etc. will know where things are

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

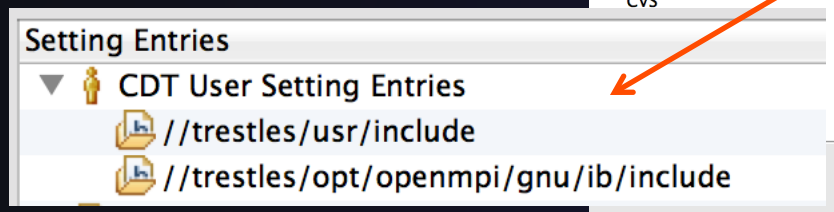


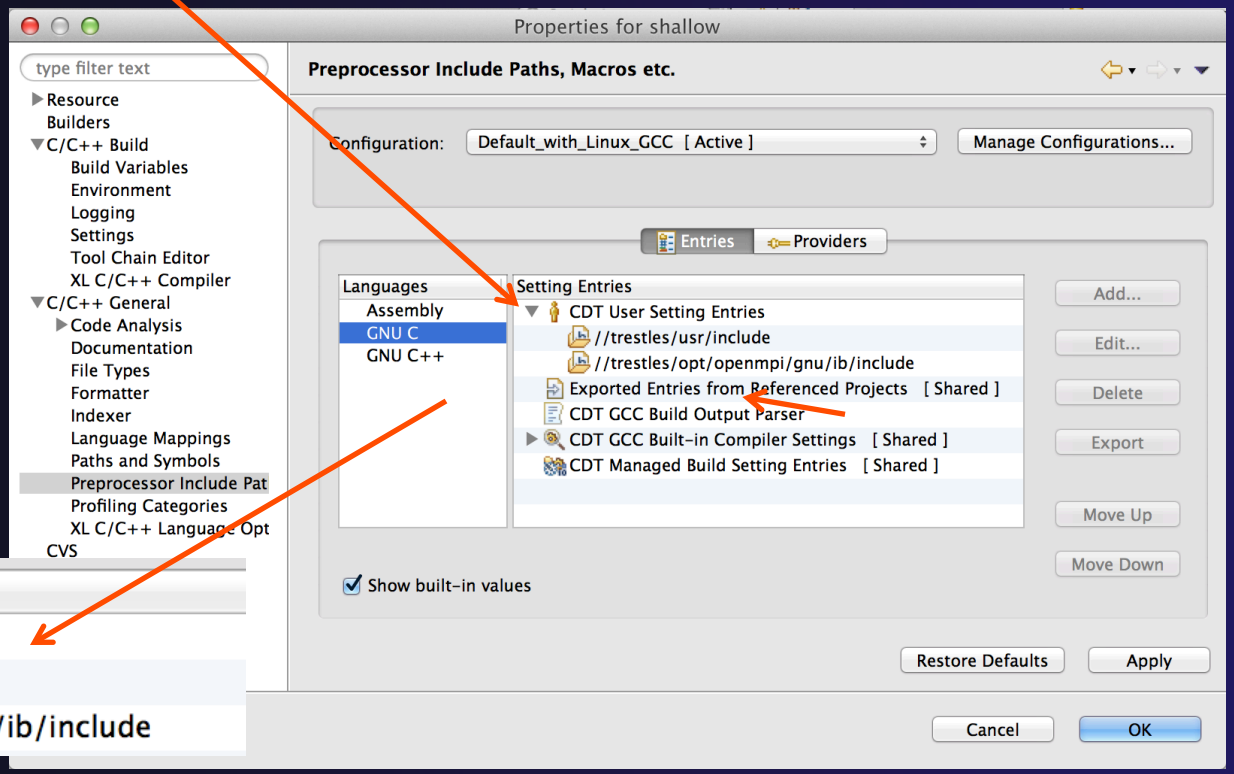
Include Paths (2)

- After adding include directory, it should appear in the list
- Add second value:

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

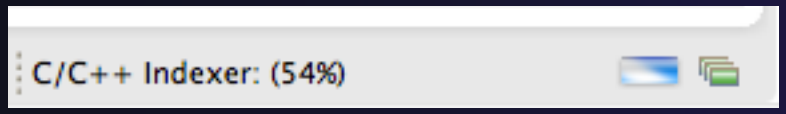
You should have two entries:





Include Paths (3)

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



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

Code Analysis (Codan)

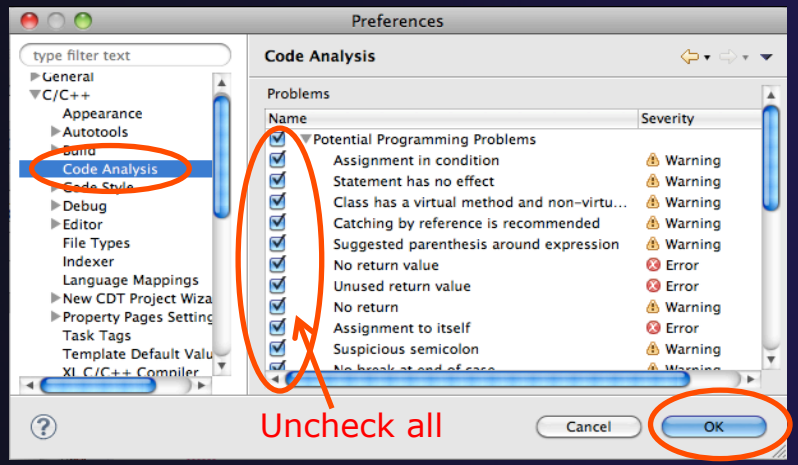
- If you see bug icons in the editor marker bar, they are likely suggestions from Codan
  - If include files are set correctly, they *should* not appear.
- Code checkers can flag possible errors, even if code is technically correct
- To turn them off, use Preferences

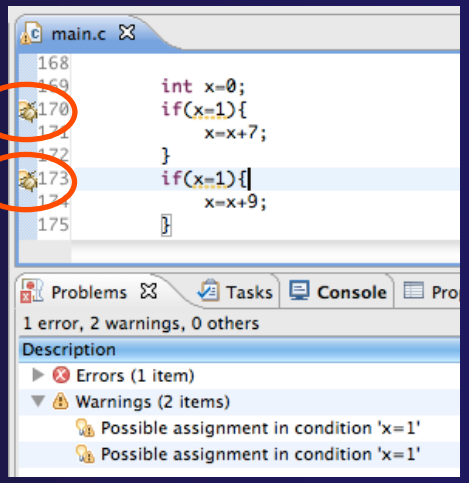
Window > Preferences or Mac: Eclipse > Preferences

C/C++ > Code Analysis

and uncheck all problems

- Select OK to close Preferences

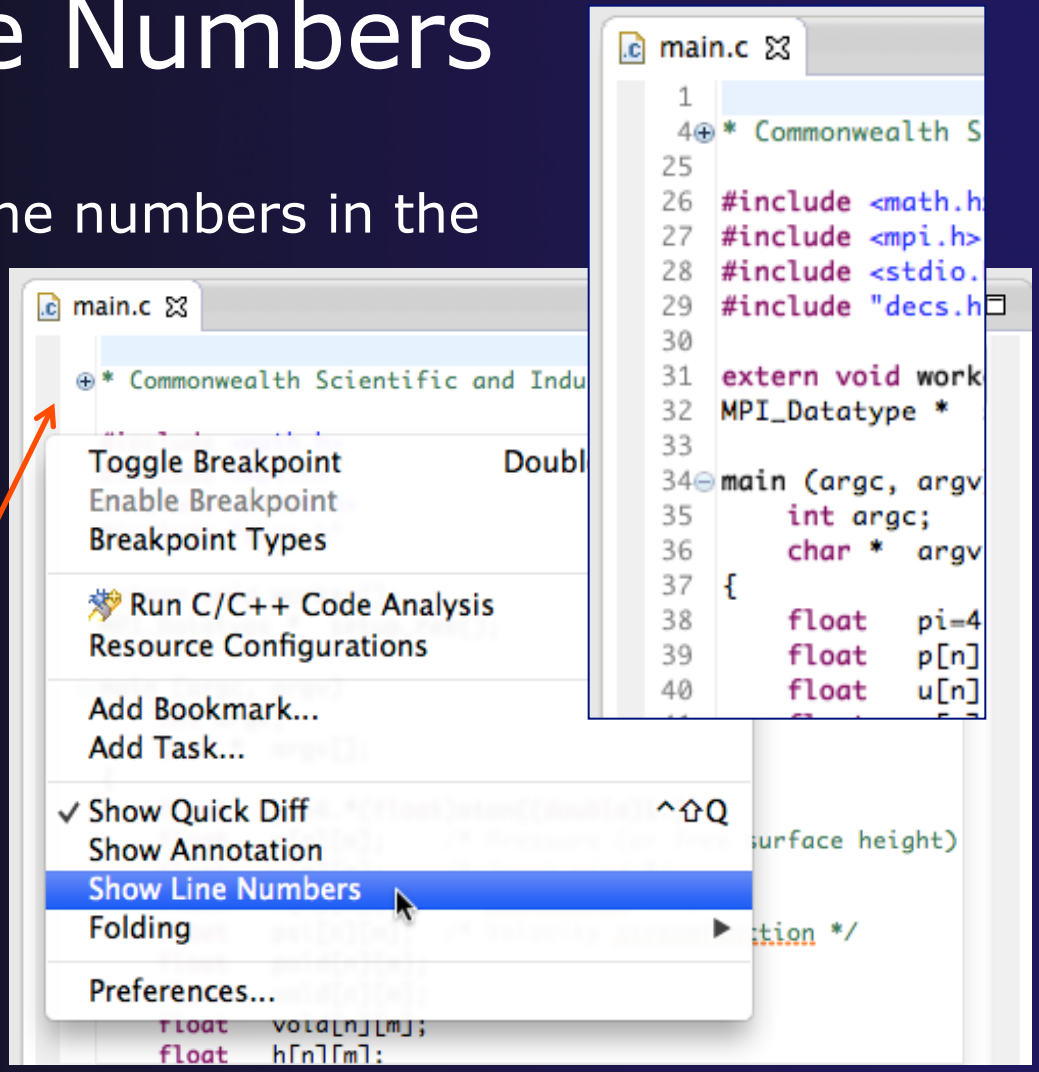




- If icons don't disappear: Right mouse on Project > **Run C/C++ Code Analysis**
- You can also enable/disable this per project in Project Properties

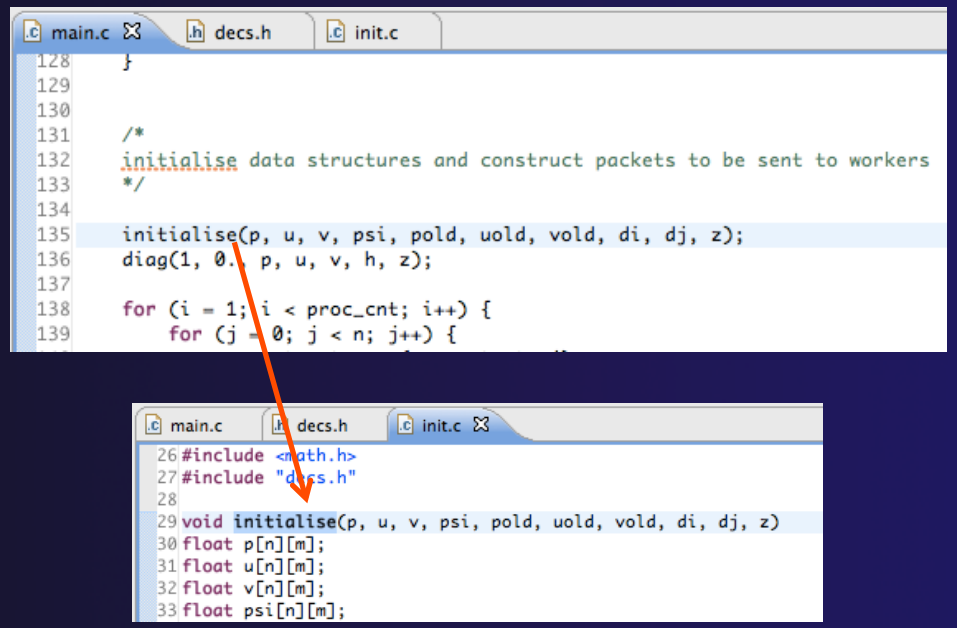
Line Numbers

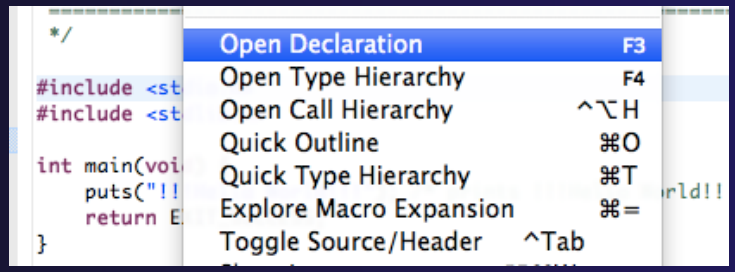
- Text editors can show line numbers in the left column
- To turn on line numbering:
  - Right-mouse click in the editor marker bar (at editor left edge)
  - Click on **Show Line Numbers**



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
- 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**

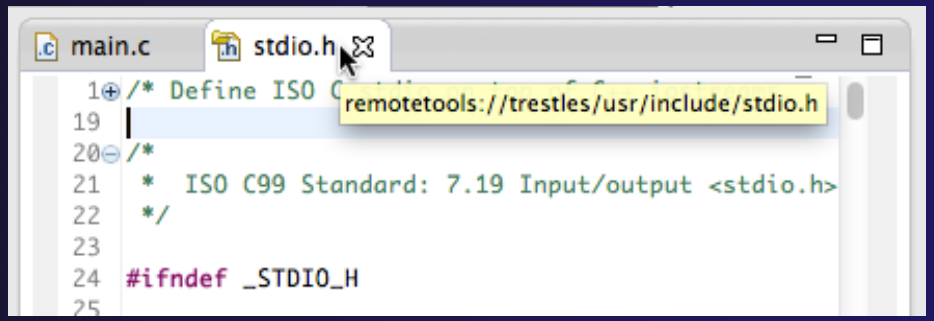




Note: may need to left-click before right-click works

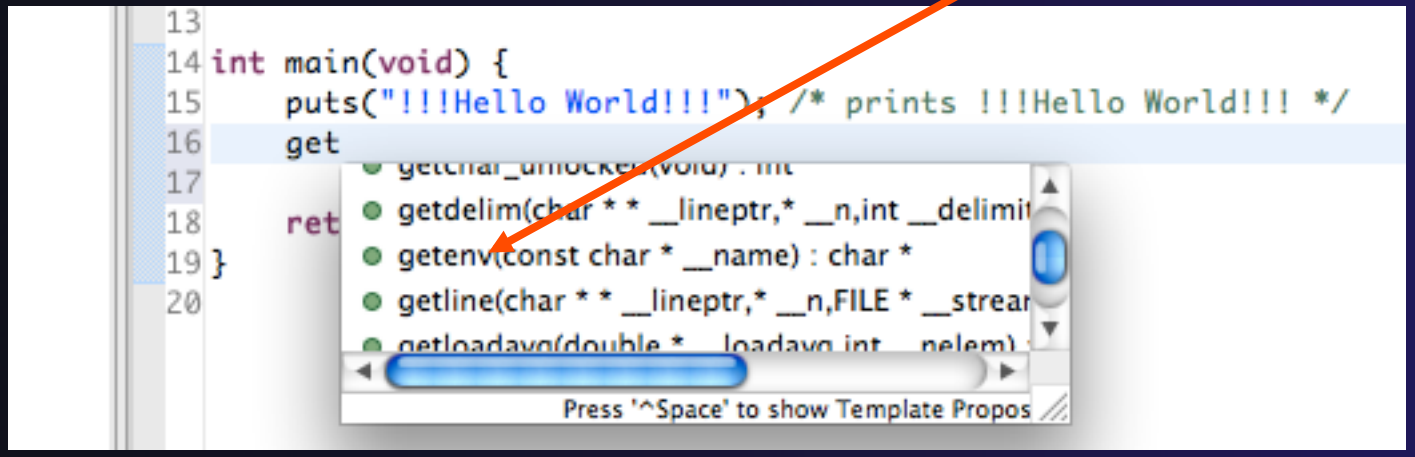
Navigating to Remote Files

- 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.



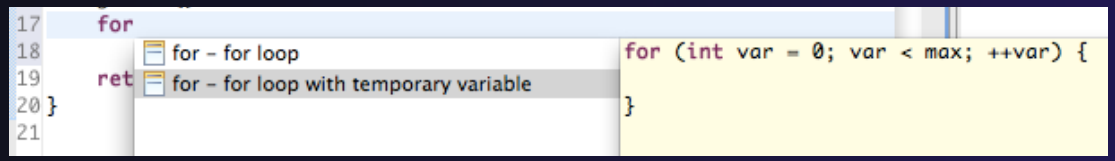
Content Assist & Templates

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



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

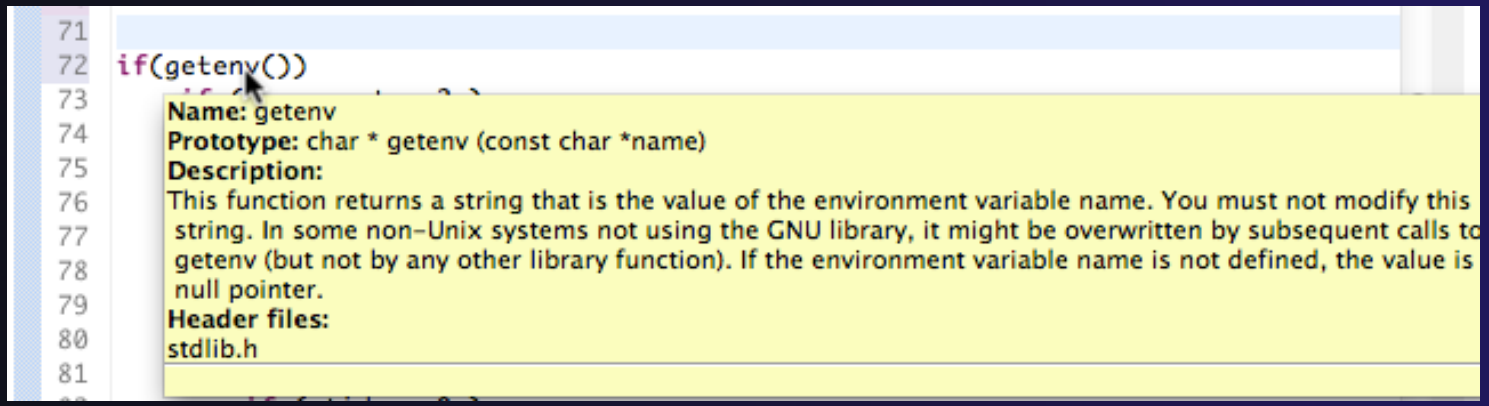
Hit ctrl-space again for code templates



More info on code templates later

Hover Help

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



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

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

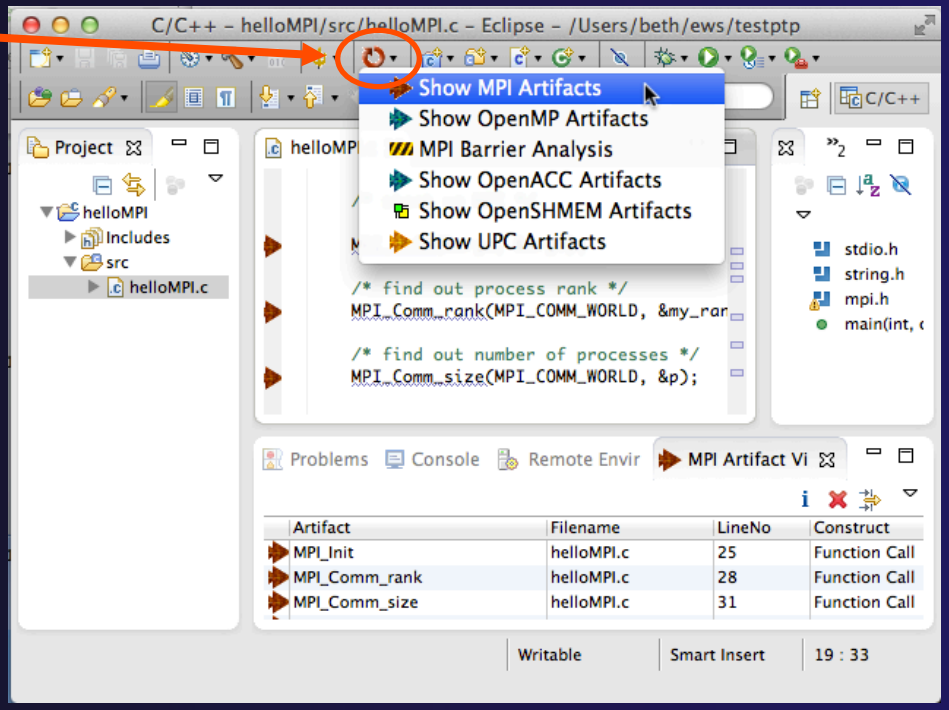
- Objective
  - Learn about MPI features for your source files
- Contents
  - Using Editor features for MPI
  - MPI Help features
  - Finding MPI Artifacts
  - MPI New Project Wizards
  - MPI Barrier Analysis

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

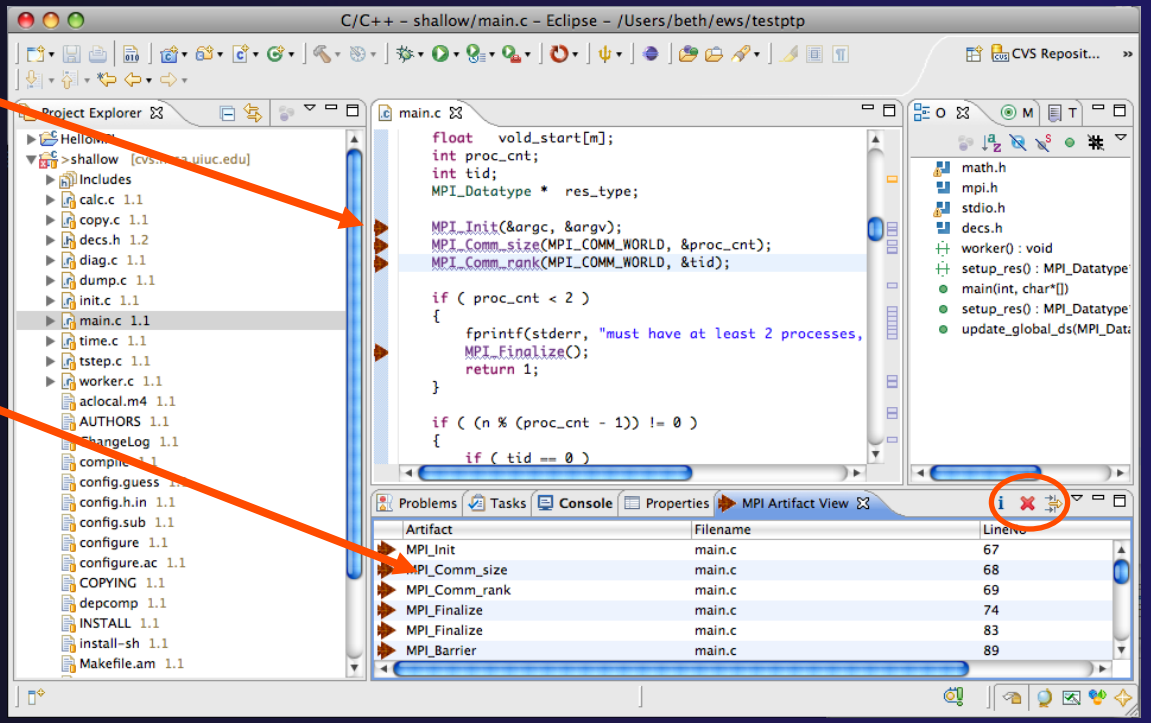
Show MPI Artifacts

- 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**

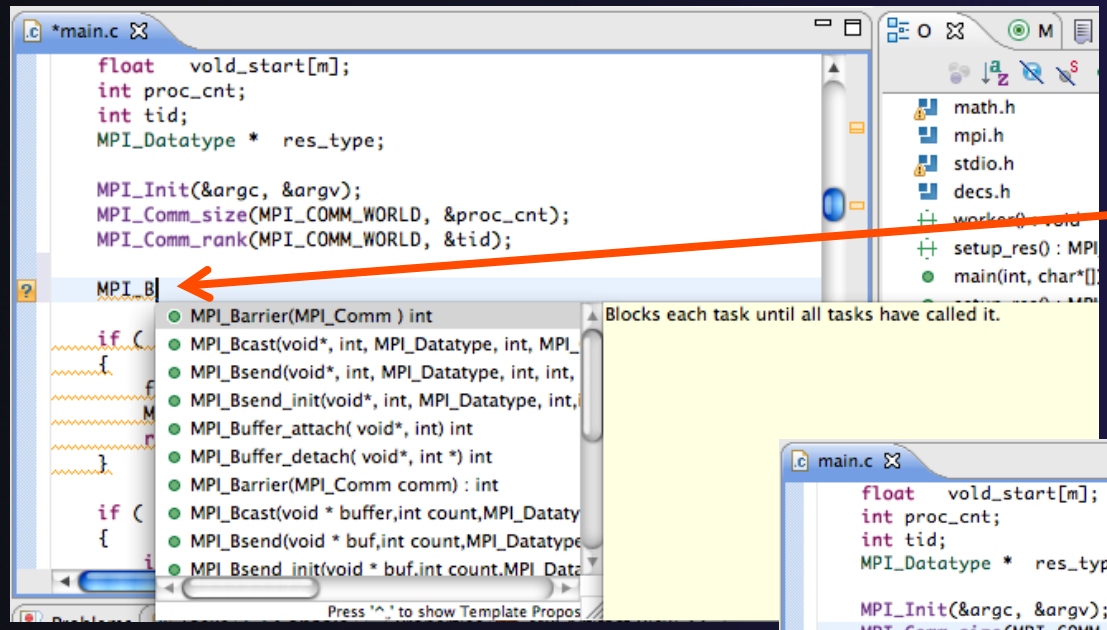


MPI Artifact View

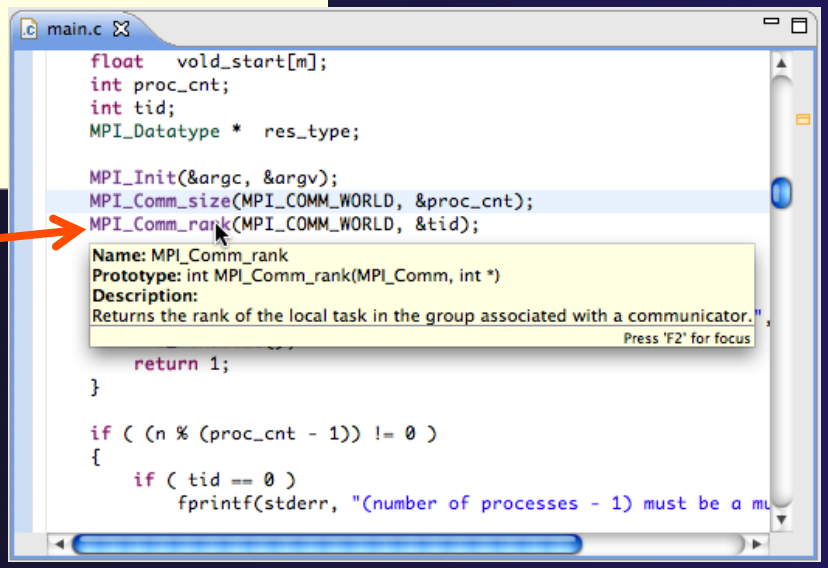
- Markers indicate the location of artifacts in editor
- The **MPI Artifact View** lists the type and location of each artifact
- Navigate to source code line by double-clicking on the artifact
- Run the analysis on another file (or entire project!) and its markers will be added to the view
- Click on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



MPI Editor Features



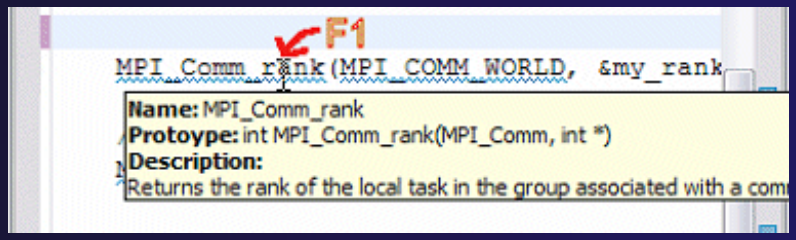
- Code completion will show all the possible MPI keyword completions
- Enter the start of a keyword then press <ctrl-space>

