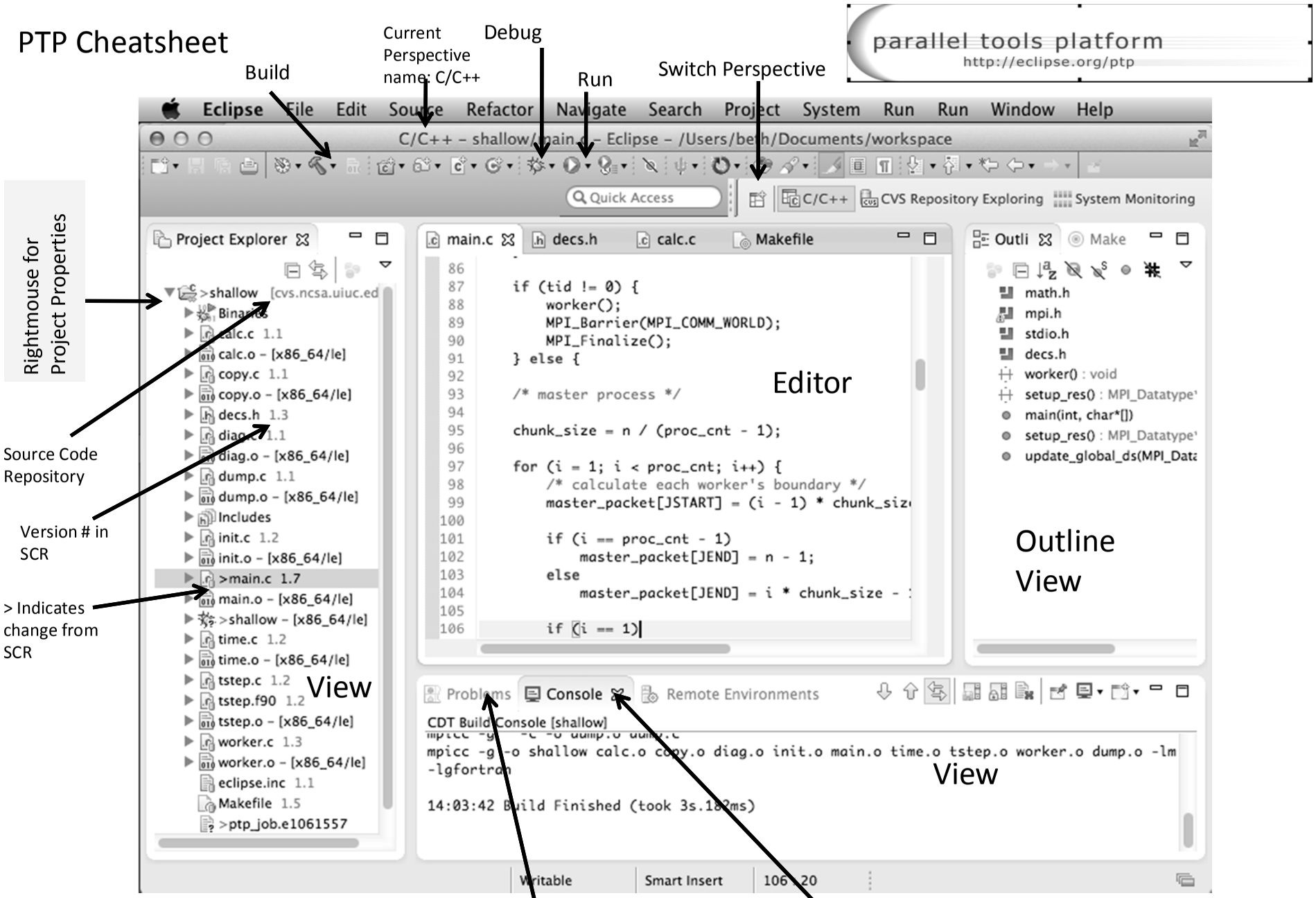
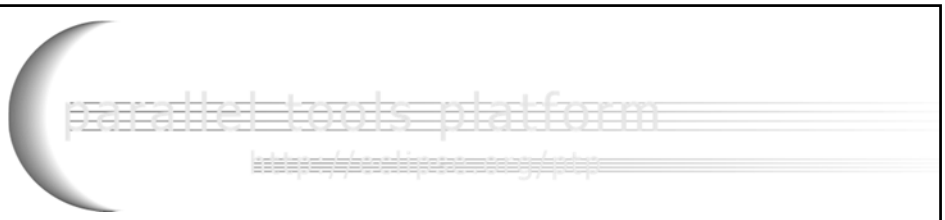
PTP Cheatsheet

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



Preferences: Menu: Window>Preferences
Mac: Eclipse>Preferences

11/09/12 tibbitts@us.ibm.com



A Unified Environment for the Development and Performance Tuning of Parallel Scientific Applications

Jay Alameda, NCSA
jalameda@ncsa.uiuc.edu

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

Galen Arnold, NCSA
arnoldg@ncsa.uiuc.edu

April 4-5, 2013

UCAR Software Engineering Assembly
University Corporation for Atmospheric Research

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

UCAR Software Engineering Assembly
University Corporation for Atmospheric Research

Tutorial Outline

Time (Tentative!)	Module	Topics	Presenter
10:30-11:00	1. Eclipse and PTP Installation	✦ Installation of Eclipse and PTP (can start early as people arrive)	Jay
11:00-11:30	2. Introduction & Overview	✦ Eclipse architecture & organization overview	Wyatt
11:30-12:00	3. Developing with Eclipse	✦ Eclipse basics; Creating a new project from CVS; Local, remote, and synchronized projects	Jay
12:00-1:00	Lunch		
1:00-2:30	3. Developing with Eclipse (continued)	Continue from before the break… ✦ Editing C files; MPI Features; Building w/Makefile ✦ Resource Managers and launching a parallel app ✦ Fortran, Refactoring, other Advanced Features	Wyatt
2:30-3:00	4. Debugging Part A	✦ Debugging an MPI program	Jay
3:00-3:30	Break		
3:30-4:00	4. Debugging Part B	✦ Debugging an MPI program	Jay
4:00-5:00	5a. Performance Tuning & Analysis Tools	✦ 1. TAU (Tuning and Analysis Utilities) ETFw (External Tools Framework)	Wyatt

Tutorial Outline

Time (Tentative!)	Module	Topics	Presenter
8:30-9:30	5b. Performance Tuning & Analysis Tools	✦ 2. Other tool integration with PTP	Wyatt/Jay
9:30-10:00	6. Developing WRF with Eclipse	✦ Overview of how to work with WRF from Eclipse	Jay
10:00-10:30	Break		
10:30-11:30	6. Developing WRF with Eclipse	Overview of how to work with WRF from Eclipse	Jay
11:30-12:00	7. Other Tools, Wrapup	✦ Other Tools, website, mailing lists, future features	Jay
12:00-1:00	Lunch		
1:00-3:00	Bring your own code/Advanced TAU		Wyatt/Jay
3:00-3:30	Break		
3:00-5:00	Bring your own code/Advanced TAU		Wyatt/Jay

PTP Tutorial – Slide Topics	**PDF page #**	**Slide page # prefix**
PTP Installation	5	Install-
PTP Introduction	21	Intro-
Eclipse Basics	30	Basic-
Adding a remote shell in Eclipse	43	Shell-
Creating a Synchronized project from existing remote dir (Project creation alt. #1)	51	Sync-
Creating a project from CVS – "Team" features – then convert to Synchronized project (Project creation alt. #2) – for SC tutorial we will do #2	73	CVS-
Eclipse Editor Features	93	Editor-
MPI Programming Features (similar features for OpenMP, UPC, OpenSHMEM, OpenACC)	110	MPI-
Using SSH tunnelling	132	Tunnel-
Building a Project on a remote target	135	Build-
Running an Application on remote target	153	Run-
Fortran	174	Fortran-
Search and Refactoring: Advanced Features	191	Advanced-
NCSA/XSEDE Features: GSI Authentication, MyProxy Login, etc	208	NCSA-
Parallel Debugging	215	Debug-
Performance Tools introduction	250	Perf-
Performance Tools – TAU (Tuning and Analysis Utilities)	253	TAU-
Performance Tools – GEM (Graphical Explorer of MPI Programs)	272	GEM-
Performance Tools – Linuxtools gcov/gprof in Eclipse	308	Linux-
Wrap-up	328	WrapUP-

Final Slides, Installation Instructions

- Please go to http://wiki.eclipse.org/PTP/tutorials/SEA2013 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, has not been tested by us but should work.
  - See http://wiki.eclipse.org/PTP/installjava

Eclipse Packages

- The current version of Eclipse (4.2) is also known as “Juno”
- 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

We often call this the “Parallel Package”

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

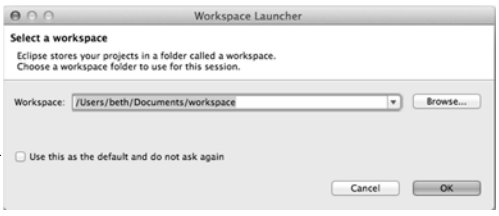
Starting Eclipse

- **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

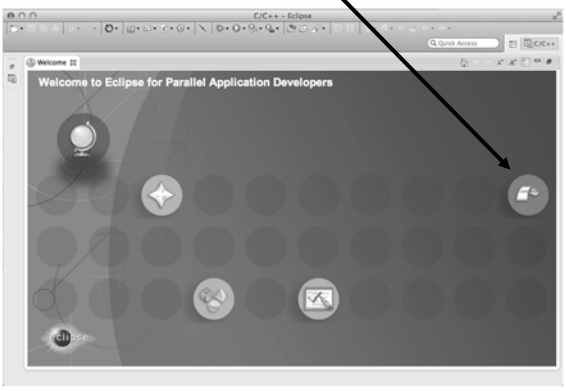
The prompt can be turned off



Eclipse Welcome Page

- Displayed when Eclipse is run for the first time

Select “Go to the workbench”