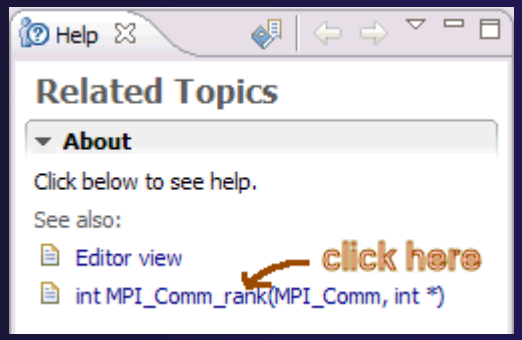


- Hover over MPI API
- Displays the function prototype and a description

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



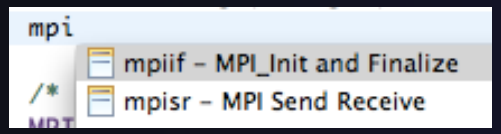


Some special info has been added for MPI APIs

MPI Templates

- Allows quick entry of common patterns in MPI programming

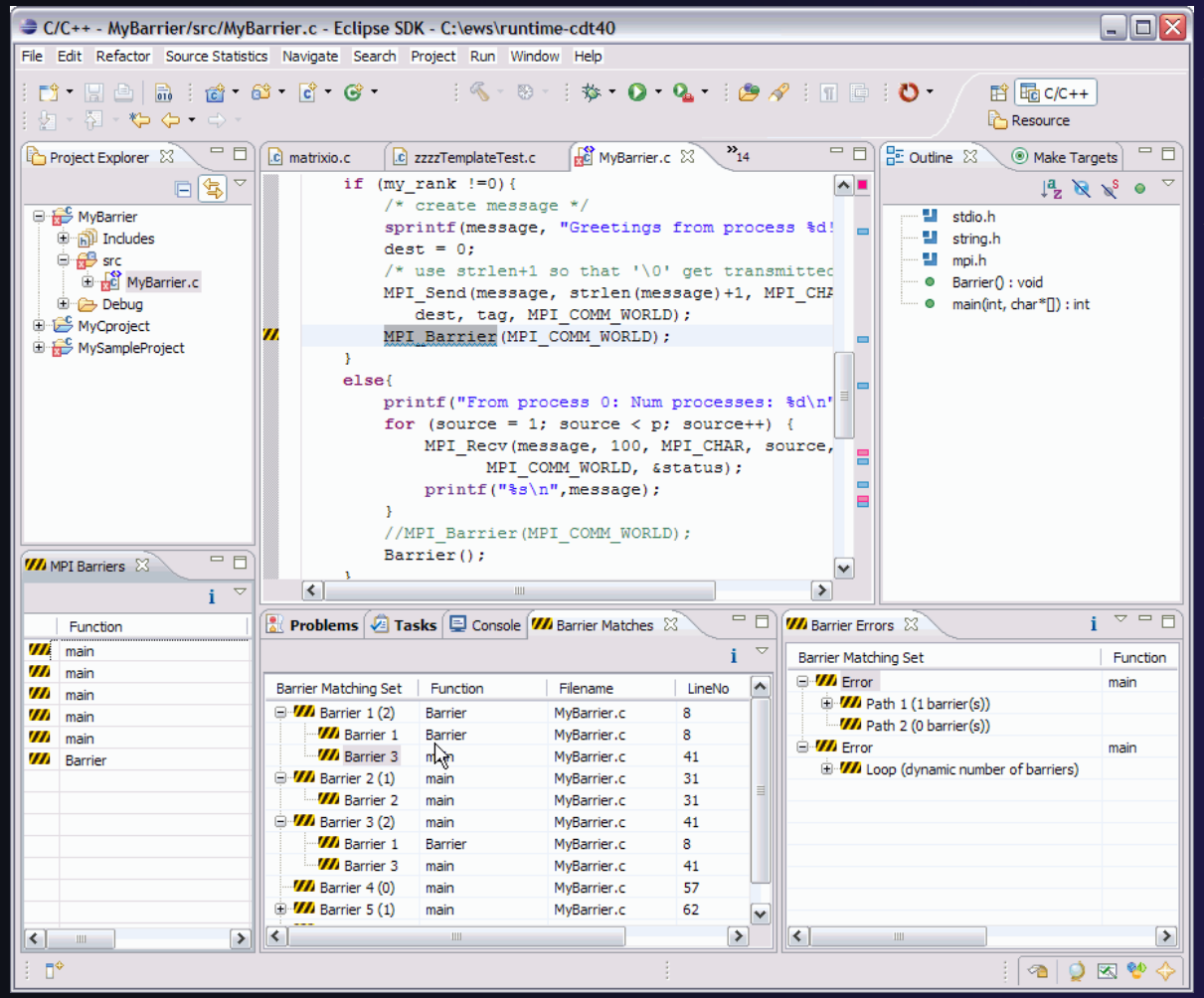
- Example:
  MPI send-receive
- Enter:
  `mpisr` <ctrl-space>
- Expands to a send-receive pattern
- Highlighted variable names can all be changed at once
- Type `mpi` <ctrl-space> <ctrl-space> to see all templates




```
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



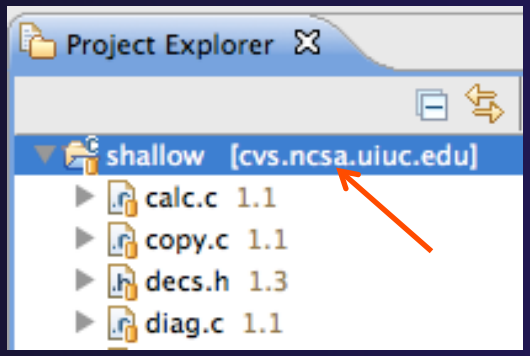
- Verify barrier synchronization in C/MPI programs
- For verified programs, lists barrier statements that synchronize together (match)
- For synchronization errors, reports counter example that illustrates and explains the error

Local files only

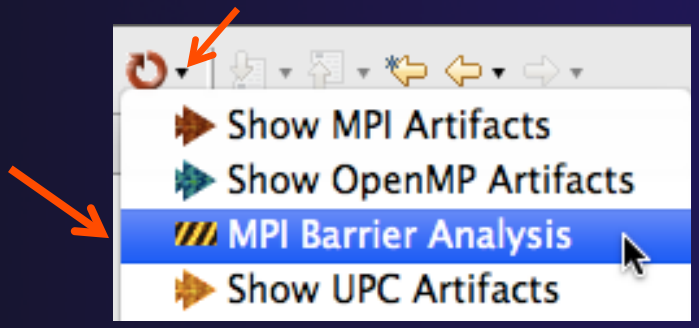
MPI Barrier Analysis (2)

Run the Analysis:

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

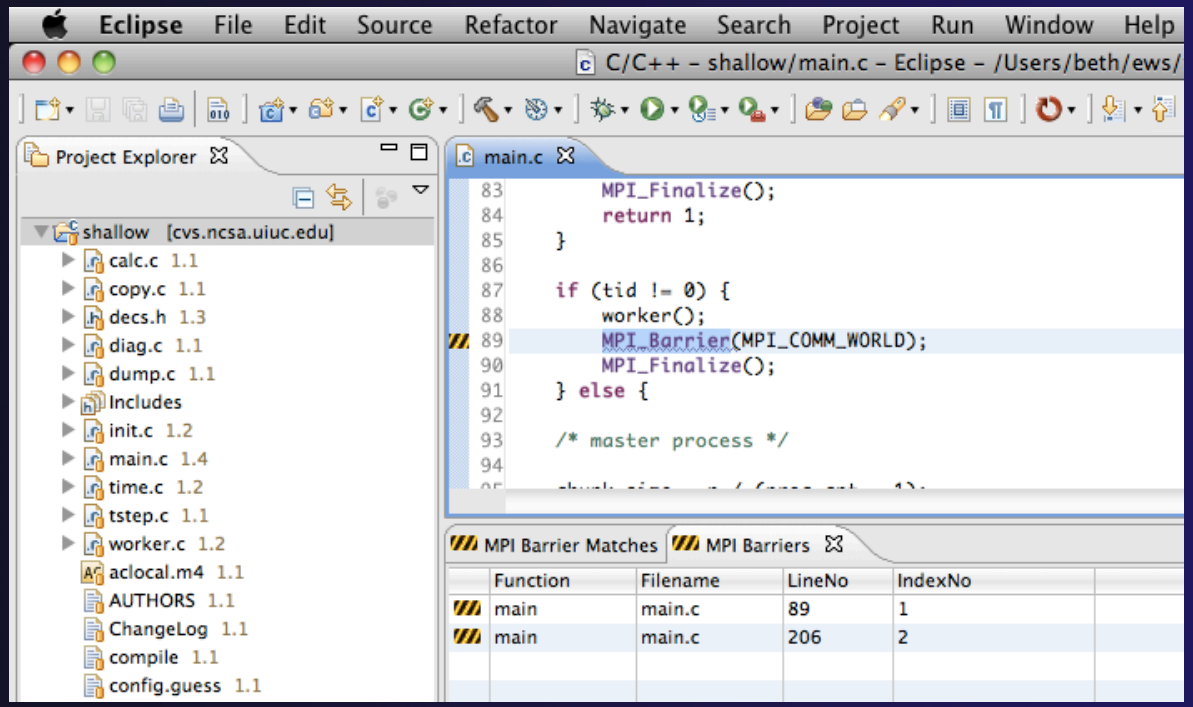


- Select the MPI Barrier Analysis action in the pull-down menu

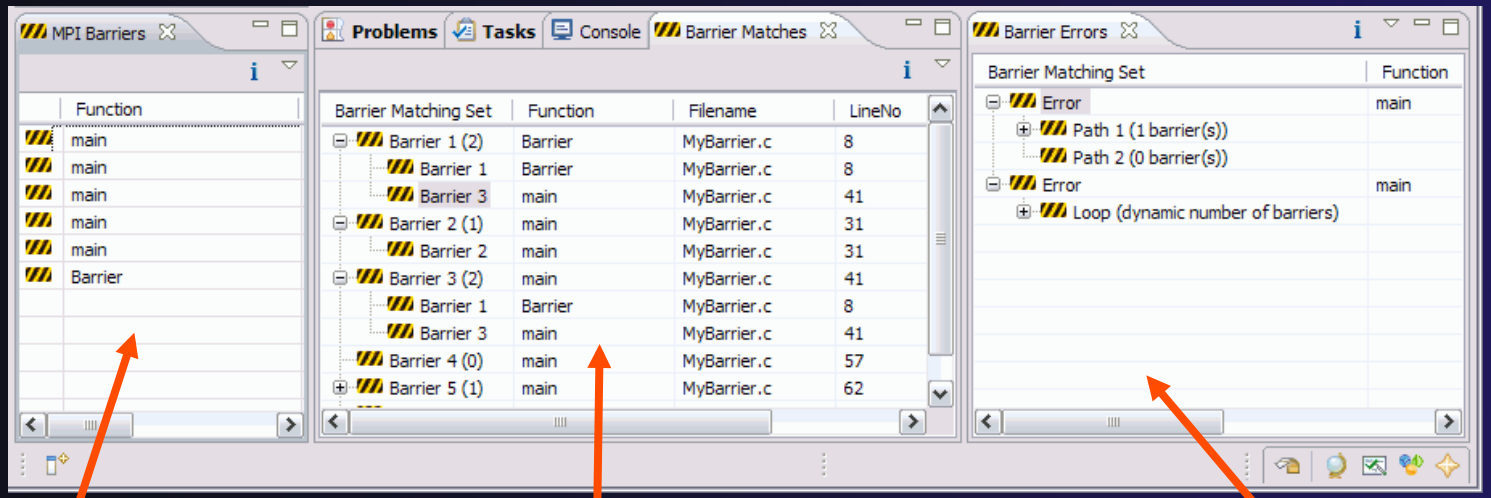


MPI Barrier Analysis (3)

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



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

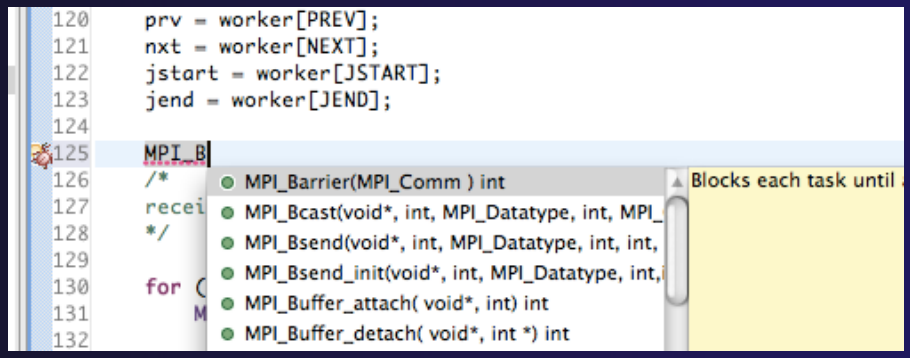
Groups barriers that match together in a barrier set – all processes must go through a barrier in the set to prevent a deadlock

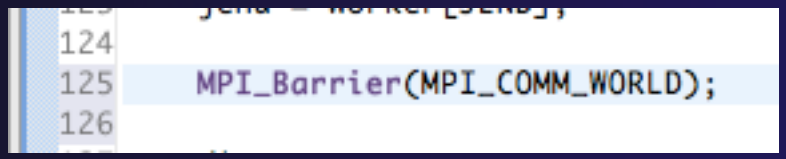
Barrier Errors view

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

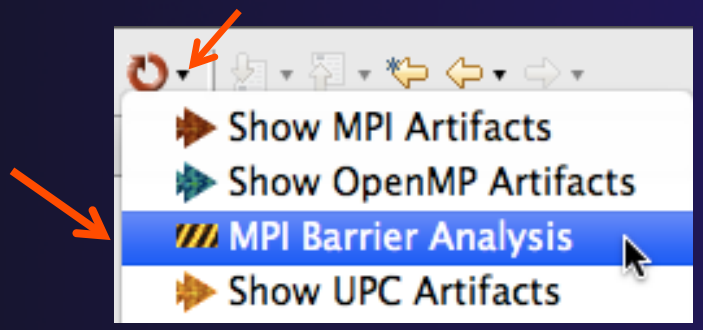




Barrier Errors (2)

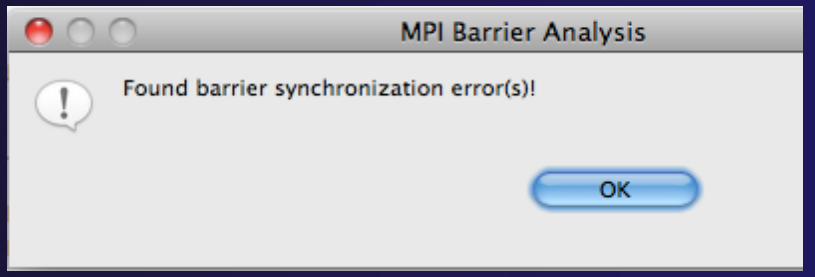
- Save the file
  - Ctl-S (Mac Command-S) or File > Save
  - Tab should lose asterisk indicating file saved

- 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

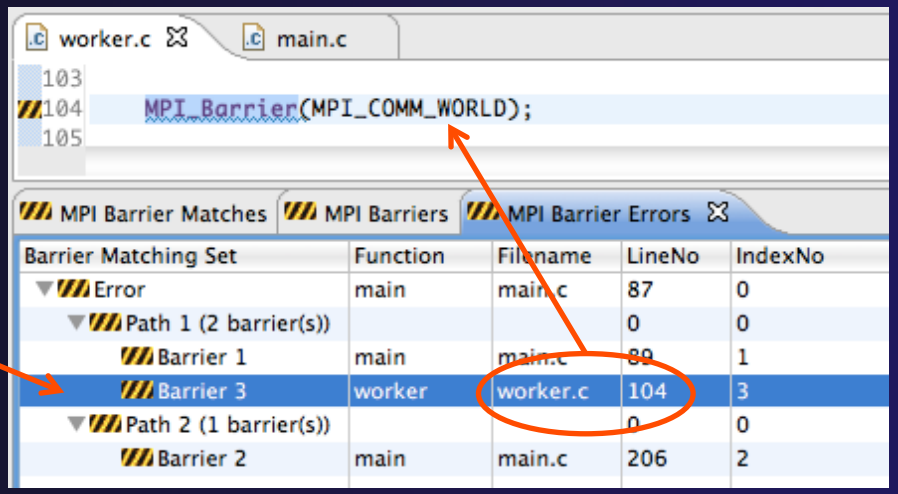
MPI Barrier Matches | MPI Barriers | MPI Barrier Errors

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

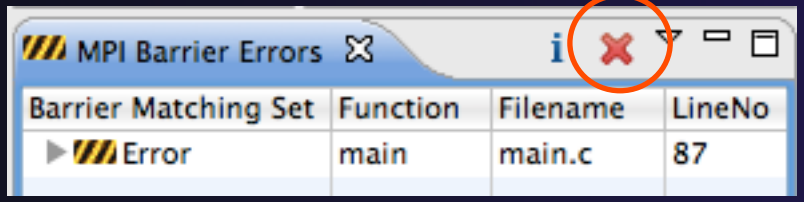
Fix Barrier Error

- 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



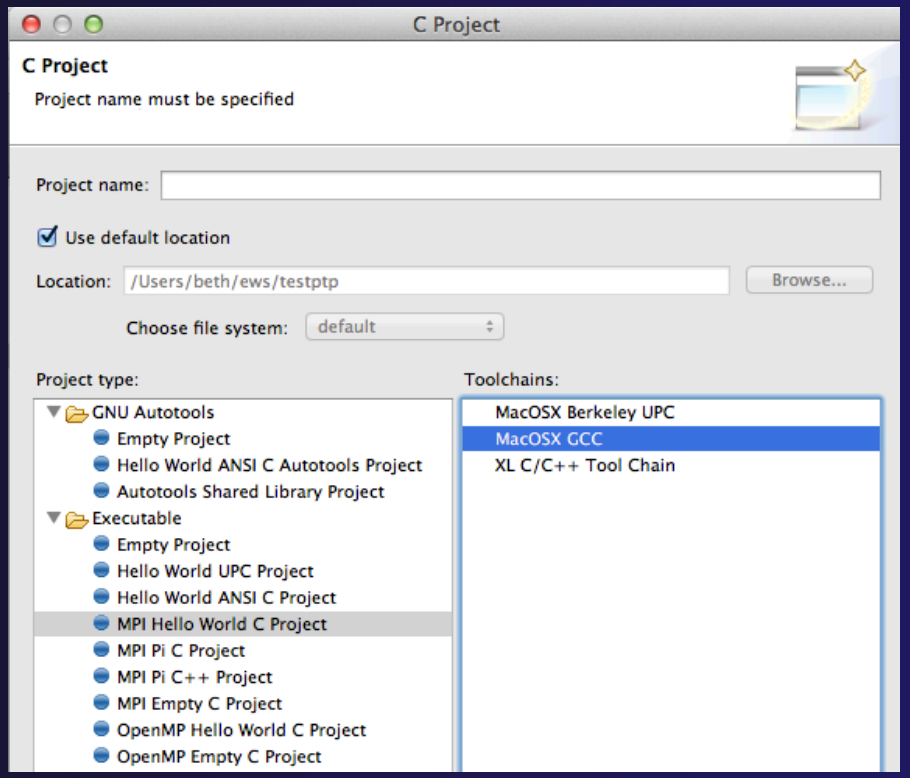
Remove Barrier Markers

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



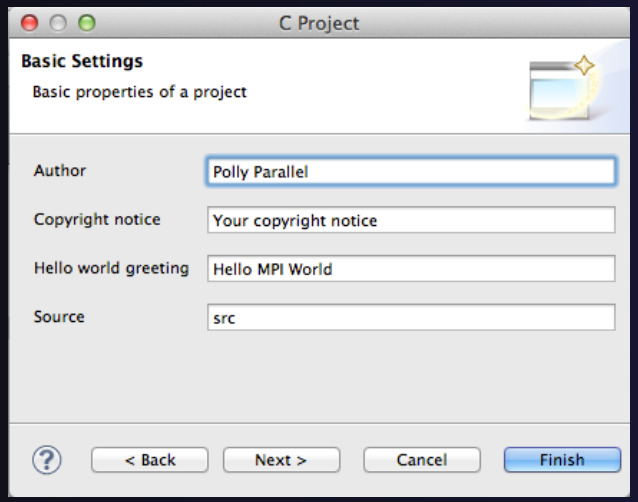
MPI New Project Wizards

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

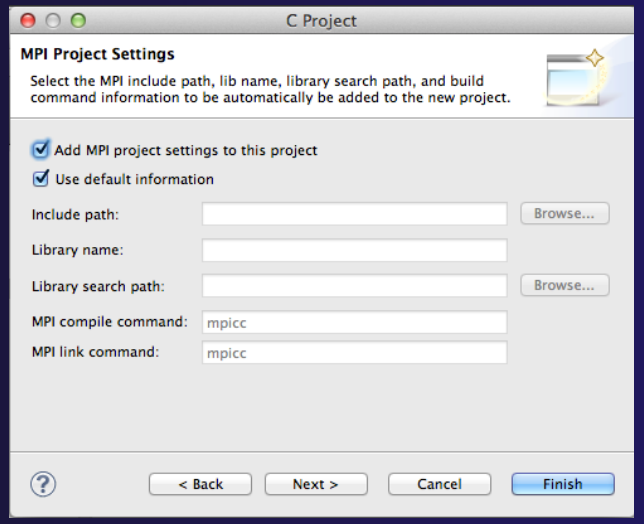


MPI New Project Wizards (2)

- Next> and fill in (optional) Basic Settings

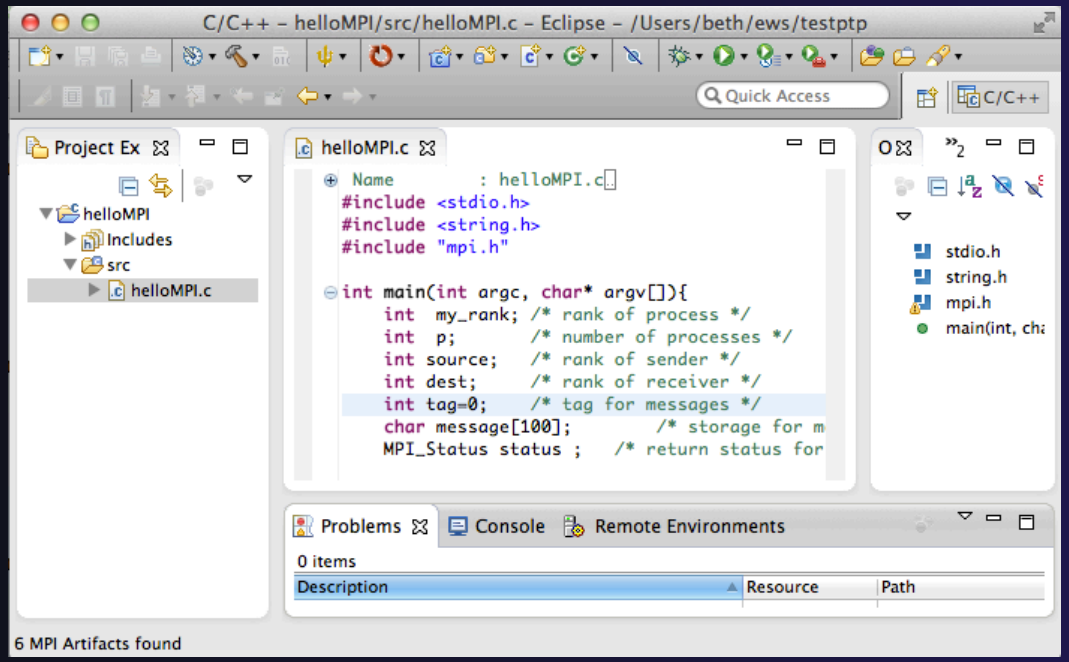


- Next> and fill in MPI Project Settings
- Include path set in MPI Preferences can be added to project



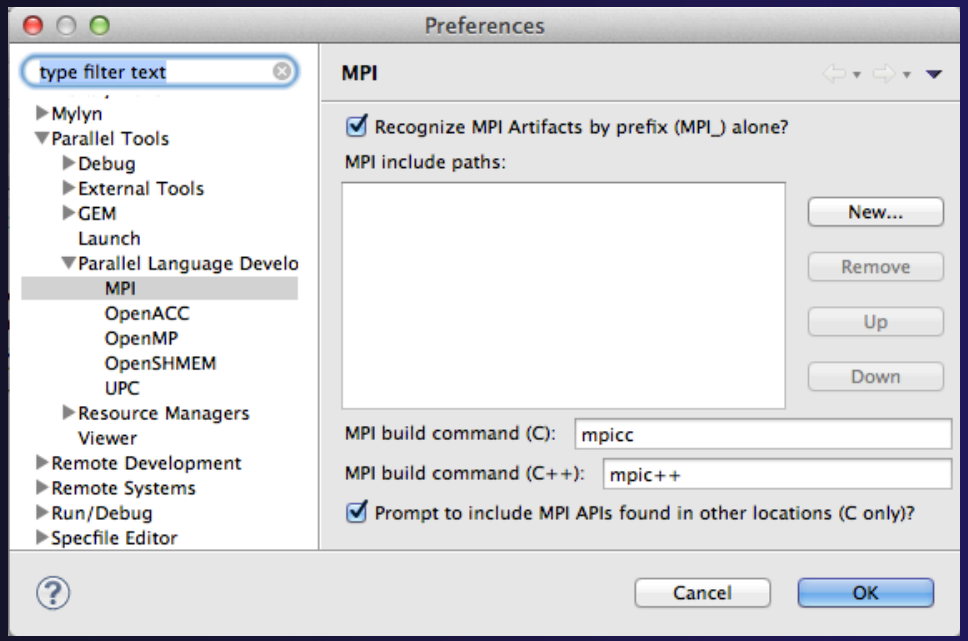
MPI New Project Wizards (3)

- Select **Finish** and "MPI Hello World" project is created



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

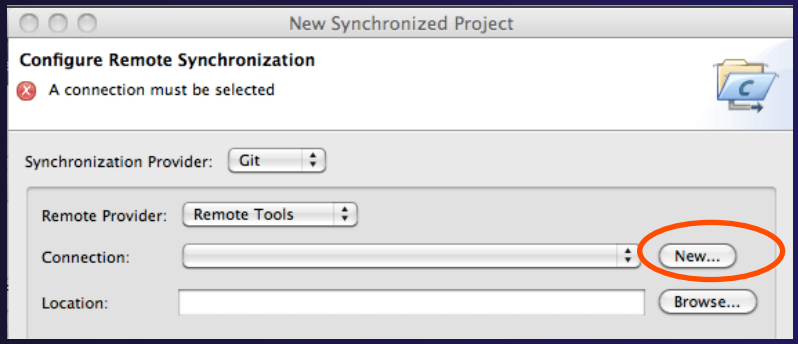
Configuring SSH Tunnel

Configure the Synchronized Project - SSH tunnel (1)

- If your machine access requires ssh access through a frontend/ intermediate node, set up an ssh tunnel before configuring the project - from command line or e.g. Windows PuTTY, e.g.
  ssh -L <port>:<target-host> <userid>@<frontend-host>
  (For details see http://wiki.eclipse.org/PTP/FAQ)

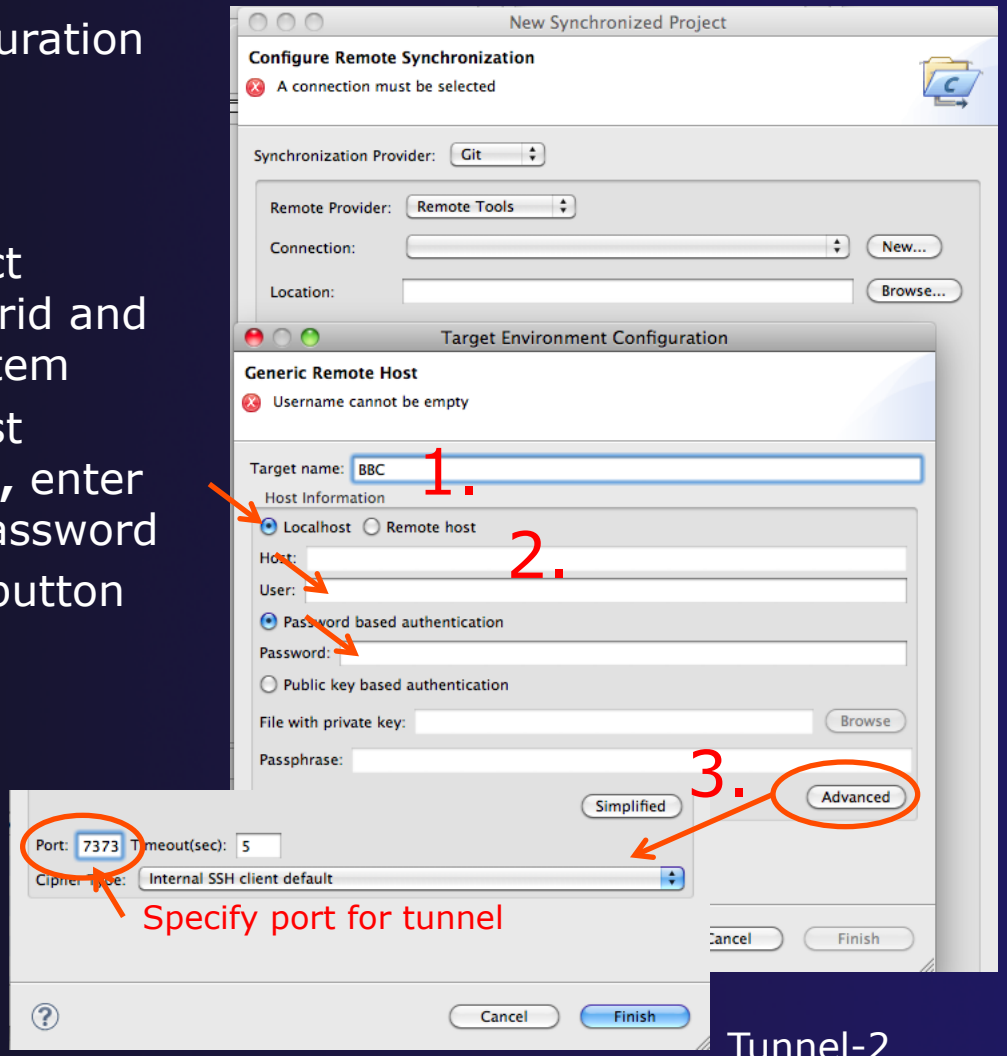
- When you configure the connection for the project
  - **Connection: New...**

- The connection will use the port for the ssh tunnel (details on next slide)



Configure connection to remote host – SSH Tunnel (2)

- In Target Environment Configuration dialog, enter target name, and host information
  - 1. Specify **Target name**
  - 2. If using a tunnel, select **Localhost** and enter userid and password for remote system
    - For direct access, just select **Remote Host,** enter hostname, userid, password
  - 3. select the **Advanced** button to specify the port
- Select **Finish**

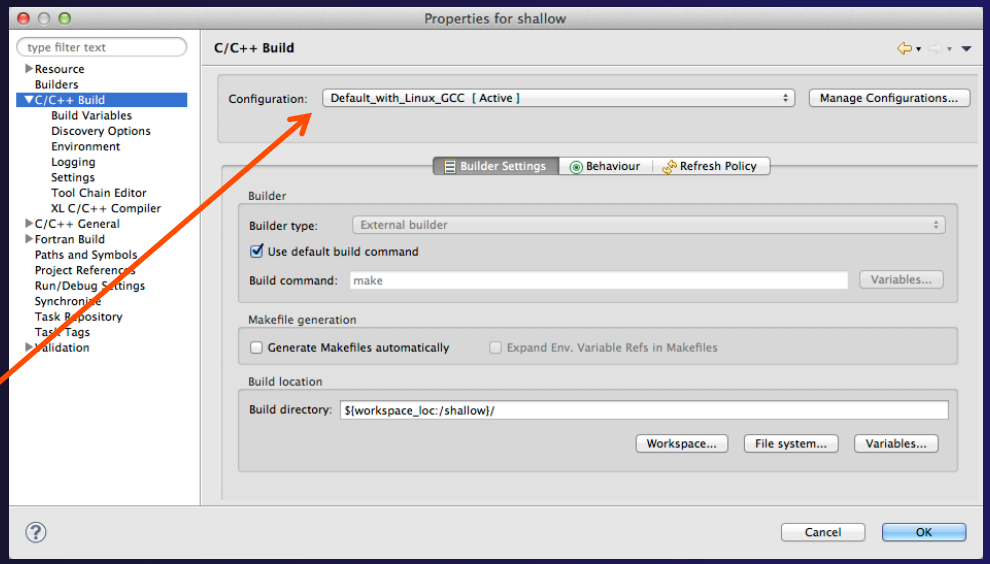


Building a Project

- 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

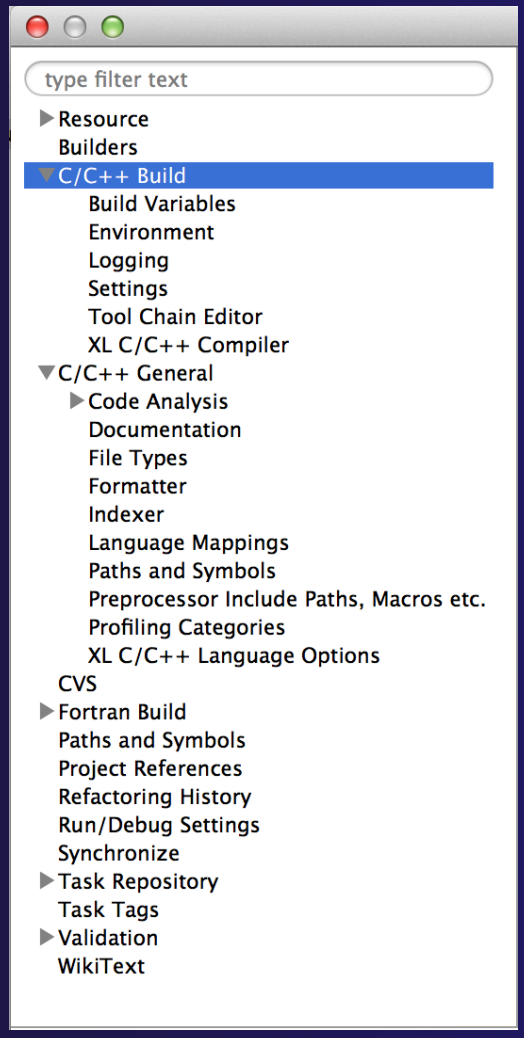
- A build configuration provides the necessary information to build the project
- The build configuration information is specified in the project properties
- Projects can have multiple build configurations, each configuration specifies a different set of options for a build
- Open the properties by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

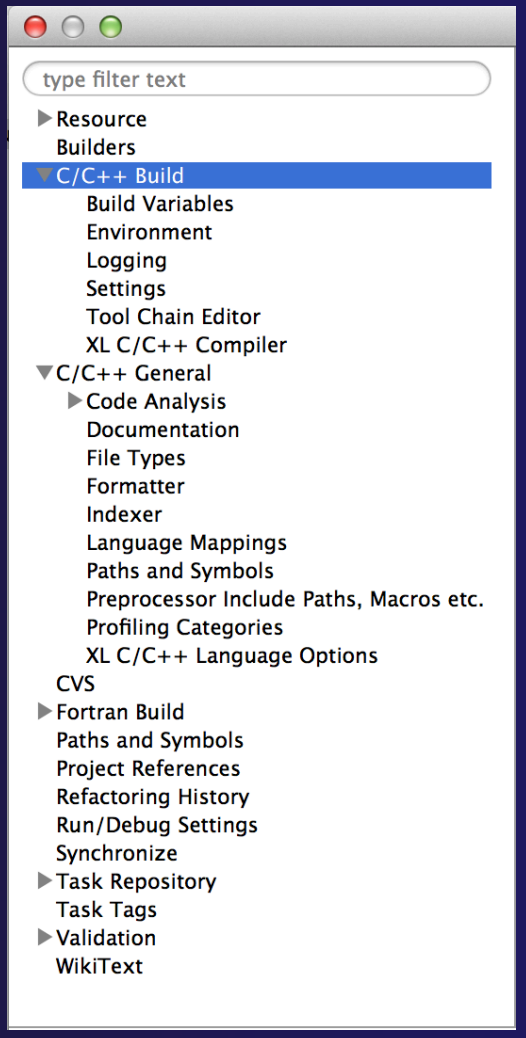
Build Properties (1)

- **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



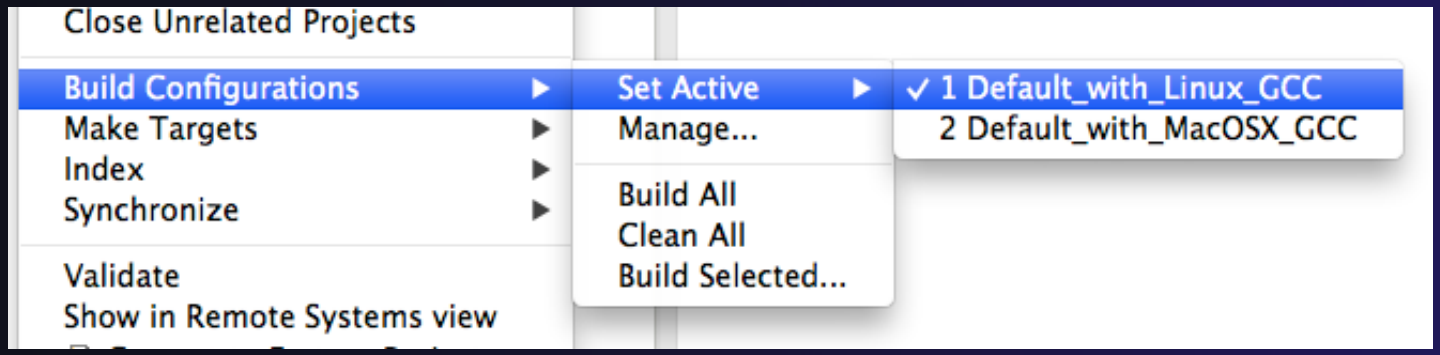
Build Properties (2)

- **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



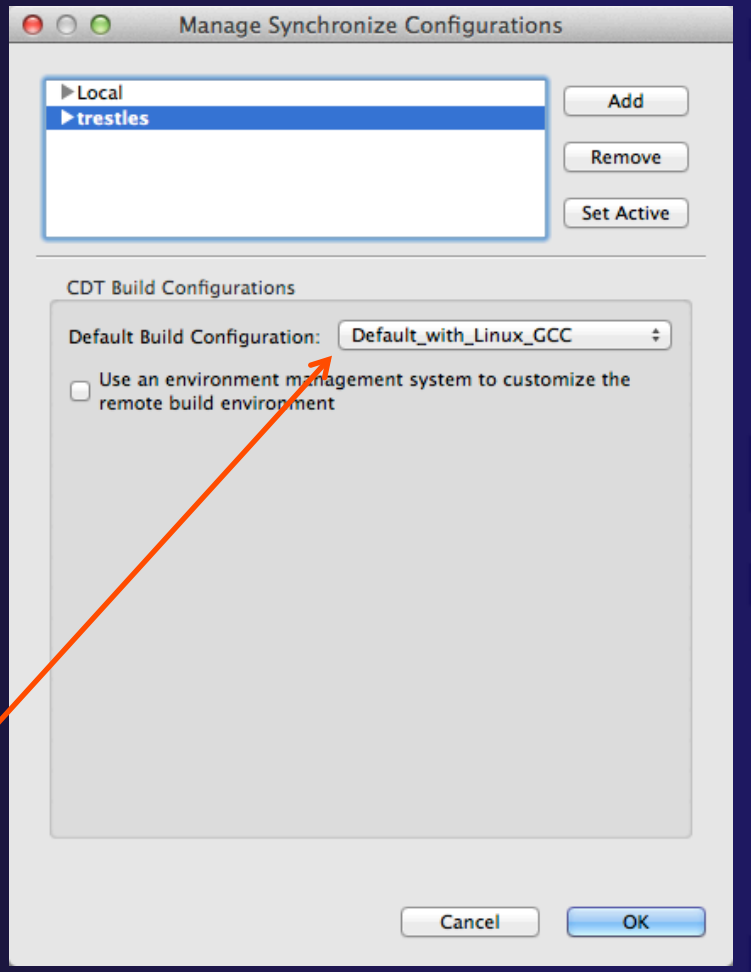
Selecting Build Configuration

- 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



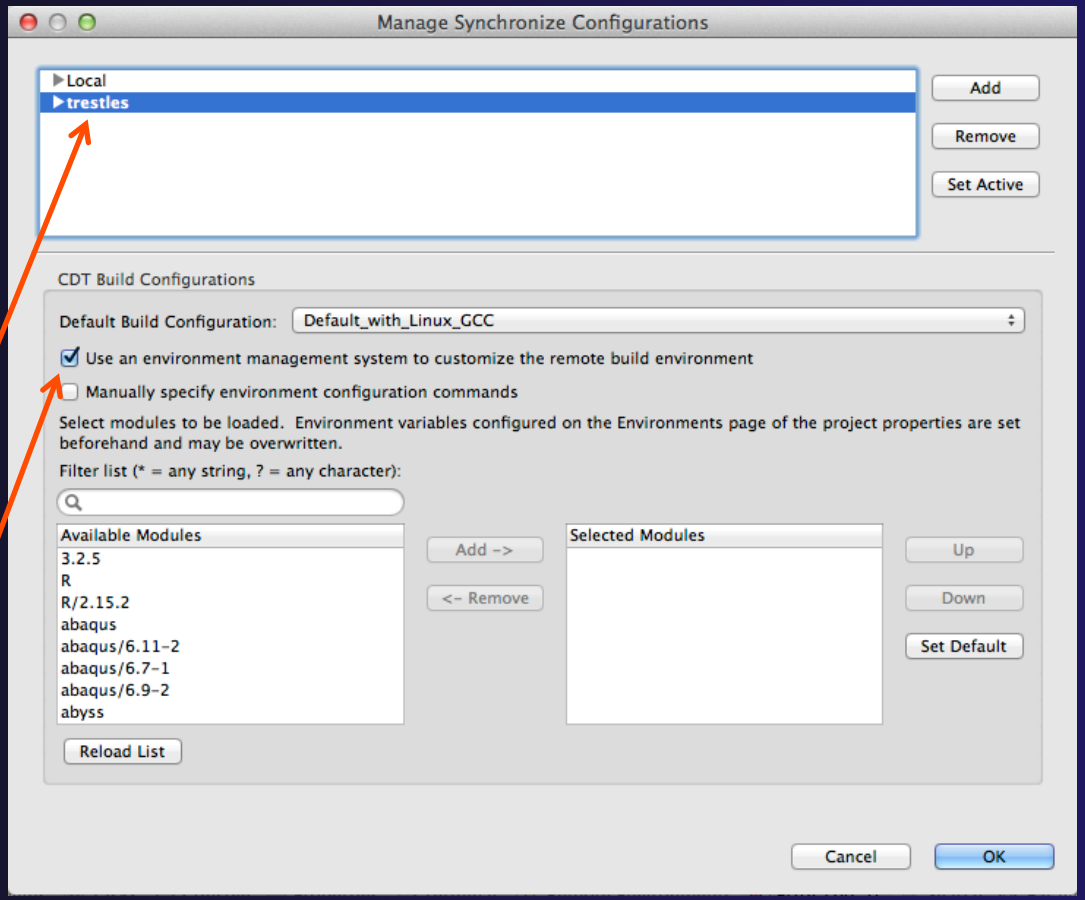
Building Synchronized Projects

- 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



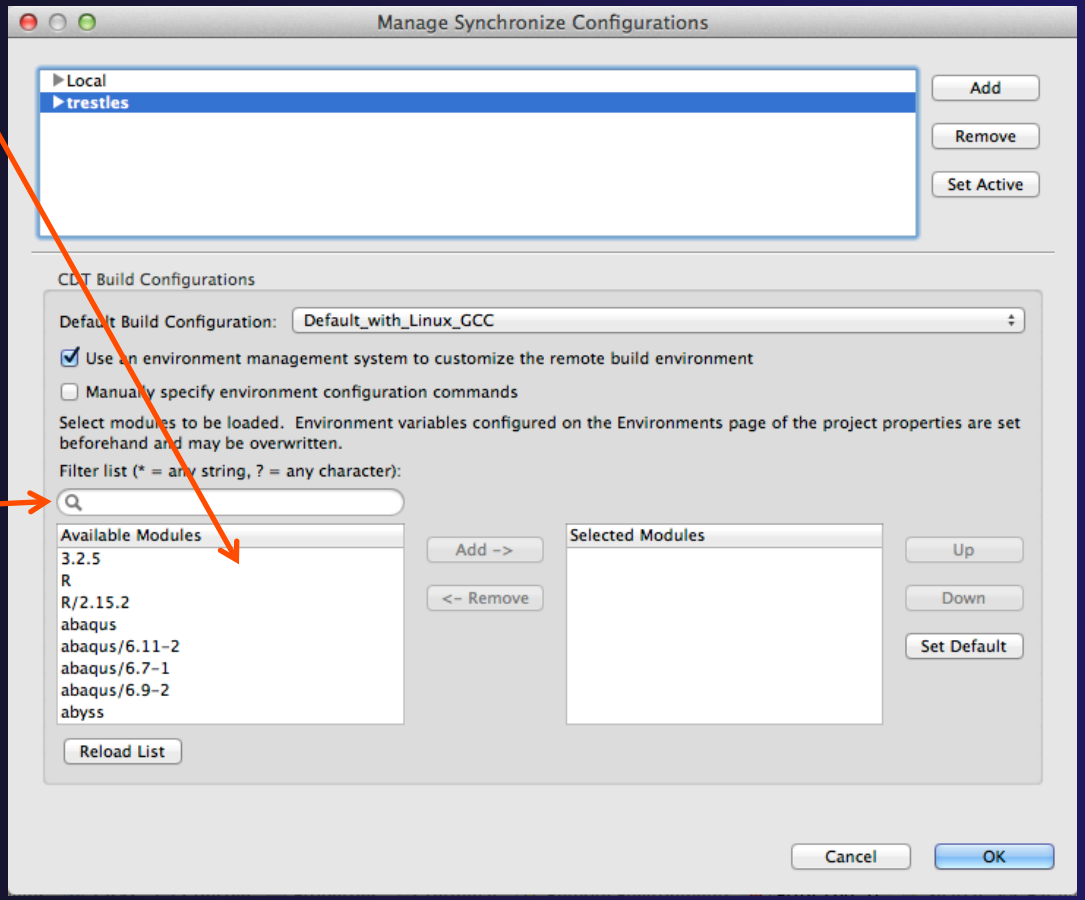
Configuring the Build Environment

- 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)

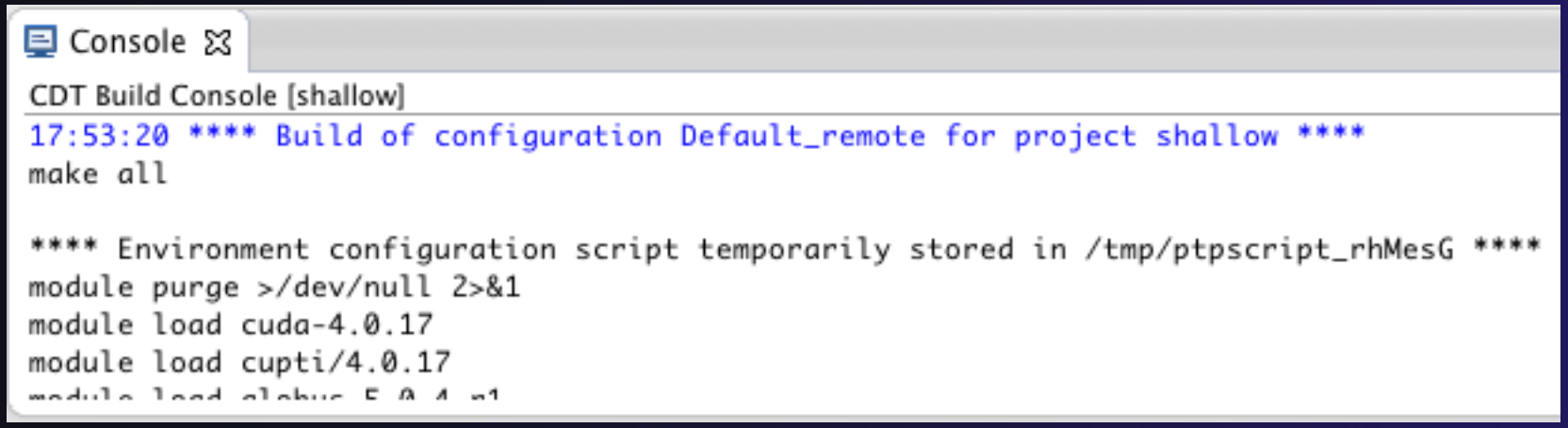
- Select a module from the **Available Modules** list and click the **Add->** button to add them to the **Selected Modules** list
- Use the **<-Remove** button to remove modules from the Selected Modules list
- Use the **Filter list** field to quickly find modules with a given name
- Use the **Up** and **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…

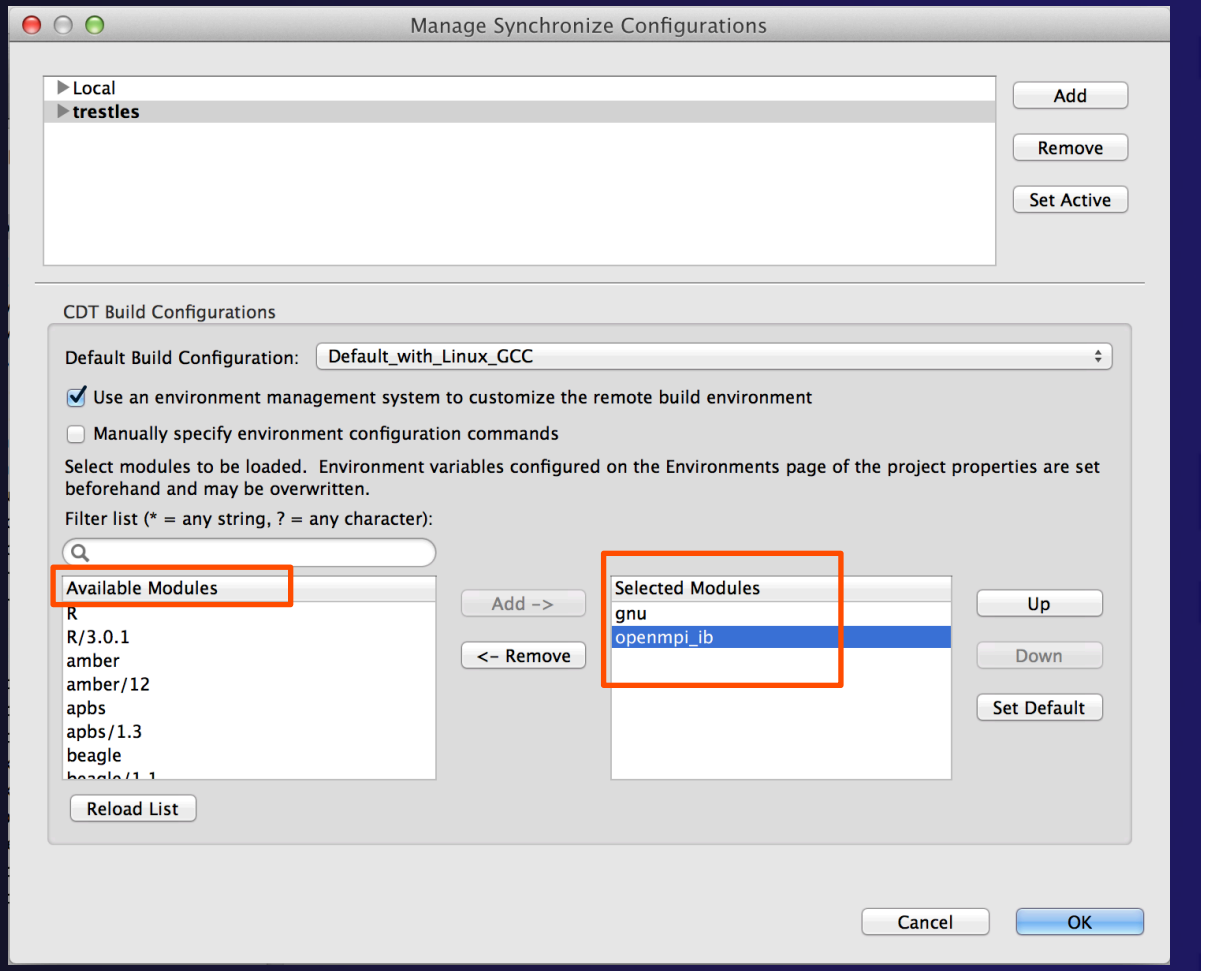
Build Environment (3)

- 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



Build Environment (4)

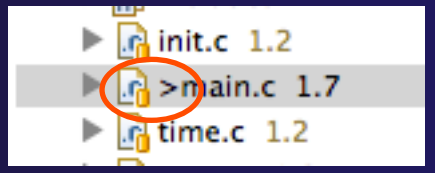
- For this tutorial, we want to use gcc and Open MPI
- To get to this dialog: Right mouse on Project, **Synchronize > Manage...**
- Navigate to **gnu** in **Available Modules** and select **Add ->**
- Navigate to openmpi_ib and select **Add ->**
- Assure the order matches this
  - If not, use **Up/Down**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

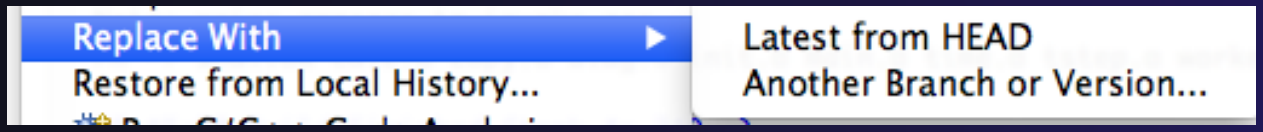


Start with original 'shallow'

- Start with original 'shallow' code:
  - Project checked out from CVS:
    - Right mouse on project,
      Replace with > Latest from HEAD

Changed file:



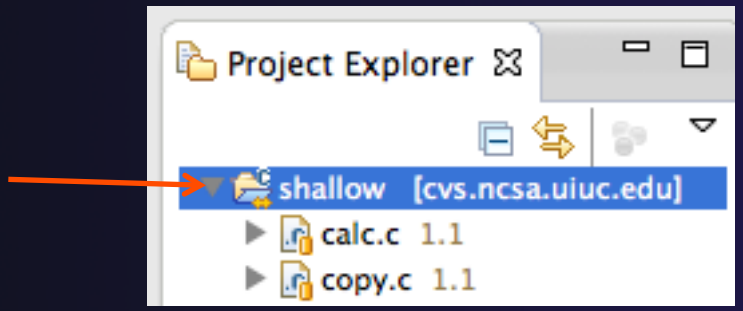


Also see Compare With …

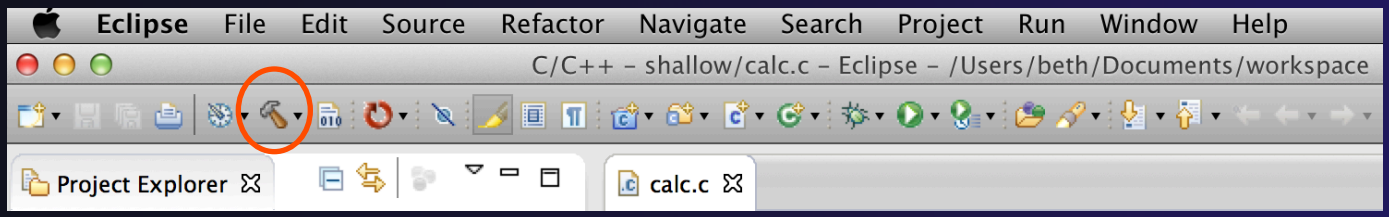
  - Other project:
    - Right mouse on project,
      Restore from local history – finds deleted files
    - Right mouse on file, **Compare With** or **Replace With**

Starting the Build 

- Select the project in Project Explorer



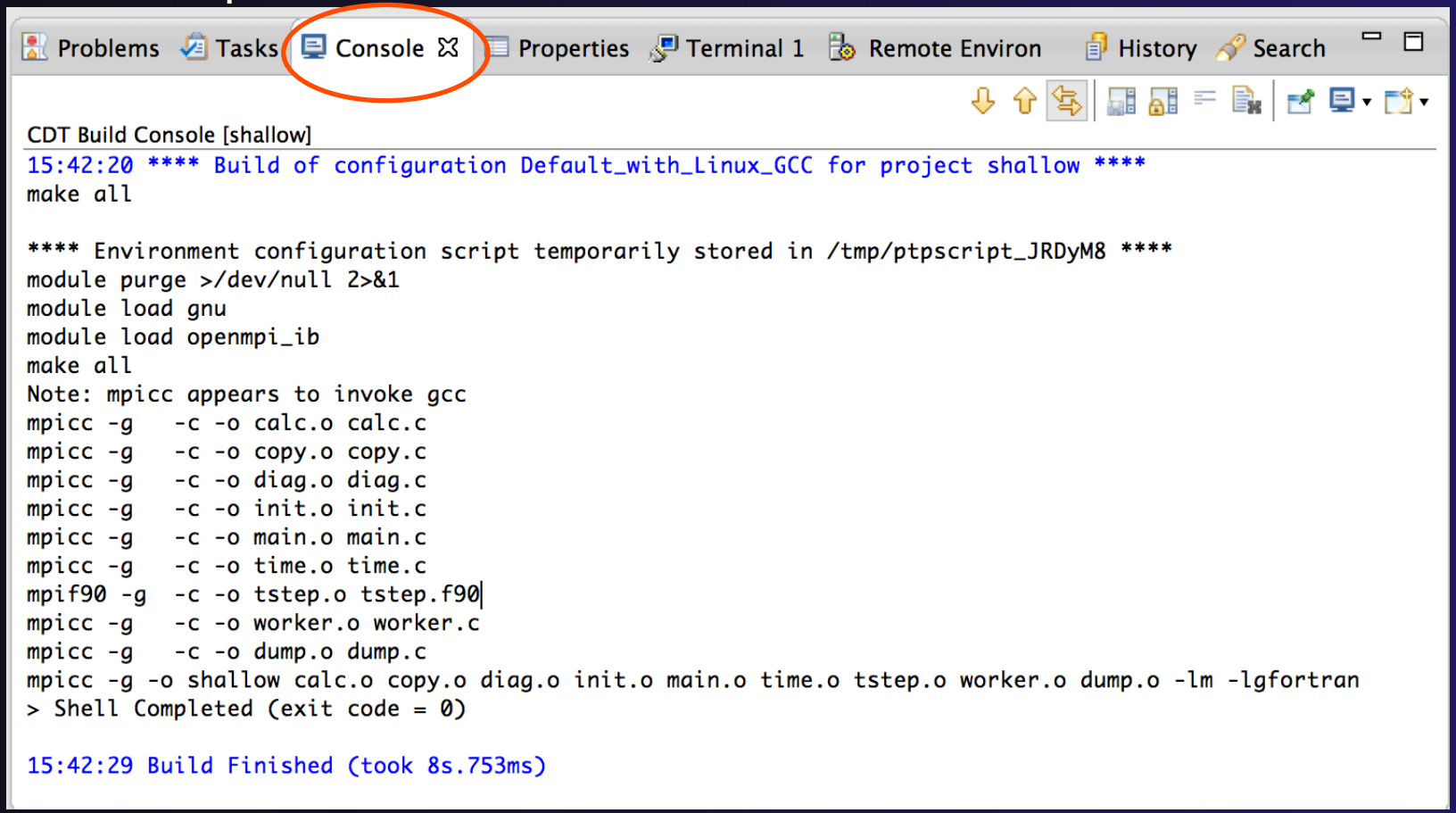
- Click on the  hammer button in toolbar to run a build using the active build configuration



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

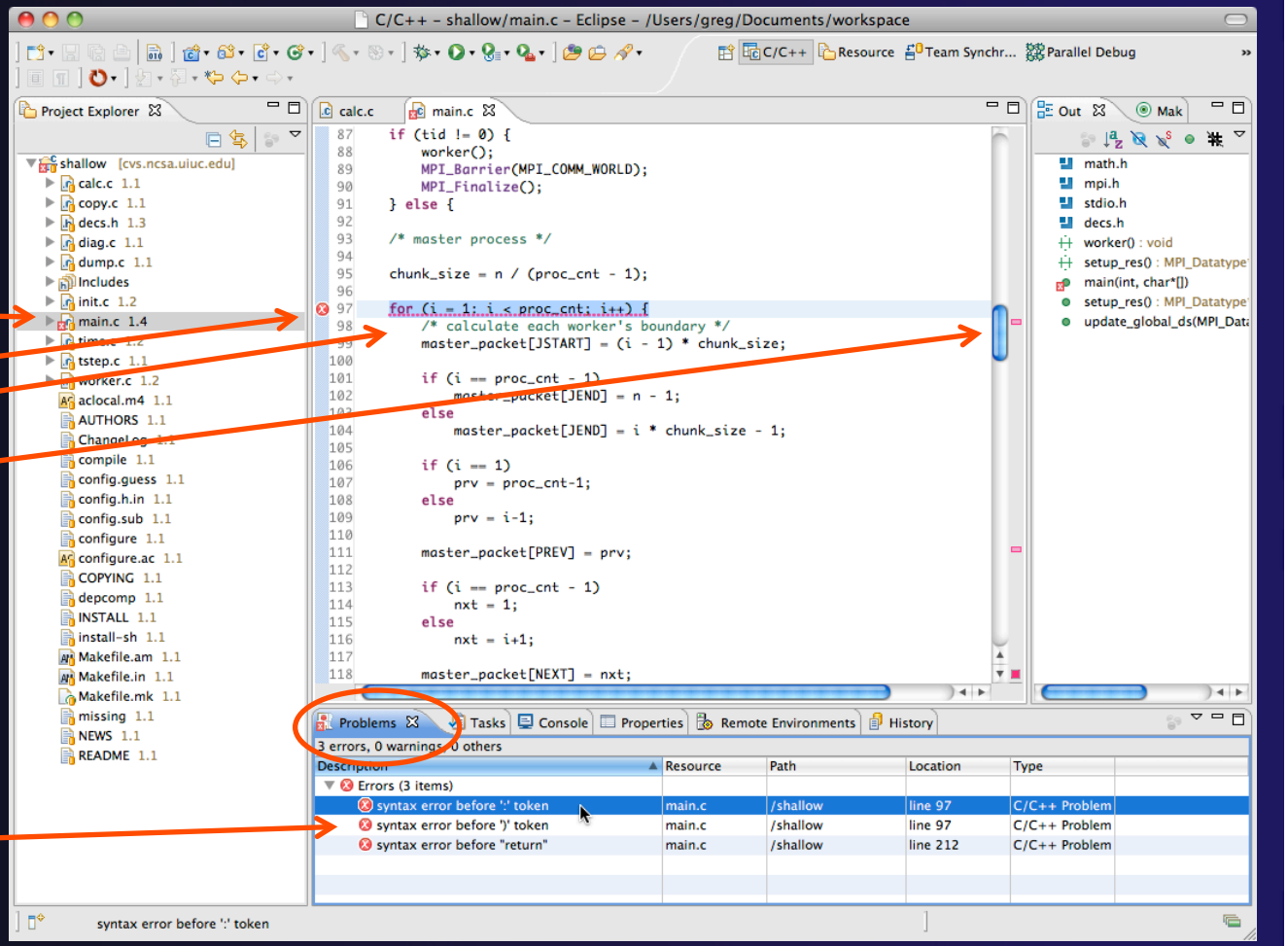
Viewing the Build Output

- Build output will be visible in console



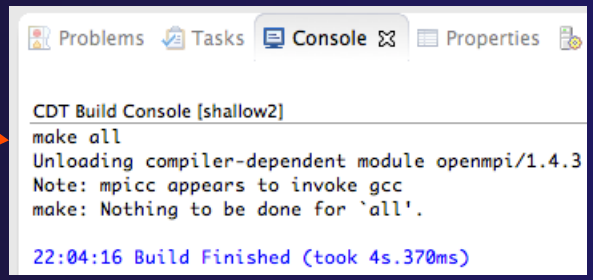
Build Problems

- 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

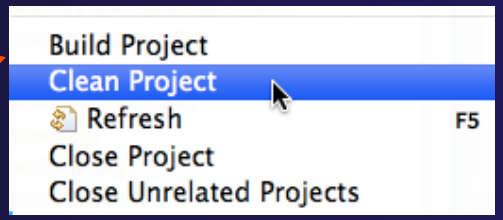


Forcing a Rebuild

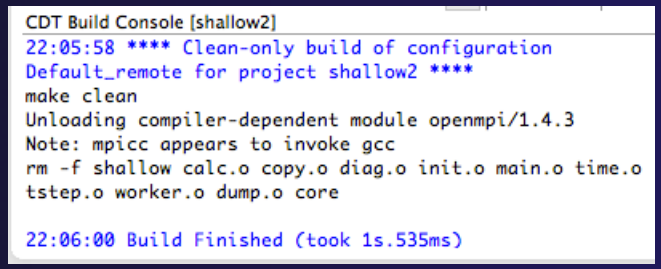
- If no changes have been made, `make` doesn't think a build is needed e.g. if you only change the Makefile



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



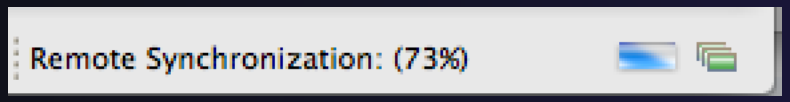
- Build console will display results

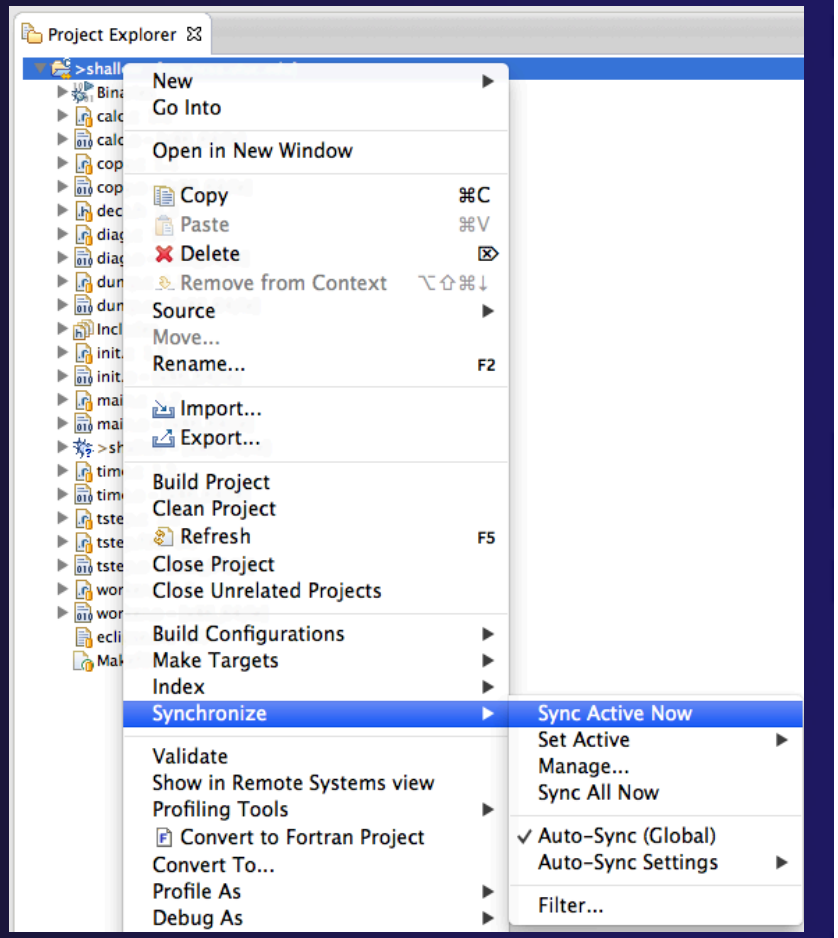


- Rebuild project by clicking on build button again 

Forcing a Resync

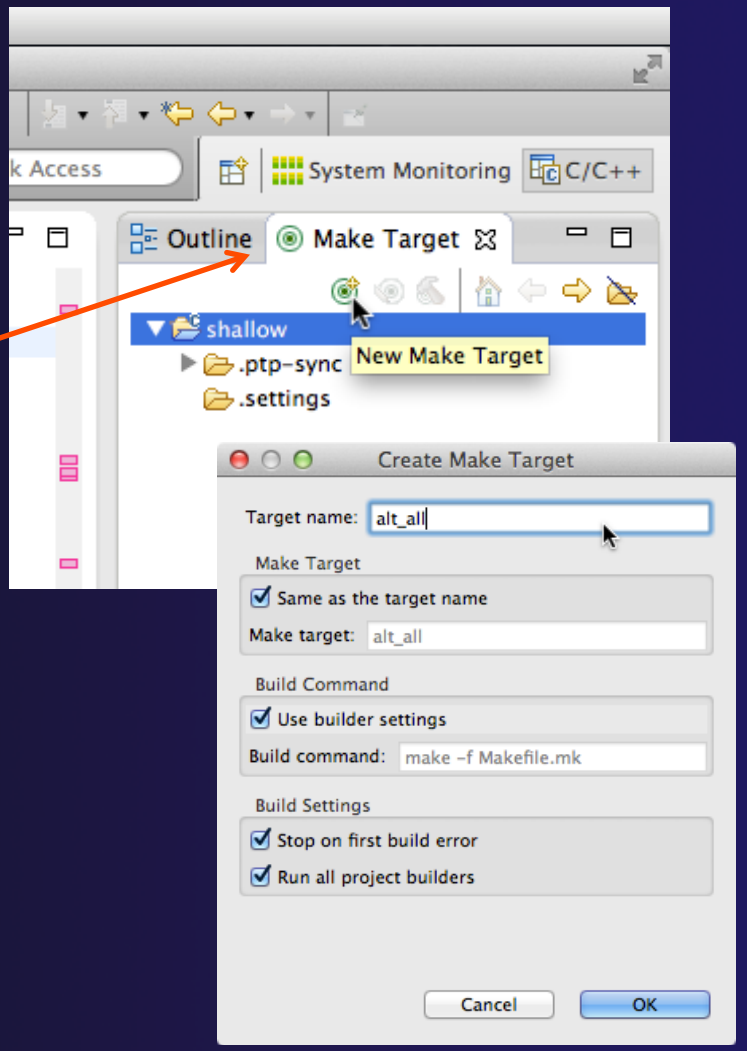
- Project should resync with remote system when things change
- Sometimes you may need to do it explicitly
- Right mouse on project, **Synchronize>Sync Active Now**
- Status area in lower right shows when Synchronization occurs





Creating Make Targets

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



Exercise

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

Optional Exercise

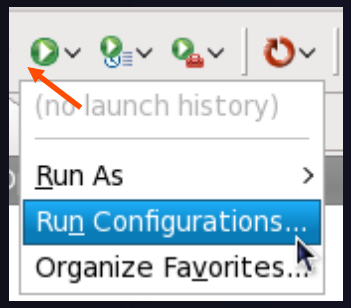
1. Open the Makefile in Eclipse. Note the line starting with "tags:" – this defines a make target named **tags**.
2. Open the **Outline** view while the Makefile is open. What icon is used to denote make targets in the Outline?
3. Right-click the **tags** entry in the Outline view. Add a Make Target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