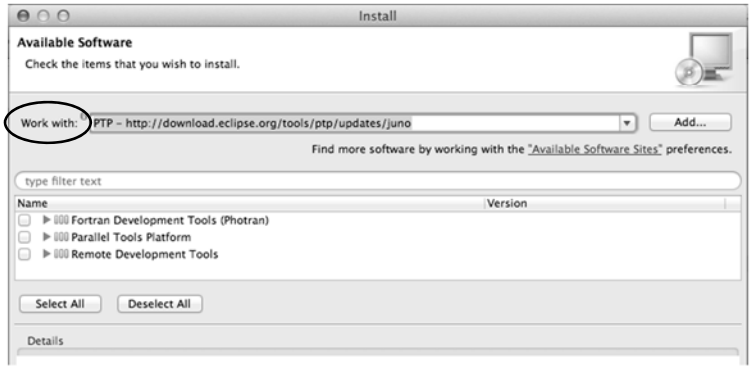


Checking for PTP Updates

- From time-to-time there may be newer PTP releases than the Juno release
  - Juno 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. Juno), and what is within the parallel package
- You must enable (and install from) the PTP-specific update site before the updates will be found
- **SEA Note**: we will show you specific instructions on applying a needed update on slide 9

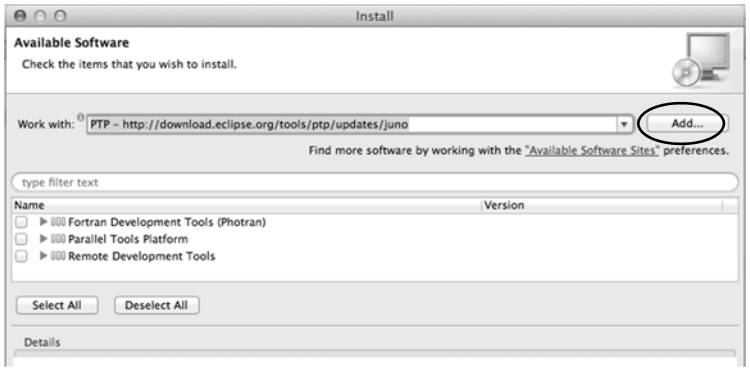
Updating PTP

- 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/juno



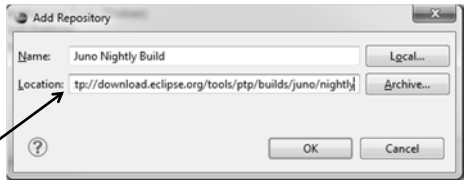
Updating PTP: SEA

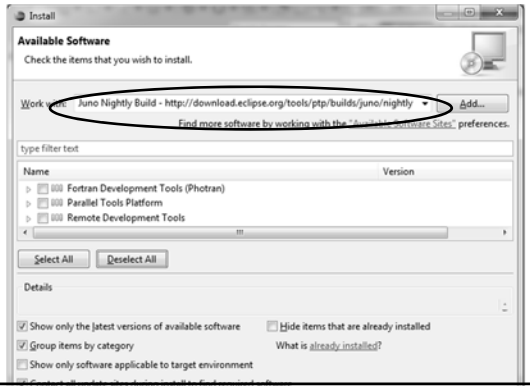
- Now select **Help>Install New Software…**
  - Next to **Work With:** dropdown box, select **Add…**



Updating PTP: SEA (2)

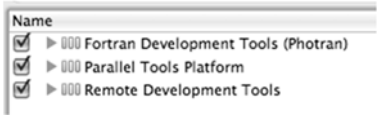
- Label the nightly build site, add location
  - http://download.eclipse.org/tools/ptp/builds/juno/nightly





Updating PTP (2)

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



Note: for our tutorial, this installs extra features you'll need later anyway (GEM, TAU)

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

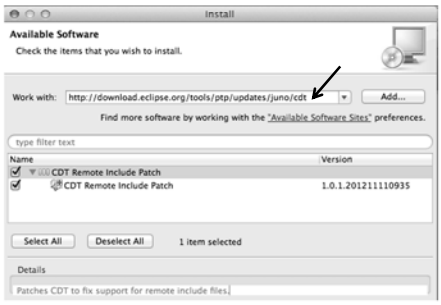
Updating PTP (3)

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

- Don't restart Eclipse yet!
  - … see next slide (for one more update) before accepting to restart Eclipse
  - … or you can restart eclipse and do it then

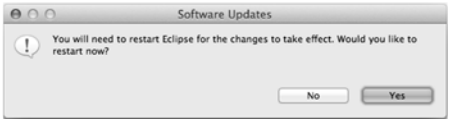
Updating PTP –CDT fix

- Currently CDT (C/C++ Development Tools) needs a fix for remote includes
- We'll install from another update site to fix this:
- In the **Work With:** dropdown box, add:
  http://download.eclipse.org/tools/ptp/updates/juno/cdt
- Select All
- Next, Next, accept license, Finish – to complete the install



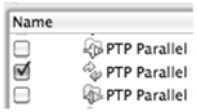
Updating PTP - restart

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

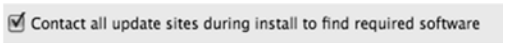


Updating Individual Features

- 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

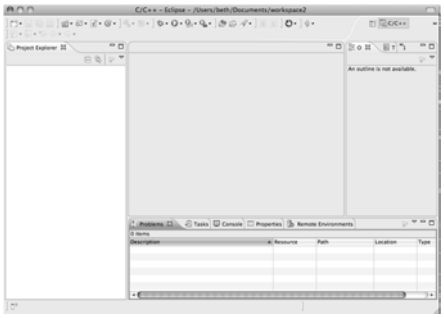


- Note: if network is slow, consider unchecking:



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

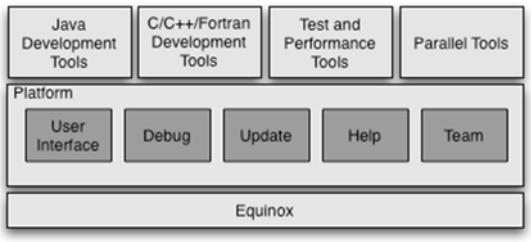
1. Launch Eclipse and select the default workspace
2. Configure Eclipse to check for PTP updates
3. Update all PTP features to the latest level
4. Install the optional features of PTP, including TAU and GEM
   - Selecting *all* features accomplishes 3. and 4.
5. Install the CDT fix
6. Restart Eclipse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

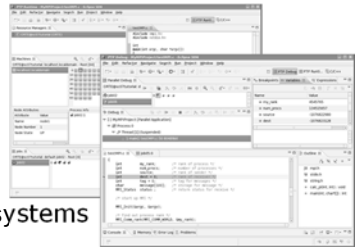


Eclipse Features

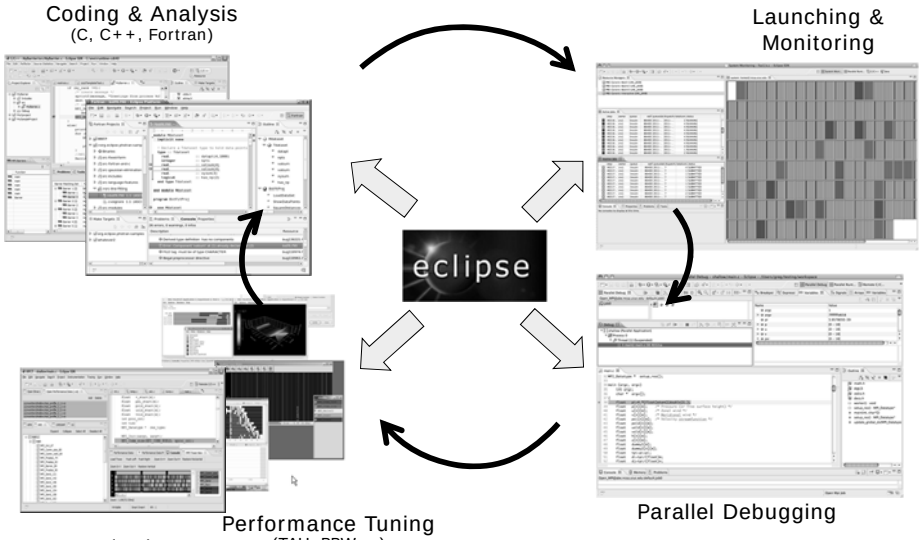
- Full development lifecycle support
- Revision control integration (CVS, SVN, Git)
- Project dependency management
- Incremental building
- Content assistance
- Context sensitive help
- Language sensitive searching
- Multi-language support
- Debugging

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

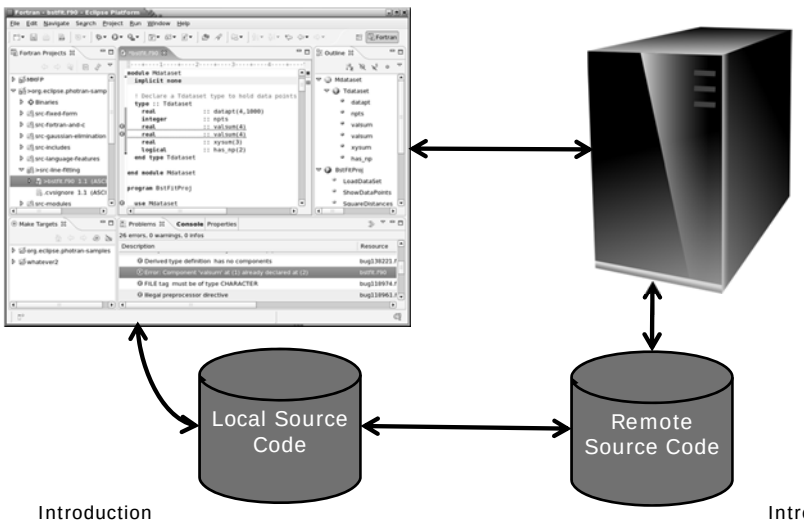


Eclipse PTP Family of Tools



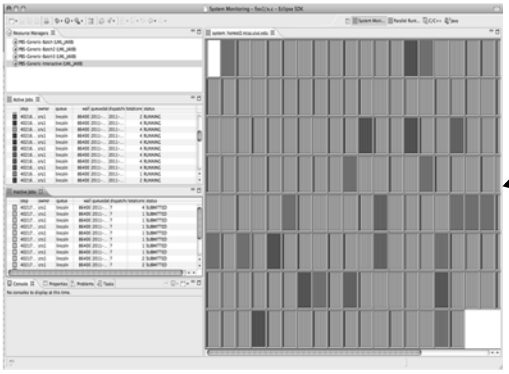
How Eclipse is Used

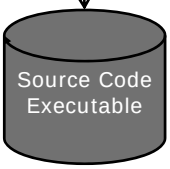
Editing/Compiling



How Eclipse is Used

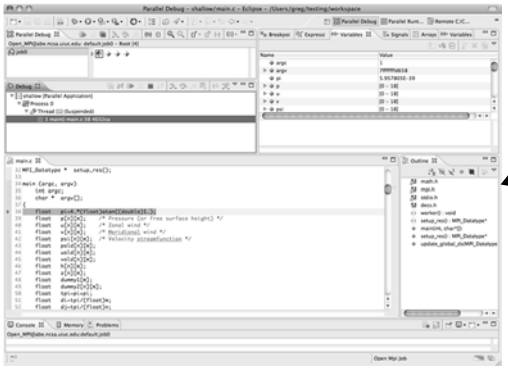
Launching/Monitoring

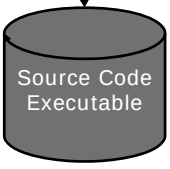




How Eclipse is Used

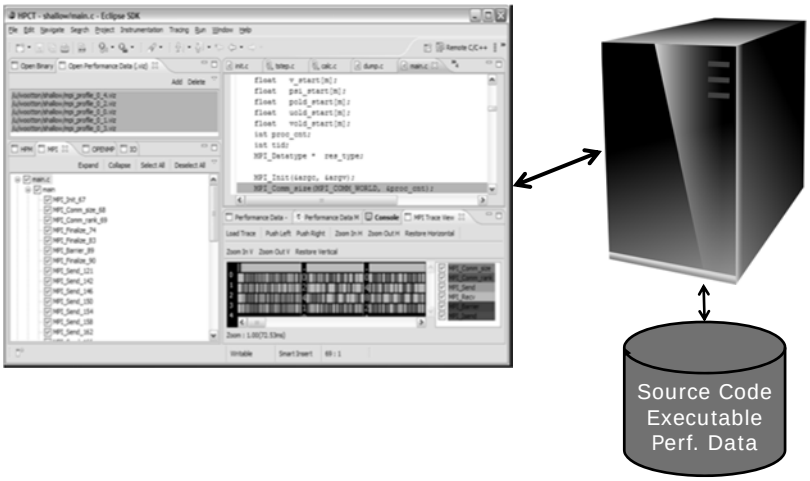
Debugging





How Eclipse is Used

Performance Tuning

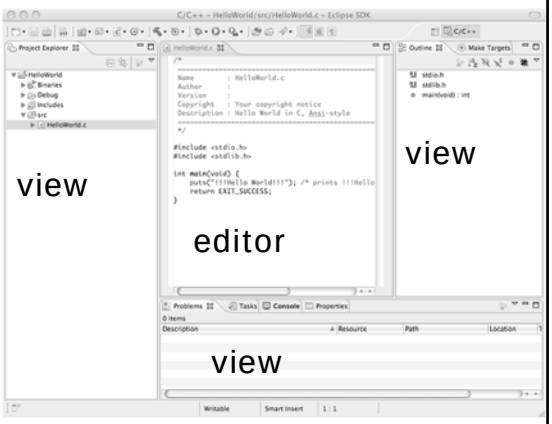


Eclipse Basics

- 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

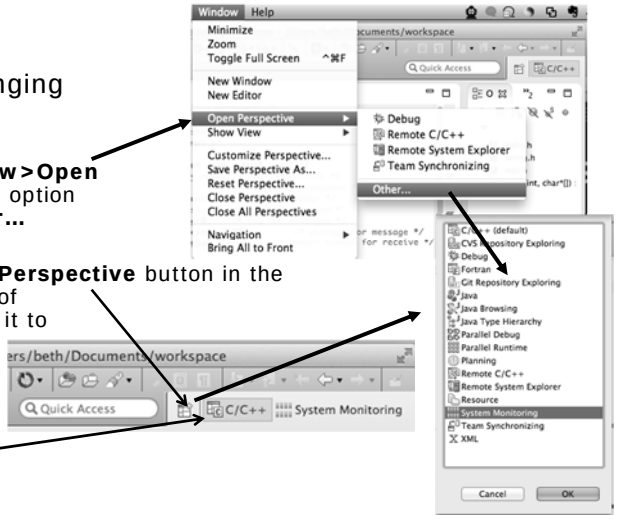


Perspectives

- 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

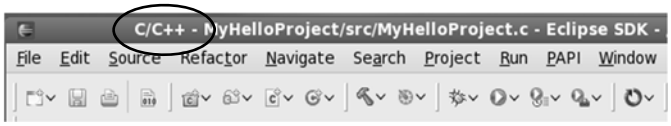
Switching Perspectives

- 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

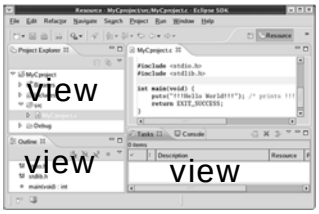


Which Perspective?

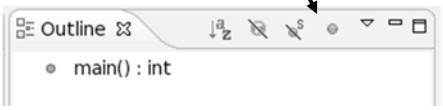
- The current perspective is displayed in the title bar



Views

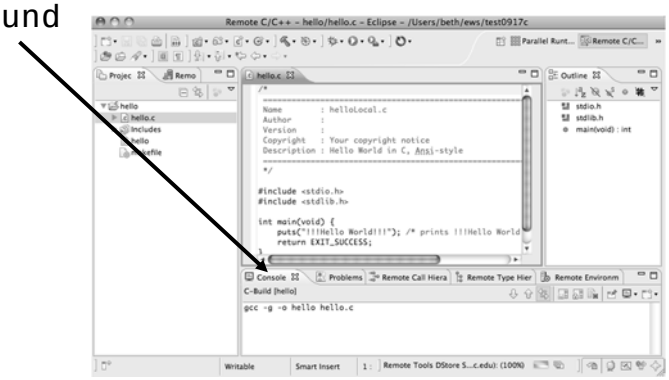


- The workbench window is divided up into Views
- The main purpose of a view is:
  - To provide alternative ways of presenting information
  - For navigation
  - For editing and modifying information
- Views can have their own menus and toolbars
  - Items available in menus and toolbars are available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



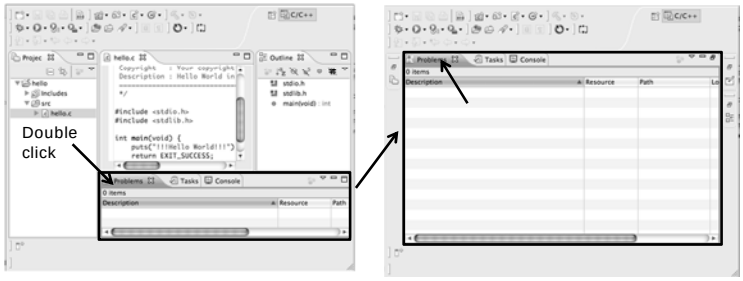
Stacked Views

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



Expand a View

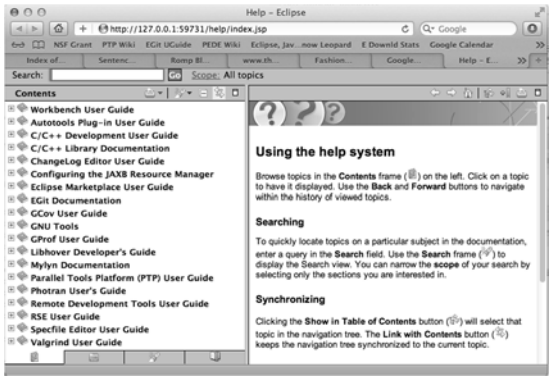
- Double-click on a view/editor's tab to fill the workbench with its content;
- Repeat to return original size



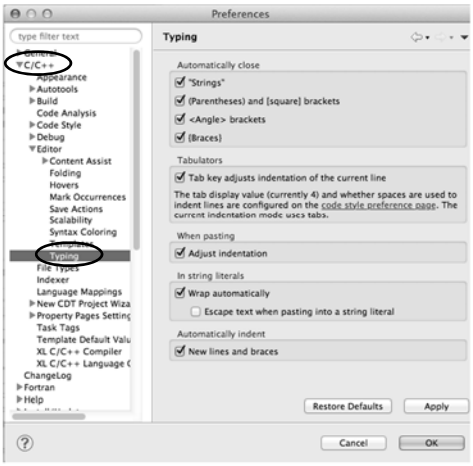
- Window > Reset Perspective returns everything to original positions

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.)

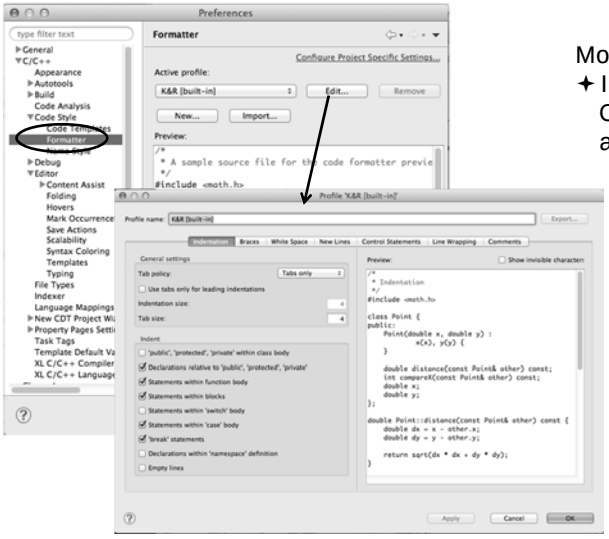


Eclipse Preferences



- 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

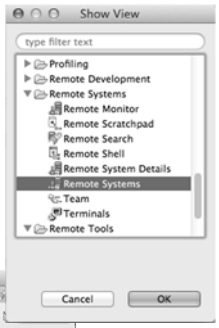
1. Use source code formatting to format a source file, or a region of a source file
   - Use Source>Format menu
2. In Eclipse Preferences, change the C/C++ source code style formatter, e.g.
   - Change the indentation from 4 to 6
   - Make line wrapping not take effect until a line has a maximum line 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