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

Running an Application

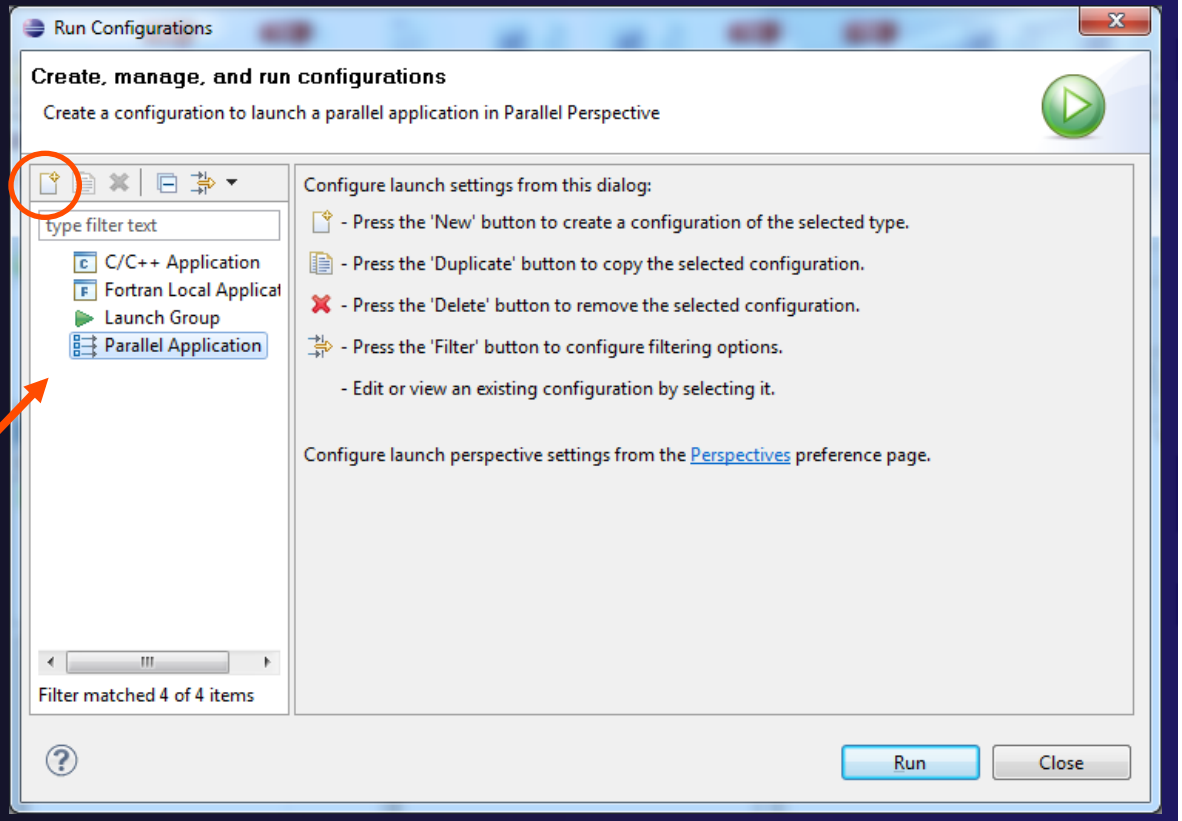
- Objective
  - Learn how to run an MPI program on a remote system
- Contents
  - Creating a run configuration
  - Configuring the application run
  - Monitoring the system and jobs
  - Controlling jobs
  - Obtaining job output

Creating a Run Configuration



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

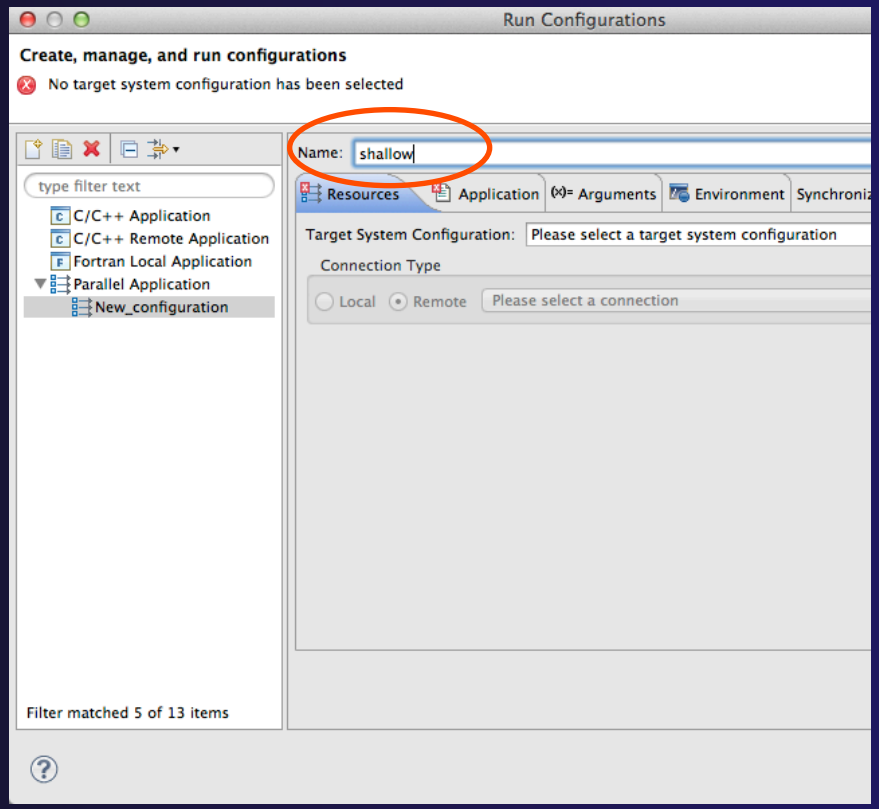
  Or, just double-click on **Parallel Application** to create a new one



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

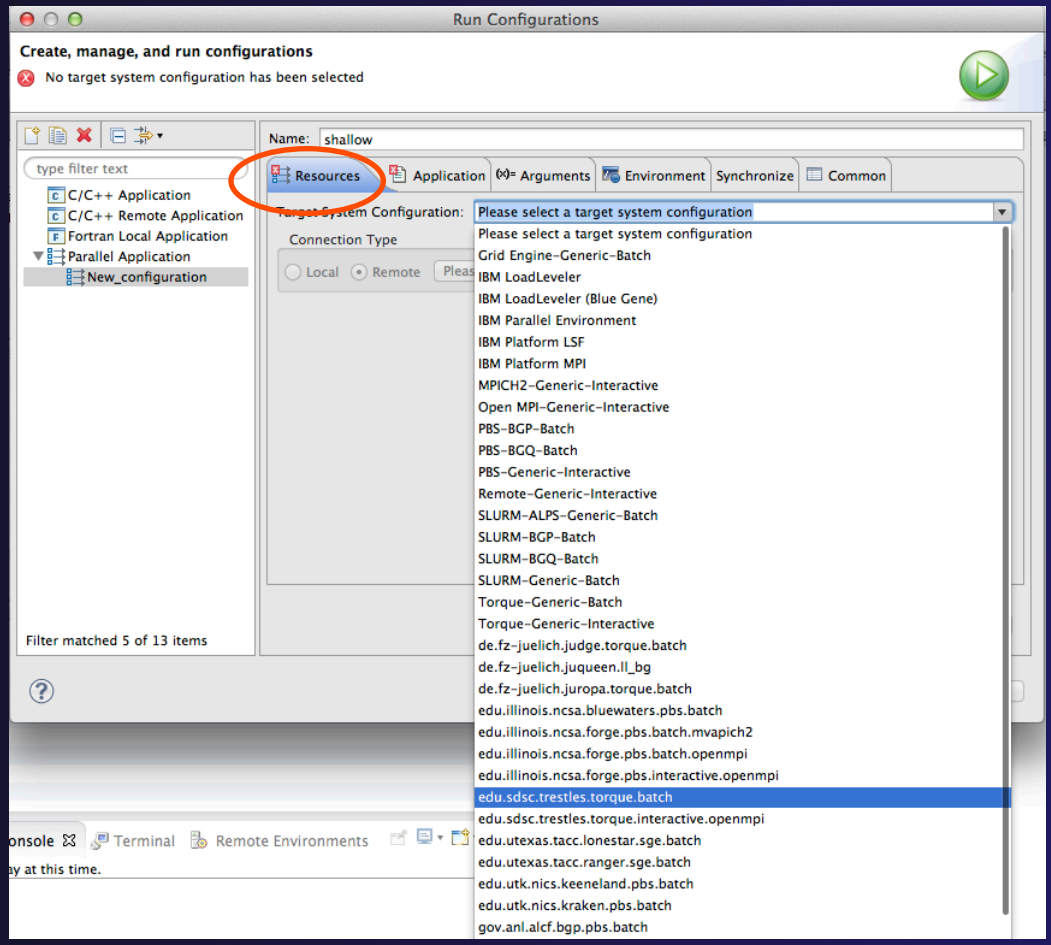
Set Run Configuration Name

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



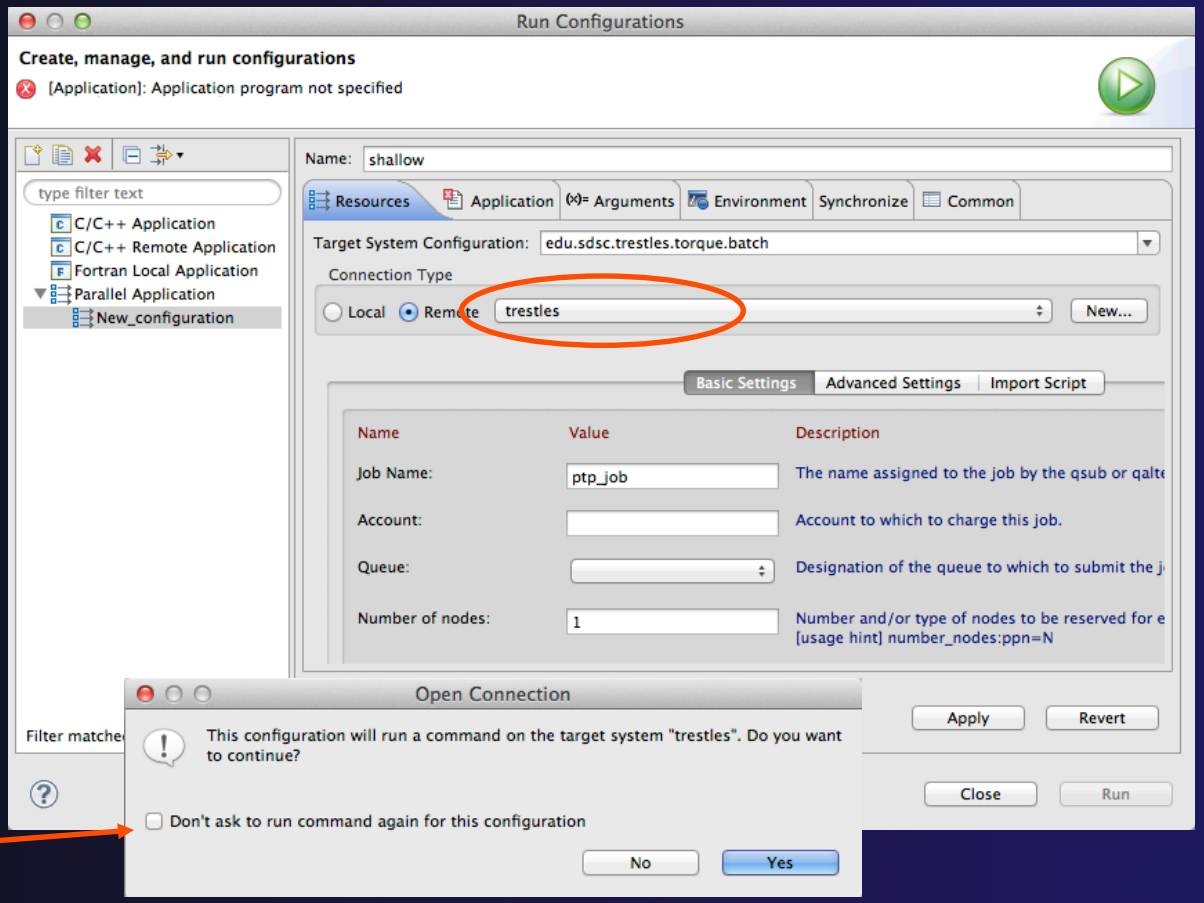
Configuring the Target System

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



Configure the Connection

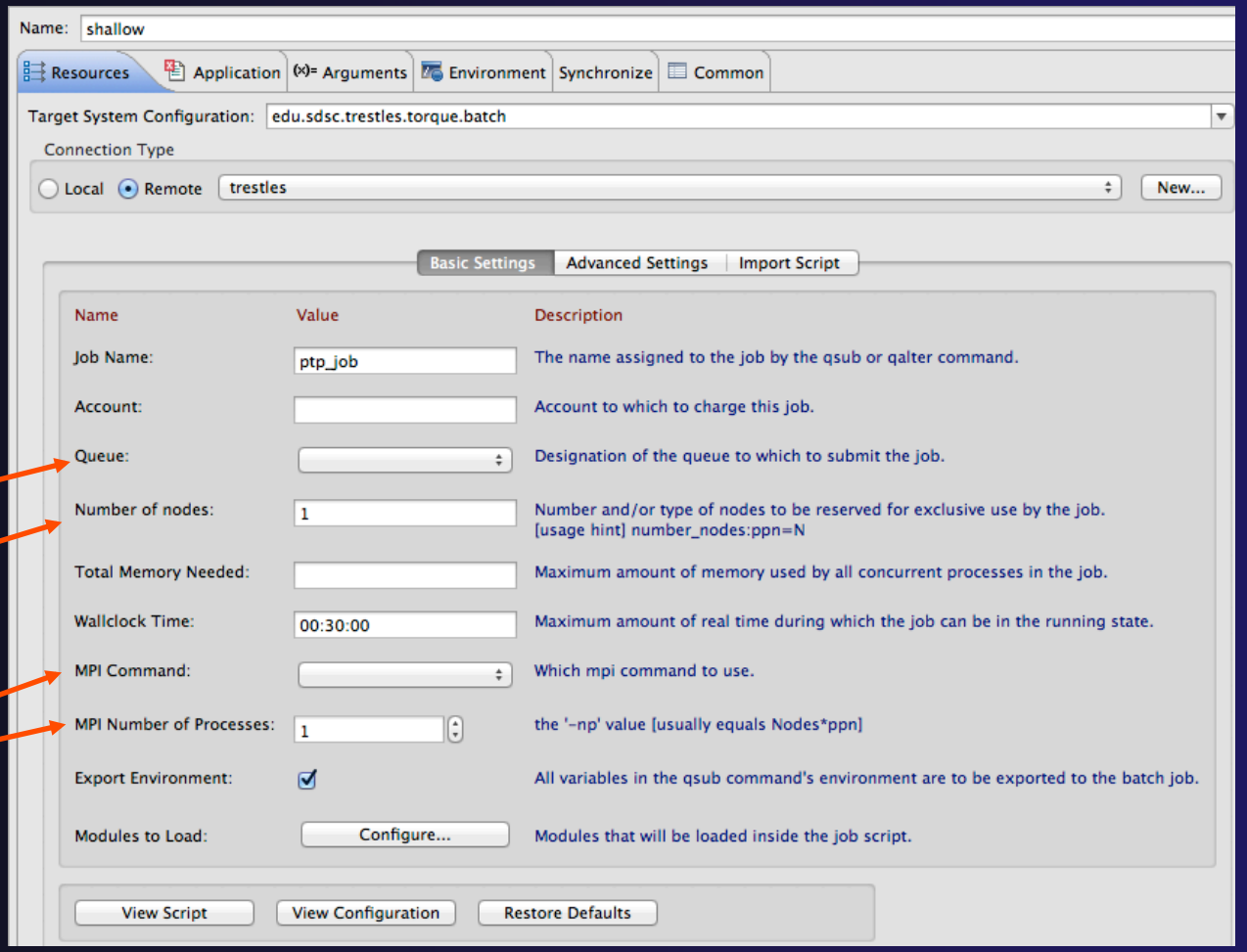
- Choose a connection to use to communicate with the target system
- If no connection has been configured, click on the **New** button to create a new one
  - Fill in connection information, then click ok
- The new connection should appear in the dropdown list
- Select the connection you already have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



Resources Tab

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

See next slide

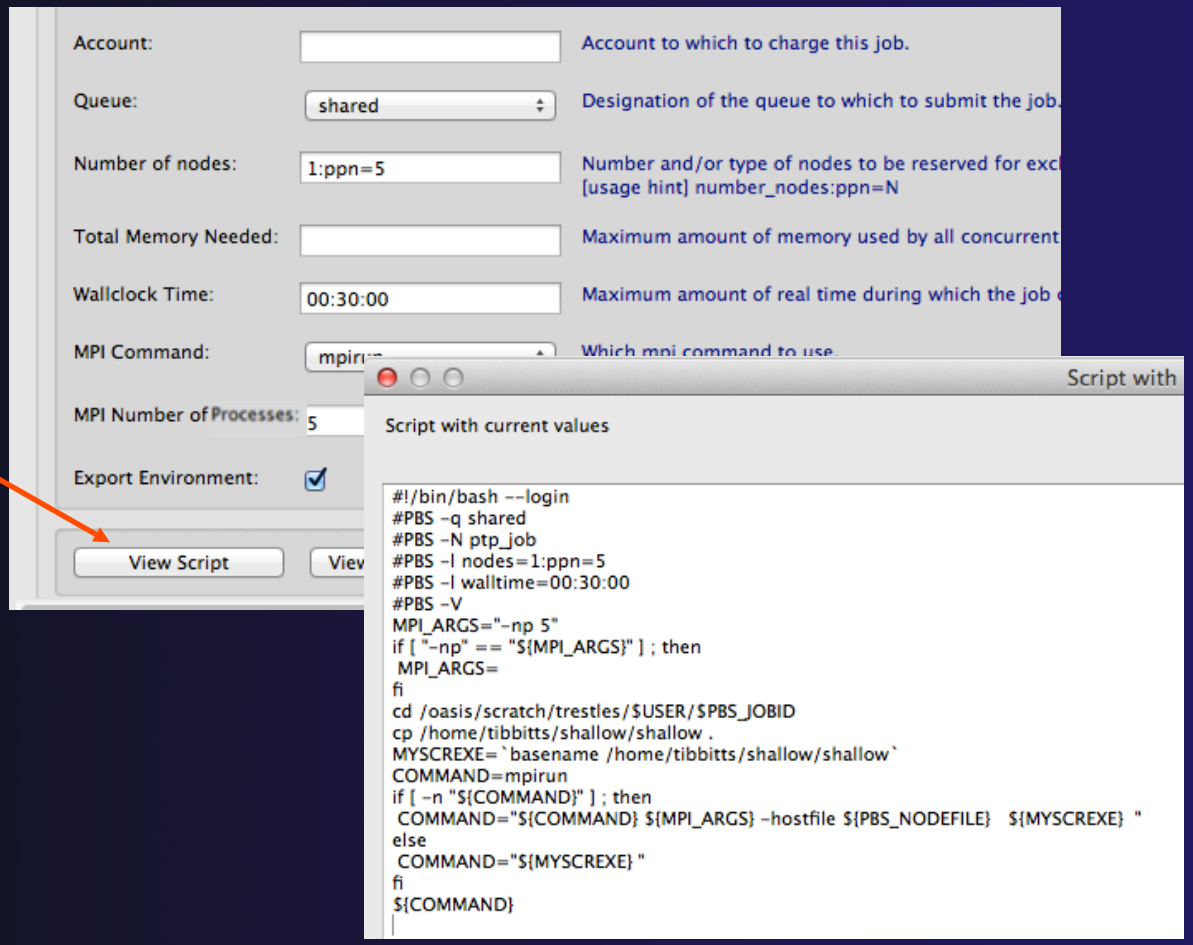


Resources Tab (2)

- For this tutorial, use the following values
  - *Queue:* **shared**
  - *Number of nodes:* **1:ppn=5**
  - *MPI Command:* **mpirun**
  - *MPI Number of Processes:* **5**
  - Leave other fields alone
- Configure modules for running the application
  - Click on the *Modules to Load:* **Configure...** button
  - Select the same modules used to build
    - gnu
    - openmpi_ib

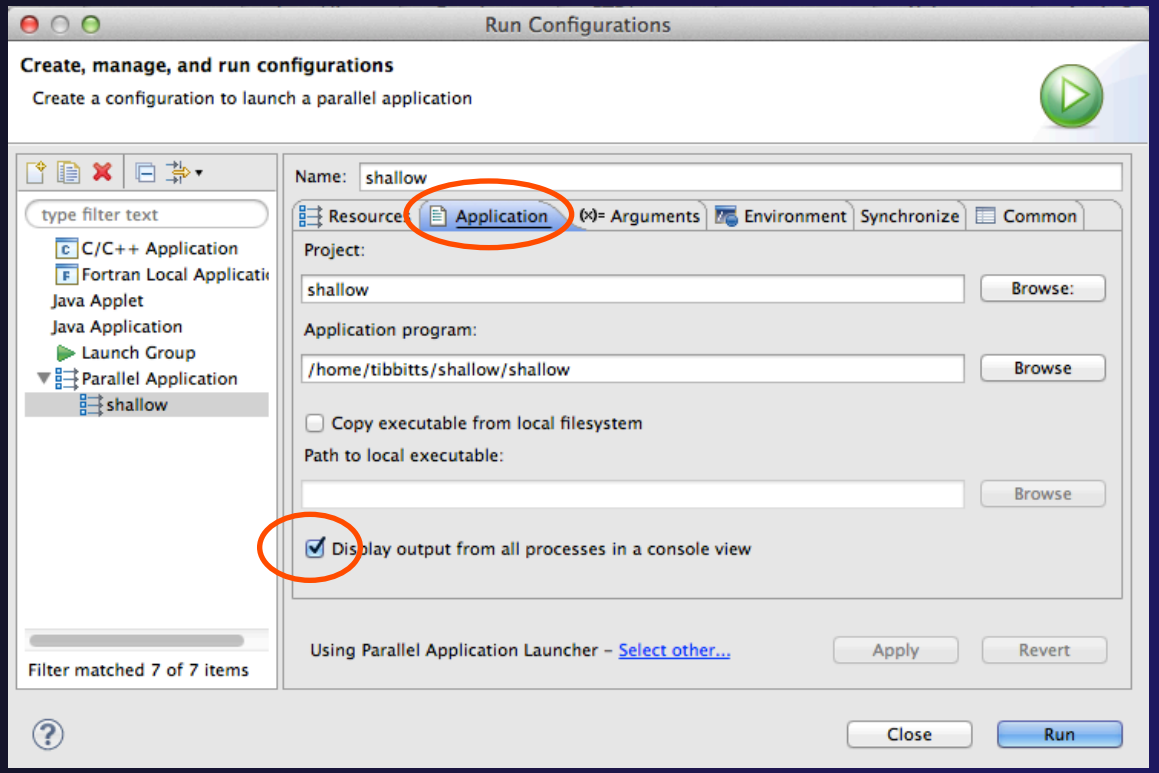
Viewing the Job Script

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



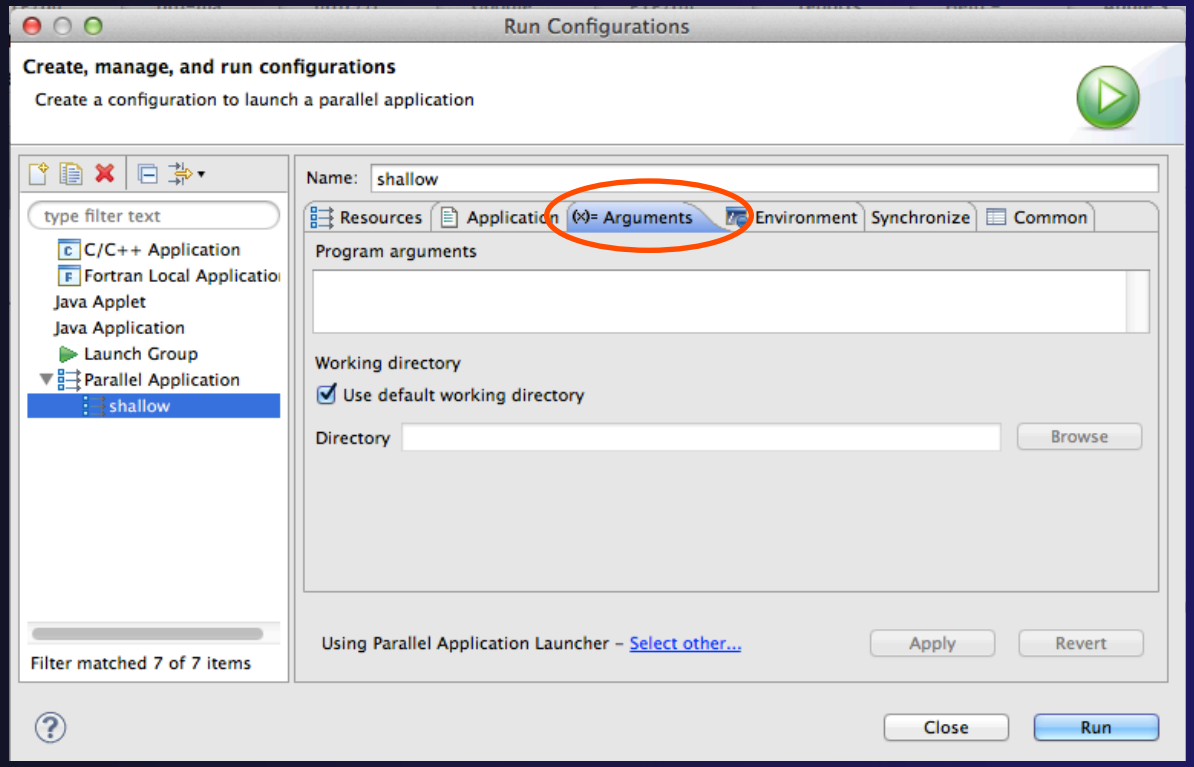
Application Tab

- 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**



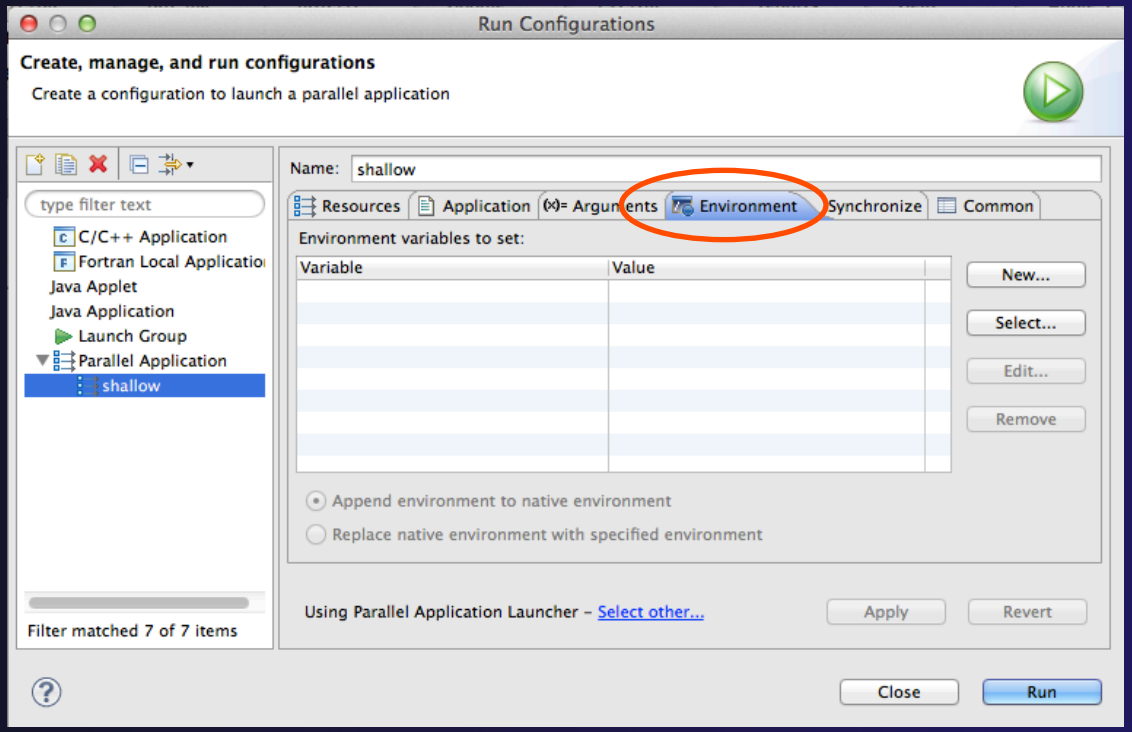
Arguments Tab (Optional)

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



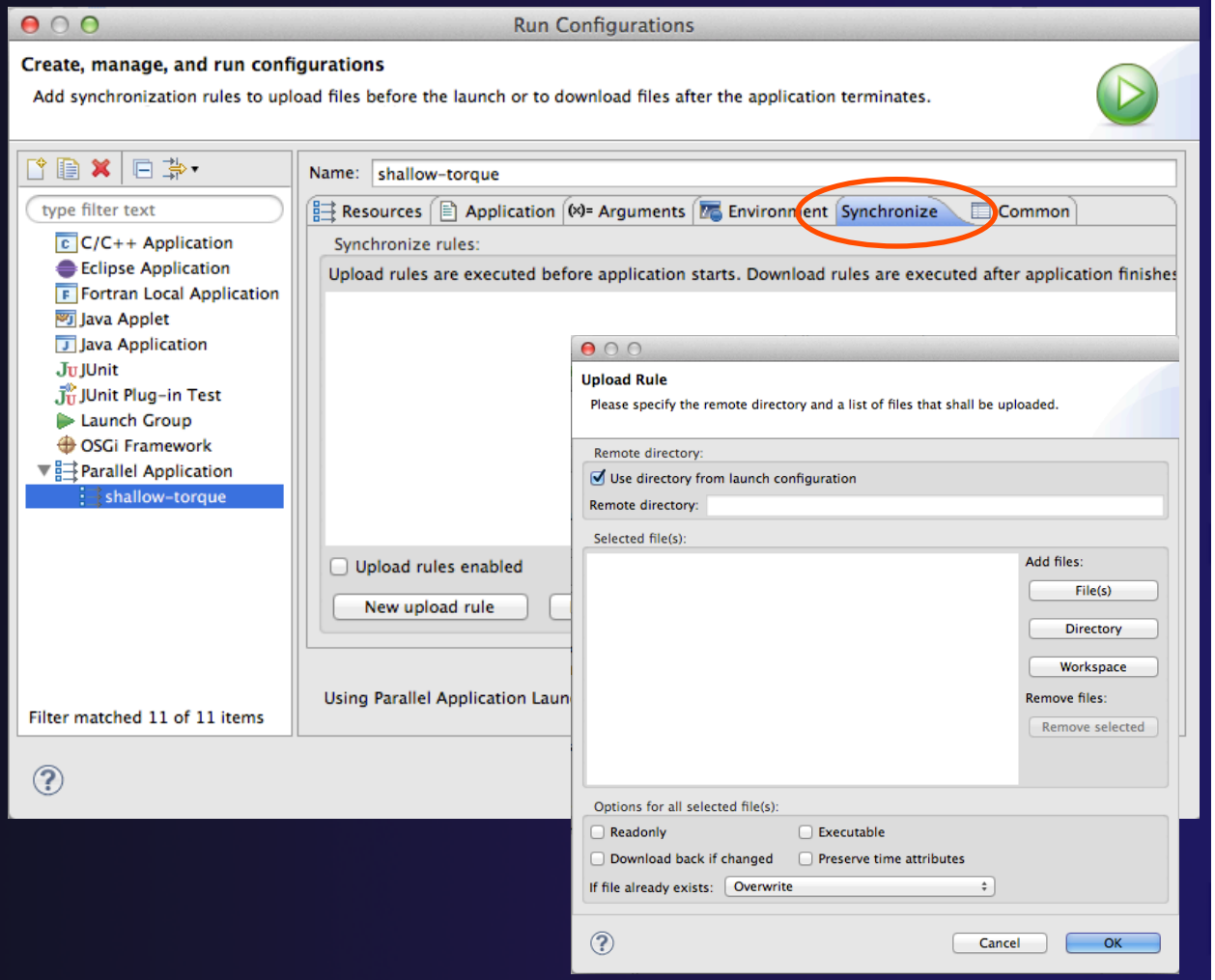
Environment Tab (Optional)

- 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



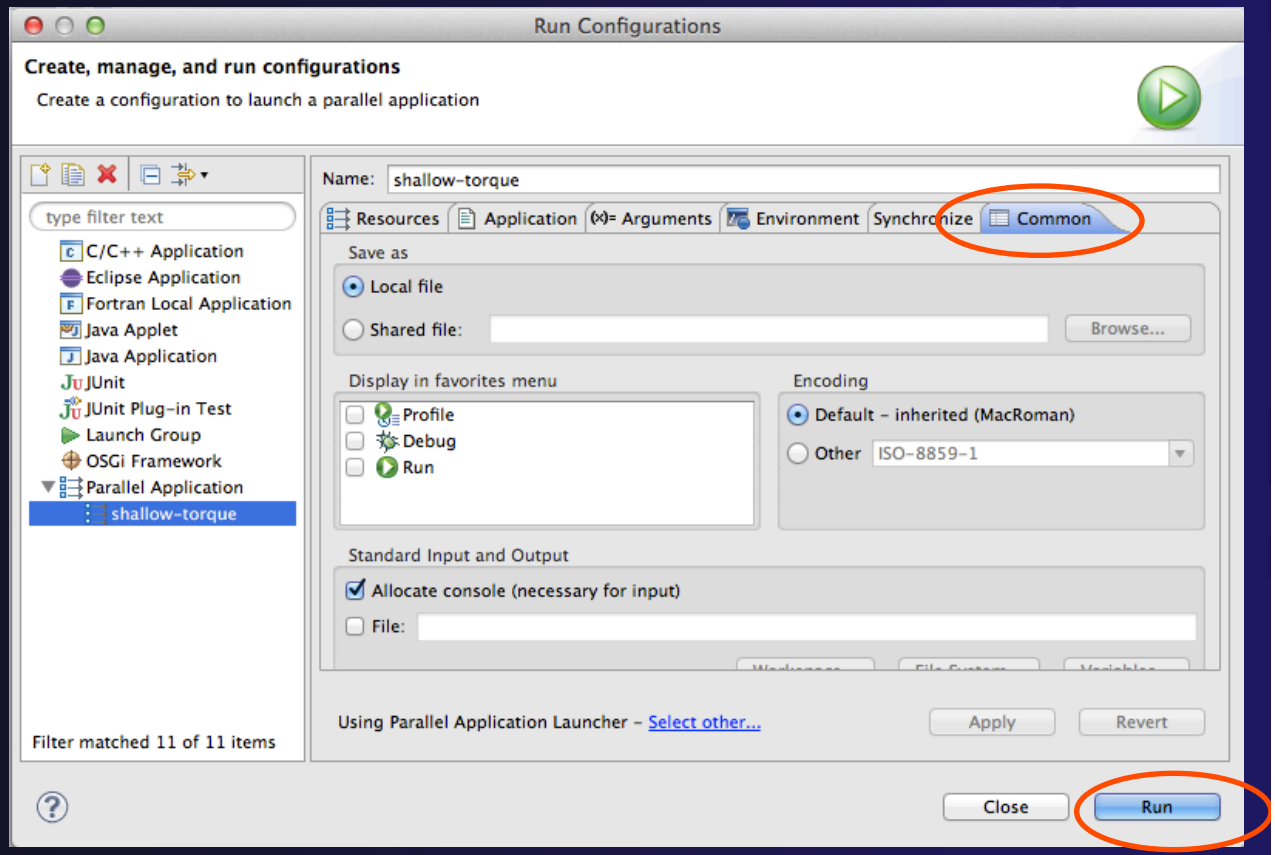
Synchronize Tab (Optional)

- 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



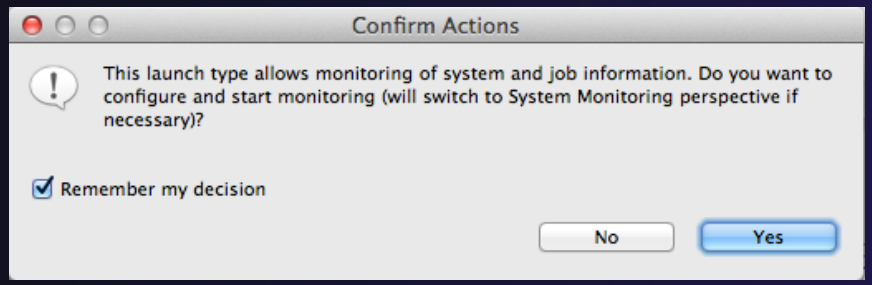
Common Tab (Optional)

- 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

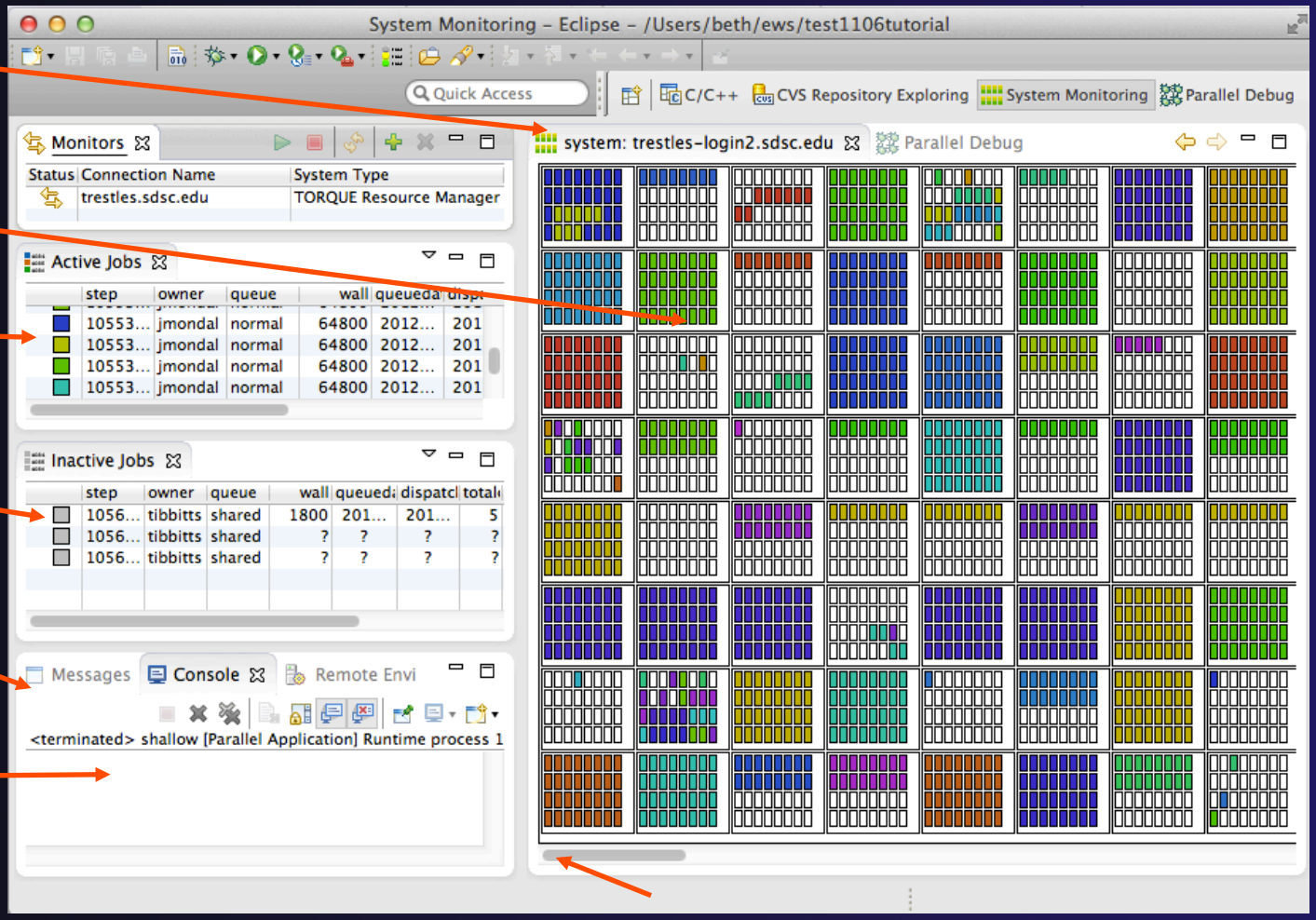
- Select **Run** to launch the job
- You may be asked to switch to the System Monitoring Perspective



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

System Monitoring Perspective

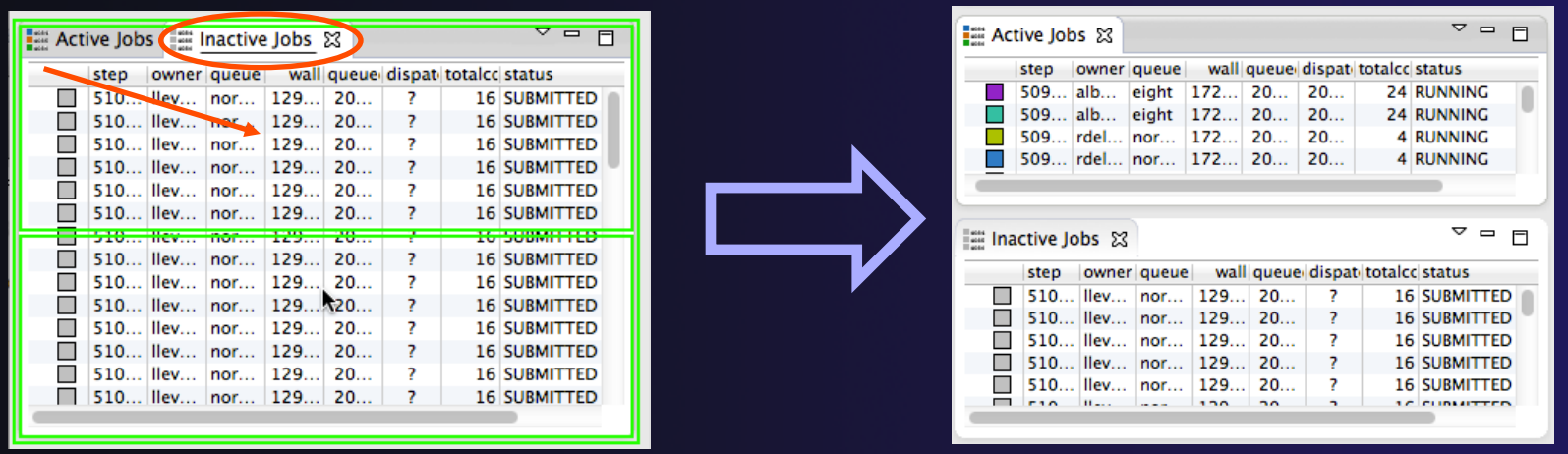
- System view
- Jobs running on system
- Active jobs
- Inactive jobs
- Messages
- Console



Scroll to see more

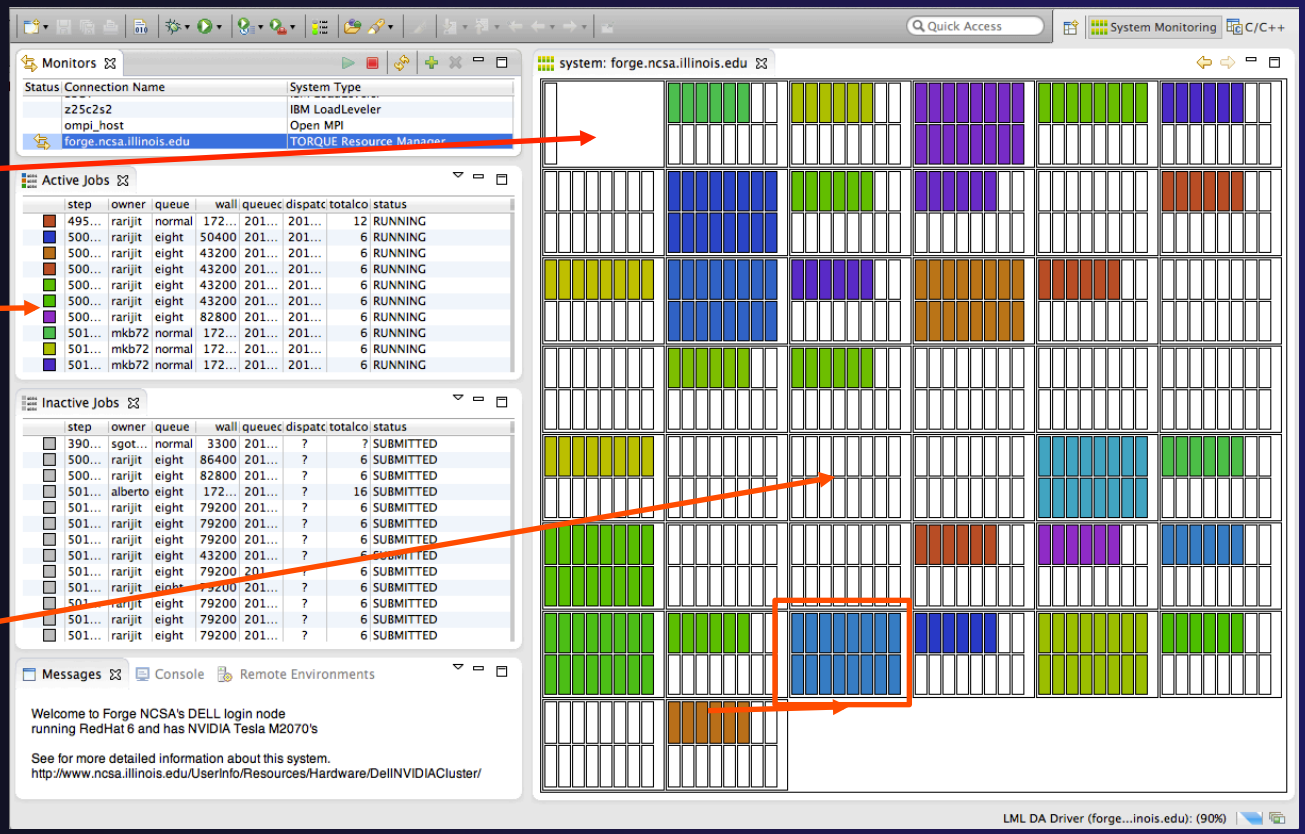
Moving views

- 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



System Monitoring

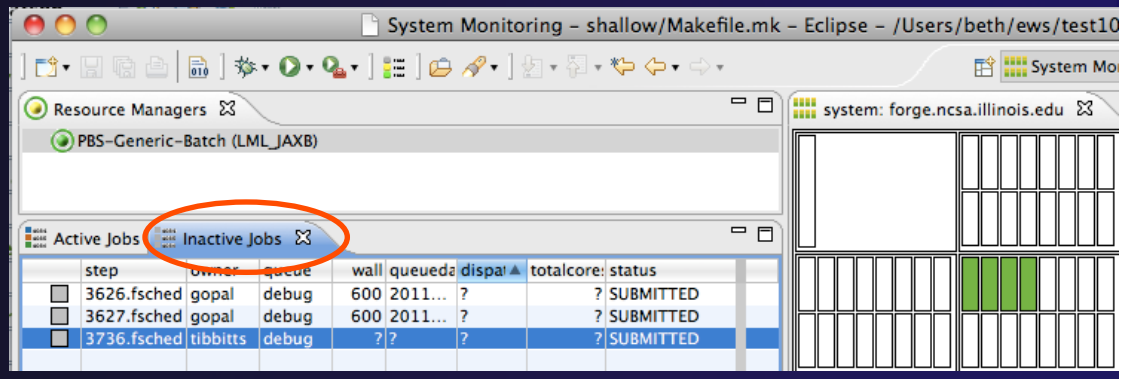
- **System** view, with abstraction of system configuration
- Hold mouse button down on a job in **Active Jobs** view to see where it is running in **System** view
- Hover over node in **System** view to see job running on node in **Active Jobs** view

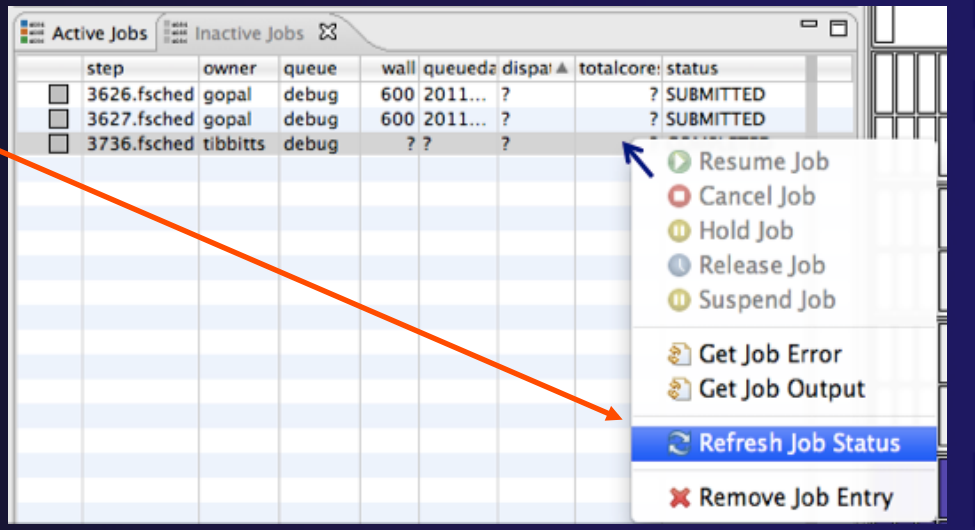


One node with
16 cores

Job Monitoring

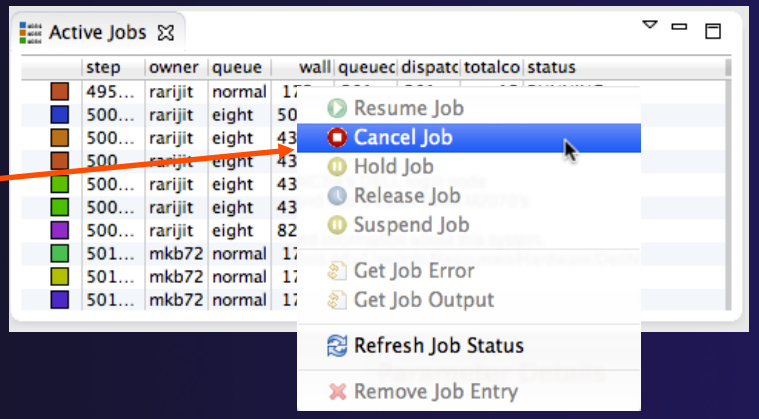
- Job initially appears in **Inactive Jobs** view
- Moves to the **Active Jobs** view when execution begings
- Returns to **Inactive Jobs** view on completion
- Status refreshes automatically every 60 sec
- Can force refresh with menu

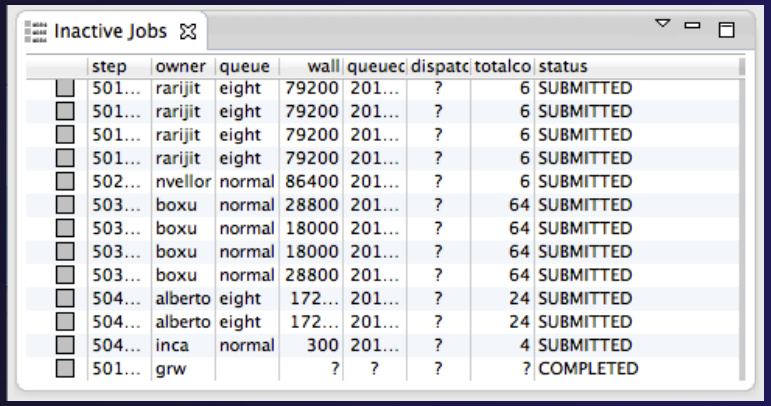




Controlling Jobs

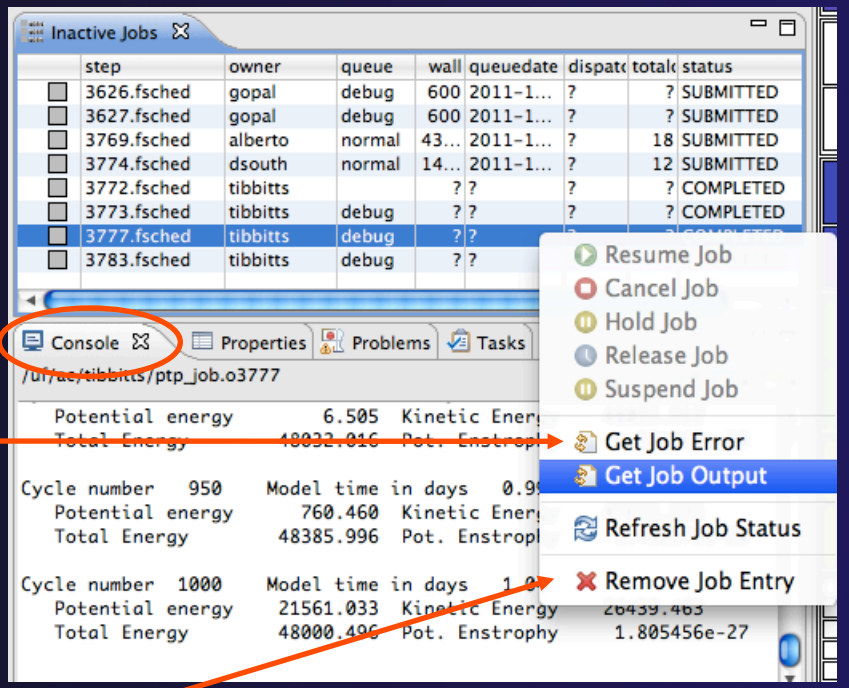
- Right click on a job to open context menu
- Actions will be enabled IFF
  - The job belongs to you
  - The action is available on the target system
  - The job is in the correct state for the action
- When job has COMPLETED, it will remain in the **Inactive Jobs** view





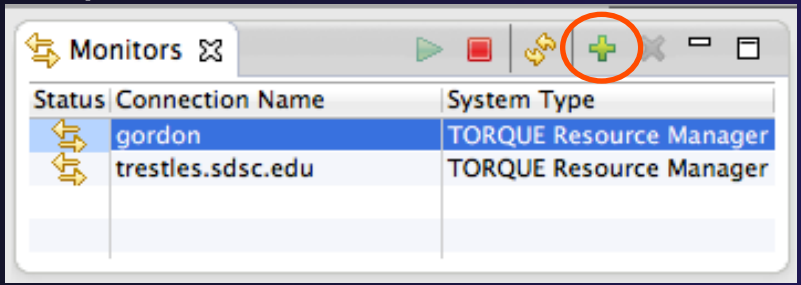
Obtaining Job Output

- 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**

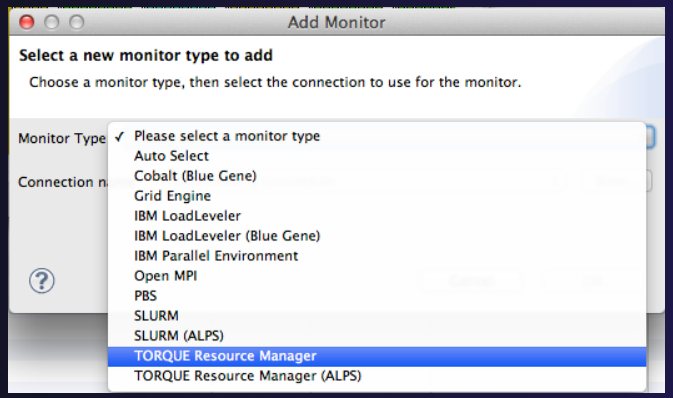


Add a Monitor

- 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

Fortran

- Objectives
  - Learn how to create and convert Fortran projects
  - Learn to use Fortran-specific editing features
  - Learn about Fortran-specific properties/preferences
- Contents
  - Fortran projects
  - Using the Fortran editor
  - Fortran project properties and workbench preferences
- Prerequisites
  - Basics (for exercises)

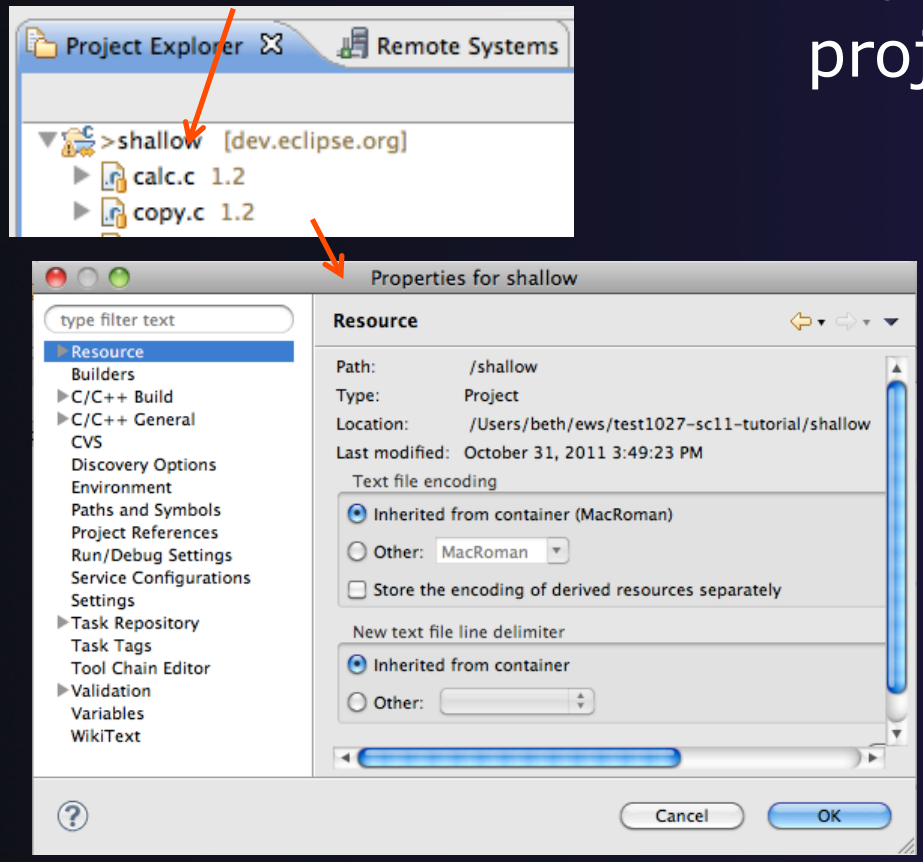
Ralph Johnson's research group at UIUC used to meet at Pho-Tran…

…which became the name of their Fortran IDE.

Configuring Fortran Projects

Project Properties

- Right-click Project
- Select **Properties**…

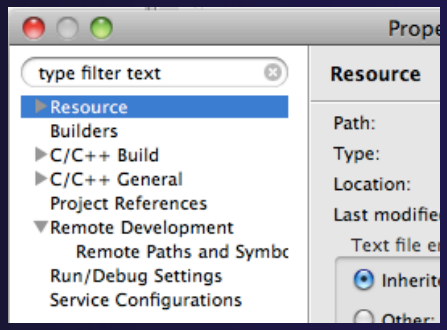


- *Project properties* are settings that can be changed for each project
  - Contrast with *workspace preferences*, which are the same regardless of what project is being edited
    - e.g., editor colors
    - Set in **Window ▸ Preferences** (on Mac, **Eclipse ▸ Preferences**)
    - Careful! Dialog is very similar

Converting to a Fortran Project

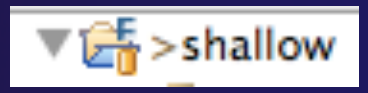
- Are there categories labeled **Fortran General** and **Fortran Build** in the project properties?

No Fortran categories →



Do this once

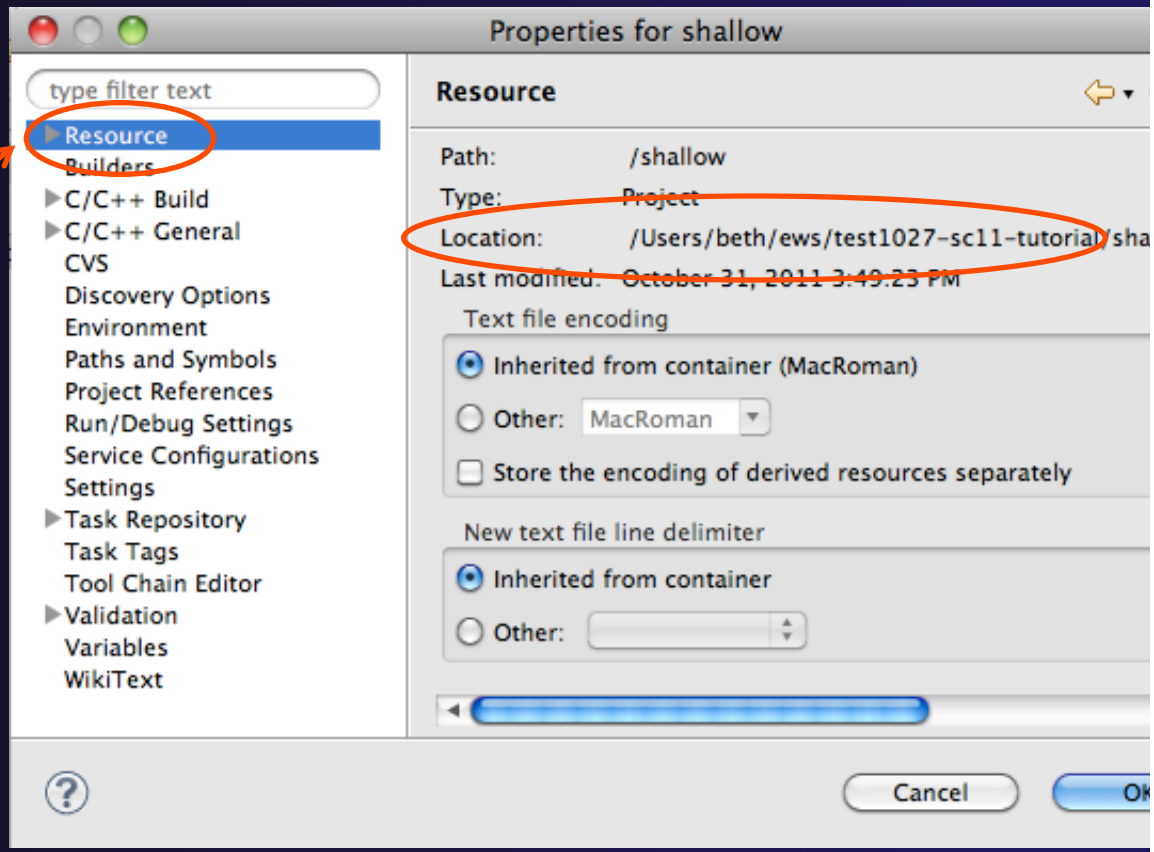
- If not, the project is not a Fortran Project
  - Switch to the Fortran Perspective
  - In the Fortran Projects view, right-click on the project, and click **Convert to Fortran Project**
  - Don't worry; it's still a C/C++ project, too



- *Every* Fortran project is also a C/C++ Project

Project Location

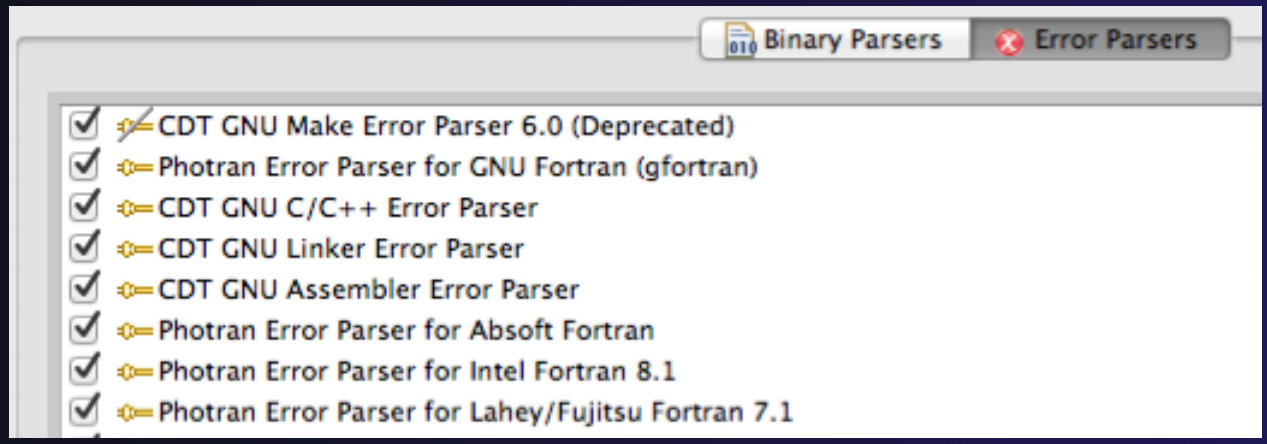
- How to tell where a project resides?
- In the project properties dialog, select the **Resource** category



Error Parsers

Do this once

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



Fortran Source Form Settings

- Fortran files are either *free form* or *fixed form;* some Fortran files are *preprocessed* (#define, #ifdef, etc.)
  - Source form determined by filename extension
  - Defaults are similar to most Fortran compilers:

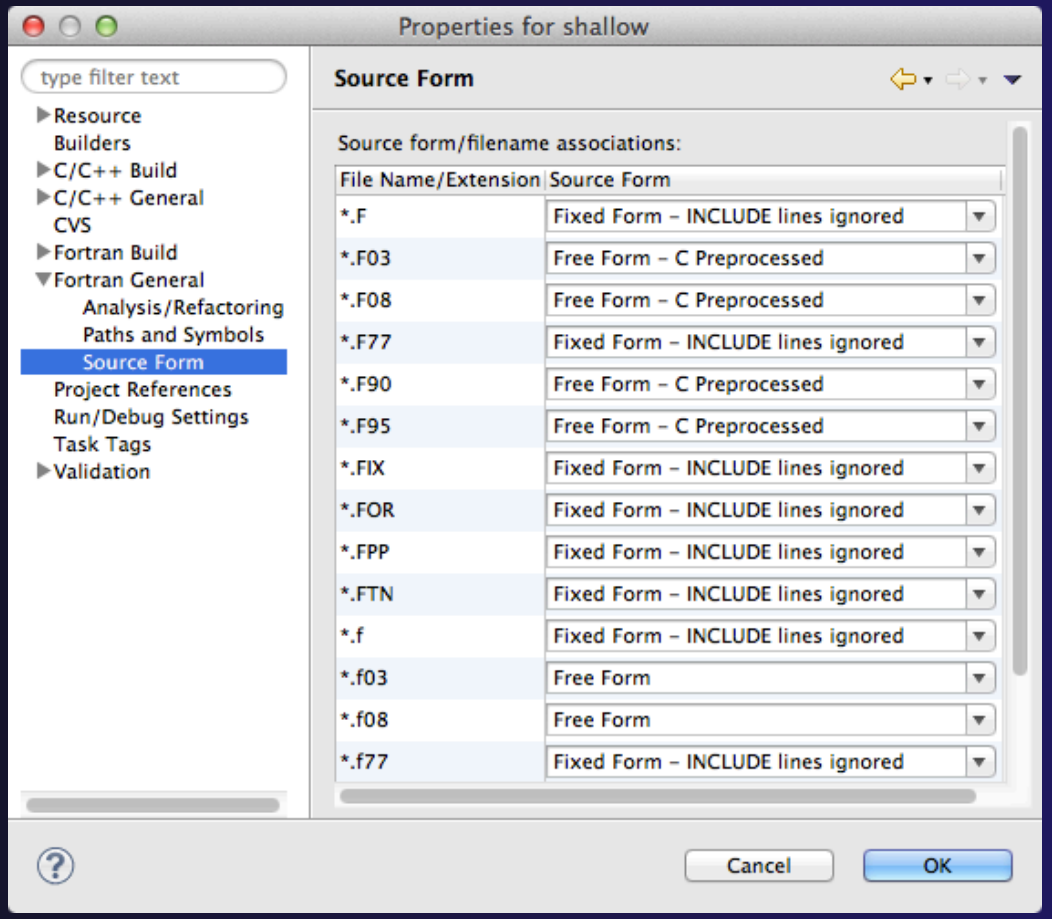
Fixed form:	.f	.fix	.for	.fpp	.ftn	.f77
Free form:	.f08	.f03	.f95	.f90		< unpreprocessed
	.F08	.F03	.F95	.F90		< preprocessed

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

Fortran Source Form Settings

Do this once

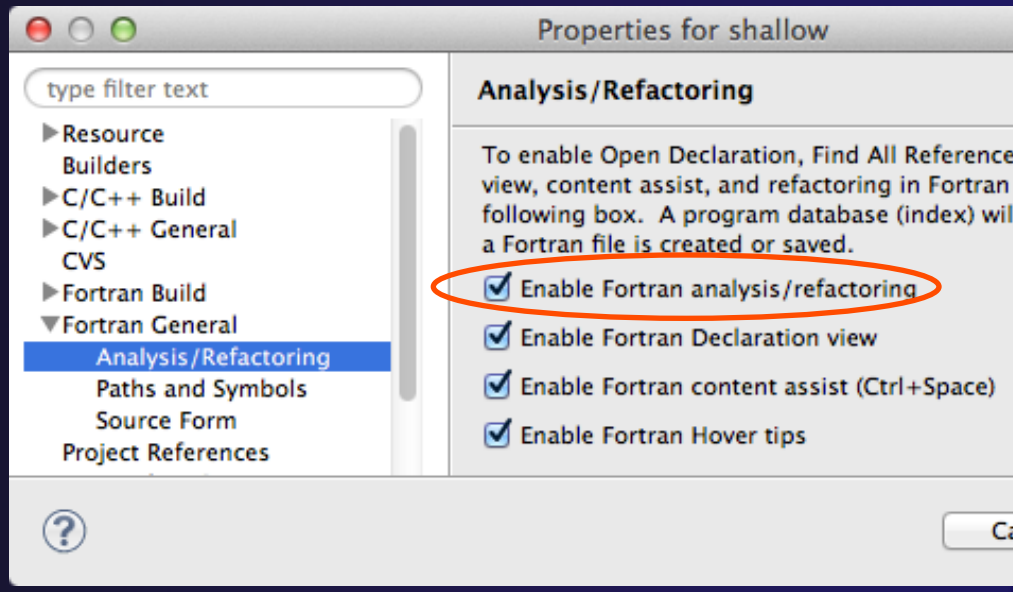
- In the project properties, select **Fortran General ▶ Source Form**
- Select source form for each filename extension
- Click **OK**



Enabling Fortran Advanced Features

Do this once

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



Exercise

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

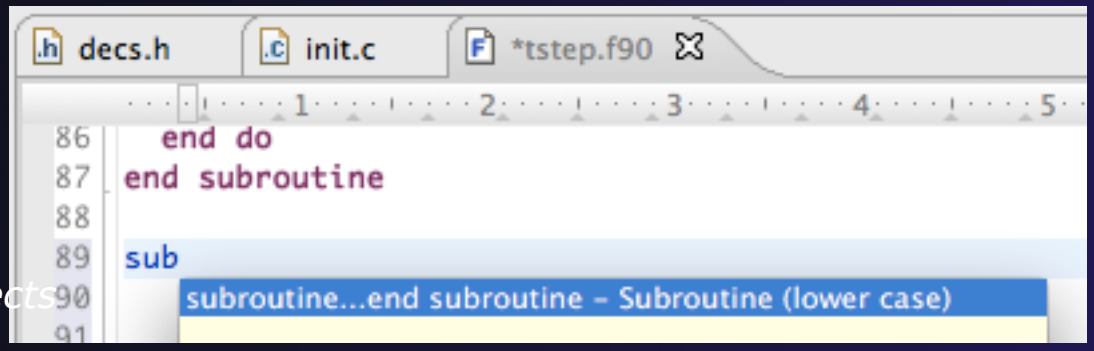
Advanced Editing

Code Templates

Code Templates

(C/C++ and Fortran)

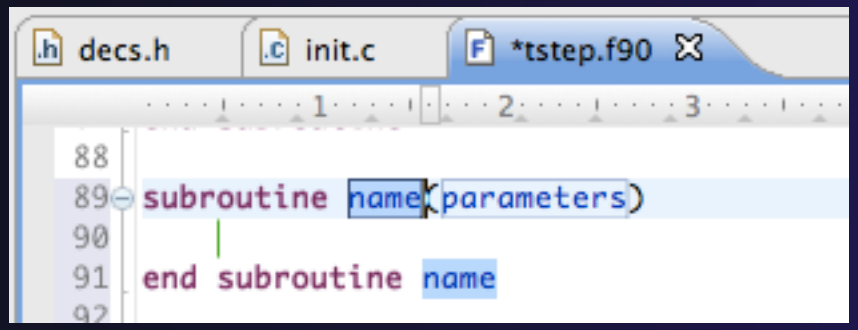
- Auto-complete common code patterns
  - For loops/do loops, if constructs, etc.
  - Also MPI code templates
- Included with content assist proposals (when **Ctrl-Space** is pressed)
  - E.g., after the last line in tstep.f90, type “sub” and press **Ctrl-Space**
  - Press **Enter** to insert the template



Code Templates (2)

(C/C++ and Fortran)

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



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

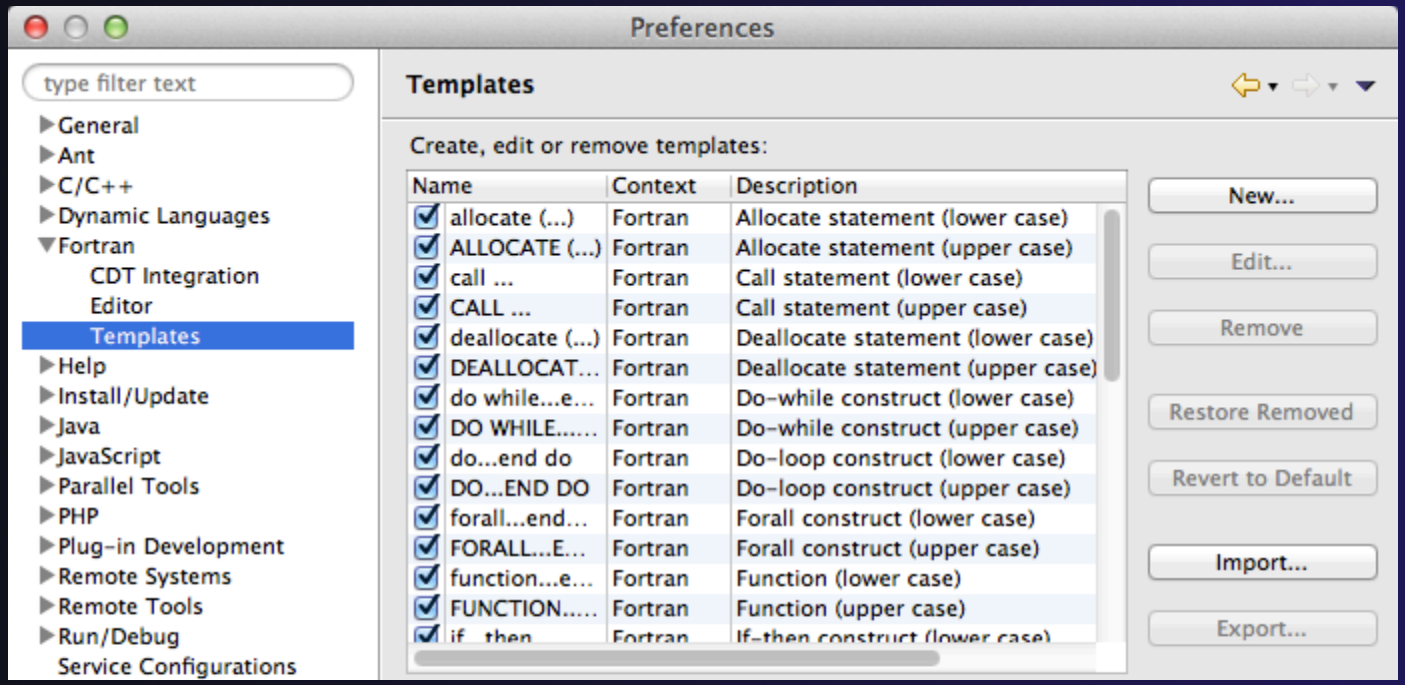
Exercise

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

Custom Code Templates

(Fortran)

- Customize code templates in **Window ▸ Preferences ▸ Fortran ▸ Templates**



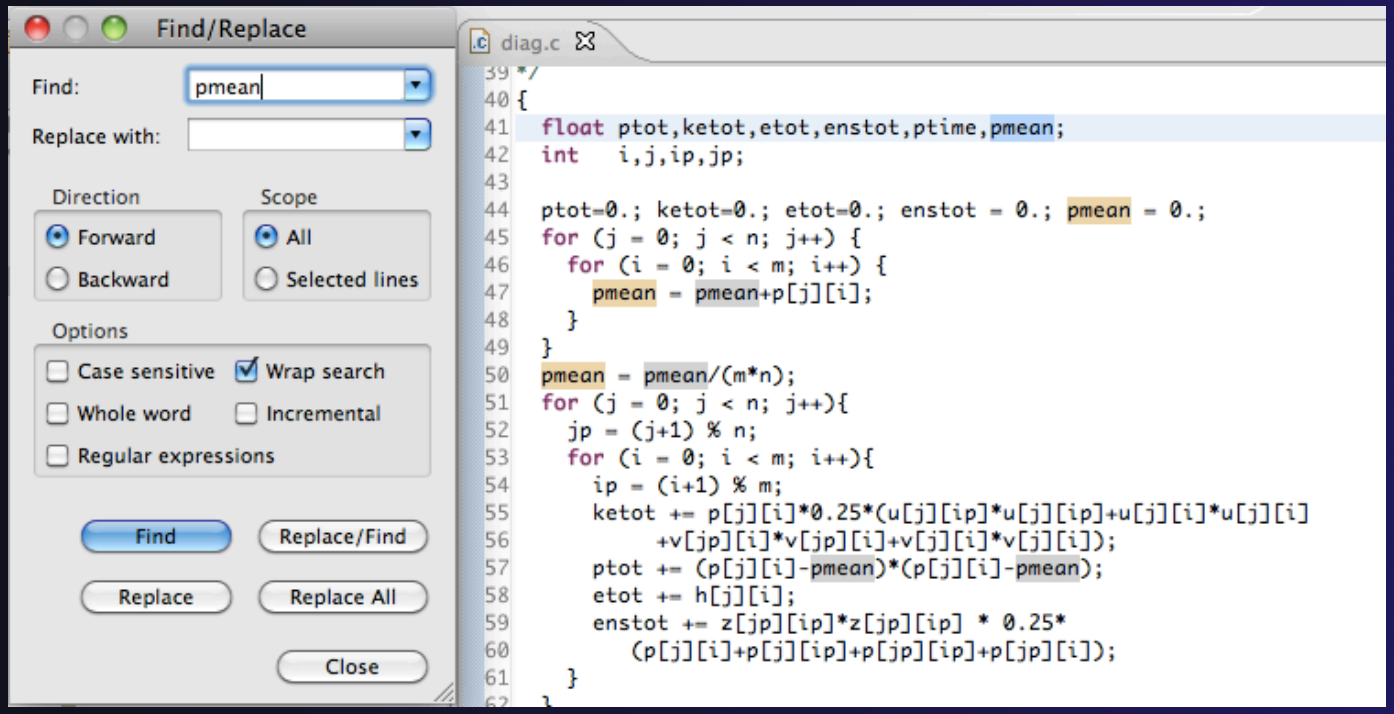
- Can import/export templates to XML files

Search & Refactoring

- Objectives
  - Develop proficiency using Eclipse's textual and language-based search and navigation capabilities
  - Introduce common automated refactorings
- Contents
  - Searching
  - Refactoring and Transformation
- Prerequisites
  - Basics
  - Fortran

Find/Replace within Editor

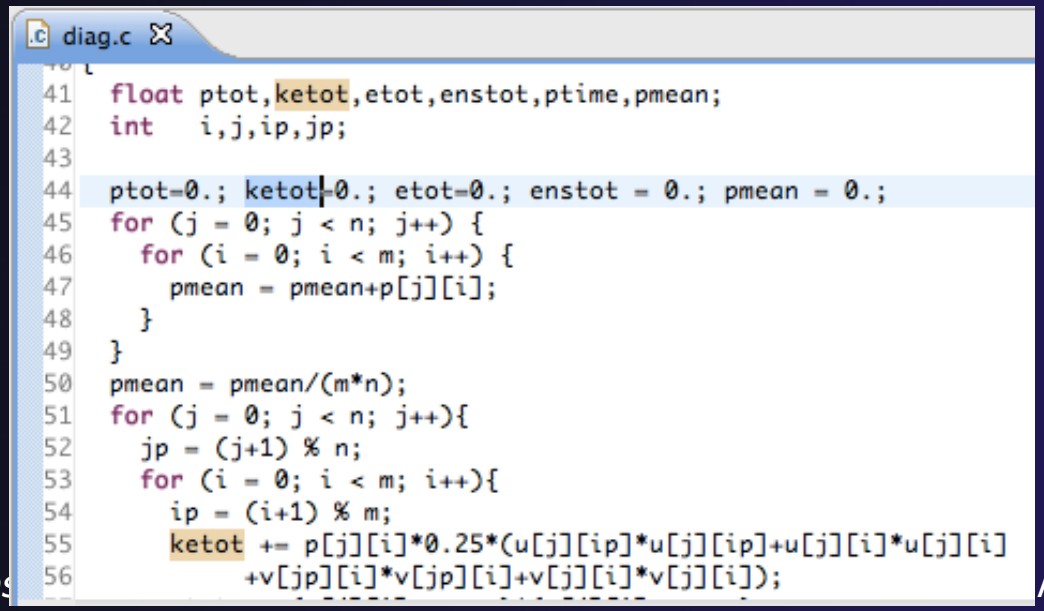
- Simple Find within editor buffer
- Ctrl-F (Mac: Command-F)



Mark Occurrences

(C/C++ Only)

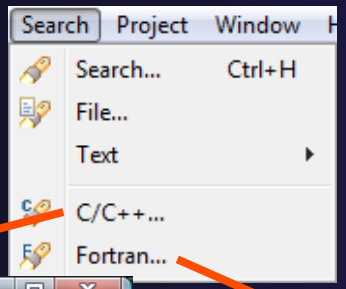
- Double-click on a variable in the CDT editor
- All occurrences in the source file are highlighted to make locating the variable easier
- Alt-shift-O to turn off (Mac: Alt-Cmd-O)



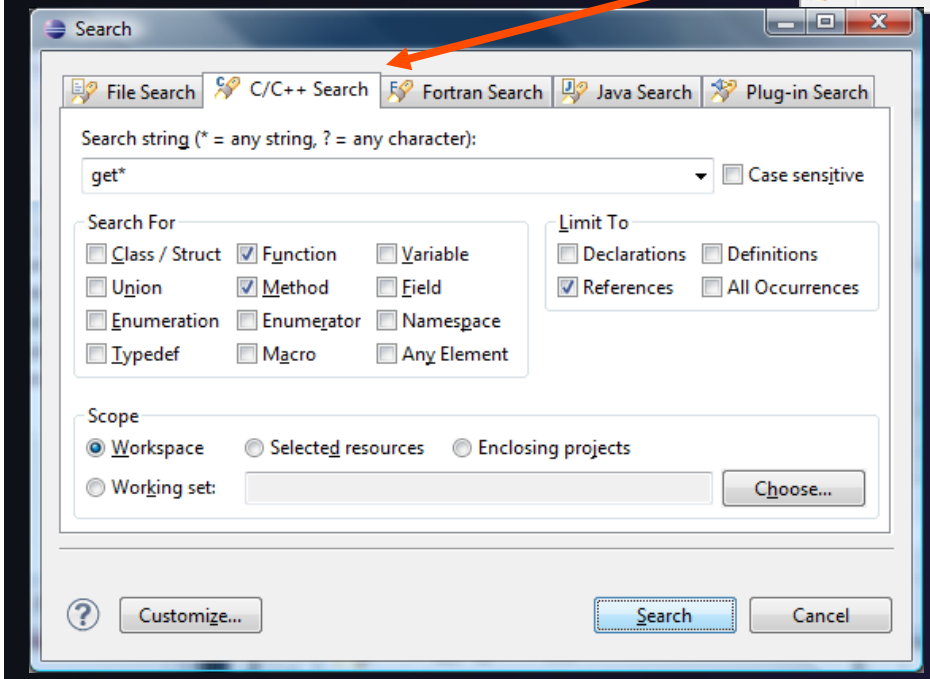
Language-Based Searching

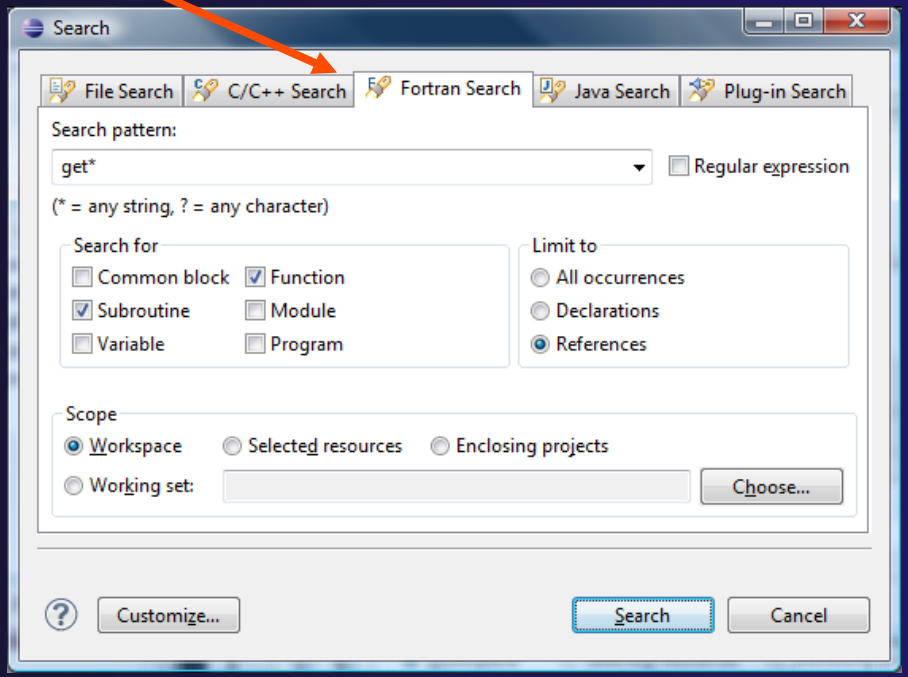
(C/C++ and Fortran)

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



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



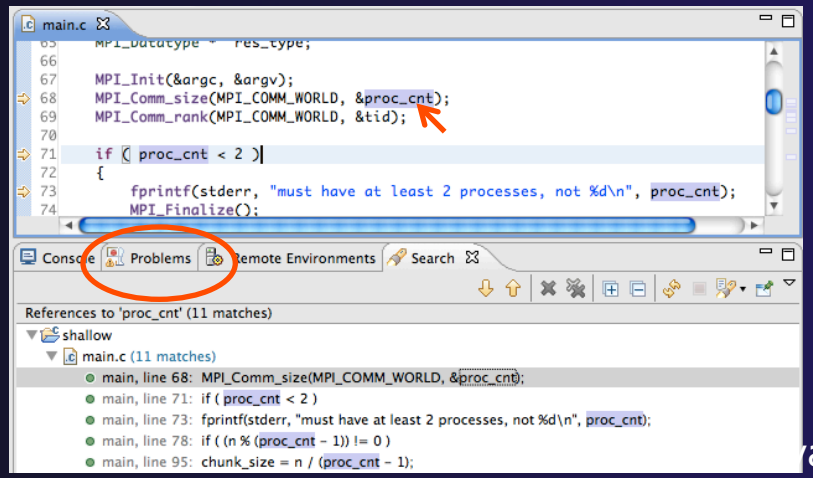


Find References

(C/C++ and Fortran)

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

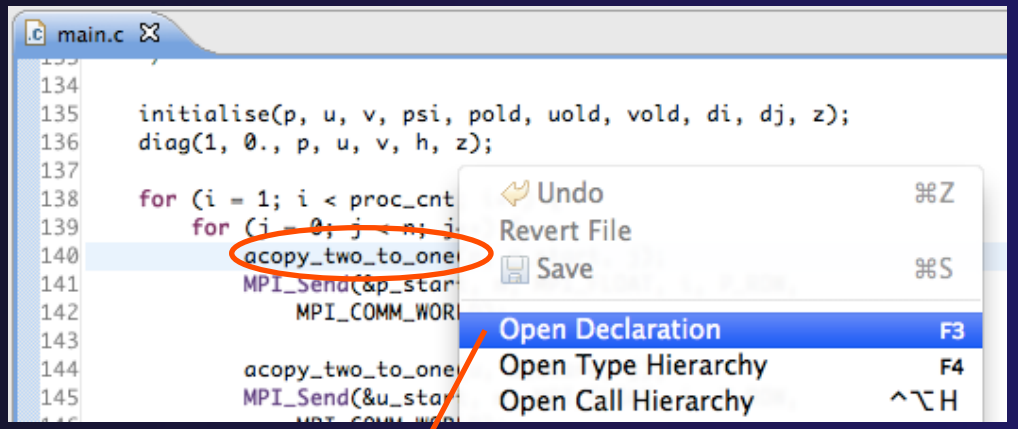
  - **Search** view shows matches



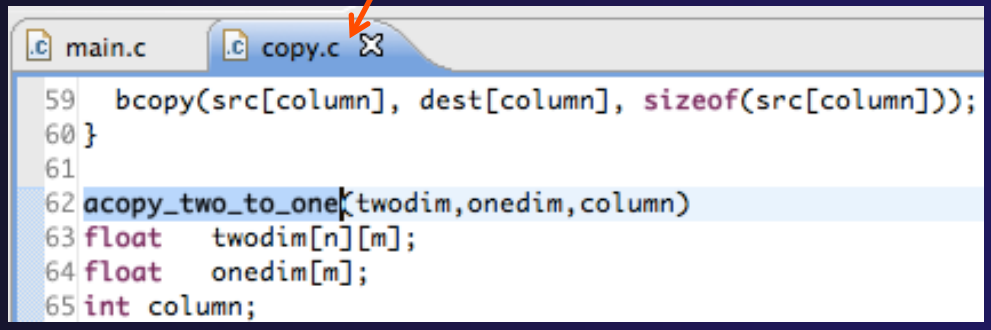
Open Declaration

(C/C++ and Fortran)

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



Goes to its declaration in copy.c



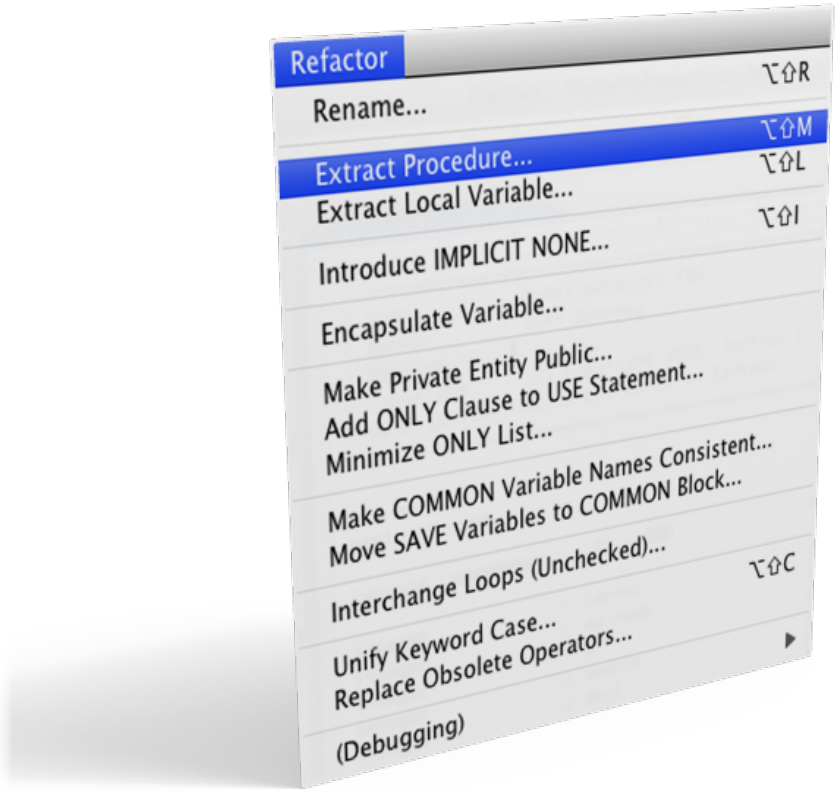
Search – Try It!

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

Refactoring and Transformation

Refactoring

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



- 39 automated refactorings in Photran

Refactoring Caveats

- Photran can only refactor free form code that is *not* preprocessed
  - Determined by Source Form settings
    (recall from earlier that these are configured in
    Project Properties: Fortran General▸Source Form)

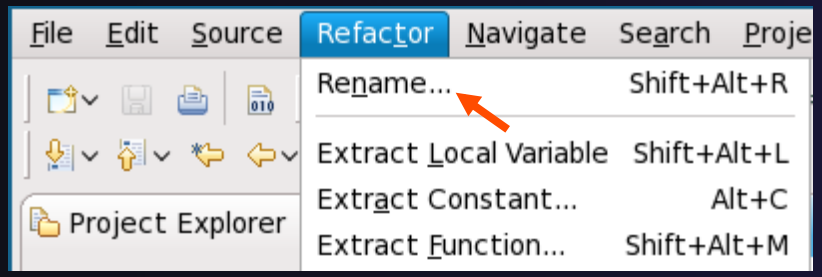
✔	**Free Form, Unpreprocessed:**	.f08	.f03	.f95	.f90	
✖	**Free Form, Preprocessed:**	.F08	.F03	.F95	.F90	
✖	**Fixed Form:**	.f	.fix	.for	.fpp	.ftn	.f77

- Refactor menu will be empty if
  - Refactoring not enabled in project properties
    (recall from earlier that it is enabled in
    Project Properties: Fortran General▸Analysis/Refactoring)
  - The file in the active editor is fixed form
  - The file in the active editor is preprocessed

Rename Refactoring

(also available in Fortran)

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



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

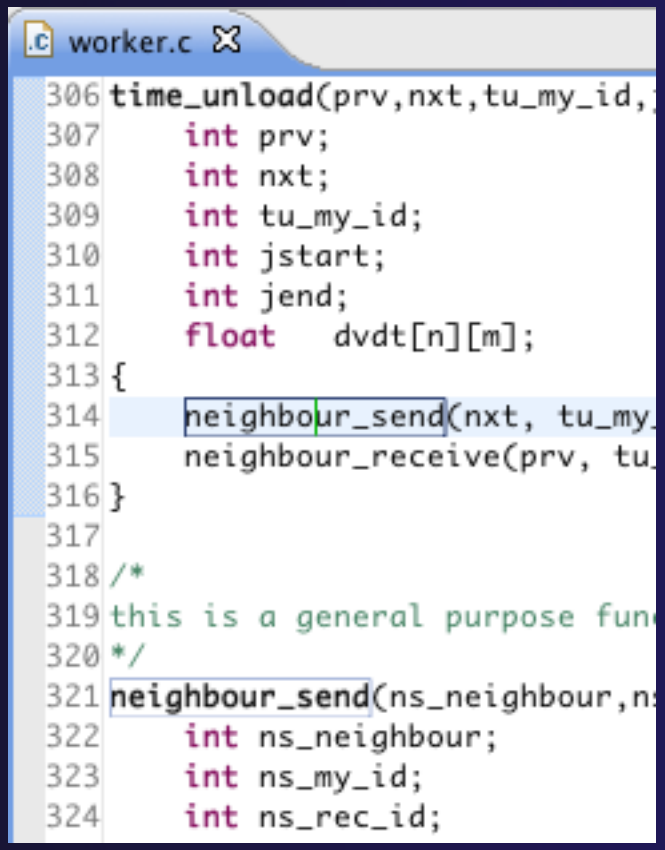
Refactoring	Uses	Percentage
Rename	179,871	74.8%
Extract Local Variable	13,523	5.6%
Move	13,208	5.5%
Extract Method	10,581	4.4%
Change Method Signature	4,764	2.0%
Inline	4,102	1.7%
Extract Constant	3,363	1.4%
(16 Other Refactorings)	10,924	4.5%

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

Rename in File

(C/C++ Only)

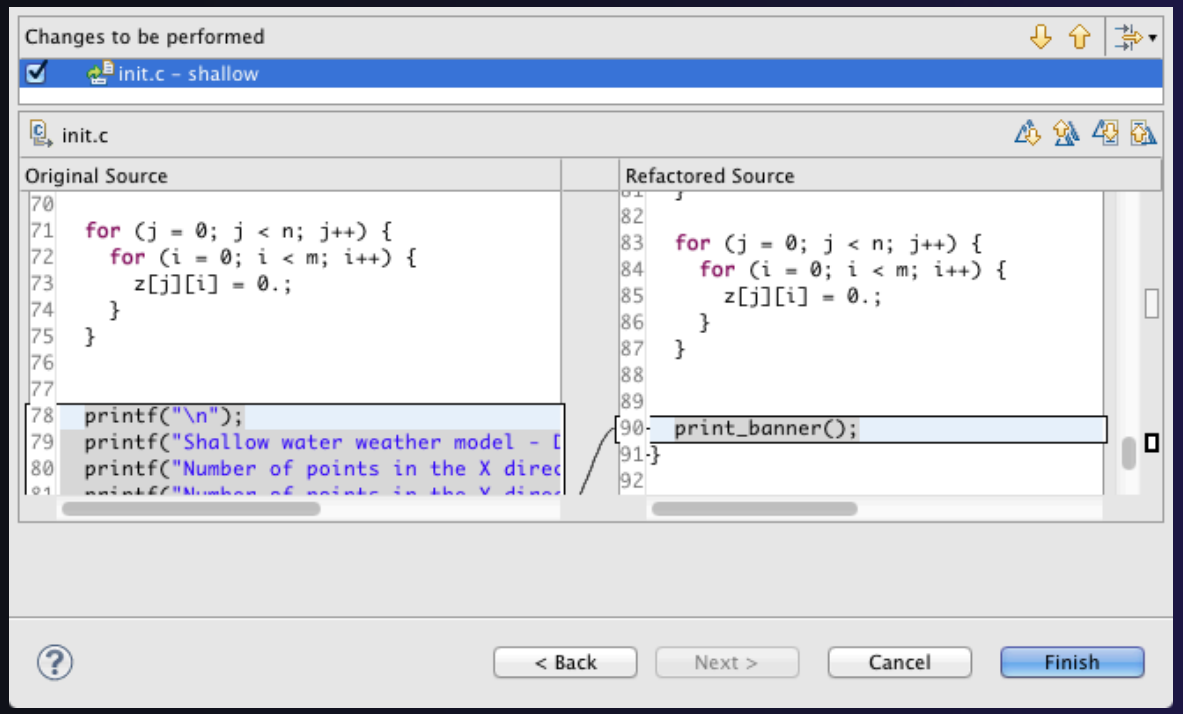
- Position the caret over an identifier.
- Press **Ctrl-1** (**Command-1** on Mac).
- Enter a new name. Changes are propagated within the file as you type.



Extract Function Refactoring

(also available in Fortran - “Extract Procedure”)

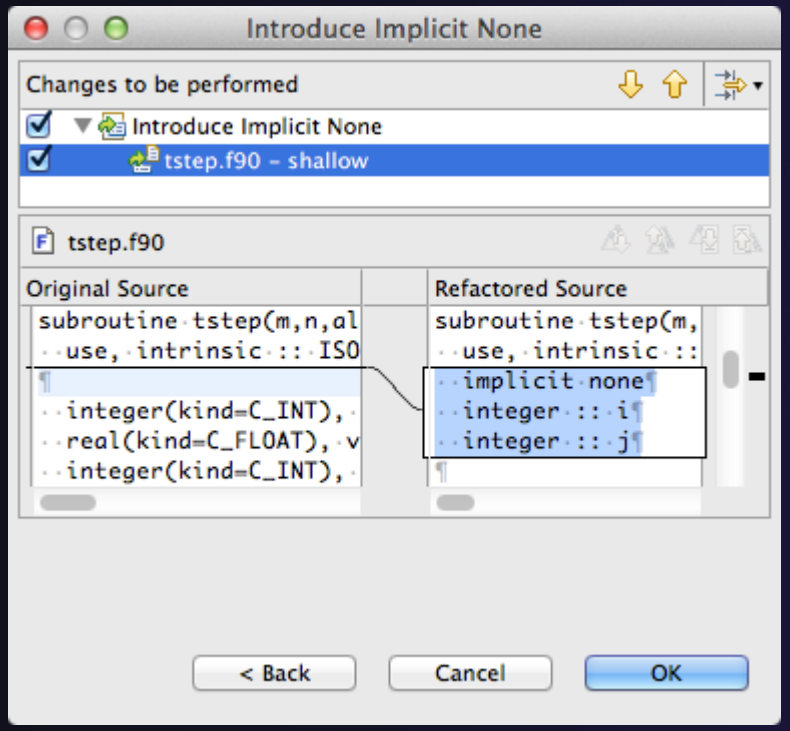
- Moves statements into a new function, replacing the statements with a call to that function
- Local variables are passed as arguments



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

Introduce IMPLICIT NONE Refactoring

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

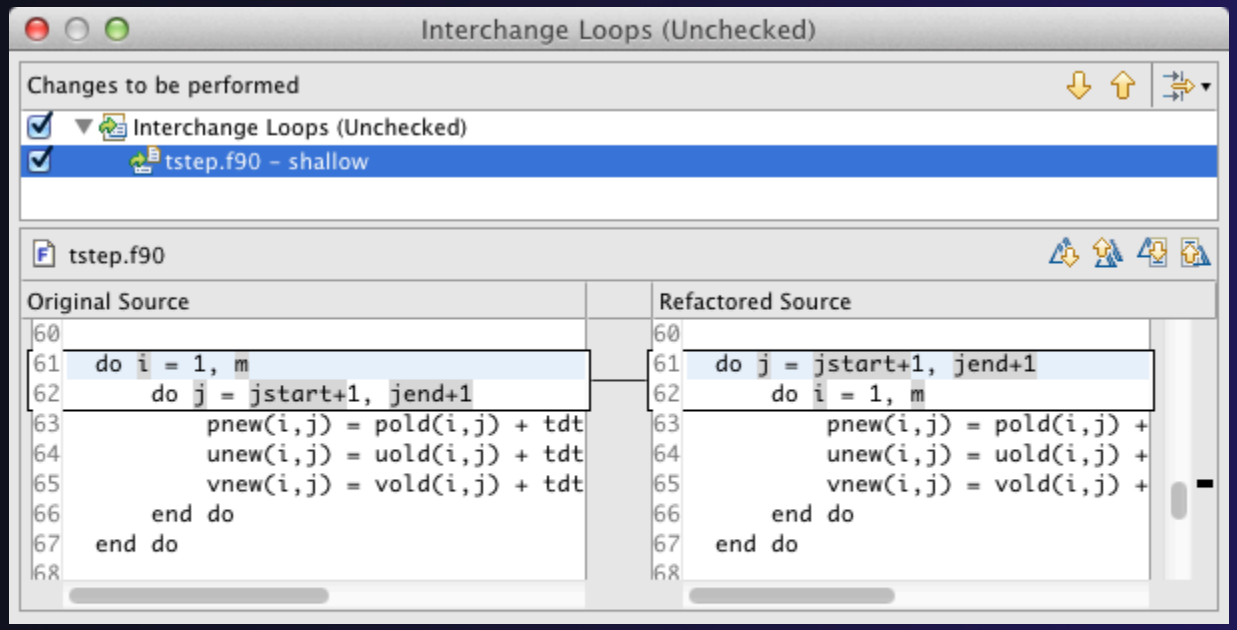


- Introduce in a single file by opening the file and selecting **Refactor▸Coding Style▸ Introduce IMPLICIT NONE...**
- Introduce in multiple files by selecting them in the Fortran Projects view, right-clicking on the selection, and choosing **Refactor▸Coding Style▸Introduce IMPLICIT NONE...**

Loop Transformations

(Fortran only)

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



Old version traverses matrices in row-major order

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

Loop Transformations

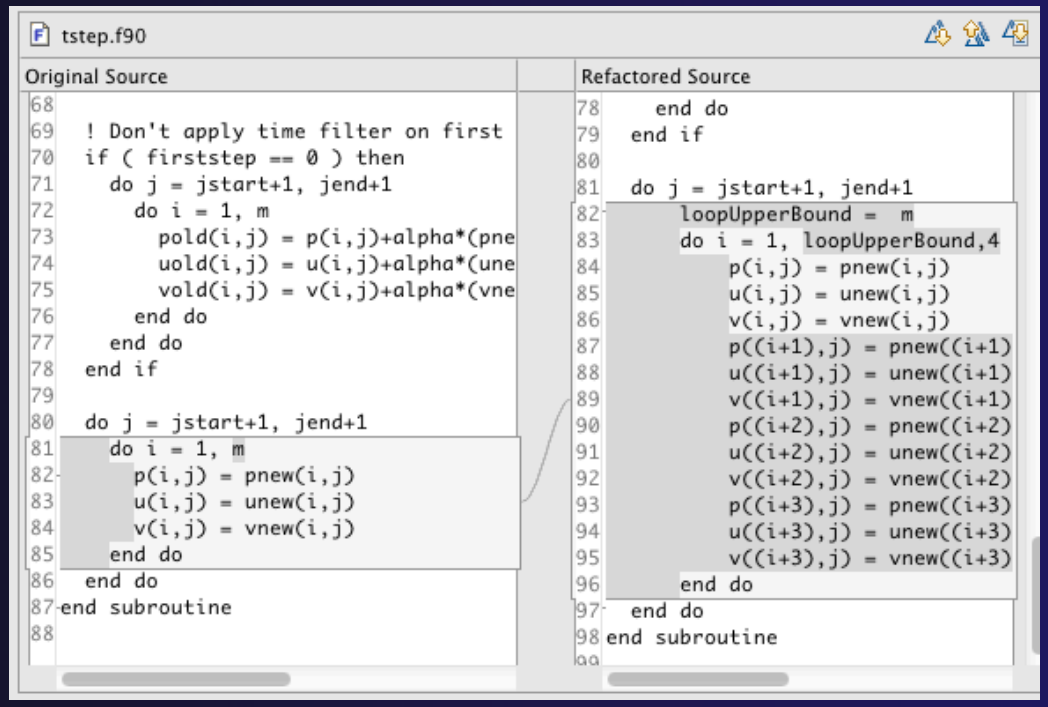
(Fortran only)

- **Unroll Loop**
- Select a loop, click **Refactor▸Do Loop▸Unroll Loop...**

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

Unroll 4×

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



Refactoring & Transformation – Exercises

In tstep.f90…

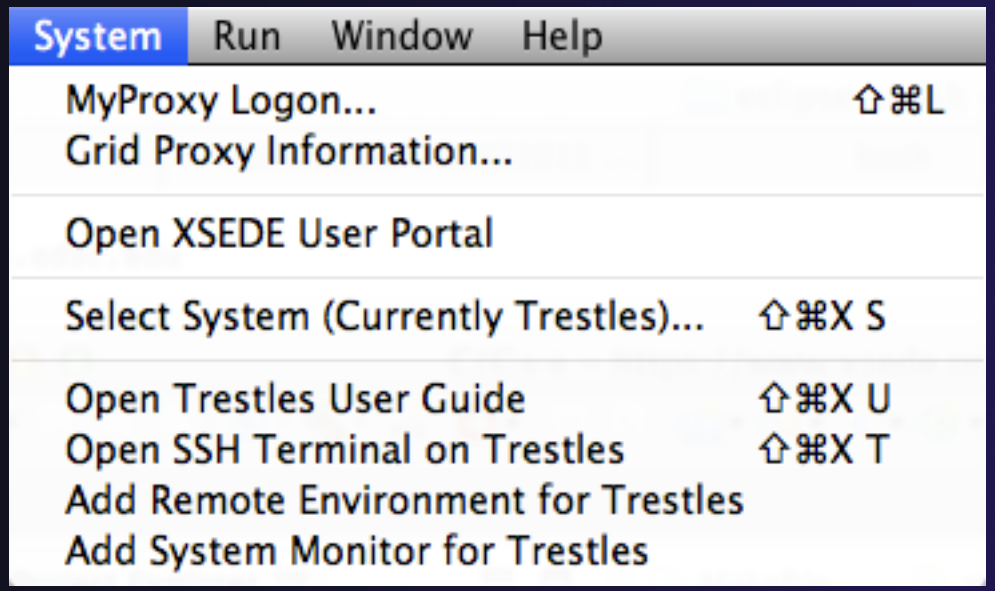
1. In init.c, extract the `printf` statements at the bottom of the file into a new function called `print_banner`
2. In worker.c, change the spellings of `neighbour_send` and `neighbour_receive` to American English
3. In tstep.f90, make the (Fortran) `tstep` subroutine IMPLICIT NONE

NCSA/XSEDE Features

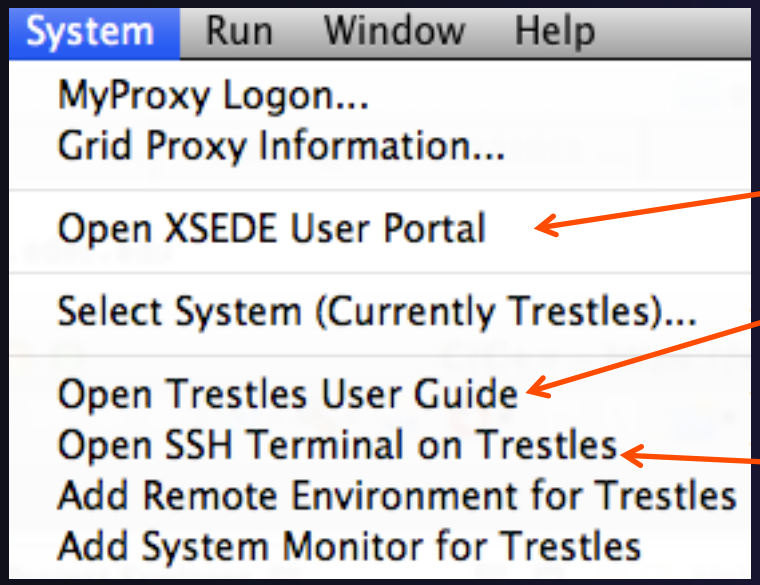
- Objectives
  - Install NCSA's GSI auth and XSEDE support plug-ins
  - Become familiar with the System menu
- Contents
  - Capabilities
  - Installation
- Prerequisites
  - (none)

Additional Plug-ins from NCSA

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



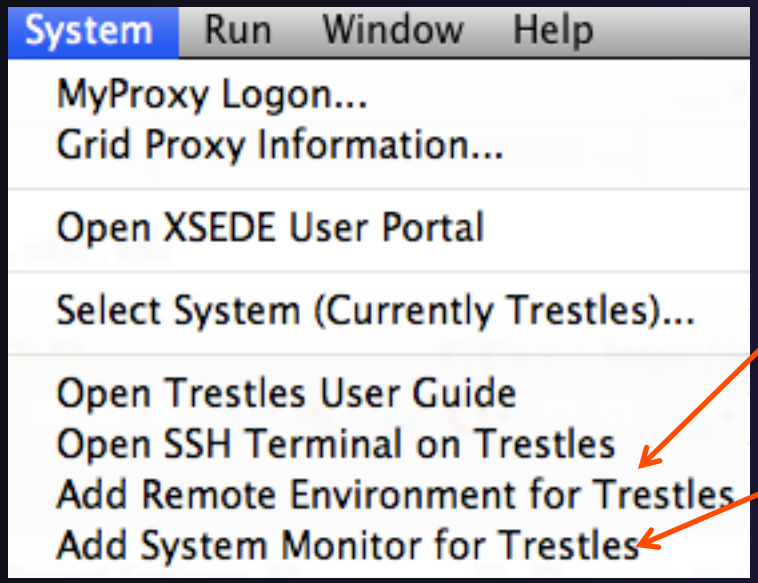
System Menu



- Open Web content in Eclipse:
  - **Open XSEDE User Portal**
  - **Open User Guide** for a machine
- Open an SSH terminal (as an Eclipse view)

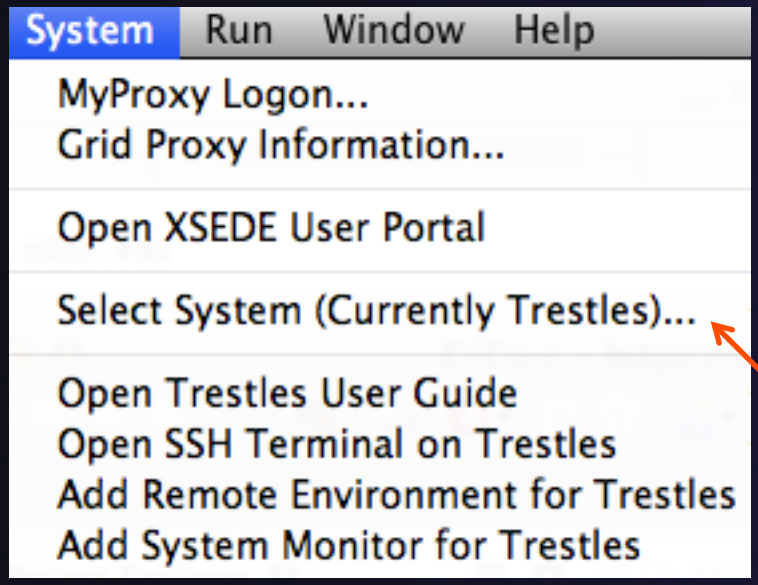
Eclipse-integrated SSH terminals are provided by the Remote System Explorer (RSE), one of the features that is included in the Eclipse for Parallel Application Developers package.

System Menu



- Shortcuts for common PTP tasks:
  - **Add Remote Environment** adds a Remote Tools connection for a particular machine
  - **Add System Monitor** opens the System Monitoring perspective and begins monitoring a particular machine

System Menu



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

MyProxy Logon

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

Installation

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

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

   and then click **OK** to close the Add dialog.
   - Or, if you copied ncsa-update-size.zip from a USB drive, click **Archive**, select that file, and click **OK.**
4. Select the following:
   - GSI Authentication and MyProxy Logon Support
   - NCSA and XSEDE System Support
5. Click **Next** and complete the installation

Parallel Debugging

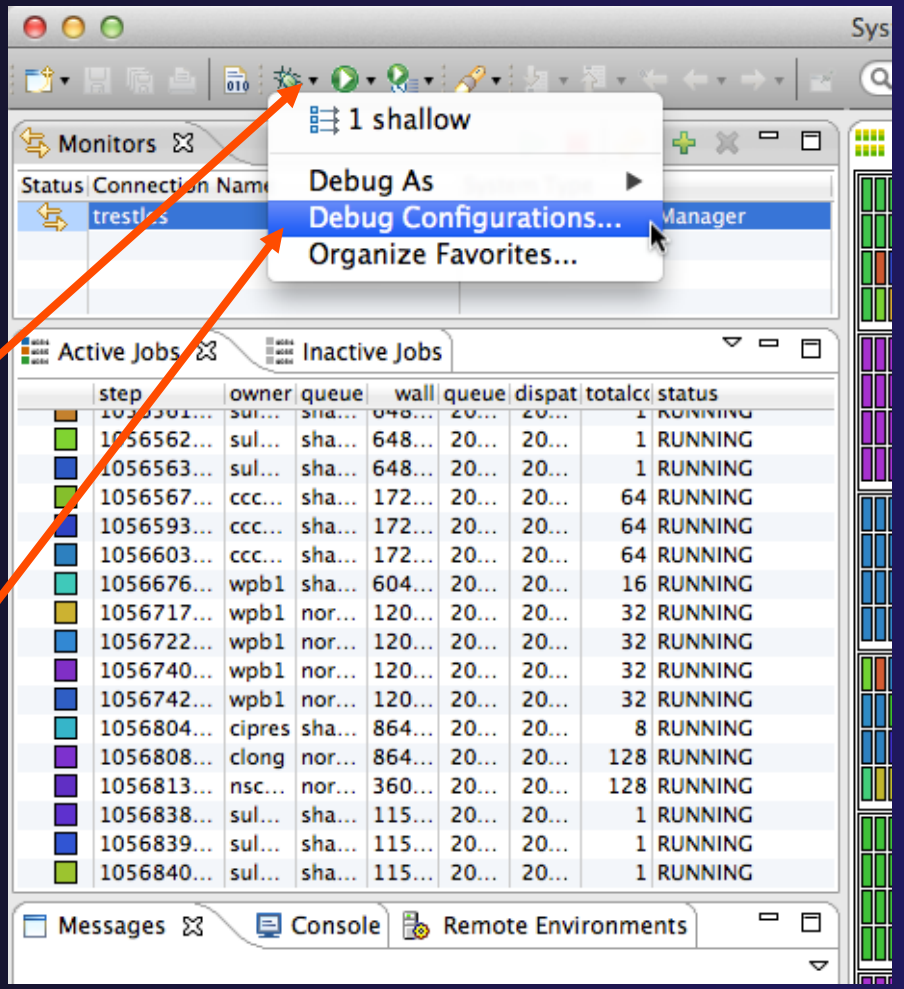
- Objective
  - Learn the basics of debugging parallel programs
- Contents
  - Launching a debug session
  - The Parallel Debug Perspective
  - Controlling sets of processes
  - Controlling individual processes
  - Parallel Breakpoints
  - Terminating processes

Debugging Setup

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

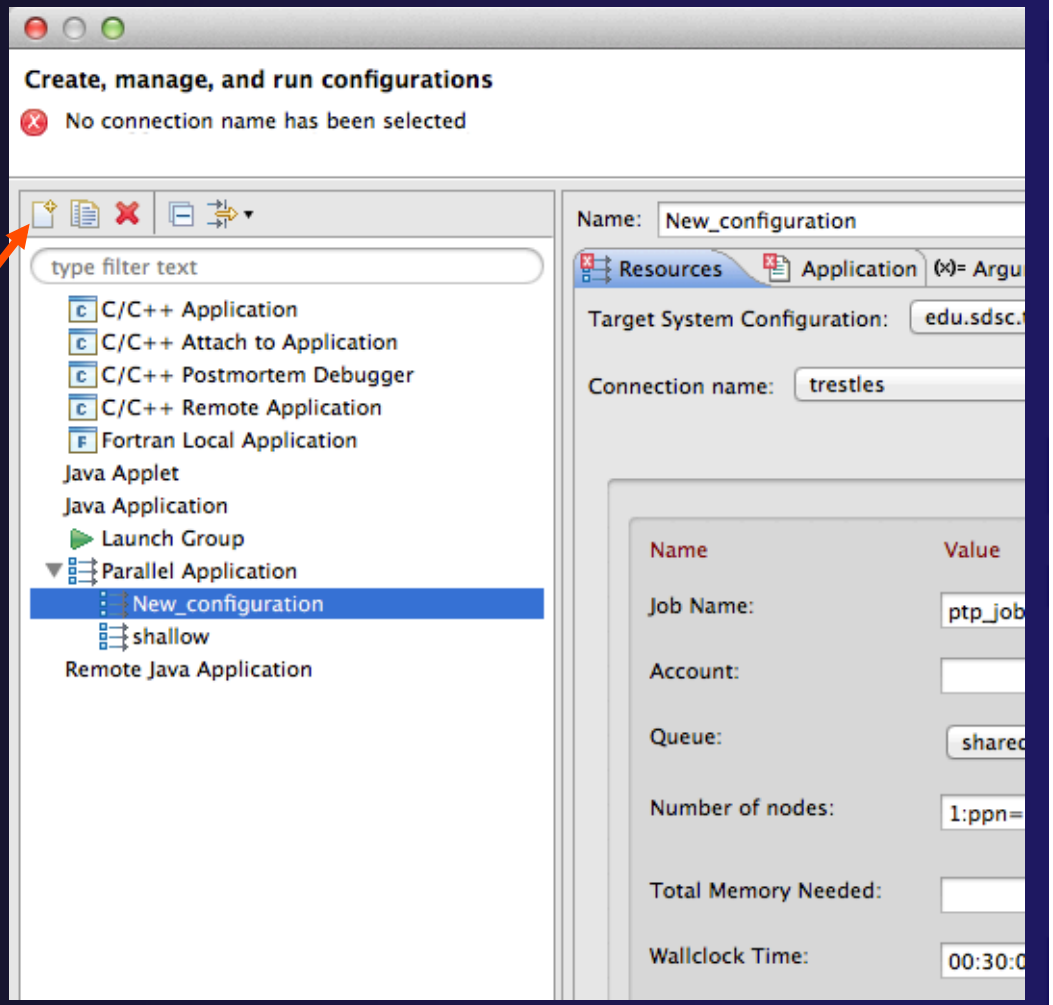
Create a Debug Configuration

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



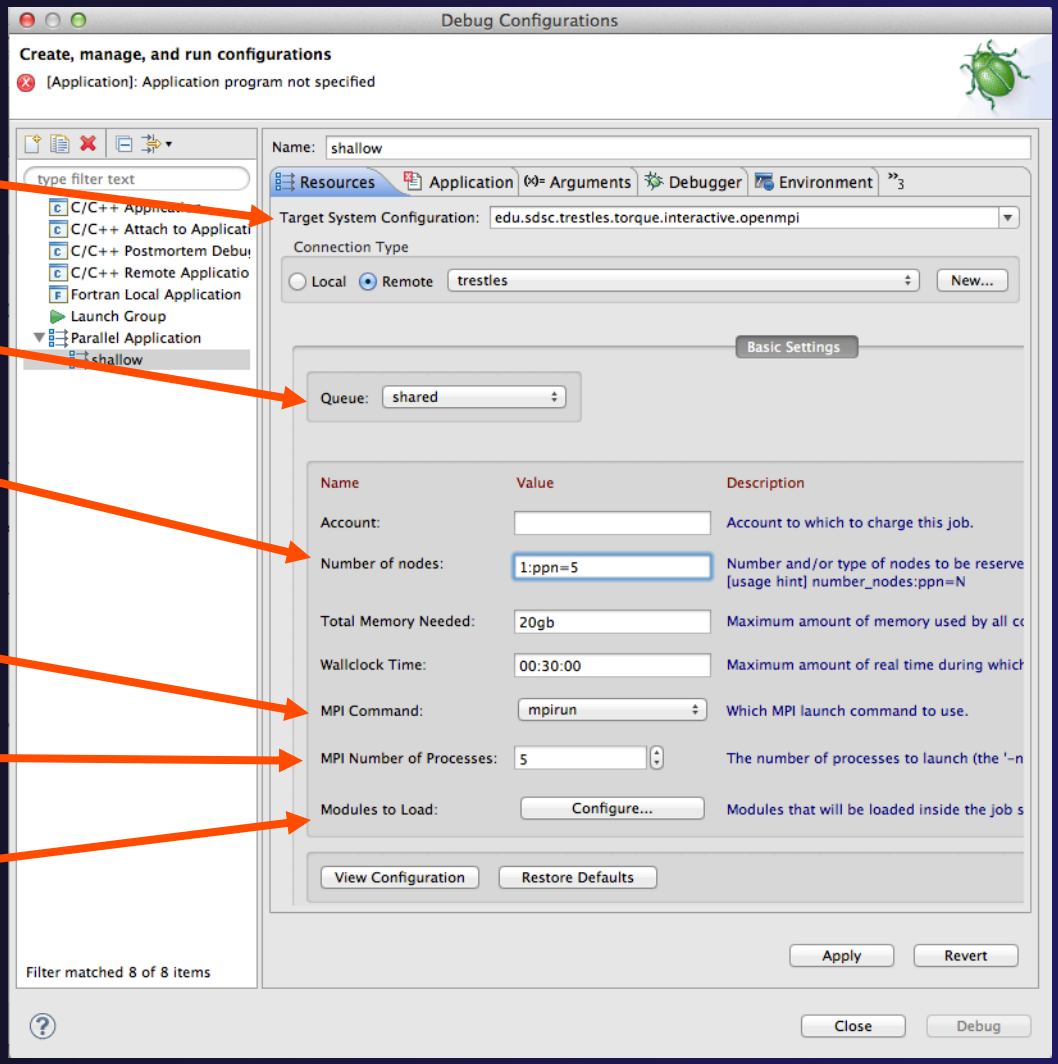
Create a New Configuration

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



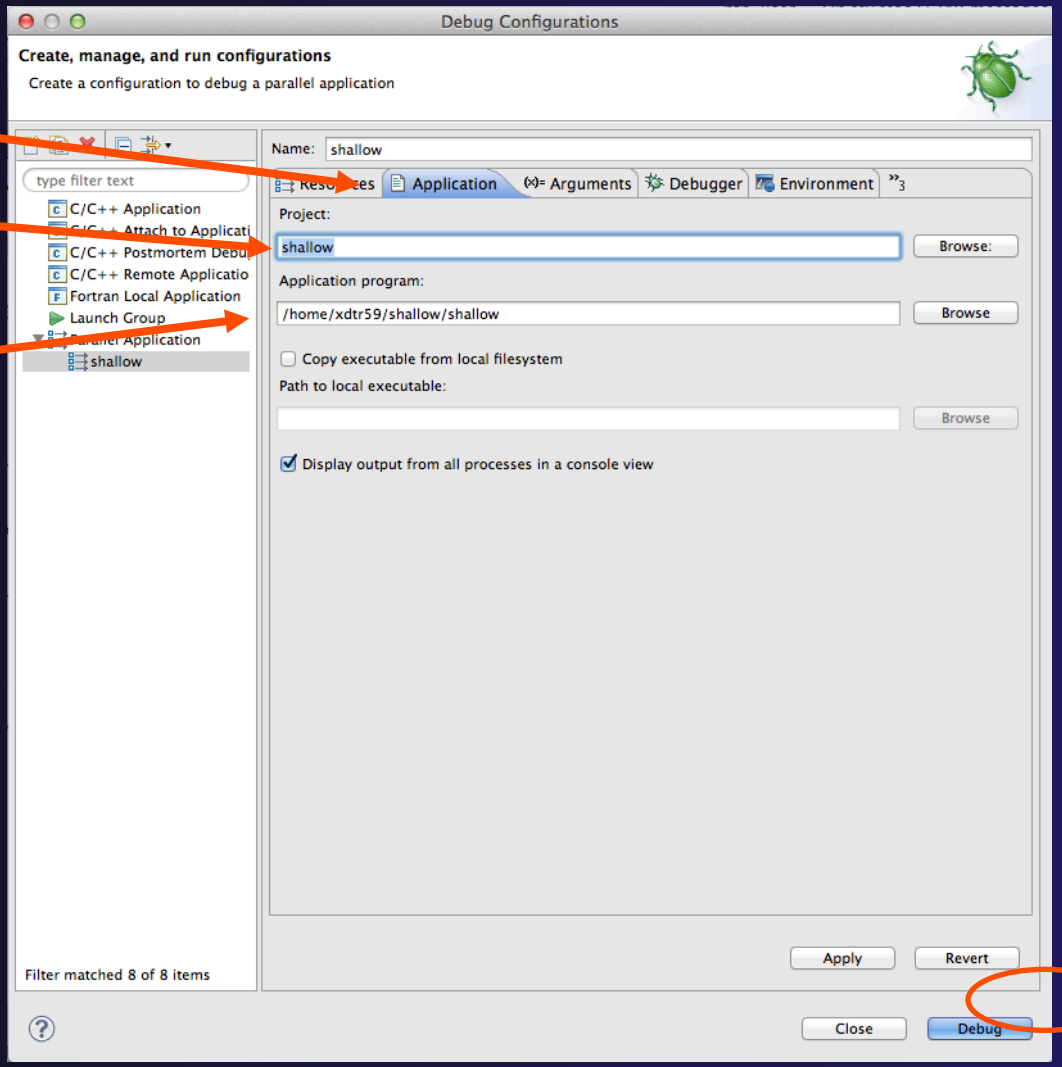
Configure the Resources Tab

- Select the new target system configuration
- Choose the queue
- Make sure number of nodes is correct
- Make sure the **mpirun** command is selected
- Select the number of processes (in this case use 5)
- Configure modules if required



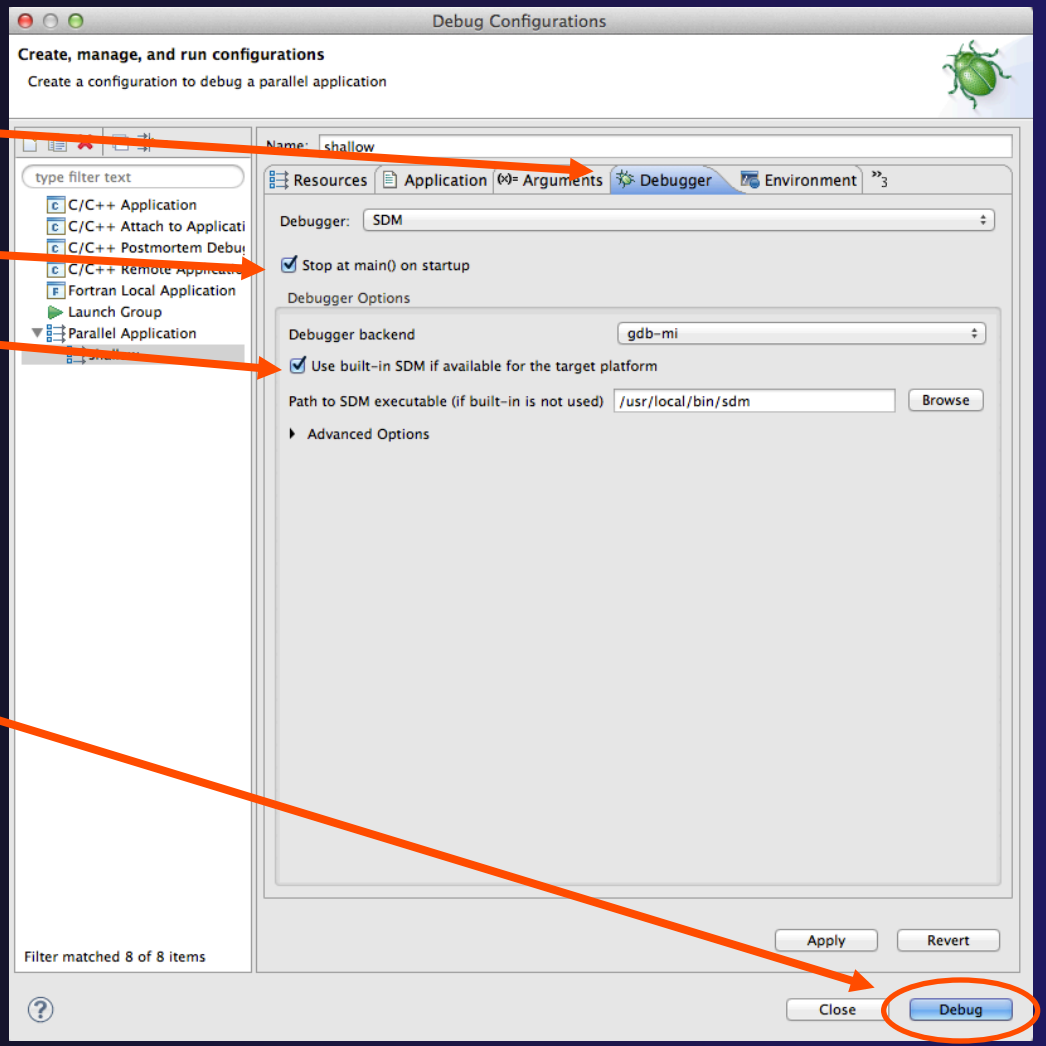
Configure the Application Tab (Optional)

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



Configure the Debug Tab (Optional)

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

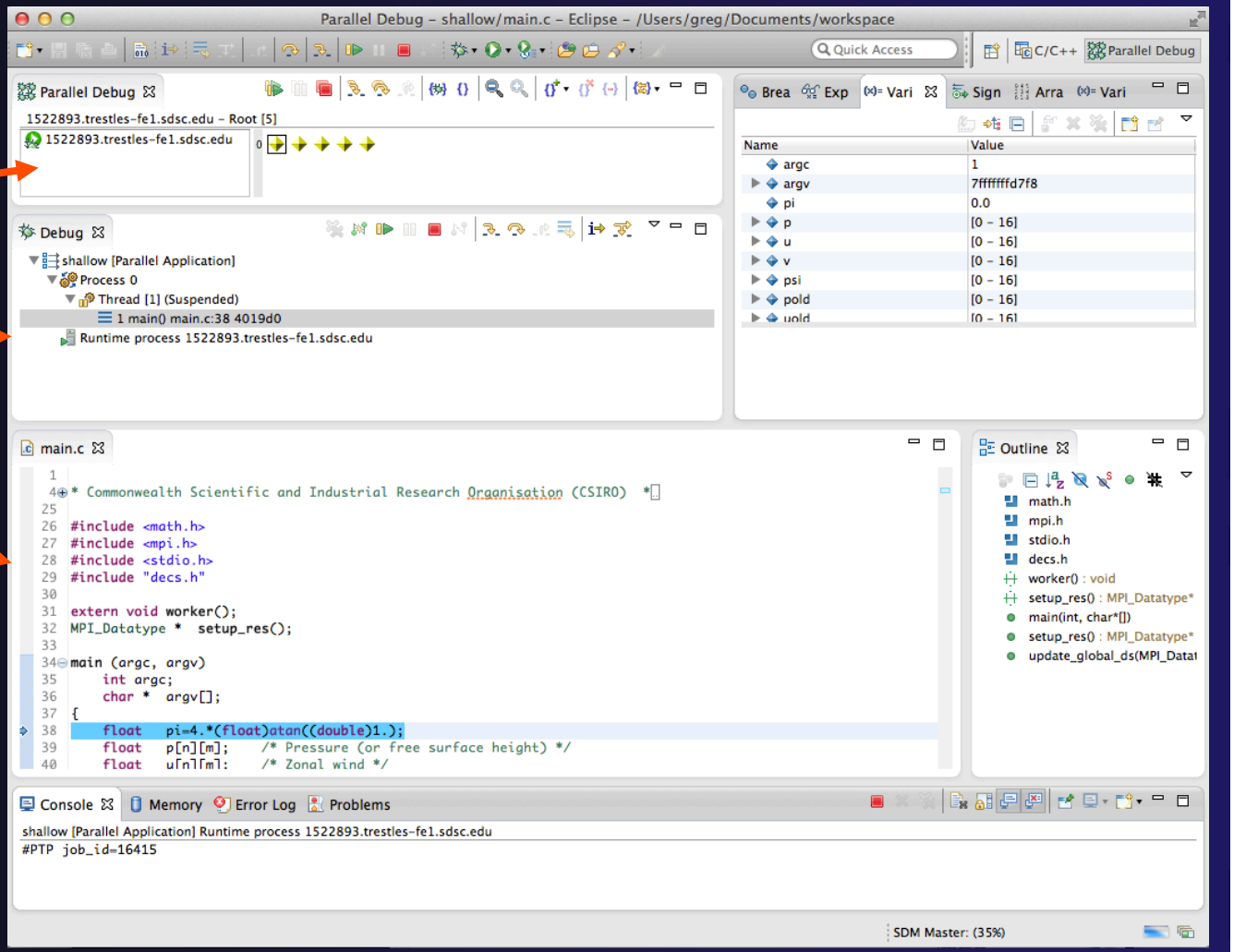


Exercise

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

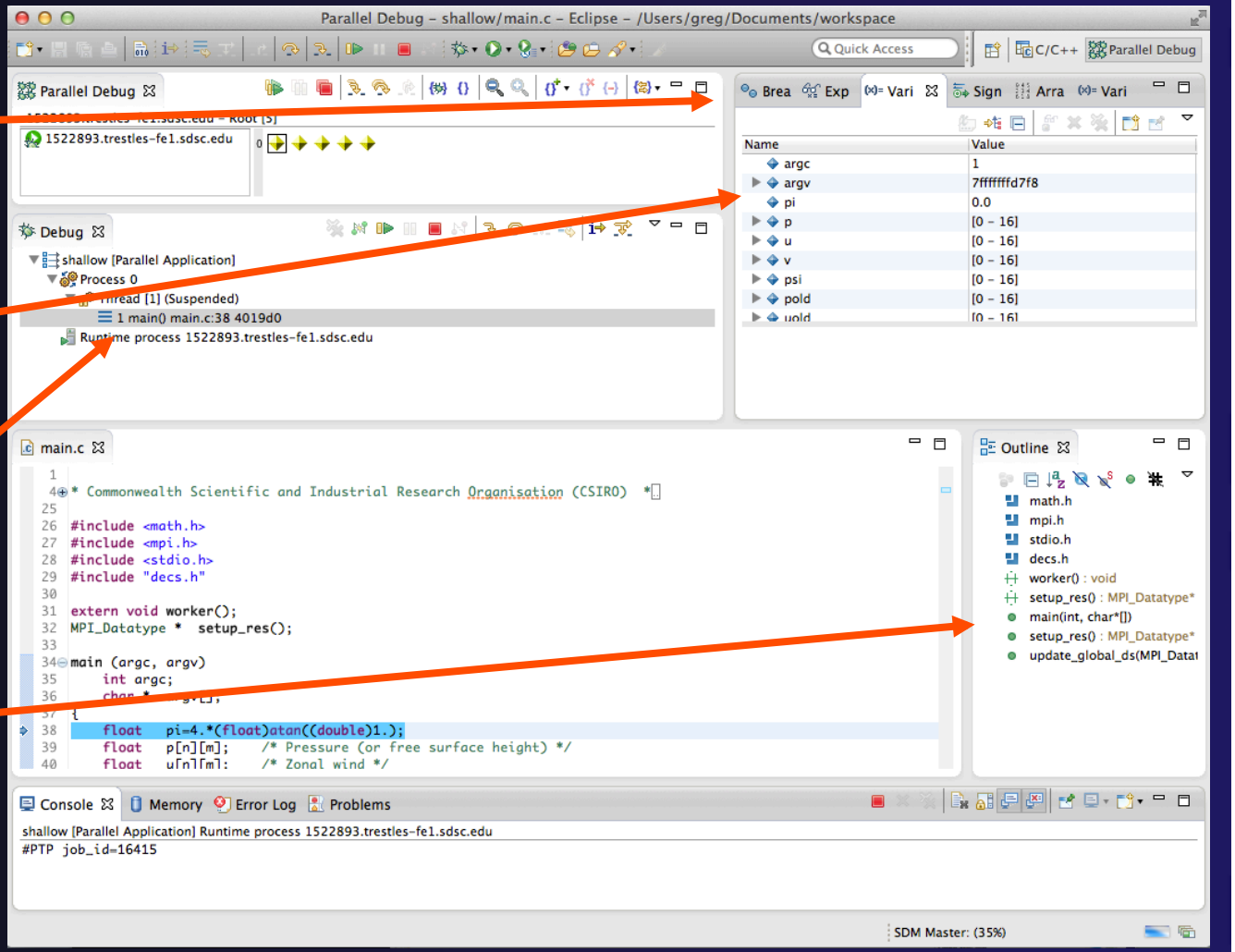
The Parallel Debug Perspective (1)

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



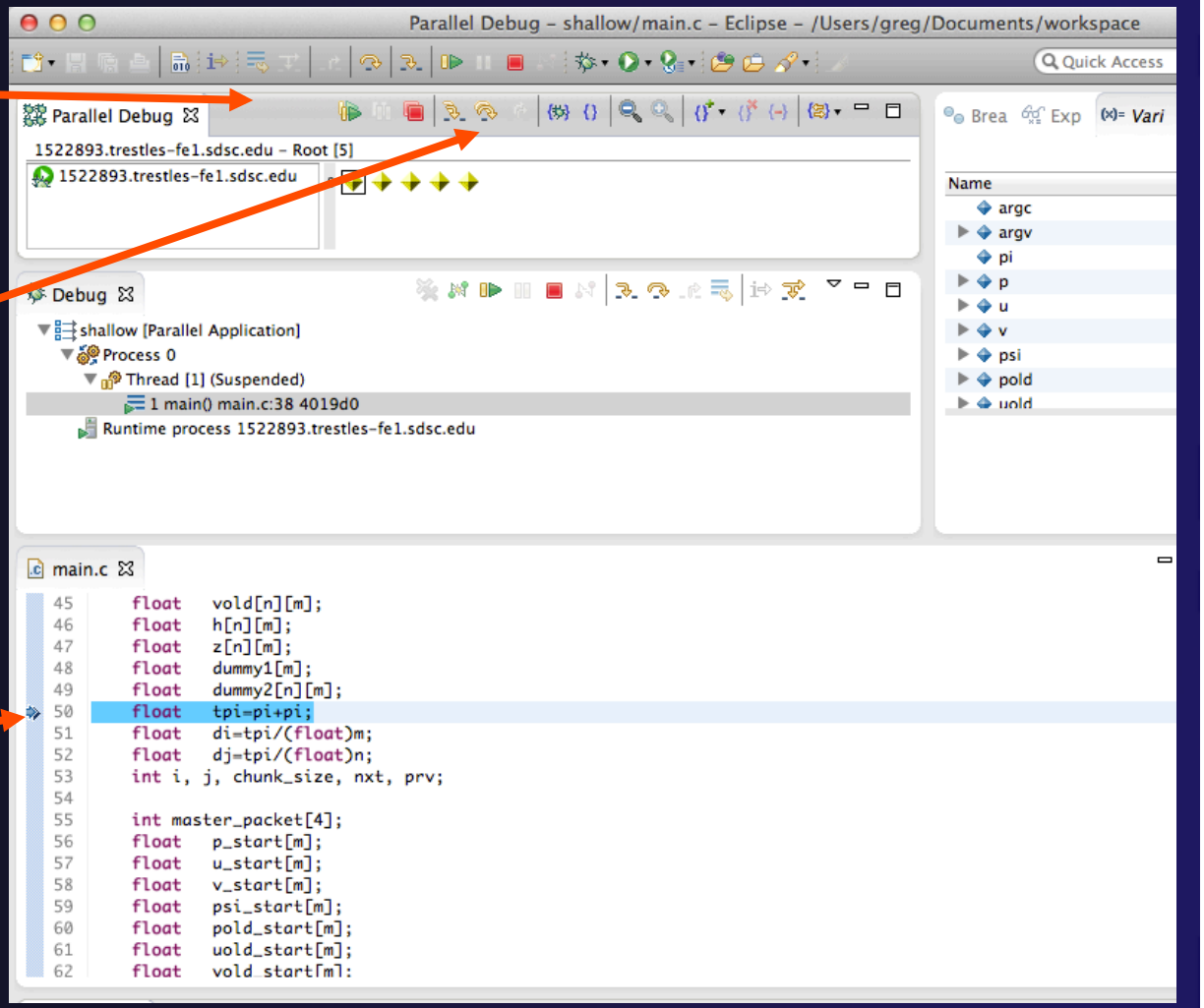
The Parallel Debug Perspective (2)

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



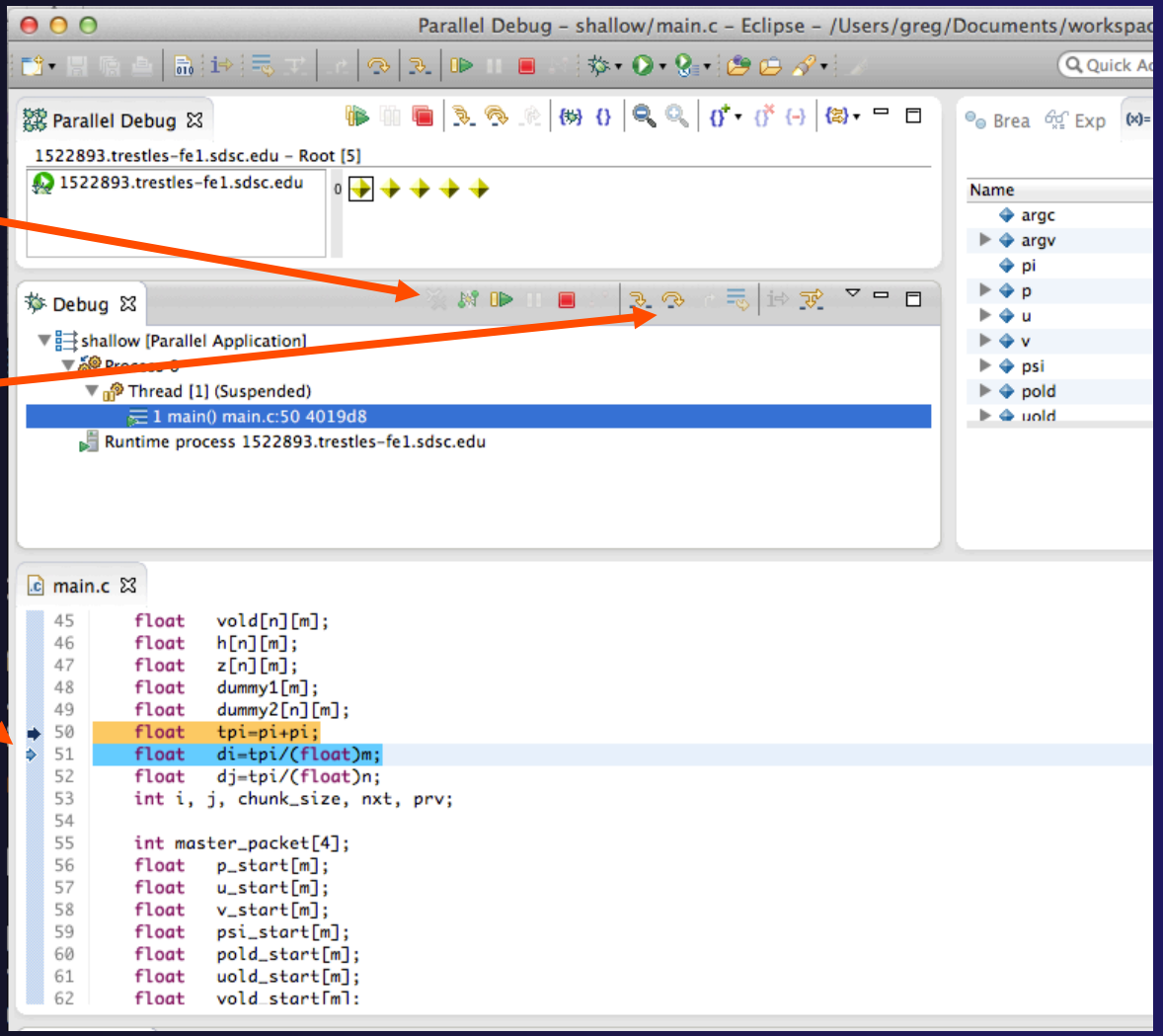
Stepping All Processes

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



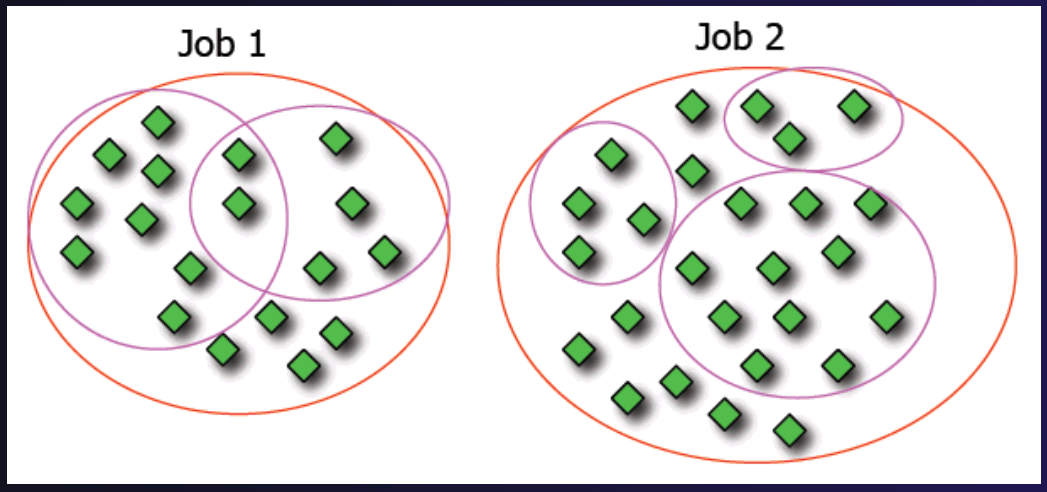
Stepping An Individual Process

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



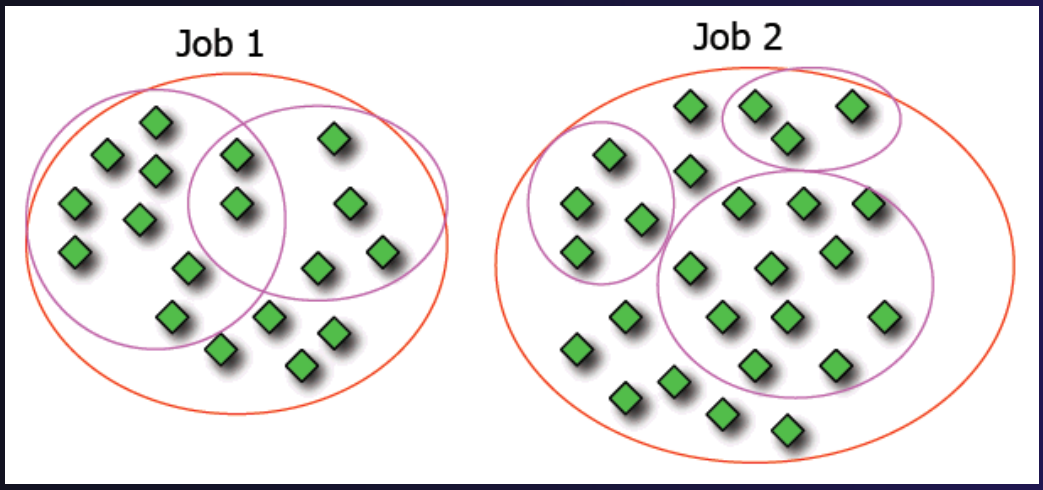
Process Sets (1)

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



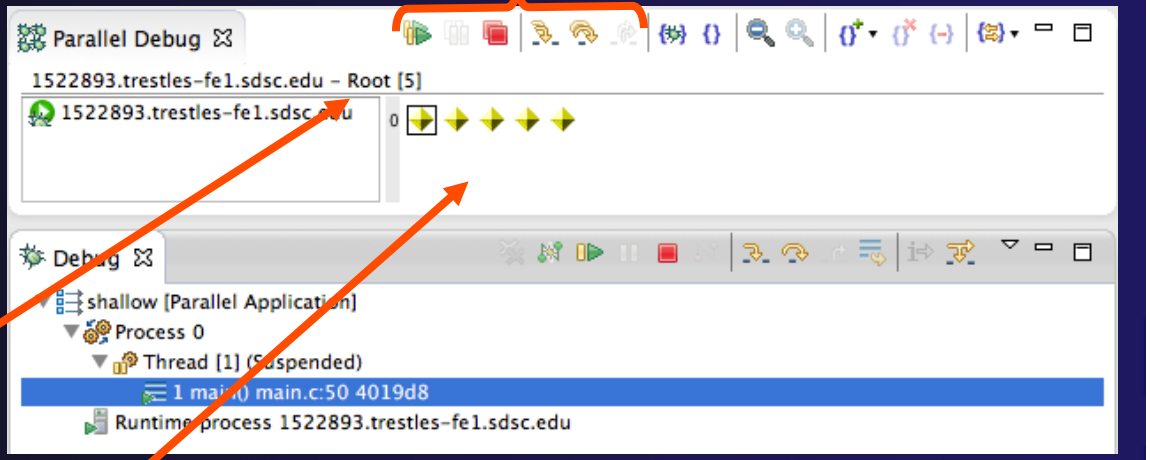
Process Sets (2)

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



Operations On Process Sets

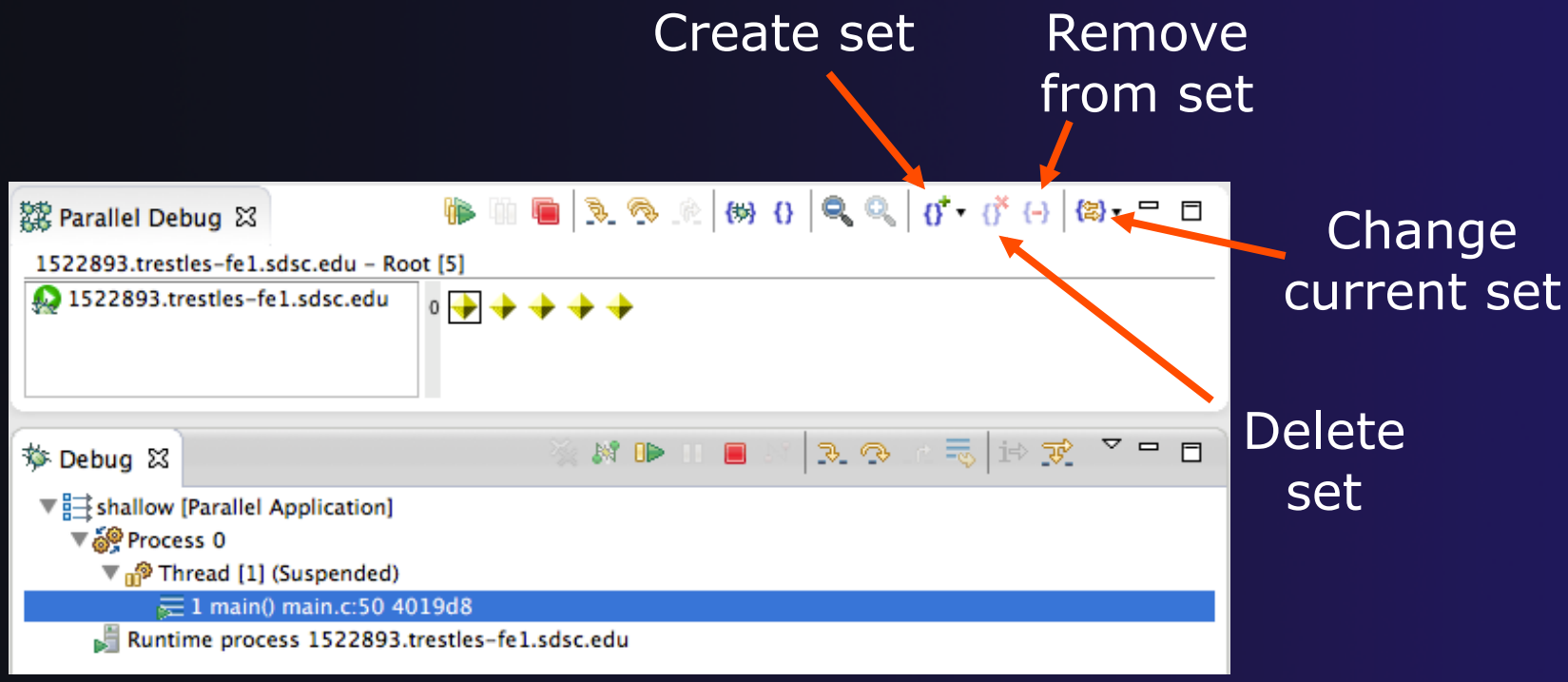
- Debug operations on the **Parallel Debug view** toolbar always apply to the current set:
  - Resume, suspend, stop, step into, step over, step return
- The current process set is listed next to job name along with number of processes in the set
- The processes in process set are visible in right hand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



Root set = all processes