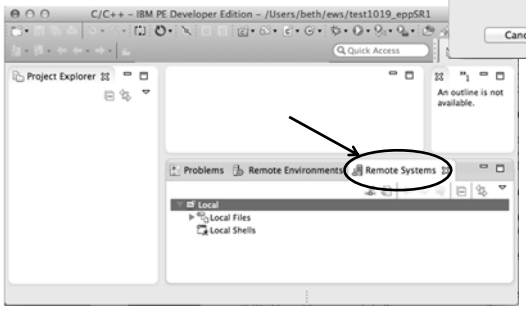
Adding a Remote Shell

- Objective
  - Learn how to add an additional Eclipse View with a shell to a remote system
  - Learn how to do command line interaction with the remote system right from Eclipse
- Contents
  - Set up Remote Systems Explorer (RSE) connection
  - Add Remote Shell view
  - Connect to remote system
  - Allow opportunity to inspect remote system, copy file, etc as needed

Add the Remote Systems View

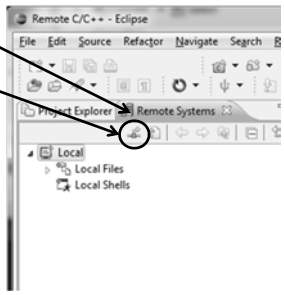
- Open the **Remote Systems** view
  - Open it via Window>Show View>Other… Under **Remote Systems**, select **Remote Systems**
  - Or … it can be found in the Remote C/C++ perspective
- Probably gets added at the bottom





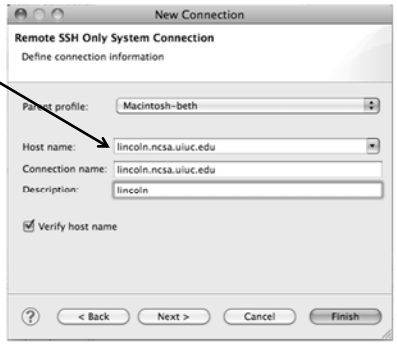
Define a new RSE connection

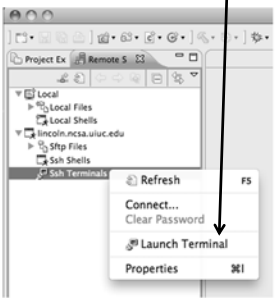
- Select the Remote Systems view
  - Define a new connection
    (buttons may be on the far right side of the toolbar)
  - In the **New Connection** dialog, Select “SSH Only”
  - Then **Next**



Define a new RSE connection (2)

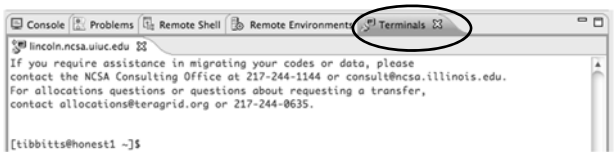
- Add system’s host info
  - Then **Finish**
- Right click on **Ssh terminals**, under your new connection
- Select **Launch Terminal**





Define a new RSE connection (3)

- Add your userid and password
- Click through any RSA messages
- And now you have a terminal to the remote system

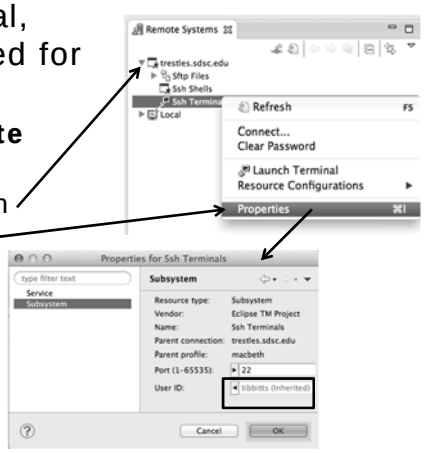


Notes

- **Why did we do this?**
  To show you can gain “traditional” access to a remote host through Eclipse
- **Why did I have to specify the connection information again?**
  Note: RSE “Connection” information is not shared with PTP Synchronized projects.
  PTP will allow using existing connections with terminal consoles in a future release.

RSE Connection Properties

- To alter the properties e.g. of the remote Terminal, such as the userid used for the login…
  - Open/select the **Remote Systems** view
  - Expand your connection
  - Right mouse on **Ssh Terminals** and select **Properties**
  - In **Properties** dialog, select **Subsystem**



Exercise

1. Add the **Remote Systems** view to your workbench
2. Connect to the remote machine and open an ssh terminal
3. Inspect home directory

Creating a Synchronized Project

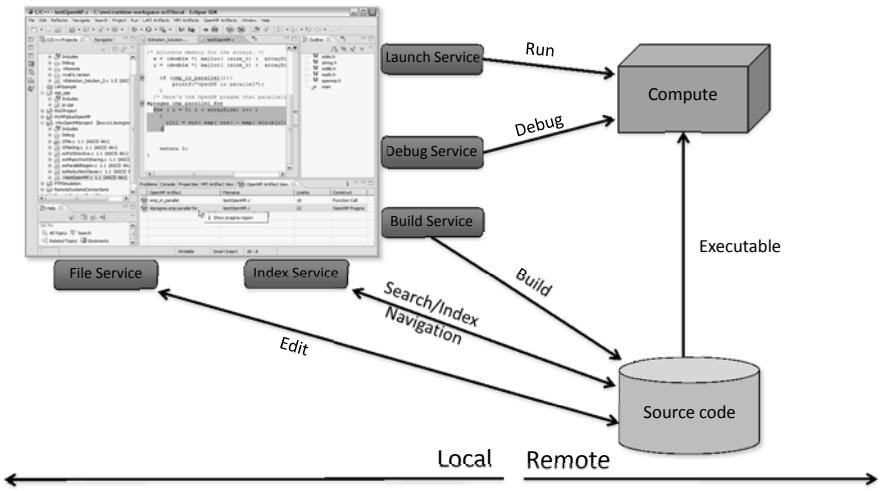
- Objective
  - Learn how to create and use synchronized projects
  - Learn how to create a sync project directly from source that already exists on a remote machine
- Contents
  - Eclipse project types
  - Creating a synchronized project
  - Using synchronize filters
  - Converting an existing project to synchronized
- Project Creation Alternative #1

  In this scenario, we will use code existing on a remote host, and create a synchronized project which copies it to the local machine
  - (Project Creation Alternative #1 is to check out code from a CVS source code repository and then convert to a Sync project)

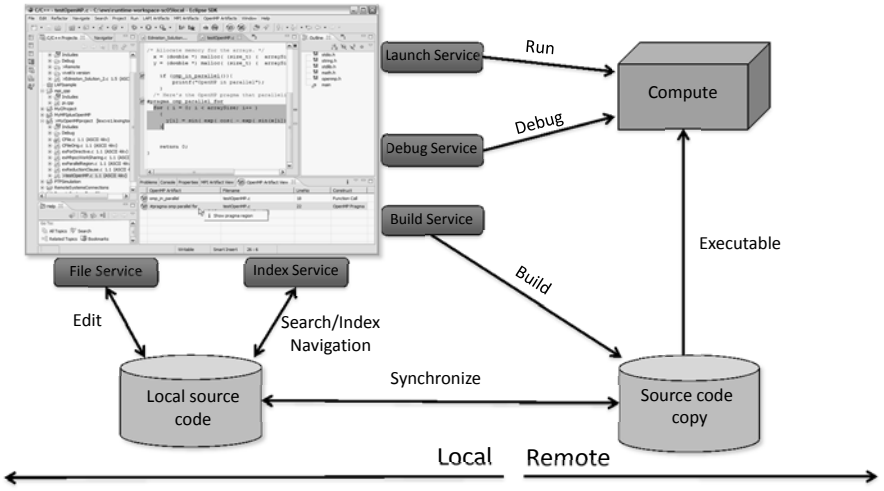
Project Types

- Local
  - Source is located on local machine, builds happen locally
- Remote
  - Source is located on remote machine(s), build and launch takes place on remote machine(s)
- Synchronized
  - Source is local, then synchronized with remote machine(s) (or vice-versa)
  - Building and launching happens remotely (can also happen locally)

Remote Projects



Synchronized Projects



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

Create Project

- This module creates a Synchronized project from source code already existing on the remote system

  - or –

- The CVS module creates a Synchronized project from a CVS source code repository

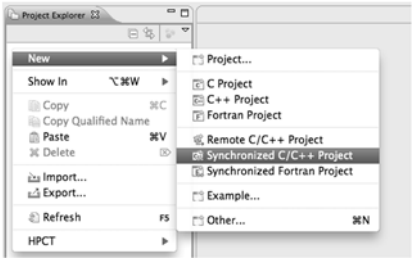
Source Code for project

- Source code exists on remote target

```
$ pwd
/gpfs/ibmu/tibbitts/shallov
$ 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
$
```


Create Synchronized Project

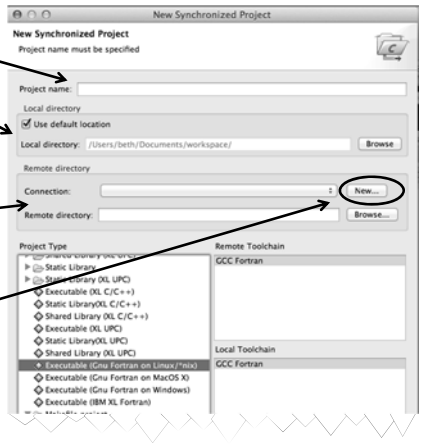
- In the Project Explorer, right click then choose
  - **New>Synchronized C/C++ Project** if your project is C/C++ only



  - **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

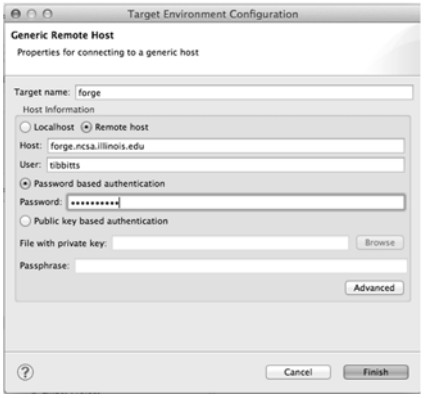
- Enter the **Project Name**
  - E.g. “shallow”
- The **Local Directory** specifies where the local files are located
  - Leave as default
- The **Remote Directory** specifies where the remote files are located
  - Select a connection to the remote machine, or click on **New…** to create a new one
    (See next slide)
  - Browse for the directory on the remote machine