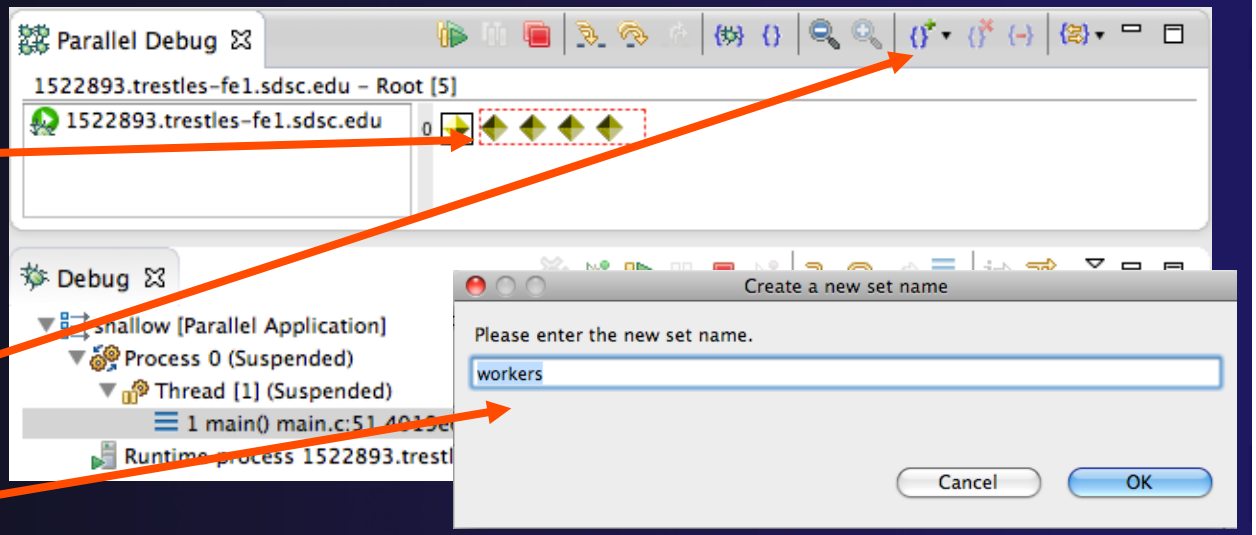
Managing Process Sets

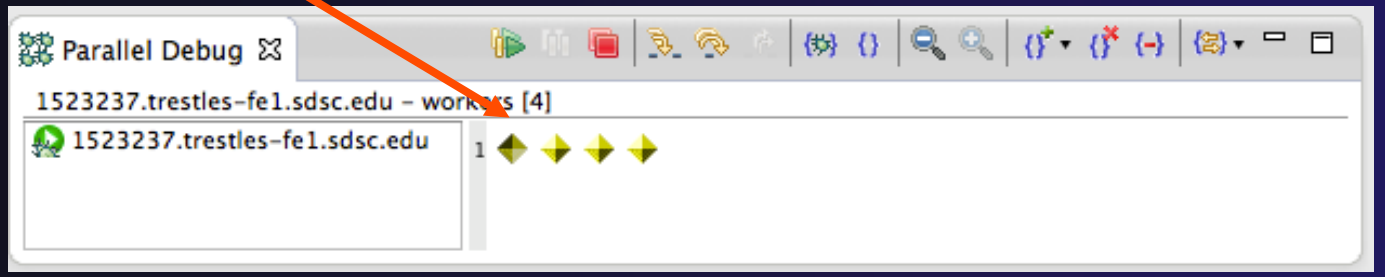
- The remaining icons in the toolbar of the **Parallel Debug view** allow you to create, modify, and delete process sets, and to change the current process set



Creating A New Process Set

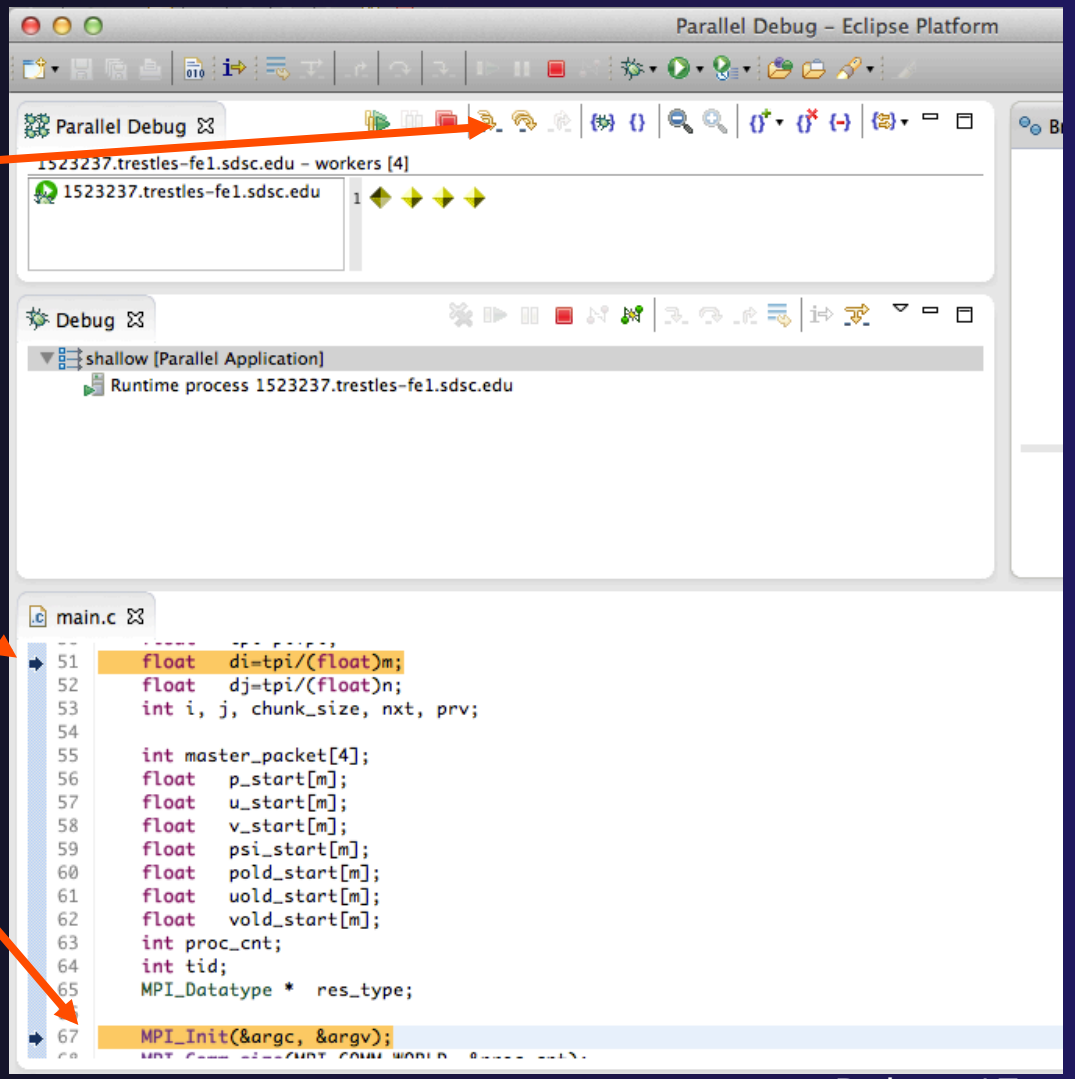
- Select the processes in the set by clicking and dragging, in this case, the last three
- The **Create Set** button enables a new process set to be created
- The set can be given a name, in this case **workers**
- The view is changed to display only the selected processes





Stepping Using New Process Set

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

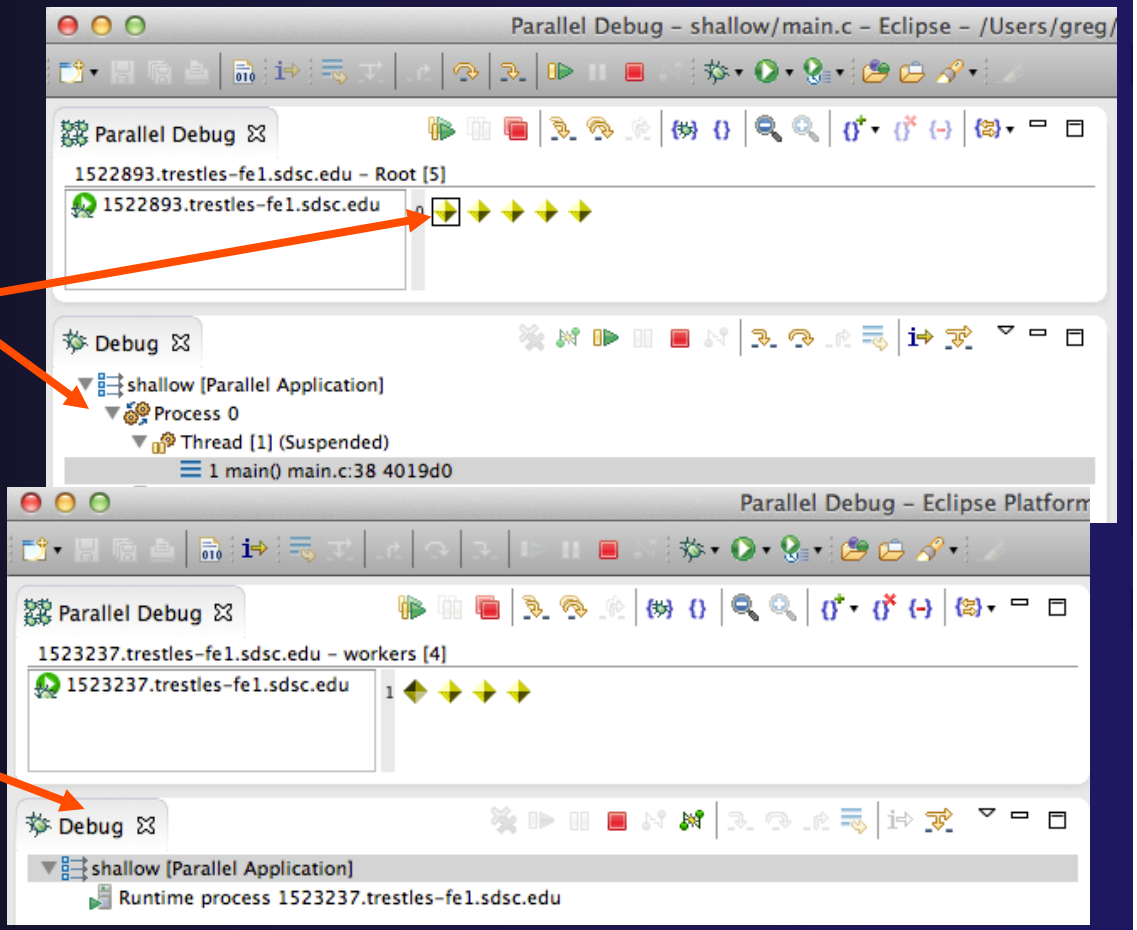


Process Registration

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

Process Registration (2)

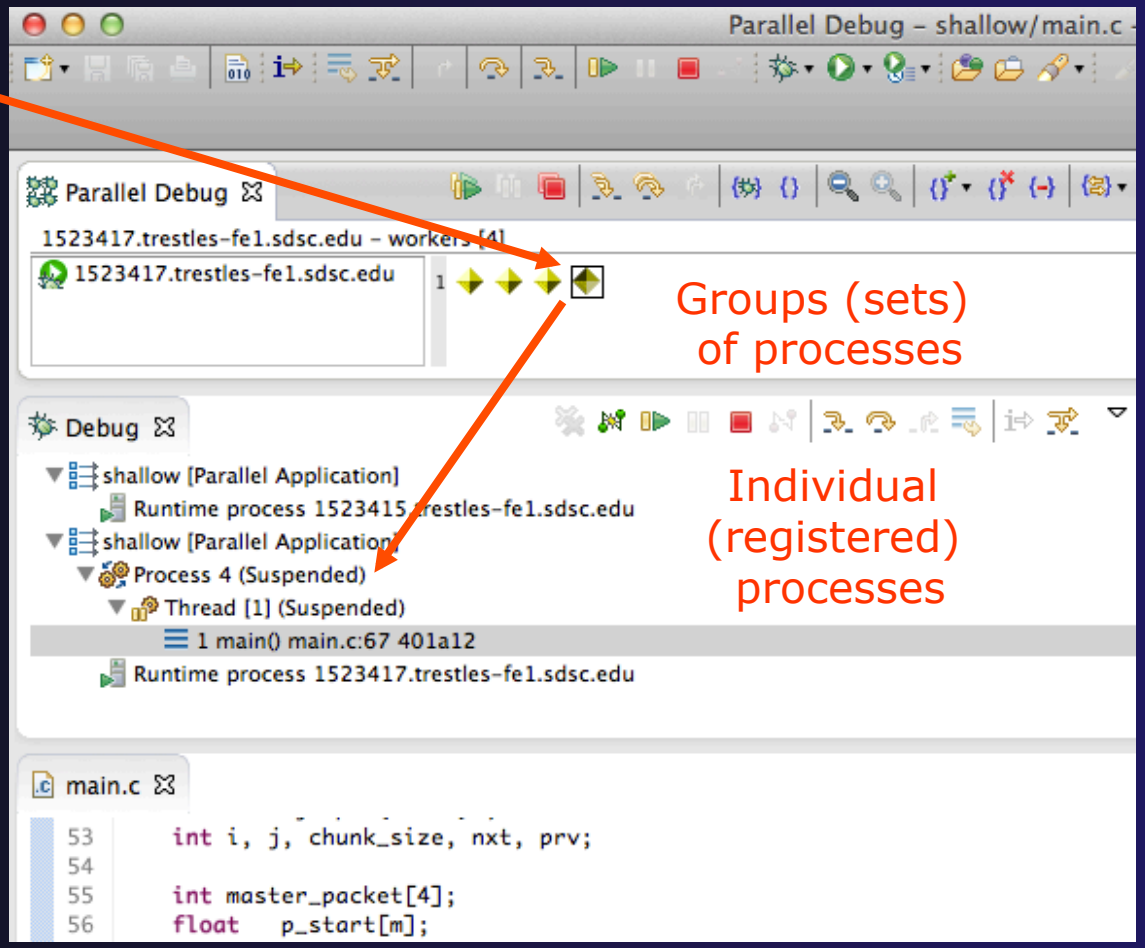
- By default, process 0 was registered when the debug session was launched
- Registered processes are surrounded by a box and shown in the Debug view



- The Debug view only shows registered processes in the current set
- Since the "workers" set doesn't include process 0, it is no longer displayed in the Debug view

Registering A Process

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

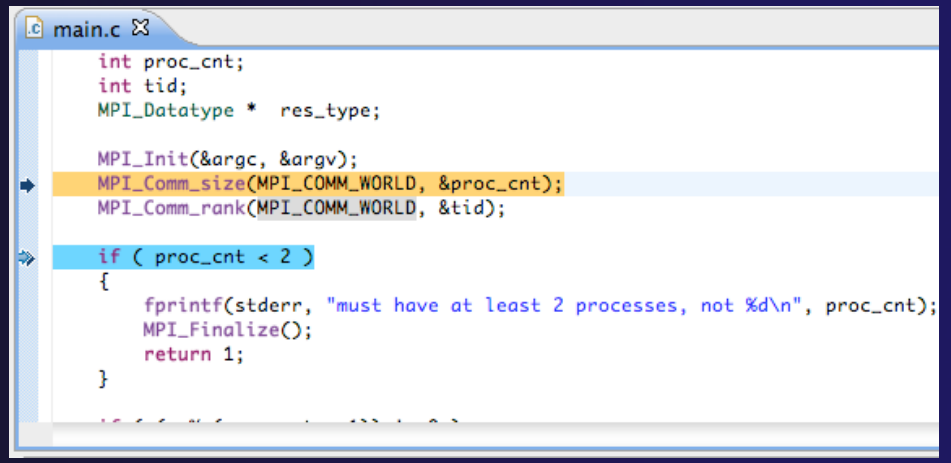


Current Line Marker

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

Colors And Markers

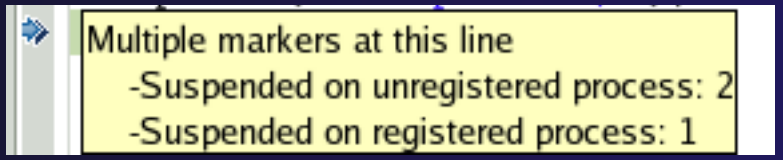
- The highlight color depends on the processes suspended at that line:
  - **Blue:** All registered process(es)
  - **Orange:** All unregistered process(es)
  - **Green:** Registered or unregistered process with no source line (e.g. suspended in a library routine)
- The marker depends on the type of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



 Multiple processes marker

 Registered process marker

 Un-registered process marker

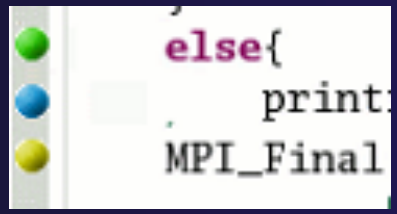


Exercise

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

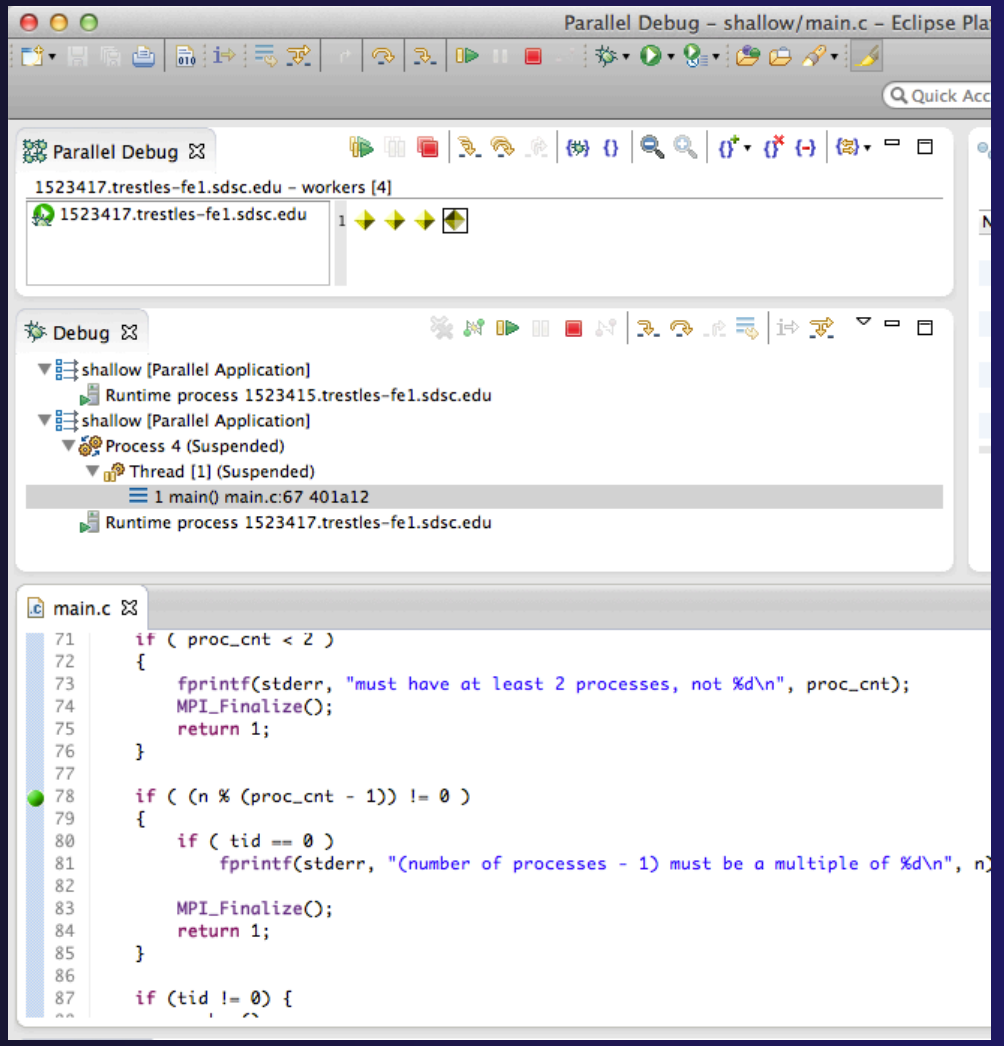
Breakpoints

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



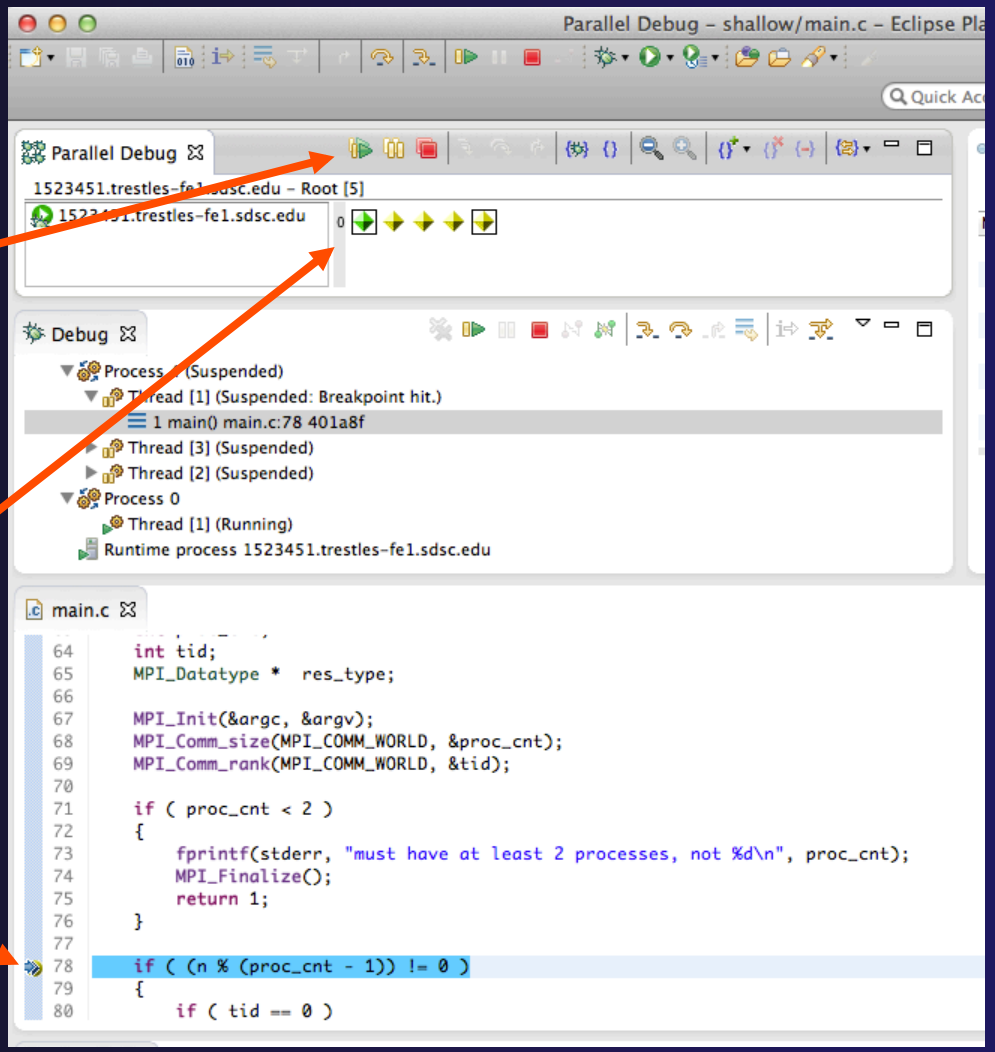
Creating A Breakpoint

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



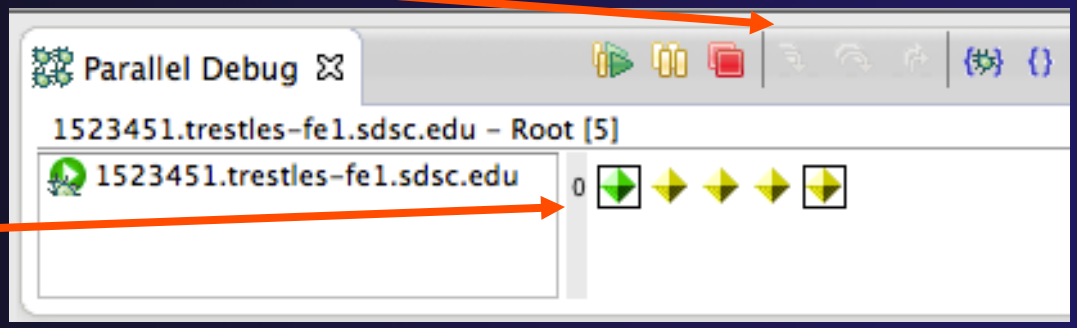
Hitting the Breakpoint

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

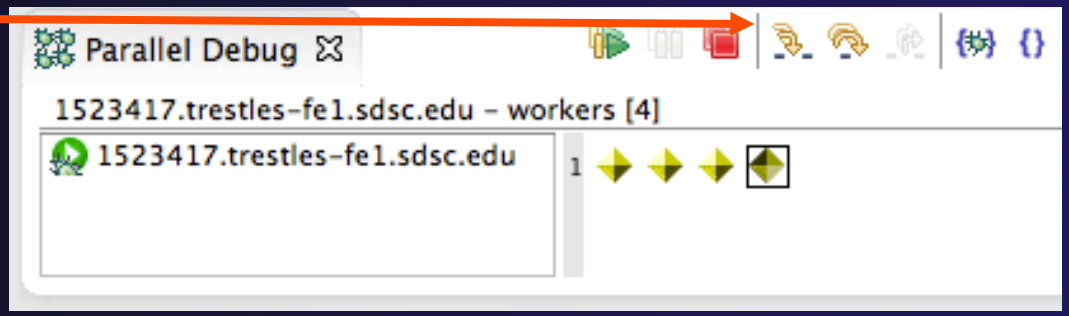


More On Stepping

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

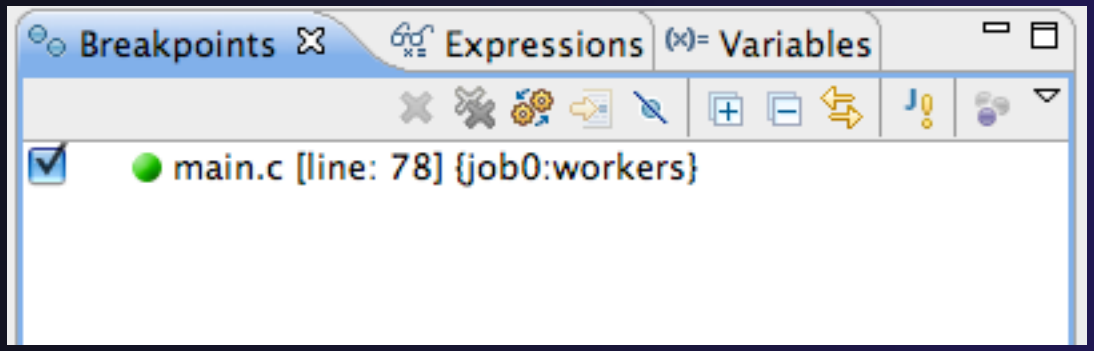


- Switch to the set of suspended processes (the **workers** set)
- You will now see the **Step** buttons become enabled



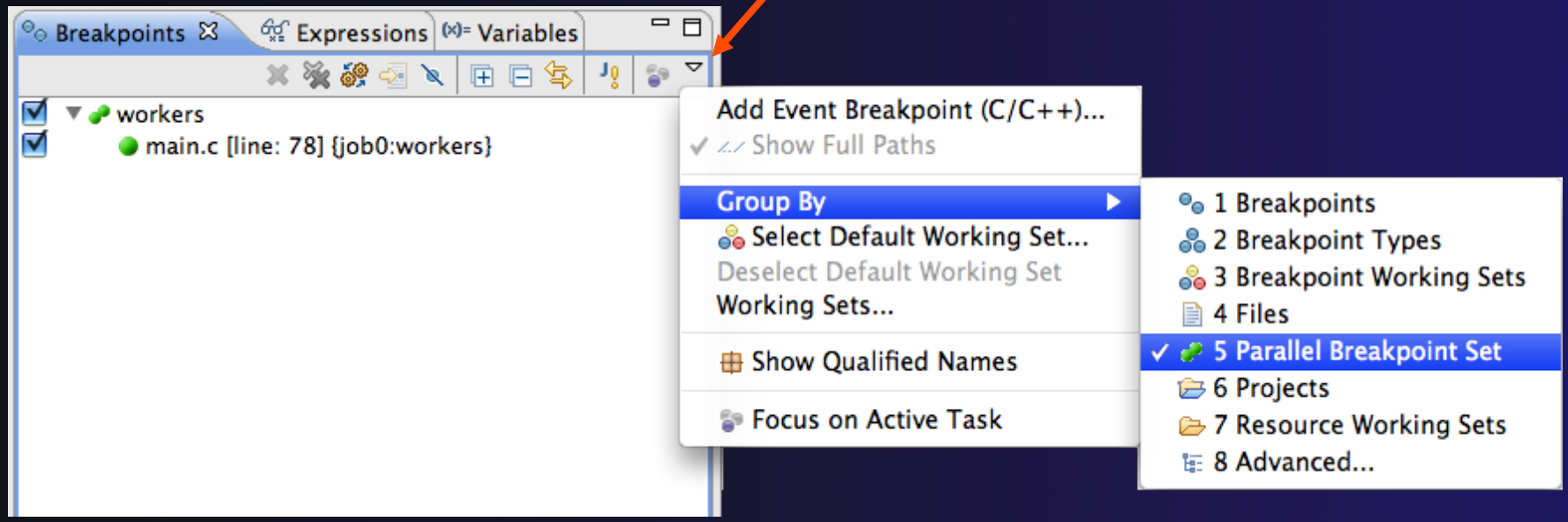
Breakpoint Information

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



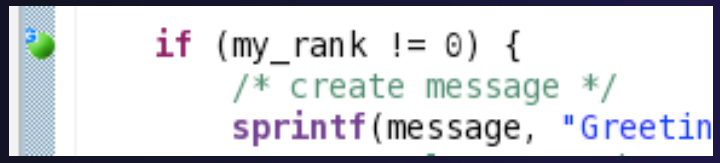
Breakpoints View

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



Global Breakpoints

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

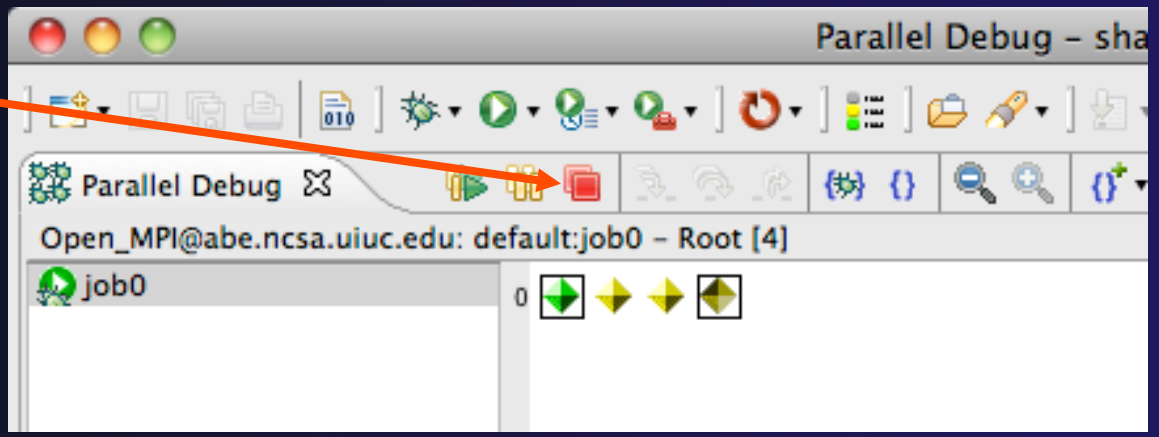


Exercise

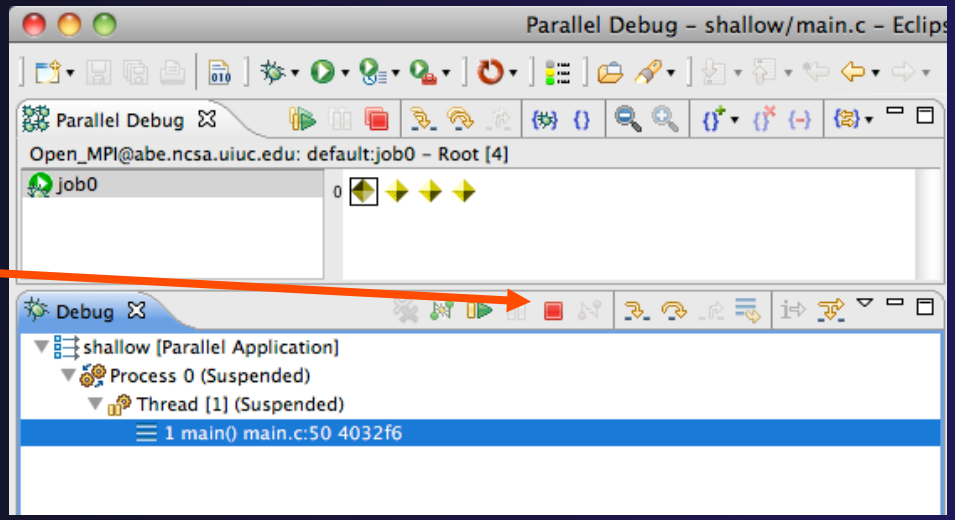
1. Select the “worker” process set
2. Create a breakpoint by double-clicking on right hand bar at line 88 (worker function)
3. Hover over breakpoint to see properties
4. Switch to “root” process set
5. Observer breakpoint color changes to blue
6. Resume all processes
7. Observe “worker” processes at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

Terminating A Debug Session

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

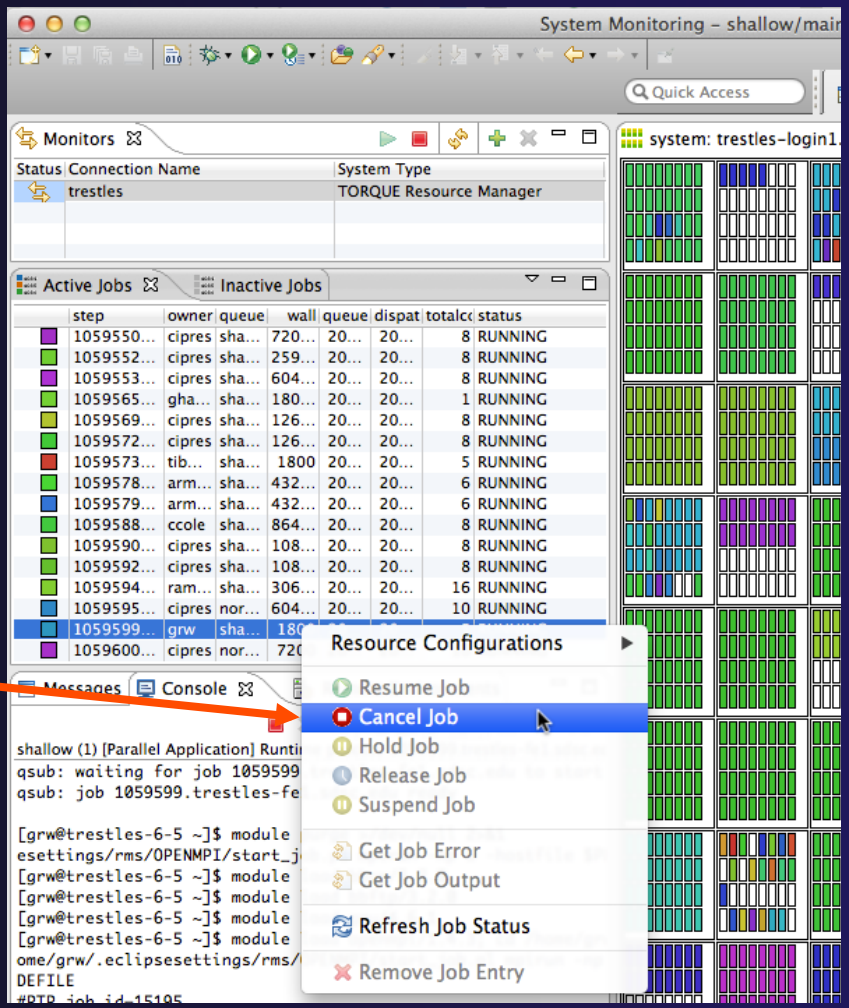


- You can also use the terminate icon in the **Debug** view to terminate the currently selected process



Cancelling The Job

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



Exercise

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

Optional Exercise

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

Performance Tuning and Analysis Tools

- Objective
  - Become familiar with tools integrated with PTP, to help enhance performance of parallel applications
- Contents
  - Overview of ETFw and Performance Tools

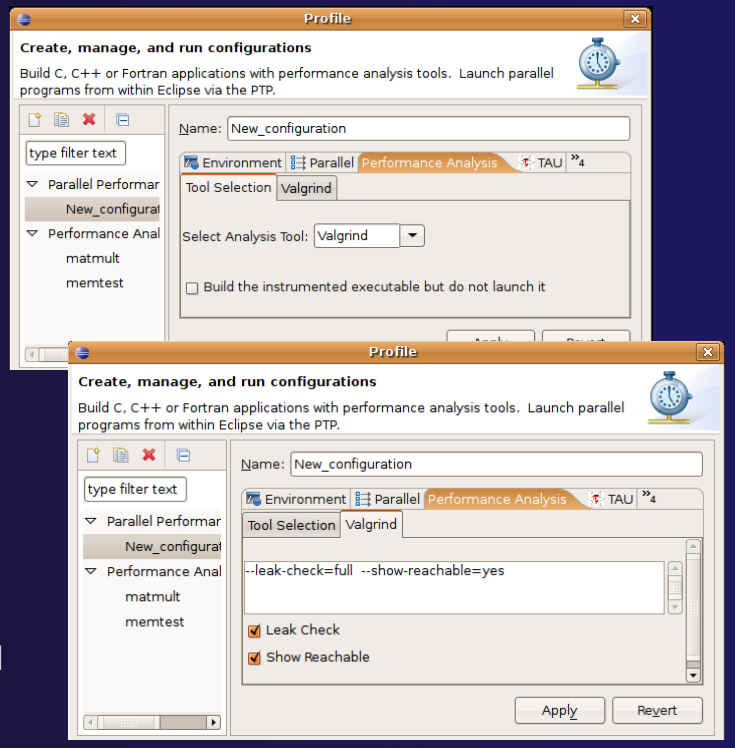
PTP/External Tools Framework

formerly “Performance Tools Framework”

Goal:

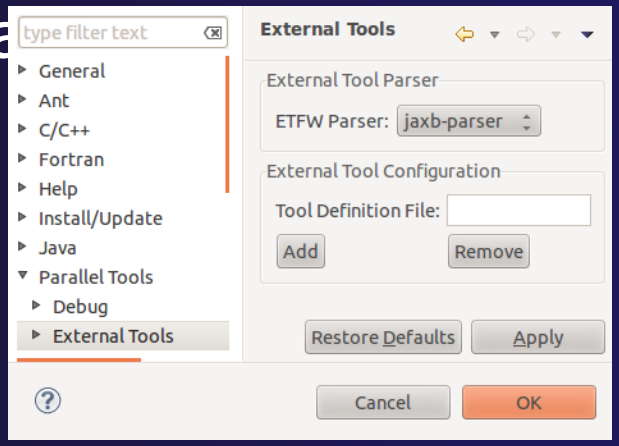
- **Reduce the “eclipse plumbing” necessary to integrate tools**
- Provide integration for instrumentation, measurement, and analysis for a variety of performance tools
  - Dynamic Tool Definitions: Workflows & UI
  - Tools and tool workflows are specified in an XML file
  - Tools are selected and configured in the launch configuration window
  - Output is generated, managed and analyzed as specified in the workflow
  - One-click ‘launch’ functionality
  - Support for development tools such as TAU, PPW and others.
  - Adding new tools is much easier than developing a full Eclipse plug-in

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



SAX and JAXB Tool Definitions

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



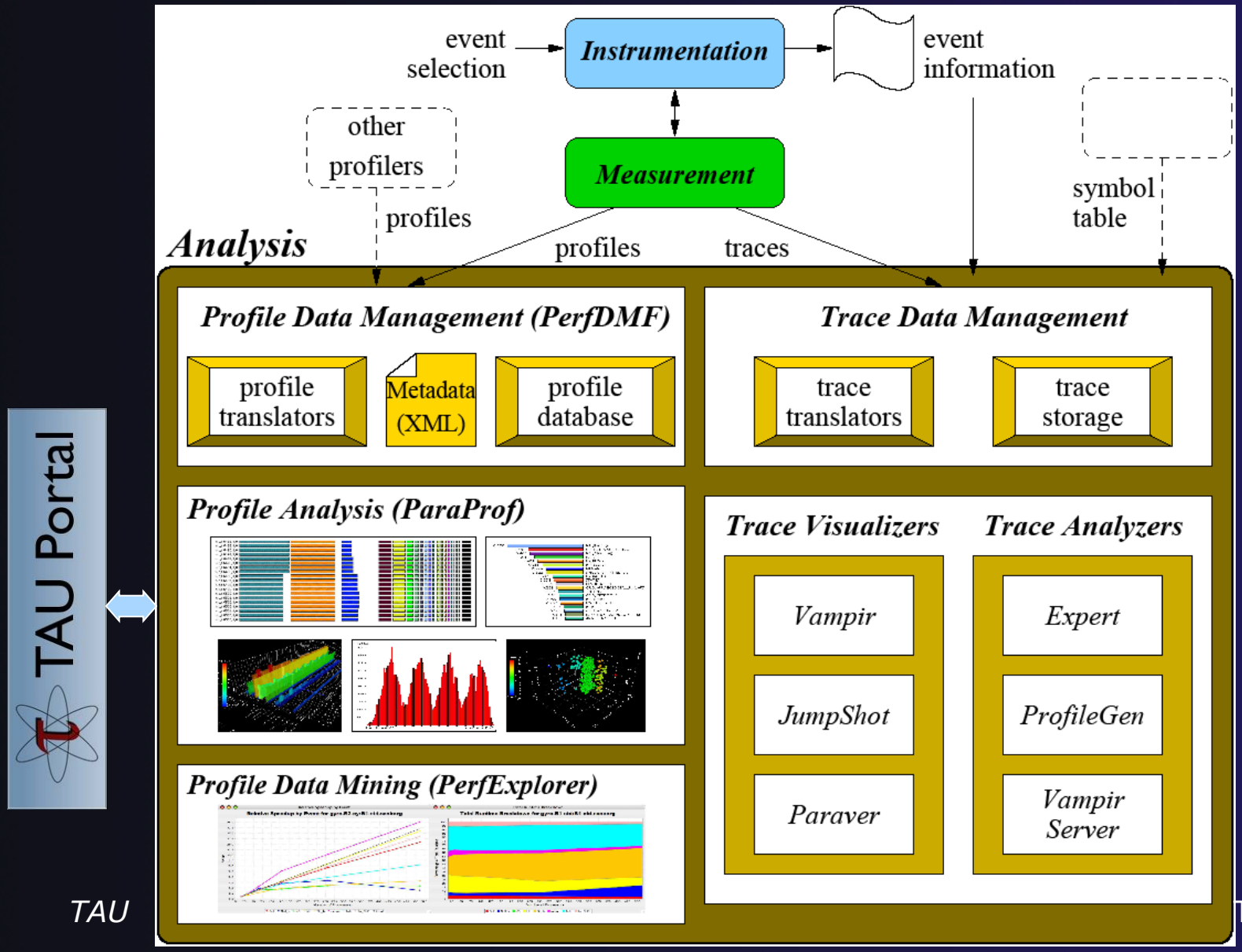
Performance Tuning and Analysis Tools - TAU

- Objective
  - Become familiar with tools integrated with PTP, to help enhance performance of parallel applications
- Contents
  - Performance Tuning and external tools:
    - PTP External Tools Framework (ETFw), TAU Hands-on exercise using TAU with PTP

TAU: Tuning and Analysis Utilities

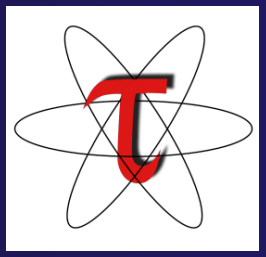
- TAU is a performance evaluation tool
- It supports parallel profiling and tracing
  - Profiling shows you how much (total) time was spent in each routine
  - Tracing shows you *when* the events take place in each process along a timeline
- TAU uses a package called PDT (Performance Database Toolkit)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

TAU Performance System Architecture

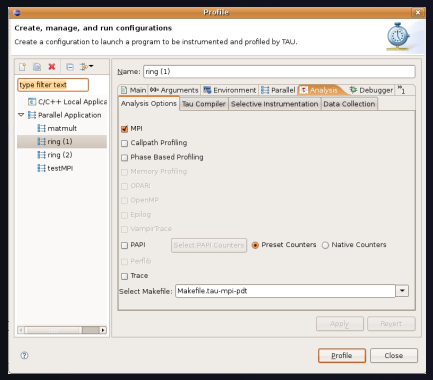


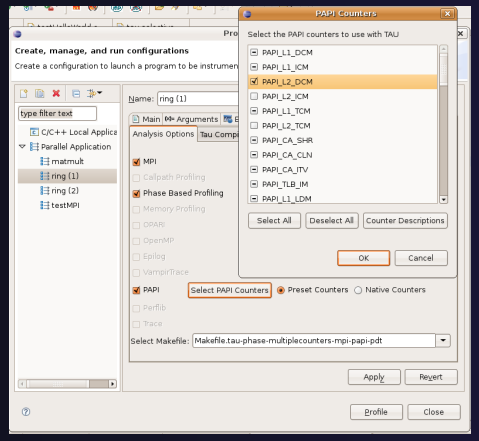
PTP TAU plug-ins

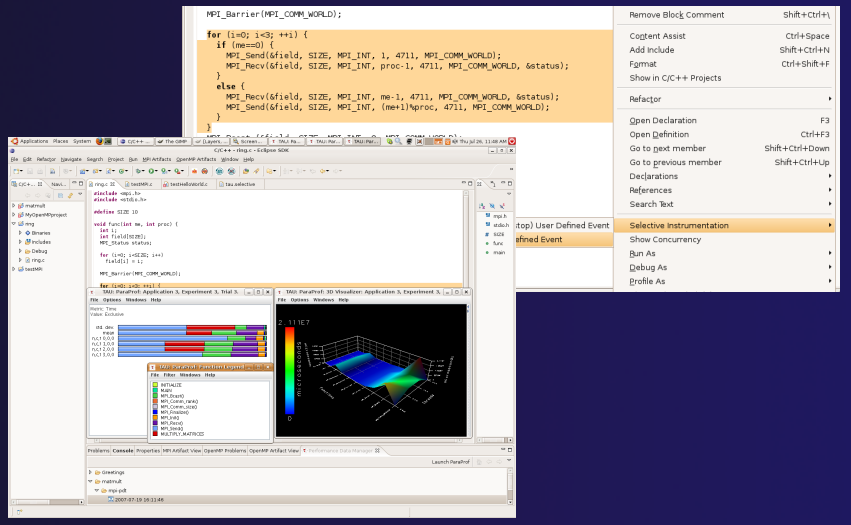
http://www.cs.uoregon.edu/research/tau



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

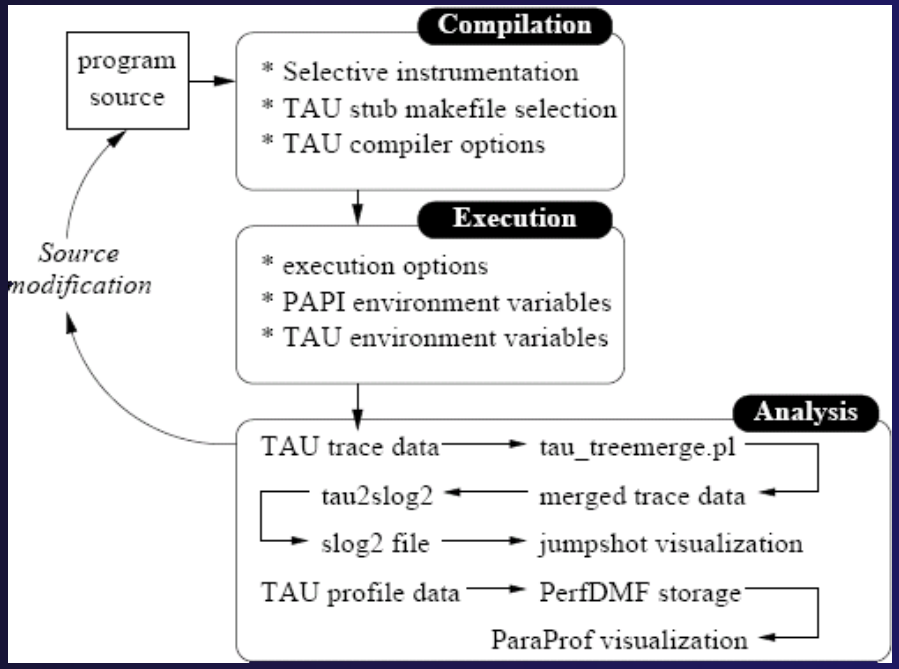






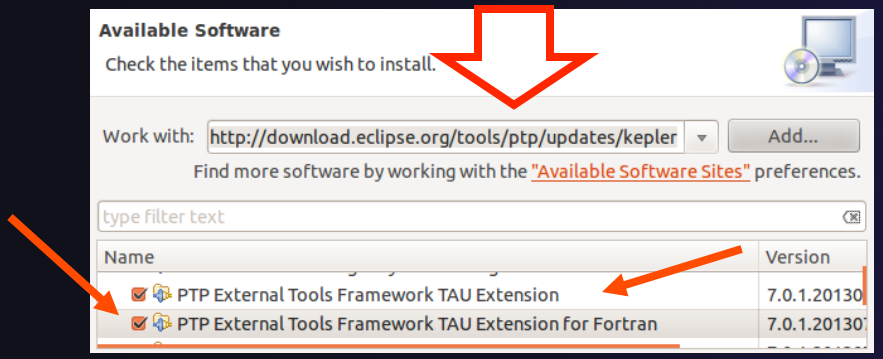
TAU Integration with PTP

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



TAU PTP Installation

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



To confirm:
- Help>Install New Software…
- Select the link "What is already installed" at the bottom of the dialog
- You should see the TAU Extension

Installing TAU Analysis Tools

- The TAU plugin can use ParaProf for visual analysis and TauDB for organization of profiles
- To install these utilities on Mac or Linux platforms:
  - Download (browser, curl or wget)

    tau.uoregon.edu/tautools-latest.tgz
  - tar -zxf tautools-latest.tgz
  - cd tautools-latest
  - ./configure
  - Set path as shown (launch eclipse from this environment)
  - Run taudb_configure and follow the instructions
- Java WebStart: tau.uoregon.edu/paraprof
- TAU Installation, downloads and instructions: tau.uoregon.edu

Assumptions

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

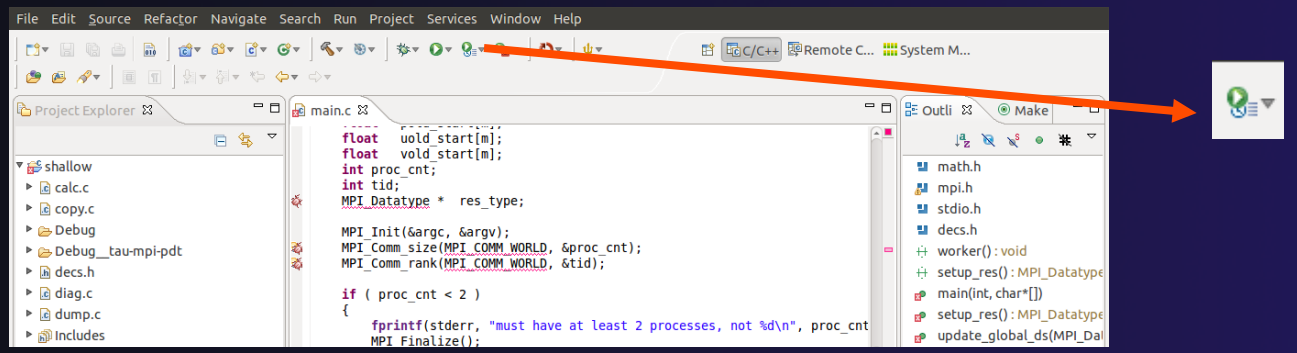
* SC tutorial: this has been done for you

Selective Instrumentation

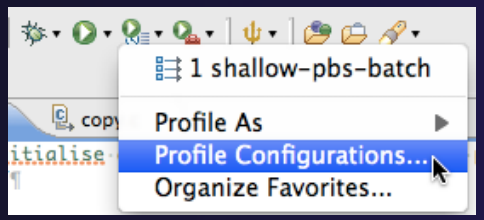
- By default tau provides timing data for each subroutine of your application
- Selective instrumentation allows you to include/ exclude code from analysis and control additional analysis features
  - Include/exclude source files or routines
  - Add timers and phases around routines or arbitrary code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

Begin Profile Configuration

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



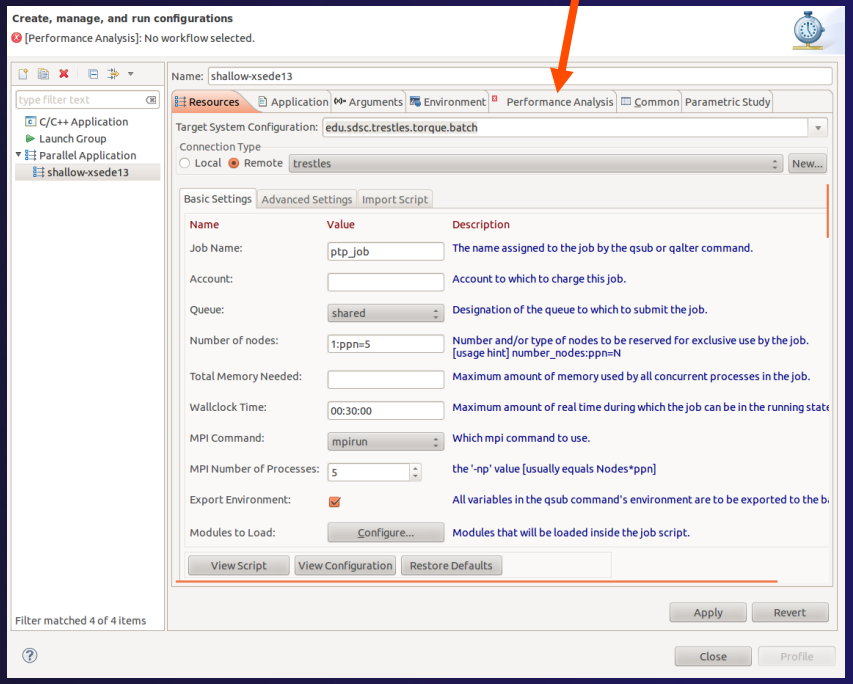
- From the dropdown menu select 'Profile configurations…'



Select Configuration

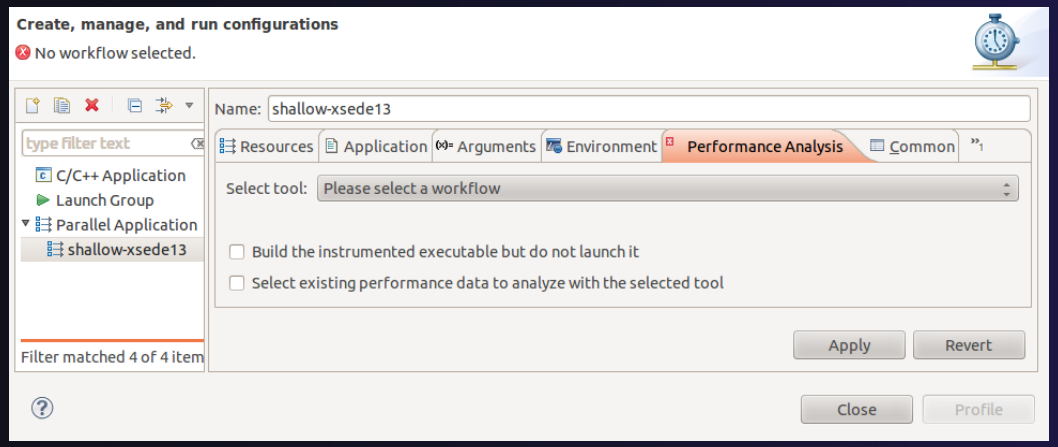
- Select the shallow configuration prepared earlier
- The Resource and Application configuration tabs require little or no modification
  - We are using the same resource manager and Torque settings
  - Since we are using a makefile project the application will be rebuilt in and run from the previously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

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



Select Tool/Workflow

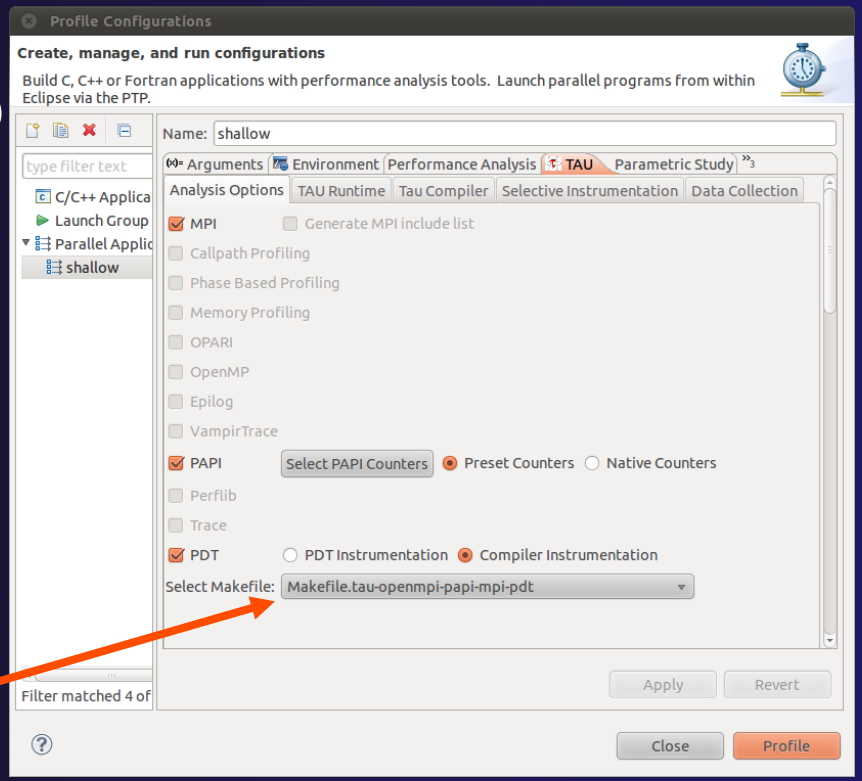
- Select the **Performance Analysis** tab and choose the TAU tool set in the 'Select Tool' dropdown box
  - Other tools may be available, either installed as plug-ins or loaded from workflow definition XML files
  - Configuration sub-panes appear depending on the selected tool



Tabs may be hidden if the window is too small

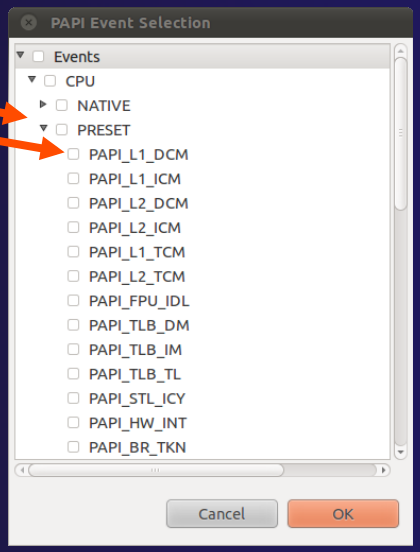
Select TAU Configuration

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



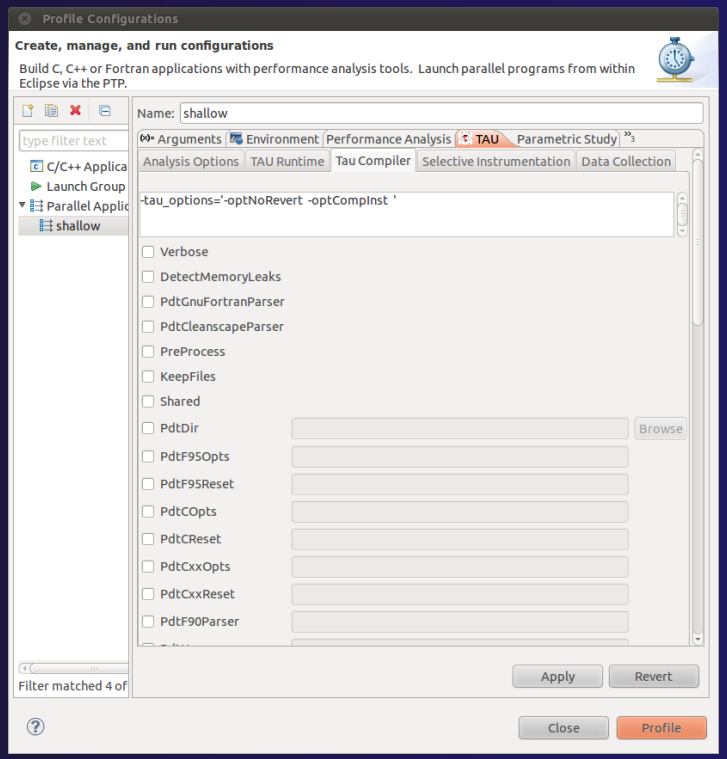
Choose PAPI Hardware Counters

- When a PAPI-enabled TAU configuration is selected the PAPI Counter tool becomes available
  - Select the 'Select PAPI Counters' button to open the tool
  - Open the PRESET subtree
  - Select PAPI_L1_DCM (Data cache misses)
  - Scroll down to select PAPI_FP_INS (Floating point instructions)
  - Invalid selections are automatically excluded
  - Select **OK**
  - **Not available on trestles.sdsc.edu**



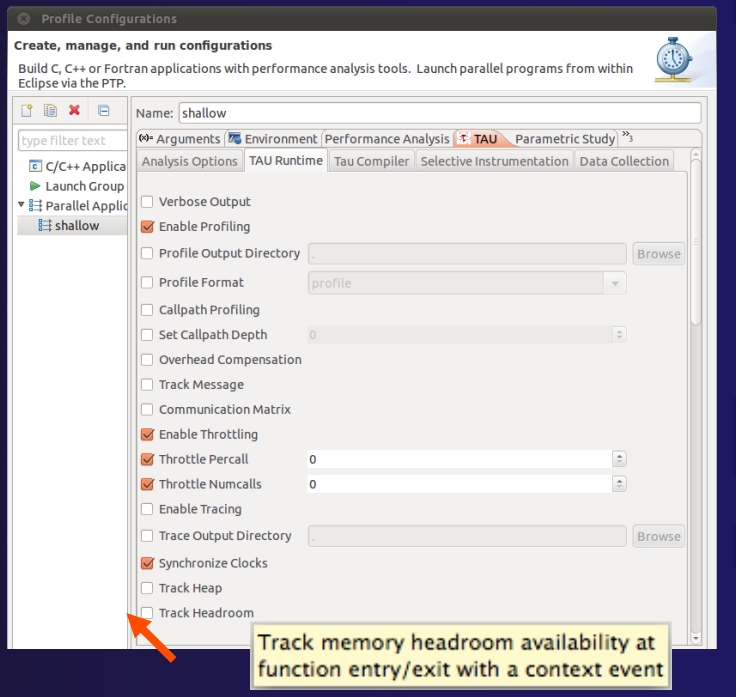
Compiler Options

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



Runtime Options

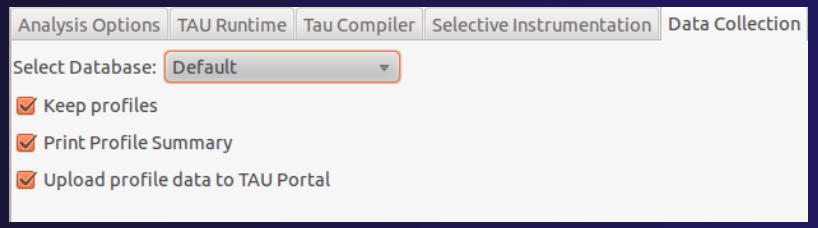
- TAU Runtime options
  - Set environment variables used by TAU
  - Control data collection behavior
  - Verbose provides debugging info
  - Callpath shows call stack placement of events
  - Throttling reduces overhead
  - Tracing generates execution timelines
- Set Profile Format to merged

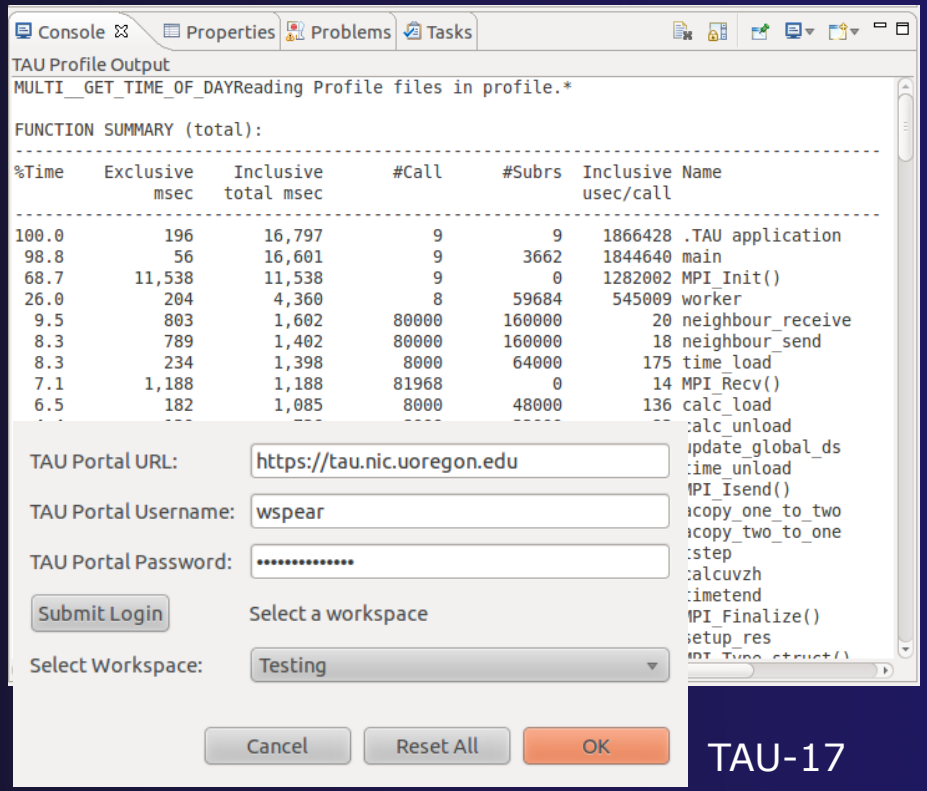


Hover help

Working with Profiles

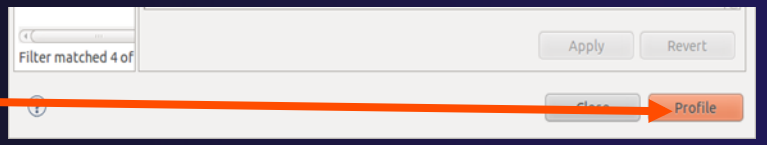
- Profiles are uploaded to selected database
- A text summary may be printed to the console
- Profiles may be uploaded to the TAU Portal for viewing online
  - tau.nic.uoregon.edu
- Profiles may be copied to your workspace and loaded in ParaProf from the command line. Select Keep Profiles



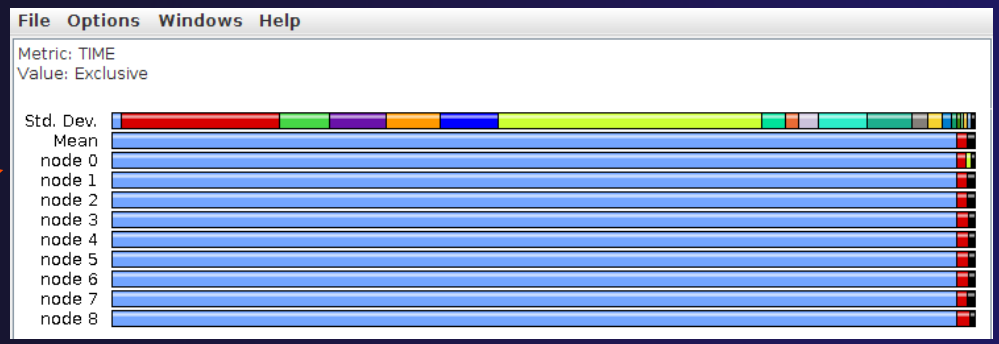


Launch TAU Analysis

- Once your TAU launch is configured select 'Profile'

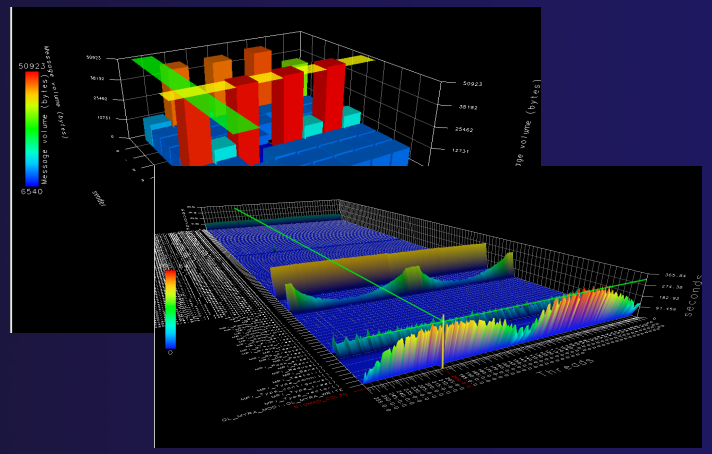


- Notice that the project rebuilds with TAU compiler commands
- The project will execute normally but TAU profiles will be generated
- TAU profiles will be processed as specified in the launch configuration.
- If you have a local profile database the run will show up in the Performance Data Management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



Paraprof

- Use ParaProf for profile visualization to identify performance hotspots
  - Inefficient sequential computation
  - Communication overhead
  - IO/Memory bottlenecks
  - Load imbalance
  - Suboptimal cache performance



- Compare multiple trials in PerfExplorer to identify performance regressions and scaling issues
- To use ParaProf, install TAU from tau.uoregon.edu or use Java webstart from tau.uoregon.edu/paraprof

Exercise

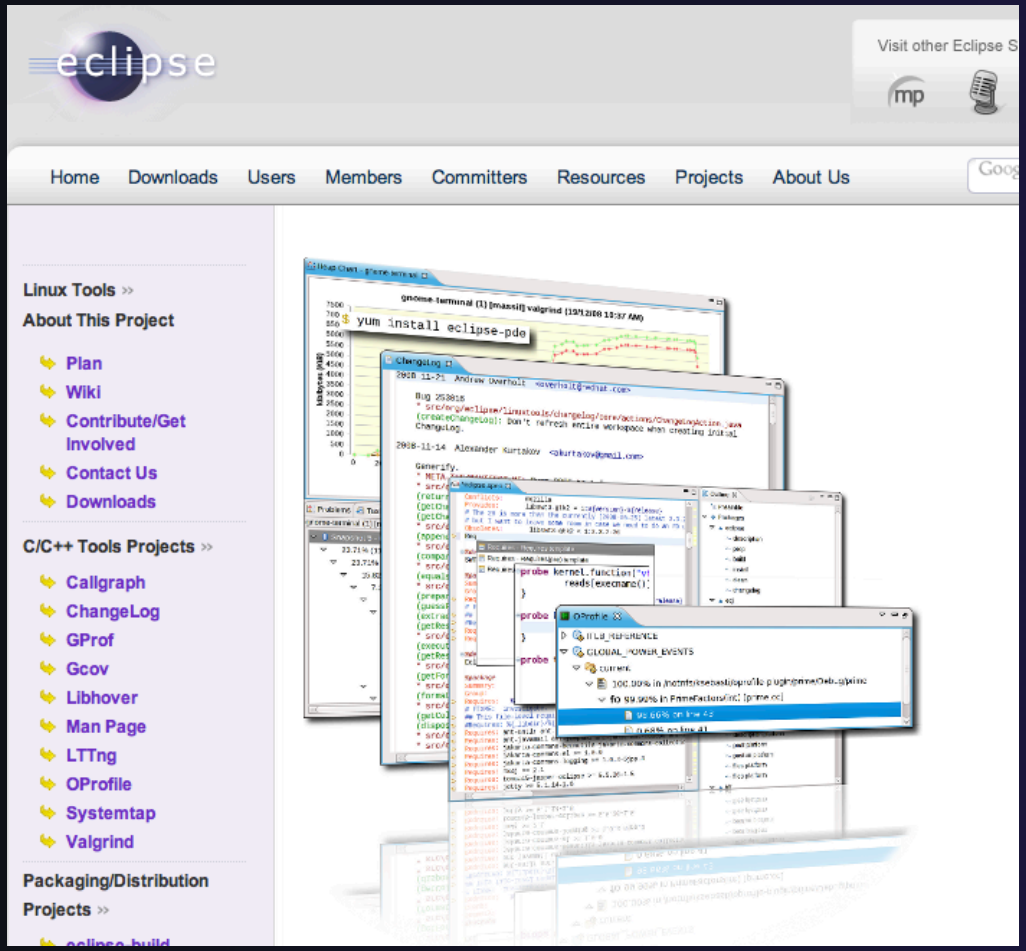
- Multi-Trial profile comparison
  1. Edit the shallow Makefile, adding -O3 to CFLAGS and FFLAGS
  2. Rerun the analysis (Run->Profile Configurations. Hit Profile)
  3. A second trial, distinguished by a new timestamp, will be generated
     - It will appear in your Performance Data Manager view if a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

Gcov and gprof support in linux tools

- Objective
  - Learn how to use Eclipse-based interfaces to GNU tools Gcov and Gprof
- Contents
  - Build with “-pg” for gprof profiling
  - Build with “-ftest-coverage –fprofile-arcs” for gcov
  - Run gcov to determine code coverage – which parts of your program are logically getting exercised
  - Run gprof to determined which parts of your program are taking most of the execution time

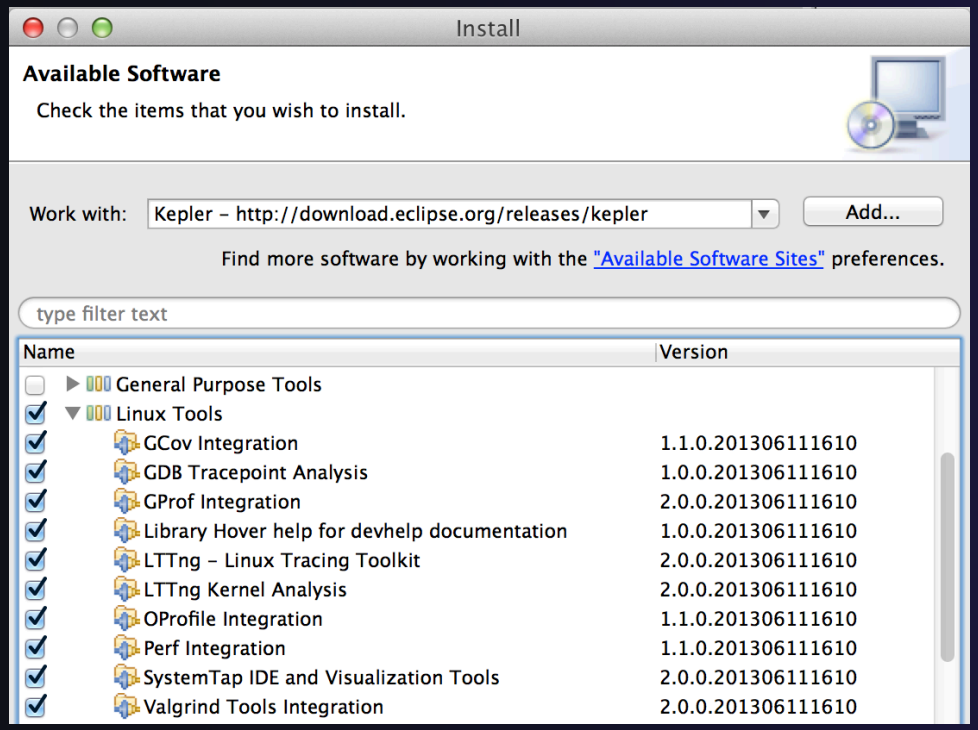
Linux Tools

http://eclipse.org/linuxtools



- What is Linux Tools?
- http://eclipse.org/linuxtools/
- Builds on CDT for C/C++
- Integrates popular native development tools such as Valgrind, OProfile, RPM, SystemTap, GCov, GProf, LTTng, etc. – into Eclipse

Linux Tools - Installation



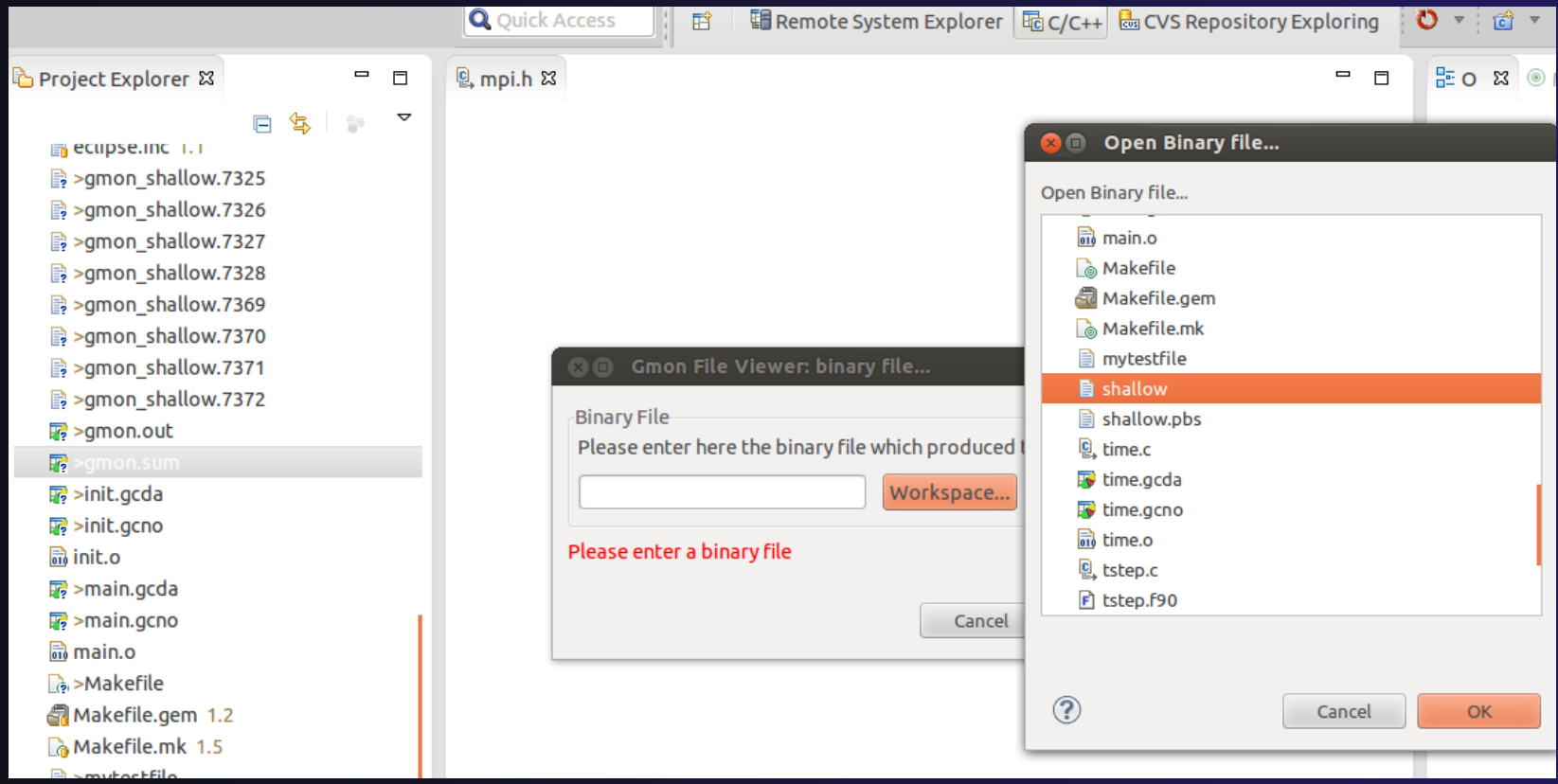
- Some of the Linux Tools are included with **Eclipse for Parallel Application Developers** package
- Everything you need for this tutorial is in the package
- To install manually:
  - Help > Install New Software
  - In **Work With:** Select Kepler update site
  - Under **Linux Tools**, Select the tools you want or just select all of them
    - Some cannot be installed on all non-Linux platforms
  - Click Next> … and continue to end of installation and restart Eclipse when prompted

Linux Tools - usage

- With a synchronized project, you only need gprof/gcov available on the remote system. (Even the Windows client can view the gprof and gcov output files.)
- Compiler flags
  - -pg to profile with gprof for the GNU compilers
  - -ftest-coverage -fprofile-arcs for gcov support
    - It's ok to use both at the same time at low optimization settings
- Re-run the application
- With synchronized projects, remember to re-sync to retrieve the resultant files
- `gprof -s shallow gmon_shallow.*` : creates a summary profile (gmon.sum) from MPI programs with a profile per rank

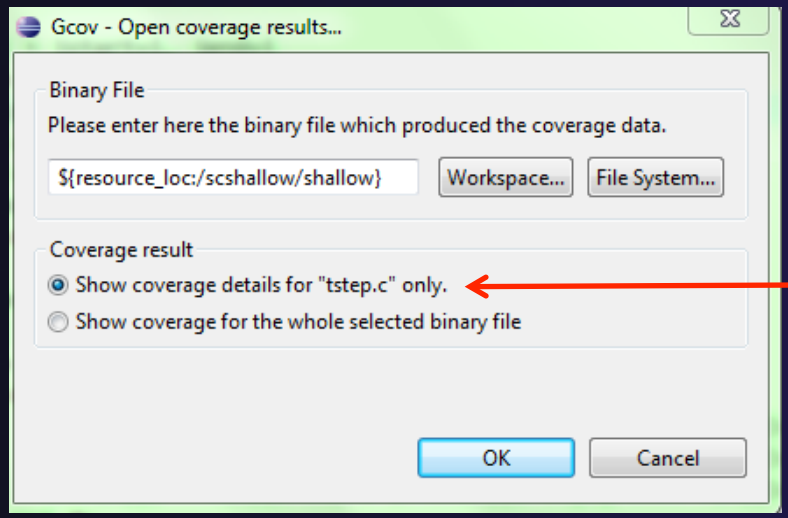
Linux Tools – click to view

- Double-click gmon.* for gprof view
- Double-click *.gcno or *.gcda for gcov view



Linux Tools – click to view

- Double-click gmon.* for gprof view
- Double-click *.gcno or *.gcda for gcov view,
  - It will ask for binary file location
  - Coverage Result: for Windows, select src file only (do not select the whole binary file in the radio button of the dialog box)

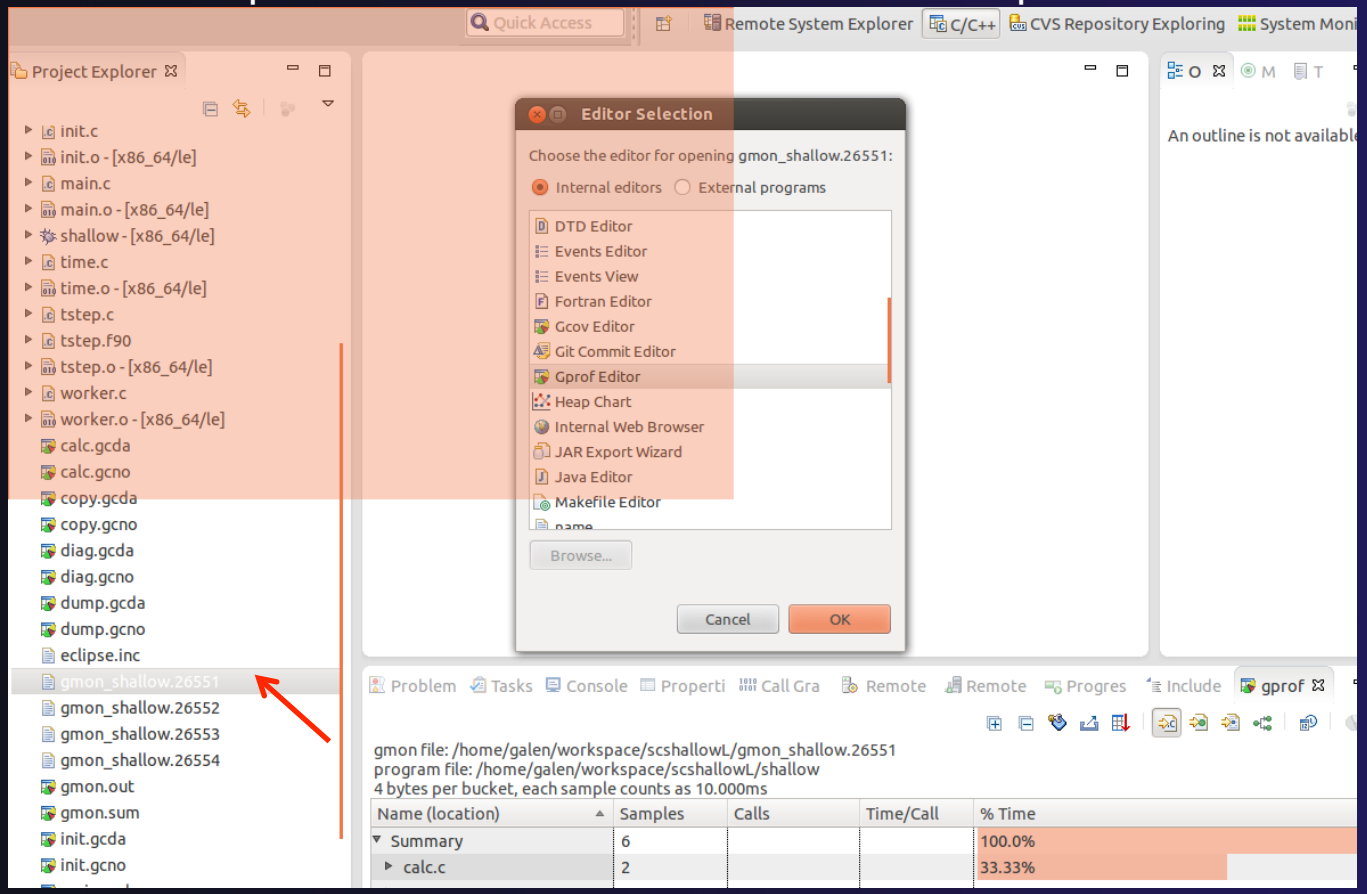


Windows only. Mac/Linux can show coverage for whole file

Windows only

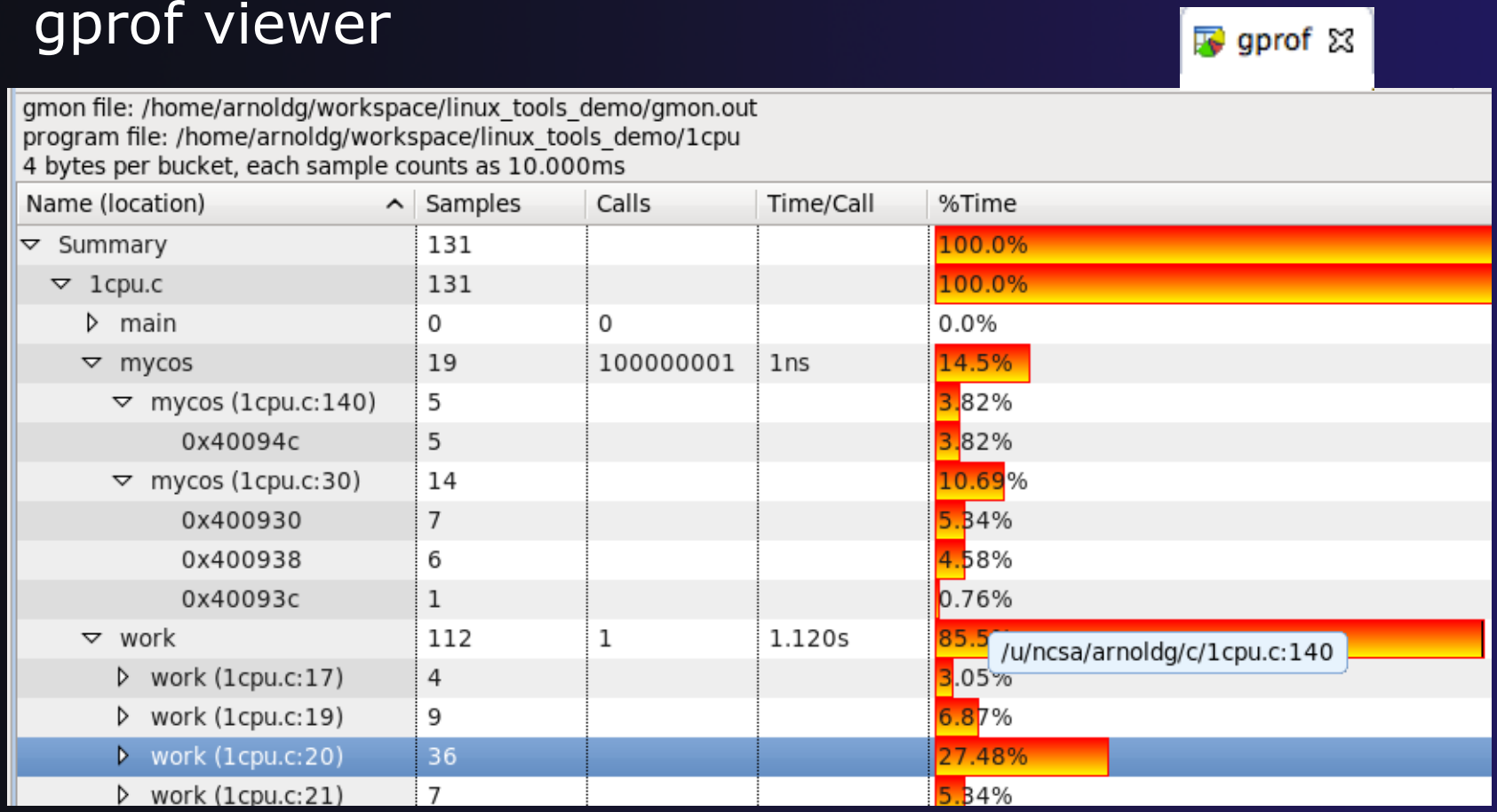
Opening a profile with gprof viewer

Note: since we've changed filename from 'gmon.out' to gmon_shallow.xxx we will force the gprof editor to be invoked for the files.
Use Right mouse > Open With… > Other … and choose Gprof Editor



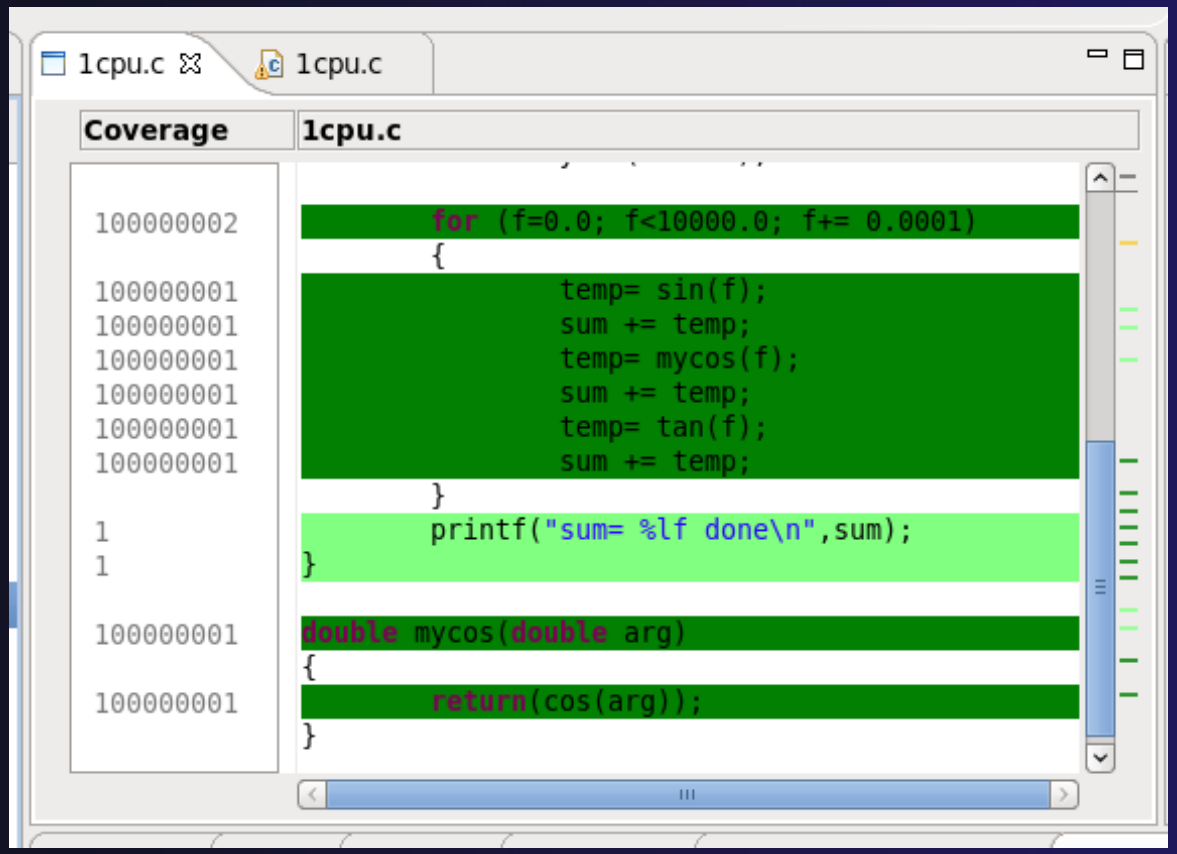
Gprof tab

Double-click on gmon.out file to open gprof viewer

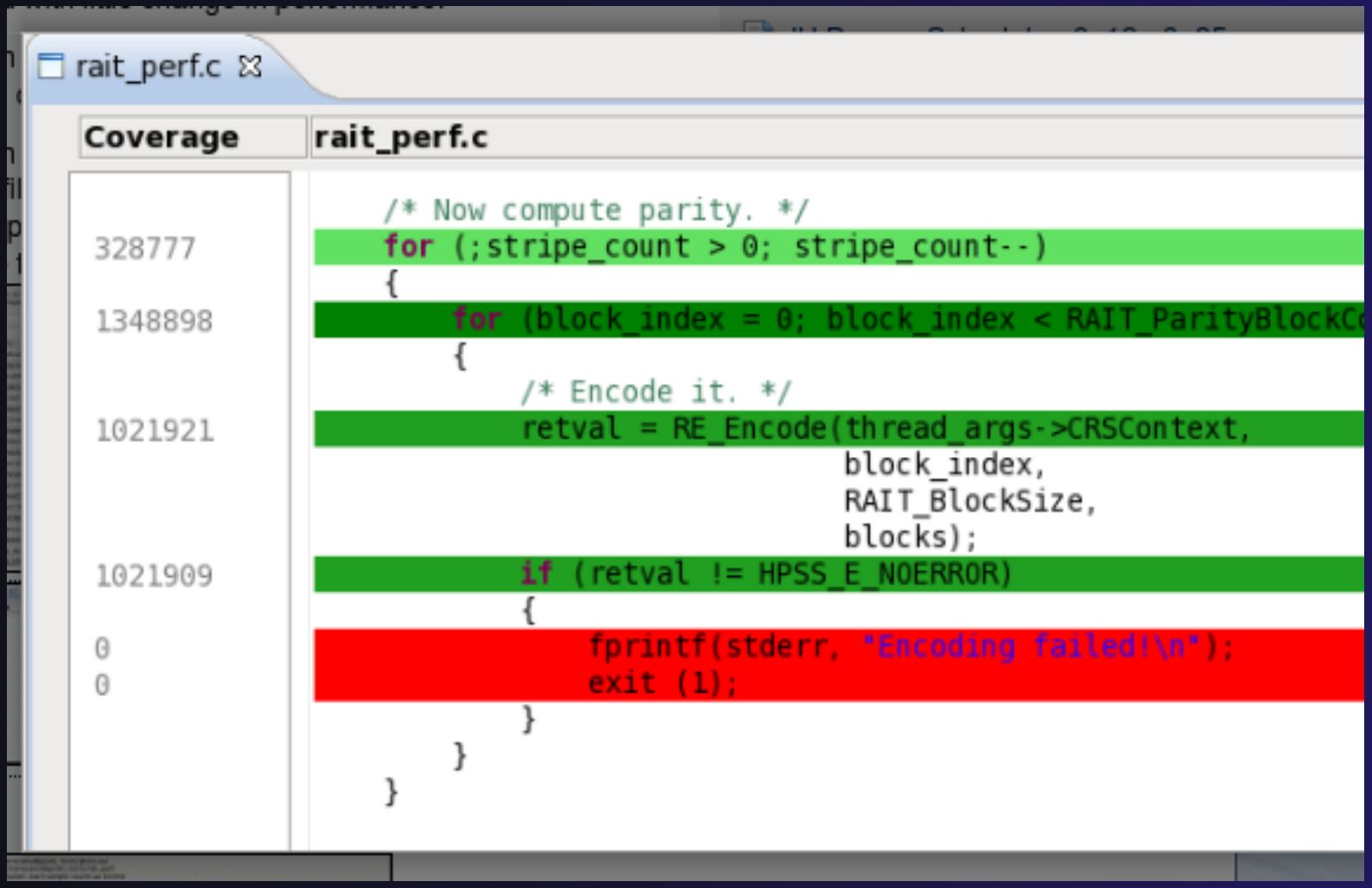


Run code, inspect gcov display

Double-click on a source file in gcov view to see code coverage highlighted in source file

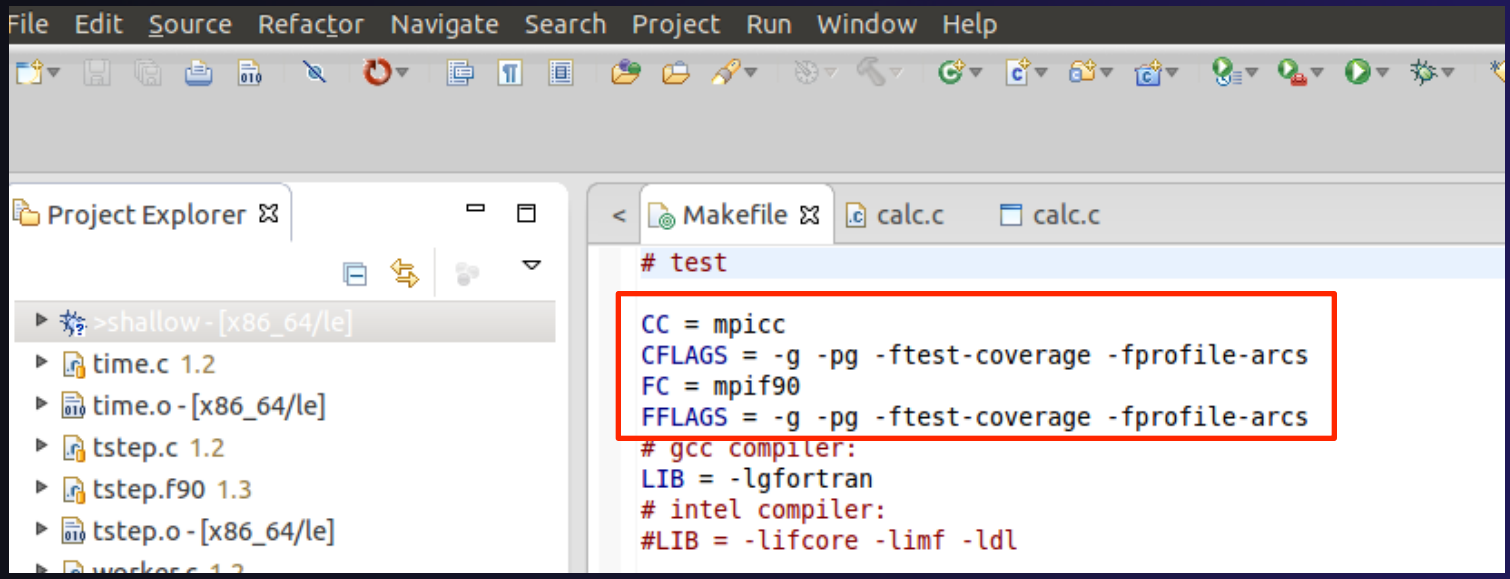


Gcov with a production code, unexecuted region



gprof with shallow project, MPI

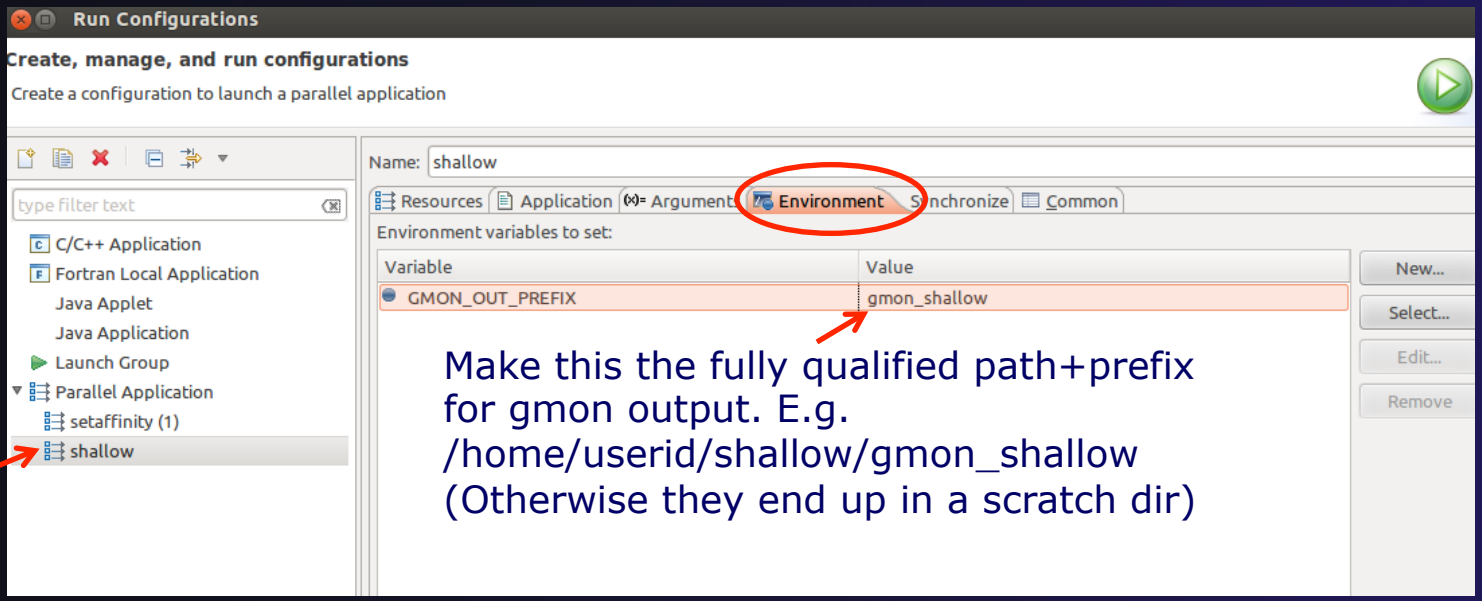
Add compiler flags to Makefile



The Makefile CFLAGS and FFLAGS are modified as shown to support profiling and coverage at the same time.
We have created the Makefile so you should just be able to uncomment these lines.

gprof with shallow project, MPI (2)

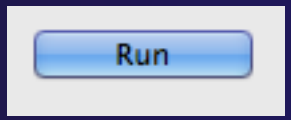
Modify Run Configuration
Run > Run Configurations … to modify existing Run Configuration



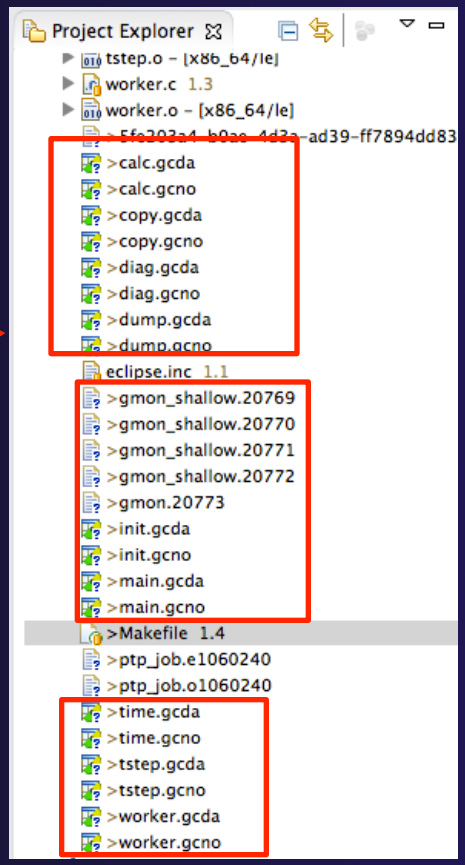
Setup the MPI run configuration with the Environment variable GMON_OUT_PREFIX defined with a /full/path/name for your individual MPI rank gmon outputs. By default gmon.out is used but MPI doesn't do that well and you end up with a profile that's missing most of the information, so by using GMON_OUT_PREFIX, each MPI rank adds its process id to its gmon output filename.

gprof with shallow project, MPI (3)

- Run from Run Configuration dialog



- See build results in Console
- See new files in Project Explorer (You may need to force a Sync)

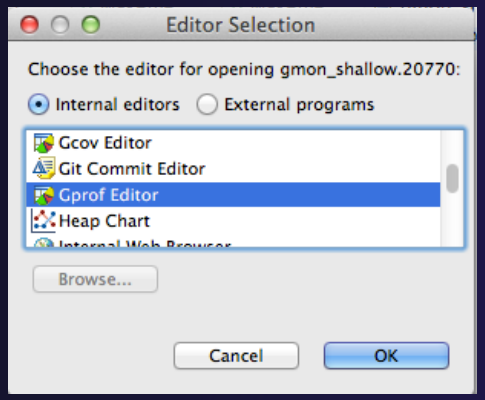


gprof with shallow project, 1 rank

- Open gmon file with gprof viewer
- Double-click on gmon.out file

  -or- since gmon_shallow.xxx has non-standard file types ….

- Right click on gmon_shallow.xxx file and select Rightmouse >Open With… Other… and select **Gprof Editor**

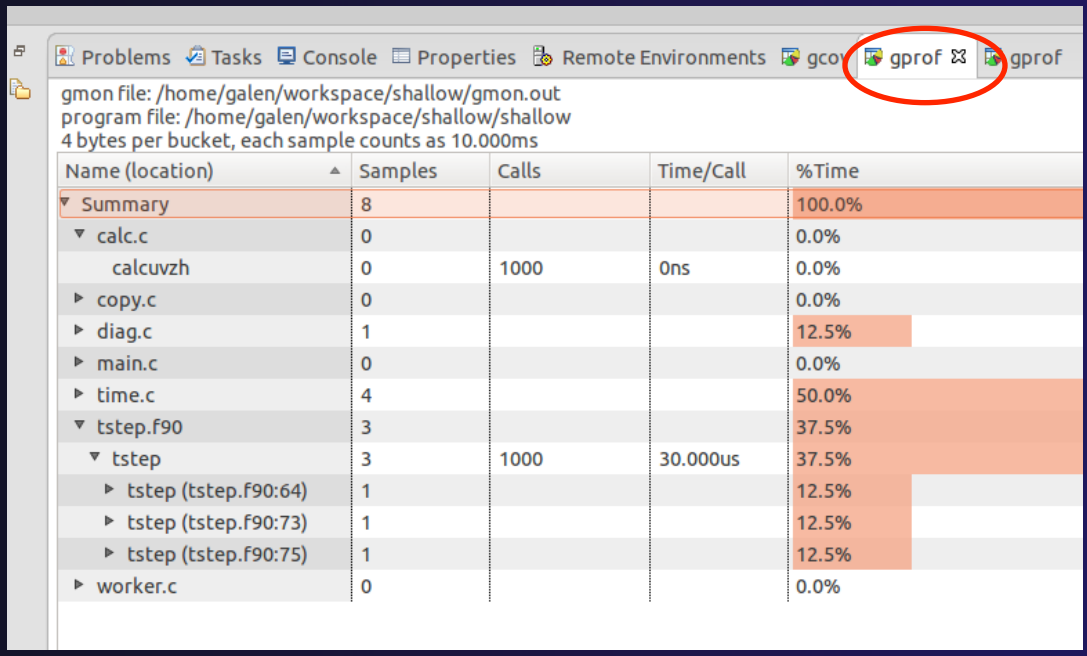


gprof with shallow project, 1 rank (2)

View gmon data with gprof viewer

It's interesting to compare the summary gmon output to that from one of the ranks.

This view shows a gmon.out file (you have a gmon_shallow.xxx file) from a single rank.



Name (location)	Samples	Calls	Time/Call	%Time
Summary	8			100.0%
calc.c	0			0.0%
calcuvzh	0	1000	0ns	0.0%
copy.c	0			0.0%
diag.c	1			12.5%
main.c	0			0.0%
time.c	4			50.0%
tstep.f90	3			37.5%
tstep	3	1000	30.000us	37.5%
tstep (tstep.f90:64)	1			12.5%
tstep (tstep.f90:73)	1			12.5%
tstep (tstep.f90:75)	1			12.5%
worker.c	0			0.0%

Gprof with shallow project, summary

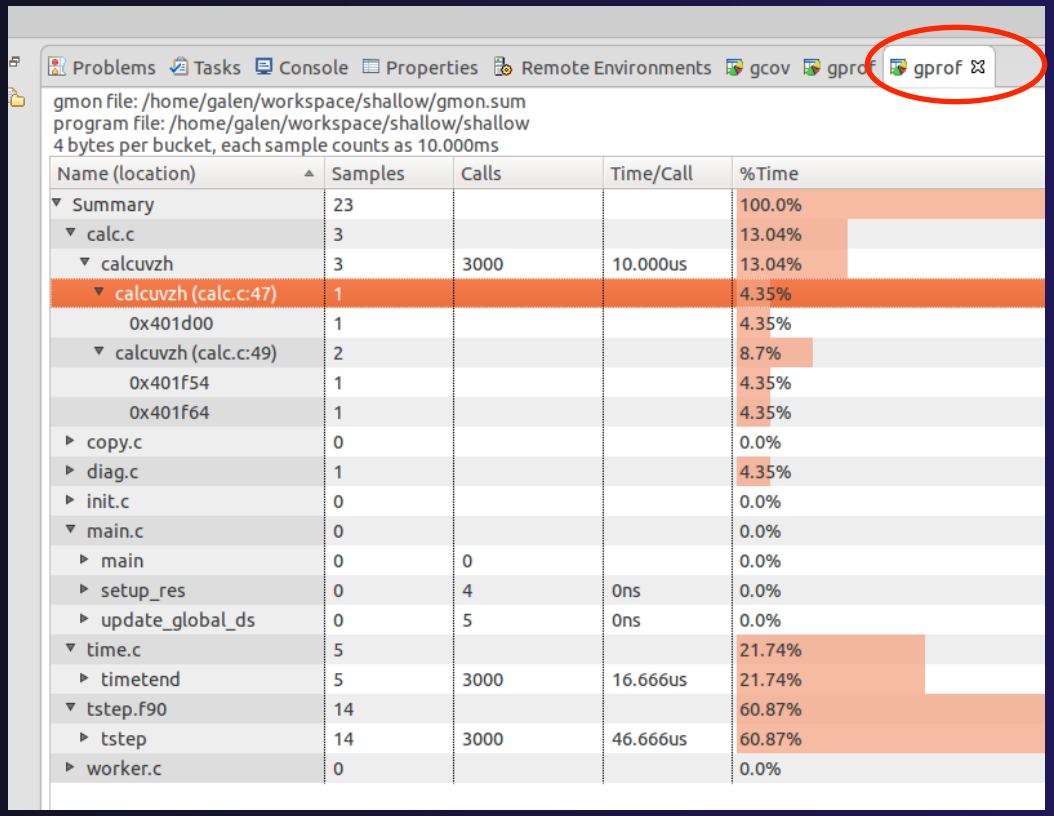
Aggregate the gmon output – invoke from a terminal:

```
gprof -s shallow gmon_shallow.*
```

This creates a gmon.sum file

Force a sync to see the file in Project Explorer

Double-click to open the gmon.sum File with the gprof viewer



Name (location)	Samples	Calls	Time/Call	%Time
Summary	23			100.0%
calc.c	3			13.04%
calcuvzh	3	3000	10.000us	13.04%
calcuvzh (calc.c:47)	1			4.35%
0x401d00	1			4.35%
calcuvzh (calc.c:49)	2			8.7%
0x401f54	1			4.35%
0x401f64	1			4.35%
copy.c	0			0.0%
diag.c	1			4.35%
init.c	0			0.0%
main.c	0			0.0%
main	0	0		0.0%
setup_res	0	4	0ns	0.0%
update_global_ds	0	5	0ns	0.0%
time.c	5			21.74%
timetend	5	3000	16.666us	21.74%
tstep.f90	14			60.87%
tstep	14	3000	46.666us	60.87%
worker.c	0			0.0%

Gprof viewer does not currently work well on Windows

Windows: gprof with shallow project, summary, use a terminal window to create text file

a.Invoke cmd line gprof
b.Sync to see file
c.Double-click to view txt file

The linuxtools team knows about the issue with Juno/Kepler and they're working on it.

Remote System Details | Tasks | Terminals

trestles.sdsc.edu