Creating a Connection

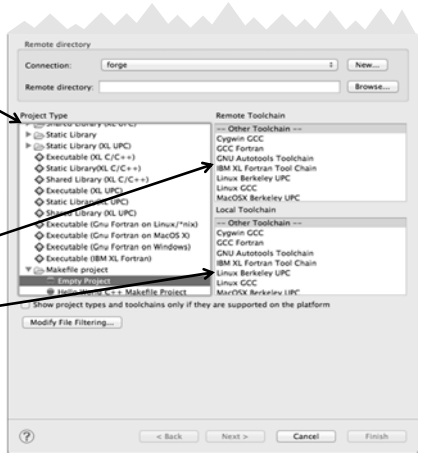
- In the **Target Environment Configuration** dialog
  - Enter a **Target name** for the remote host
  - Enter host name, user name, and user password or other credentials
  - Select **Finish**

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



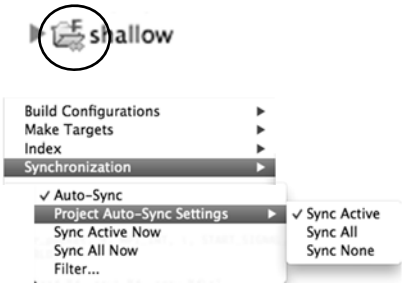
Project Type & Toolchain

- Choose the **Project Type**
  - If you are synchronizing with an existing project, 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 used for advanced editing/searching
- Use **Modify File Filtering…** if required (see later slide)
- Click **Finish** to create the project



Synchronized Project

- Synchronized projects are indicated with a “synchronized” icon
- Right click on project to access **Synchronization** menu
  - Select **Auto-Sync** to enable/disable automatic syncing
  - **Project Auto-Sync Settings** are used to determine which configurations are synchronized (**Active** only, **All** or **None**)
  - **Sync Active/All Now** to manually synchronize
  - **Filter…** to manage synchronization filters

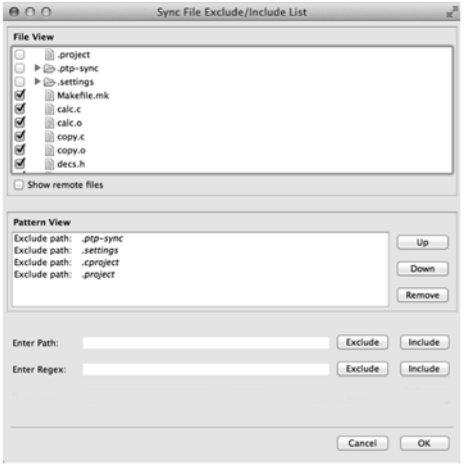


Synchronize Filters

- 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…**

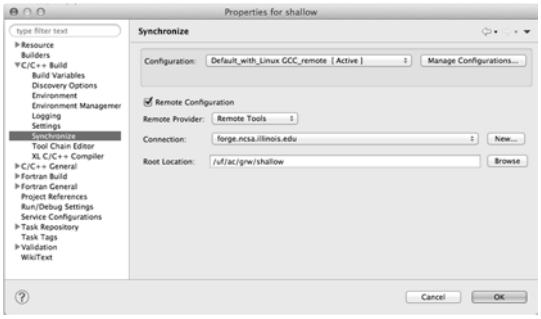
Synchronize Filter Dialog

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



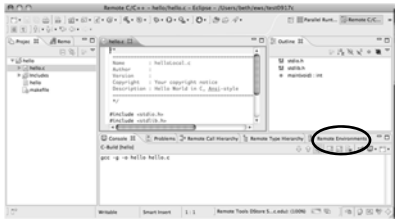
Synchronized Project Properties

- Synchronized project properties can be configured manually
- Open the project properties, then select **C/C++ Build>Synchronize**
- Each configuration is associated with a remote connection and a root directory
- Can be changed manually, but only if you know what you are doing!



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 C/C++ or Fortran Project to a Synchronized Project**
- Click **Next>**

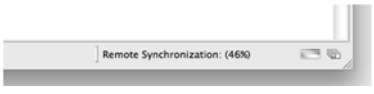
Convert Projects Wizard

- Select checkbox next to "shallow"
  - Only C/C++/Fortran projects will be in the list of candidates for conversion to Sync project
- For **Connection:**, click on **New…**
  - Unless you already have a connection
- The tutorial instructor will indicate what to enter for:
  - **Target name**
  - **Host** name of remote system
  - **User** ID
  - **Password**
- Click **Finish** to close it
- The connection name will appear in the **Convert Projects** wizard for **Connection**

Convert Projects Wizard (2)

Back in the **Convert Projects** 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

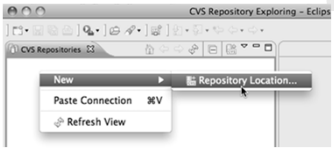
- Objective
  - Learn how to use a source code repository with Eclipse
  - Learn how to create a Synchronized project
- Contents
  - Checking out project in CVS
  - Setting up a Connection for a Synchronized Project
  - Handling changes; Comparing files (diffs)
- Project Creation Alternative #2

  In this scenario, we will check out code from a CVS source code repository, setting it up as a synchronized project on a remote host.
  - (Project Creation Alternative #1 in this PTP tutorial is to create a Synchronized project directly from code existing on a remote host)

Importing a Project from CVS

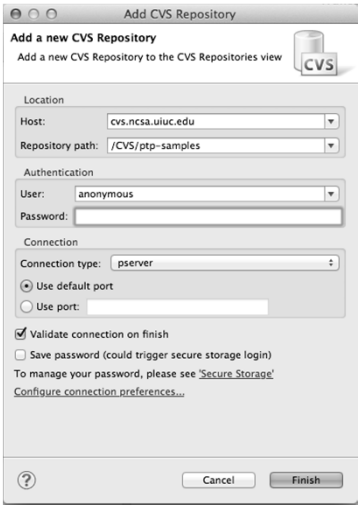
- Switch to **CVS Repository Exploring** perspective
  - Window > Open Perspective > Other…
  - Select **CVS Repository Exploring**
  - Select **OK**

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



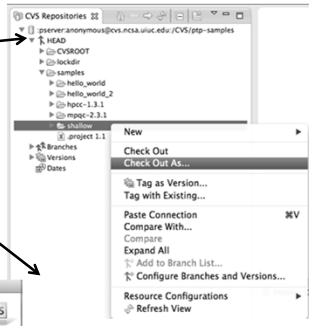
Add CVS Repository

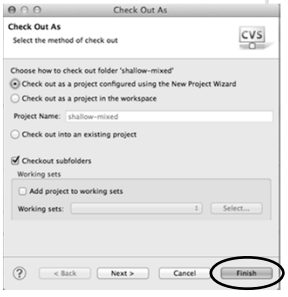
- Enter **Host:** cvs.ncsa.uiuc.edu
- **Repository path:** /CVS/ptp-samples
- For anonymous access:
  - **User**: anonymous
  - No password is required
  - **Connection type:** pserver (default)
- For authorized access:
  - **User**: your userid
  - **Password**: your password
  - **Connection type:**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



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**



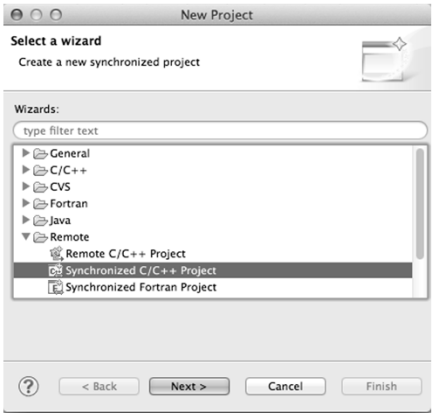


The default of "Check out as a project configured using the New Project Wizard" is what we want

New Project Wizard

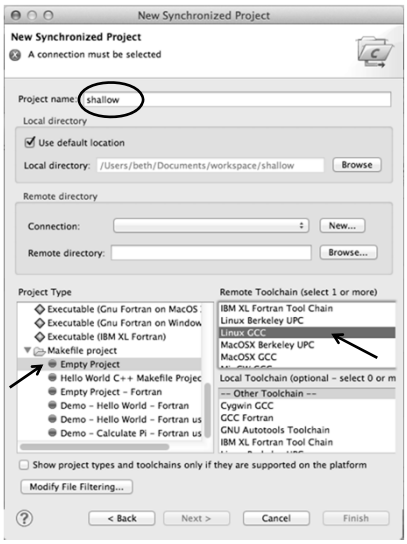
As project is checked out from CVS, the **New Project** Wizard helps you configure the Eclipse information to be added to the project

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



New Project Wizard (2)

- Enter 'shallow' as **project name**
- Under **Project type,** expand **Makefile project**
  - scroll to the bottom
- Select **Empty Project**
- For **Remote Toolchain,** Select **Linux GCC**
  - In general, choose a toolchain that matches the compiler you intend to use on the remote system
- For **Local Toolchain**
  - If you intend to build on the local machine, select this; it is optional
- Next slide … connection



Select Remote Connection: UCAR SEA Workshop

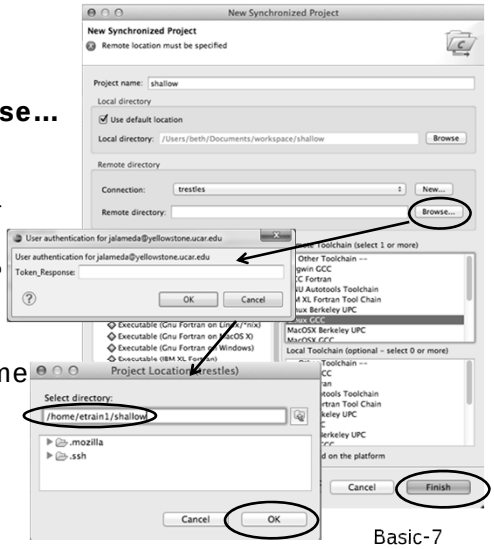
- For **Connection:**, click on **New…**
  - Unless you already have a connection
- **Target Environment Configuration**
  - Target name: **yellowstone**
  - Host name of remote system
    - **yellowstone.ucar.edu**
  - User ID
  - Password: ***do not enter***

  Note: if you need to use ssh tunneling, use **Localhost**; the **Advanced** button lets you enter the port #*
- Select **Finish**

* See http://wiki.eclipse.org/PTP/FAQ and search for 'tunnel'

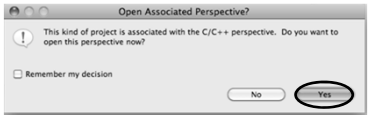
Select Remote Directory

- Back in the **New Synchronized Project** dialog, select the **Browse…** button under **Remote Directory**
  - You will be asked for your token response
- *This is the first time the connection is used*
- For **Project Location**, enter new directory name (or select existing dir)
- Hit **OK,** then **Finish**

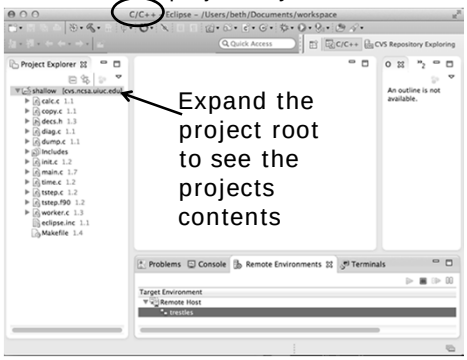


Project successfully checked out

- Switch to the C/C++ Perspective when prompted after checking out the code



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

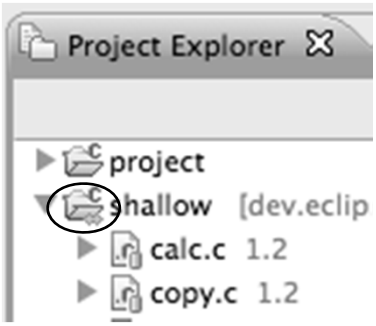


Synchronizing the Project

- Because we will be running on a remote system, we must also build on that system
- Source files must be available to build
- The synchronized project does this
- Files are synchronized automatically when they are saved
- A full synchronize is also performed prior to a build

Synchronized Project

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



- C Project w/o Sync

- Synchronized Project

Team Features

“Team” Features

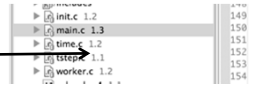
- Eclipse supports integration with multiple version control systems (VCS)
  - CVS, SVN, Git, and others
  - Collectively known as “Team” services
- Many features are common across VCS
  - Compare/merge
  - History
  - Check-in/check-out
- Some differences
  - Version numbers
  - Branching

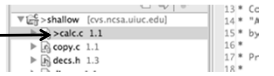
Two meanings for ‘Synchronize’

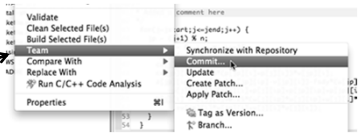
- PTP’s *synchronize*
  - *Copy files in synchronized projects between local and remote to mirror them*
- Team *synchronize*
  - *Show differences between local project and source code repository versions*

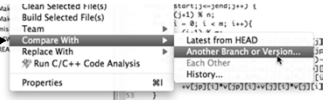
CVS Features

- 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)







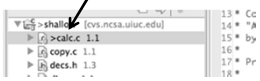


* Team synchronize

How to tell that you've changed something

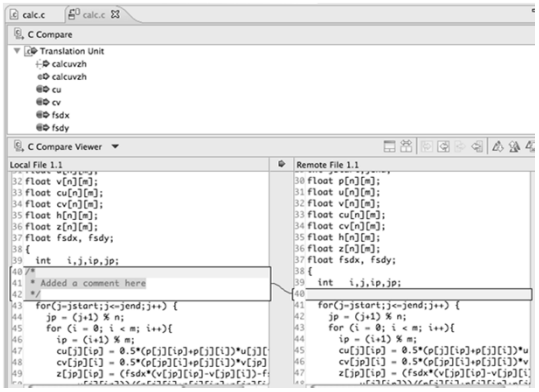
- Open "calc.c"
- Add comment at line 40
- Save file
- File will be marked ">" to indicate that it has been modified

```
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     }
```



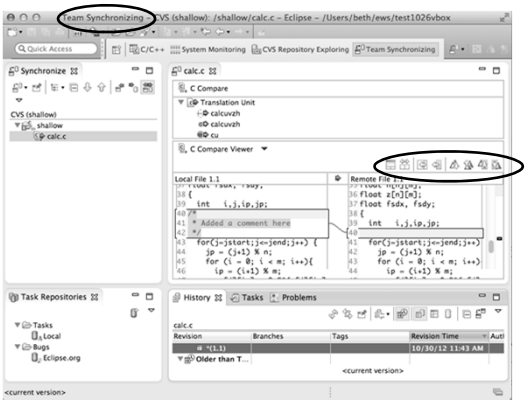
Comparing *single file* with what's in the repository

- 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

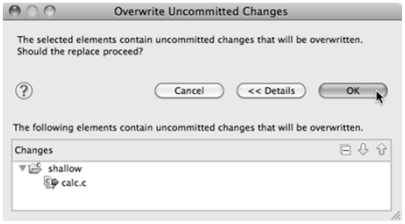
- Right-click on project name (or any subset) and select **Team>Synchronize with Repository**
- **Team Synchronizing** perspective will open
- List of changed files appears
- Double-click on a file to see the diff viewer
- Buttons allow changes to be merged from right to left
- Can also navigate between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

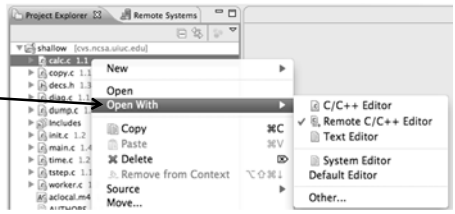
Hint: *Right-click the Makefile and select* ***Team > Show History****. Both of these can be done from the History view.*

Editor Features

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

Editors

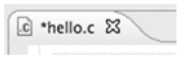
- An editor for a resource (e.g. a file) opens when you double-click on a resource
- The type of editor depends on the type of the resource
  - .c files are opened with the C/C++ editor by default
  - You can use **Open With** to use another editor
  - In this case the default editor is fine (double-click)



  - 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

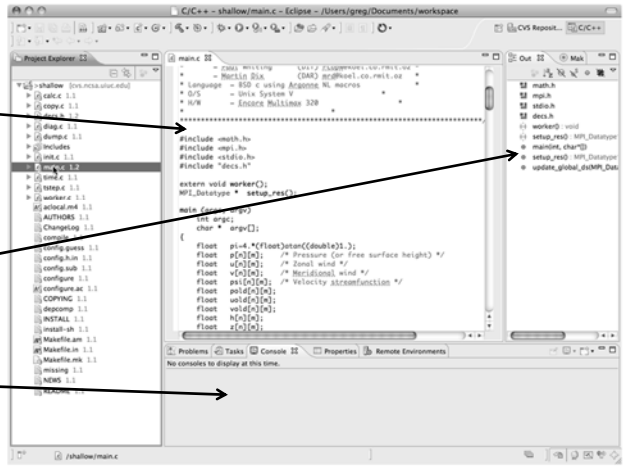
- When you change a file in the editor, an asterisk on the editor's title bar indicates unsaved changes



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

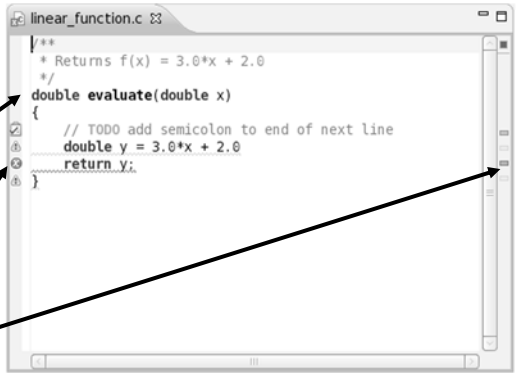
Editor and Outline View

- 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

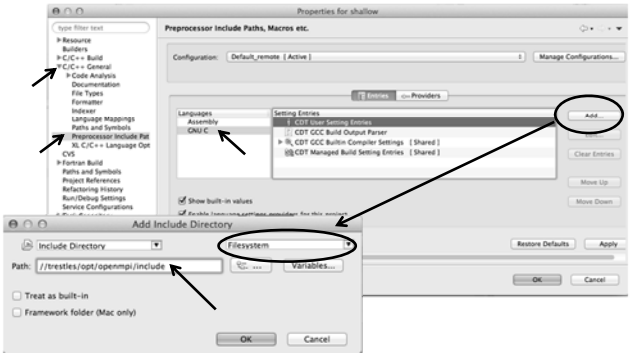


Icons:  Task tag, Warning, Error

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 **Filesystem** in pulldown
- A UNC-style path specifies //<connection>/<path>
- Enter Path //trestles/opt/openmpi/include
- Select **OK**

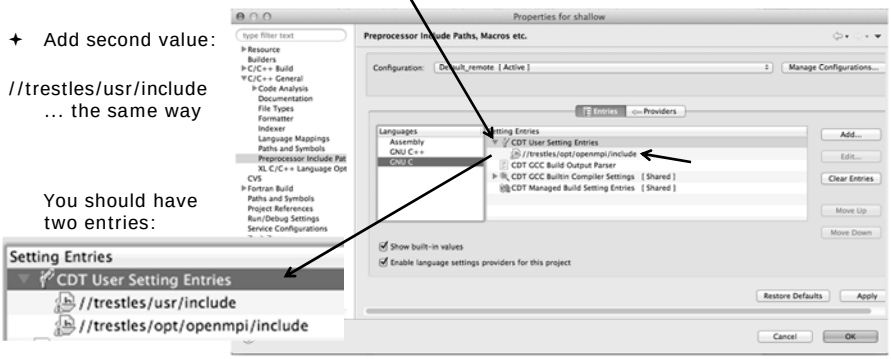


Include Paths (2)

- After adding include directory, it should appear in the list
  - Bug: on Mac, it appears as blank. Close and re-open the twisty to see the correct value.
  - Add second value:

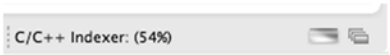
//trestles/usr/include
... the same way

You should have two entries:



Include Paths (3)