```
calc.o      diag.o       gmon_shallow.27090  main.gcda
copy.c      dump.c       gmon_shallow.27091  main.gcno
copy.gcda   dump.gcda    gmon.sum            main.o
copy.gcno   dump.gcno    gmon.sum.txt        Makefile
copy.o      dump.o       init.c              ptp_job.e
decs.h      eclipse.inc  init.gcda           ptp_job.e
-bash-3.2$ gprof -A shallow gmon.sum > gmon.sum.txt
-bash-3.2$
```

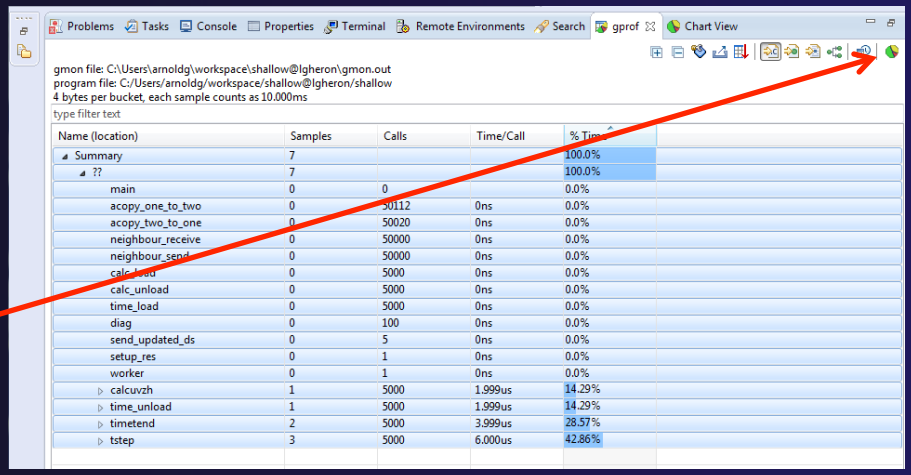
gmon.sum.txt

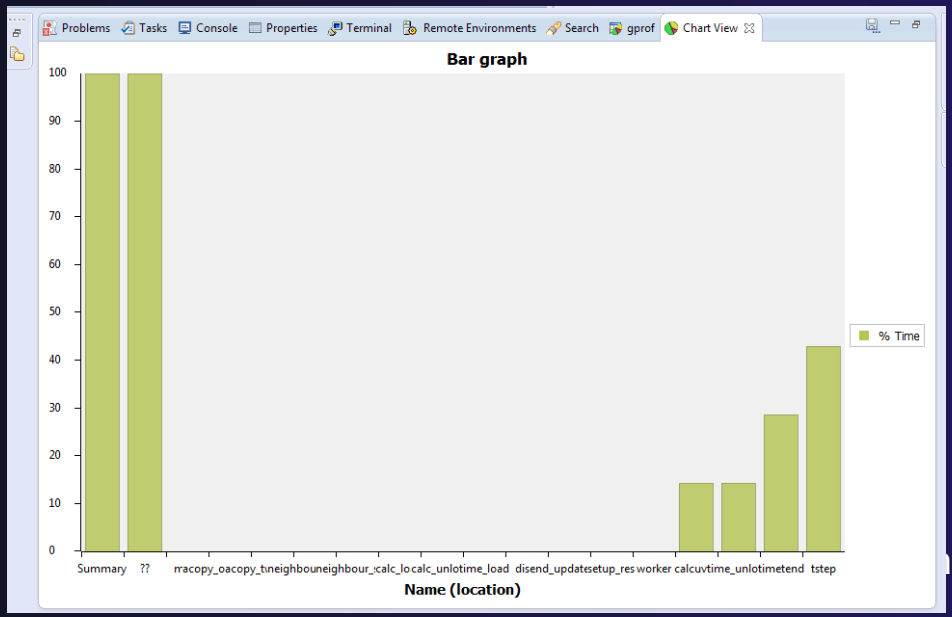
```
402                !                                          *
403                ! Programmers   = David Abramson     (DIT) rcoda@koe
404                !        = Paul Whiting       (DIT) rcopw@koel.co.rmi
405                !        = Martin Dix         (DAR) mrd@koel.co.rmit.
406                ! Language  = BSD c using Argonne NL macros
407                ! O/S       = Unix System V                        *
408                ! H/W       = Encore Multimax 320
409                !                                          *
410                !**********************************************
411
412          4000 -> subroutine tstep(m,n,alpha,jstart,jend,cpold,cuold,
413                    use, intrinsic :: ISO_C_BINDING
414                    implicit none
415
416                    integer(kind=C_INT), value :: m, n
417                    real(kind=C_FLOAT), value :: alpha
418                    integer(kind=C_INT), value :: jstart,jend
419                    type(C_PTR), value :: cpold; real(kind=C_FLOAT),
```

Gprof viewer still useful on Windows

Windows: Expand the gprof tab view of gmon.out (copy of one of the ranks output).

You can use the chart tool to make charts of the data in the gprof table you select.

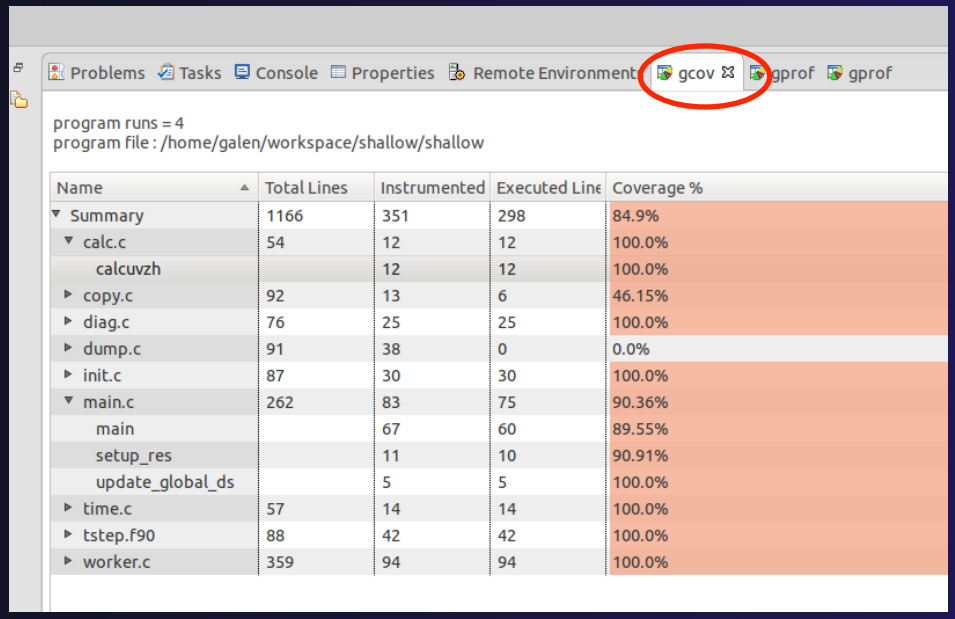




Gcov with shallow project

The gcov view is simlar to the gprof view but keep in mind that you're looking at code coverage and not necessarily performance or timing information (though there is a relationship...code not executed is performing quite well !). Also note that multiple executions will accumulate values in the gcov output files until they are removed or truncated to zero-length (2nd run to demonstrate this).

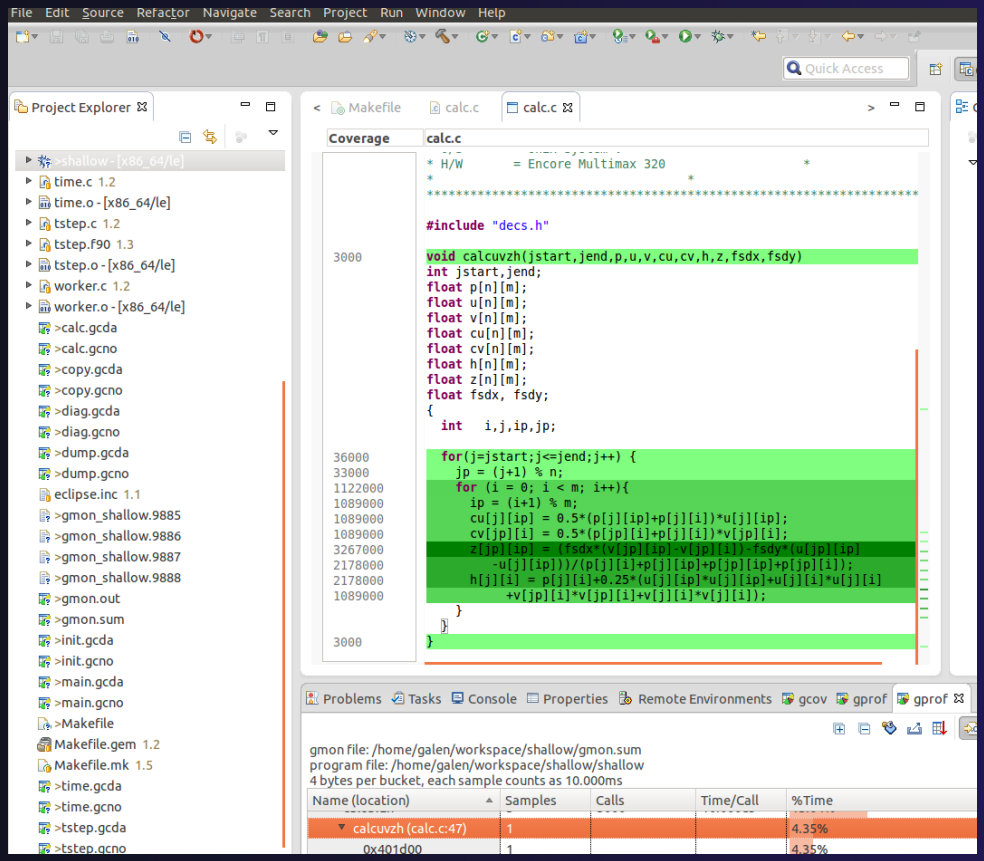
Double-click on any of the *.gcda or *.gcno to open this gcov viewer



Name	Total Lines	Instrumented	Executed Line	Coverage %
Summary	1166	351	298	84.9%
calc.c	54	12	12	100.0%
calcuvzh		12	12	100.0%
copy.c	92	13	6	46.15%
diag.c	76	25	25	100.0%
dump.c	91	38	0	0.0%
init.c	87	30	30	100.0%
main.c	262	83	75	90.36%
main		67	60	89.55%
setup_res		11	10	90.91%
update_global_ds		5	5	100.0%
time.c	57	14	14	100.0%
tstep.f90	88	42	42	100.0%
worker.c	359	94	94	100.0%

Gcov with shallow project, integration with CDT editor

Selecting (double click) a source code line from either the gcov or gprof view and you'll see the file and routine highlighted in the editor. Also notice the support for the .f90 file and its routines.



Exercise

Follow directions in previous slides to

1. Add the compiler flags to Makefile
2. Modify run configuration as described (add gmon prefix), and Run
3. View gmon and gcov files with gprof and gcov viewers

Optional Exercise

1) Run the shallow application with gcov compiler flags enabled.
a) Re-sync with Sync Active Now under Synchronization
b) View the tstep.gcno file and note the count, then repeat 1)a-b , have the counts changed?

2) Compare the tstep.f90 loops at lines 61, 70, 80 in the gprof and gcov displays .
a) Change the Makefile to use –O3 with FFLAGS and clean/rebuild/re-run
b) gprof –s shallow gmon_shallow.273* [your most recent gmon_ files from the run you just finished]
c) Re-sync the project
d) Does the gprof view of gmon.sum still exactly match up with the gcov display? If not, what happened to the missing loop(s)?

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.aptana.com/products/radrails
- Perl
  - http://www.epic-ide.org
- VI bindings
  - Vrapper (open source) - http://vrapper.sourceforge.net
  - viPlugin (commercial) - http://www.viplugin.com

Online Information

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

Mailing Lists

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

Getting Involved

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

PTP Tutorial Wrap-Up

- Please fill out the feedback form
- Your feedback is valuable!

Thanks for attending
We hope you found it useful