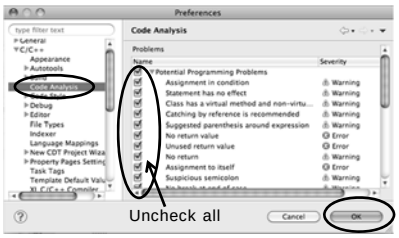
- Select **OK**
- The C/C++ Indexer should run
  - Lower right status area indicates it

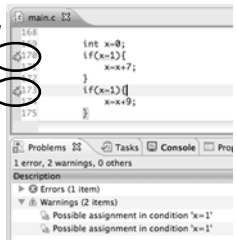


  - If not force it via Project Properties>Index>Rebuild

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

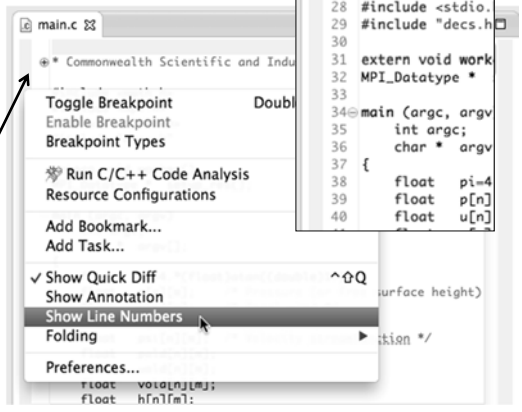




- If icons don't disappear: Right mouse on Project > **Run C/C++ Code Analysis**
- You can also enable/disable this per project in Project Properties

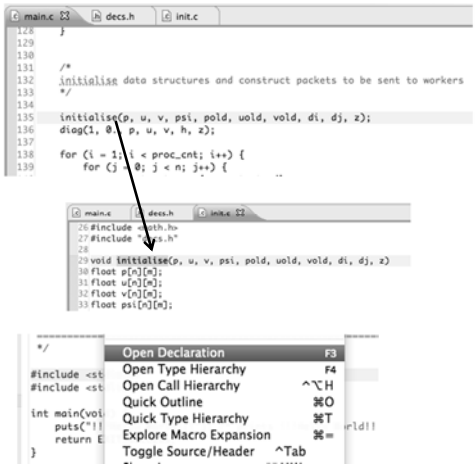
Line Numbers

- 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

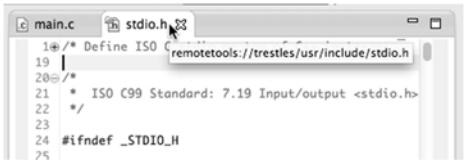
- On demand hyperlink
  - In main.c line 135:
  - Hold down Command/Ctrl key e.g. on call to `initialise`
  - Click on `initialise` to navigate to its definition in the header file (Exact key combination depends on your OS)
  - E.g. Command/Ctrl and click on initialise
- Open declaration
  - Right-click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

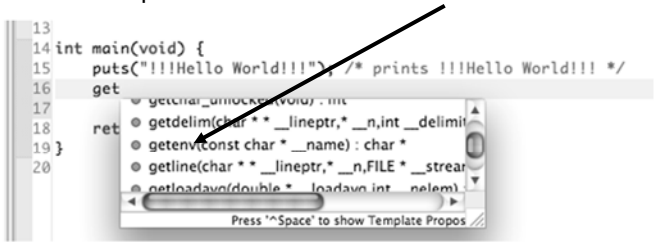
Navigating to Remote Files

- 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.



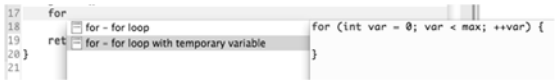
Content Assist & Templates

- 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

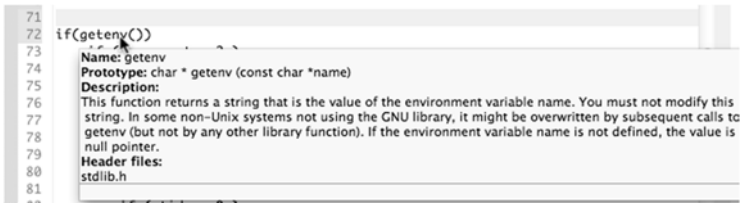
Hit ctrl-space again for code templates



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

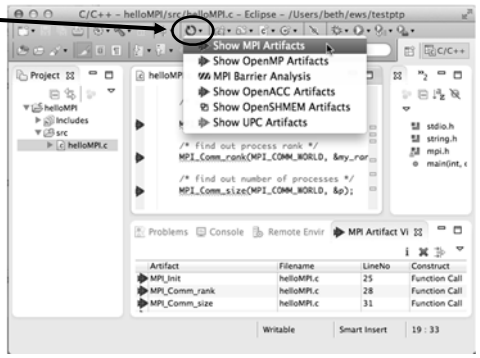
- Objective
  - Learn about MPI features for your source files
- Contents
  - Using Editor features for MPI
  - MPI Help features
  - Finding MPI Artifacts
  - MPI New Project Wizards
  - MPI Barrier Analysis

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

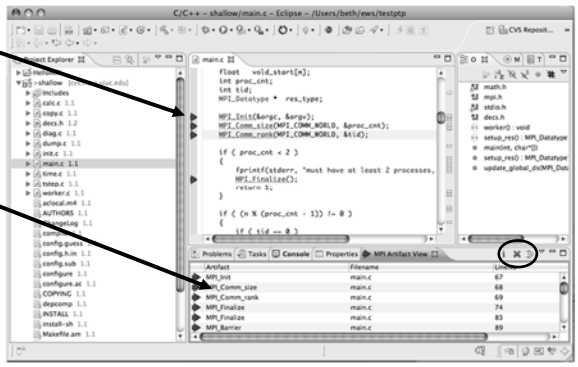
Show MPI Artifacts

- In Project Explorer, select a project, folder, or a single source file
  - The analysis will be run on the selected resources
- Run the analysis by clicking on drop-down menu next to the analysis button
- Select **Show MPI Artifacts**
- Works on local and remote files

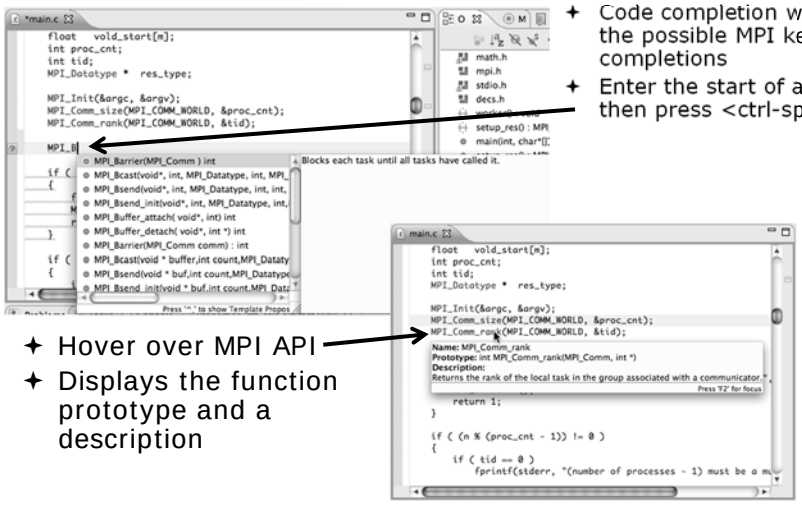


MPI Artifact View

- 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

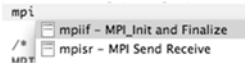
- Click mouse, then press help key when the cursor is within a function name
  - Windows: **F1** key
  - Linux: **ctrl-F1** key
  - MacOS X: **Help** key or **Help▸Dynamic Help**
- A help view appears (**Related Topics**) which shows additional information (You may need to click on MPI API in editor 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

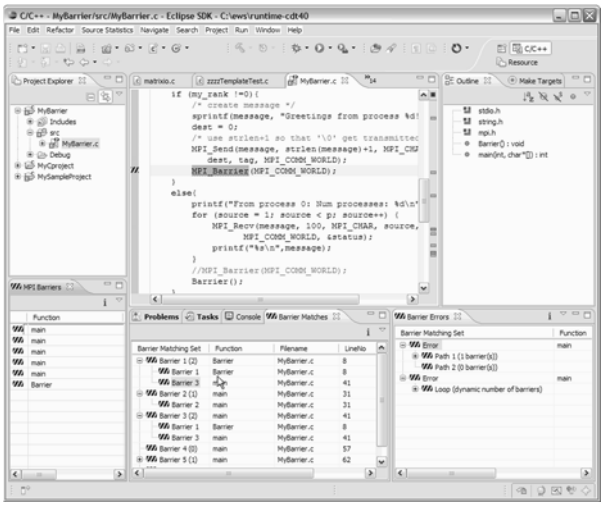
- Example:
  MPI send-receive
- Enter:
  `mpisr` <ctrl-space>
- Expands to a send-receive pattern
- Highlighted variable names can all be changed at once
- Type `mpi` <ctrl-space> <ctrl-space> to see all templates




```
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



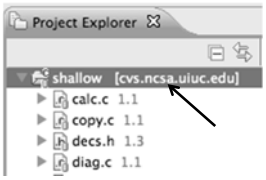
- Verify barrier synchronization in C/MPI programs
- For verified programs, lists barrier statements that synchronize together (match)
- For synchronization errors, reports counter example that illustrates and explains the error

Local files only

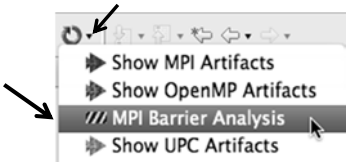
MPI Barrier Analysis (2)

Run the Analysis:

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

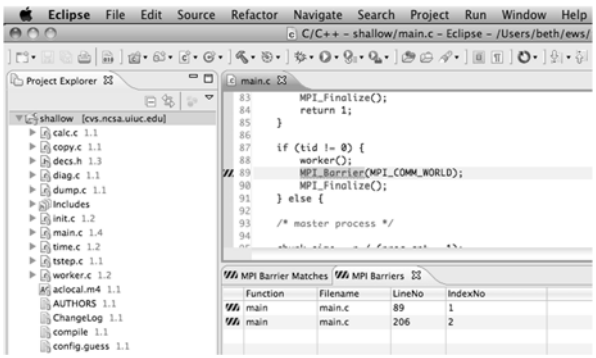


- Select the MPI Barrier Analysis action in the pull-down menu

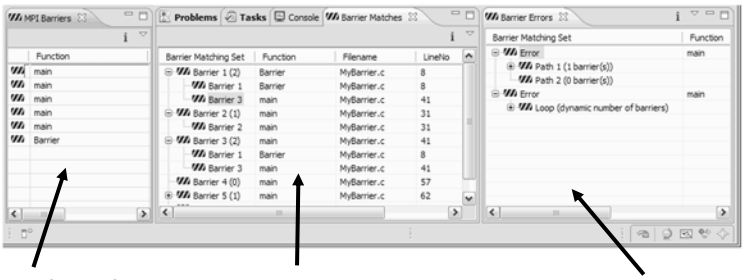


MPI Barrier Analysis (3)

- 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)

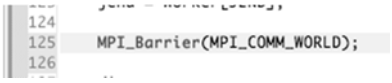
Barrier Matches view
Groups barriers that match together in a barrier set – all processes must go through a barrier in the set to prevent a deadlock

Barrier Errors view

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

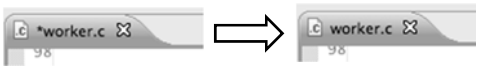
Barrier Errors

- 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

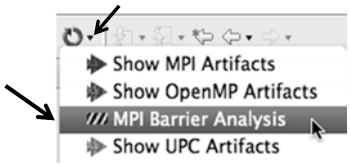


Barrier Errors (2)

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

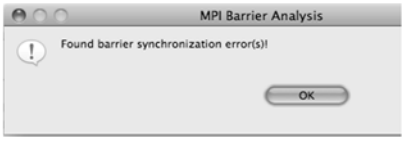


- Run barrier analysis on shallow project again
  - Select shallow project in Project Explorer first

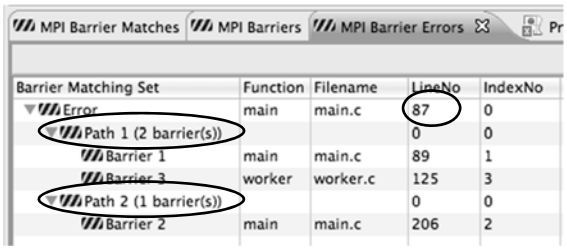


Barrier Errors (3)

- Barrier Error is found
- Hit OK to dismiss dialog



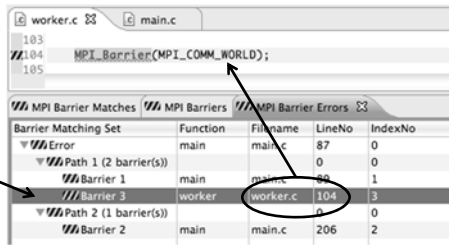
- Code diverges on line 87
  - One path has 2 barriers, other has 1



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



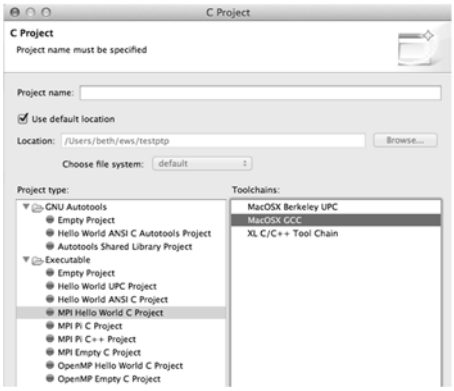
- Remove the line and save the file
- Re-run the barrier analysis to check that it has been fixed

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

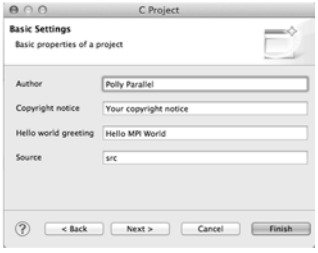
MPI New Project Wizards

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

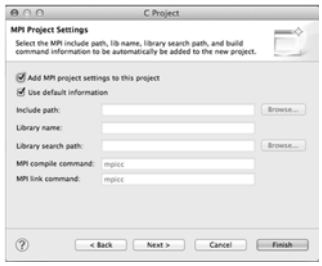


MPI New Project Wizards (2)

- Next> and fill in (optional) Basic Settings

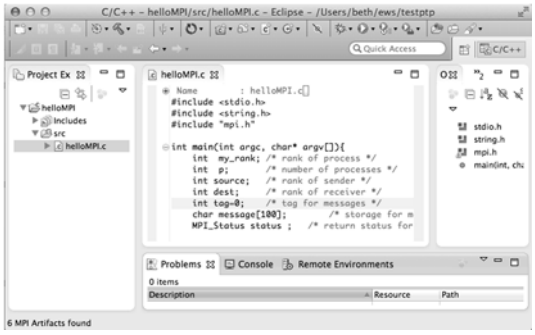


- Next> and fill in MPI Project Settings
- Include path set in MPI Preferences can be added to project



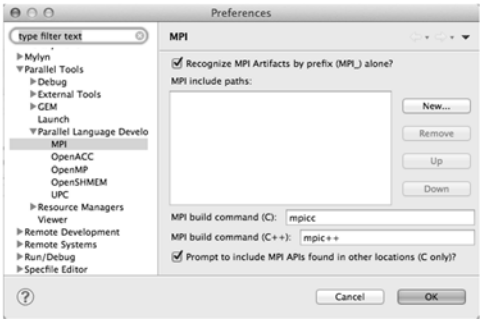
MPI New Project Wizards (3)

- 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

Building a Project

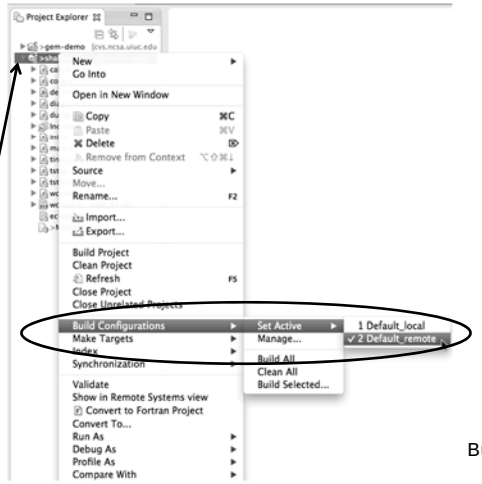
- Objective
  - Learn how to build an MPI program on a remote system
- Contents
  - How to change build settings
  - How to start a build and view build output
  - How to clean and rebuild a project
  - How to create build targets

Synchronizing the Project Prior to Build

- Because we will be running on a remote system, we must also build on that system
- Source files must be available to build
- We have already created a synchronized project to do this
- Files are synchronized automatically when they are saved
- A full synchronize is also performed prior to a build

Active Build Configuration

- The “Active” build configuration determines which system will be used for both synchronizing and building
- Since this is a Synchronized Project, the remote target will be the Active Build Configuration by default
- Right mouse on Project
- Next slide confirms where this is….

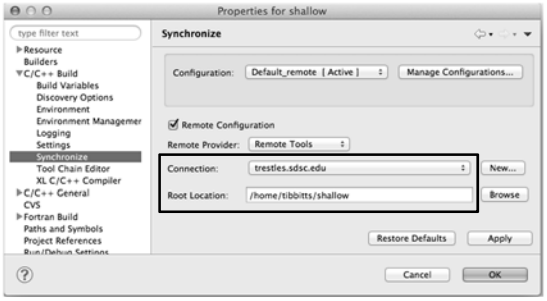


Confirm Active Build Configuration

- “Where is my project going to build?”

To confirm where each build configuration is located:

- In **Project Properties*,** under C/C++ Build, select **Synchronize.** You should see the remote connection and file location

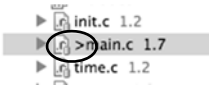


* To see Project Properties: in Project Explorer view, right mouse on project and select **Properties…** at the bottom of the context menu

Start with clean 'shallow'

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

Changed file:

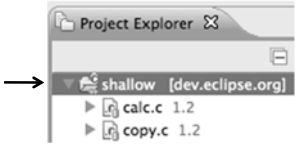


Also see Compare With ...

  - 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

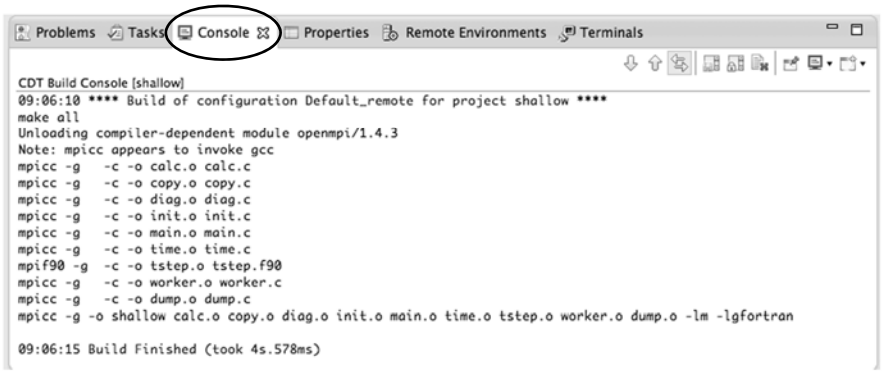


- Click on the hammer button in toolbar to run a build using the active build configuration

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

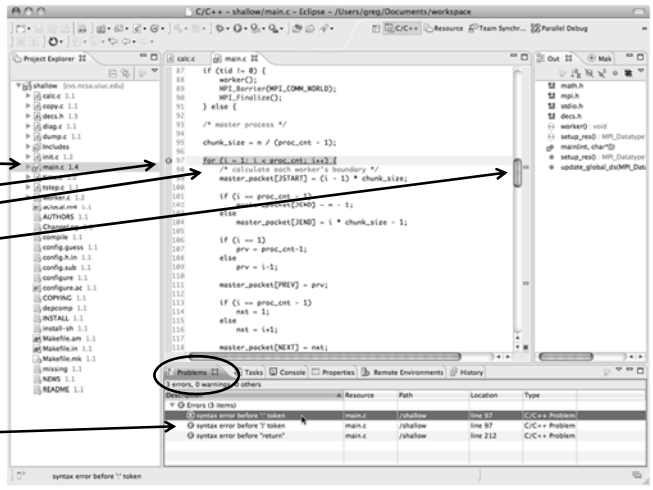
Viewing the Build Output

- Build output will be visible in console



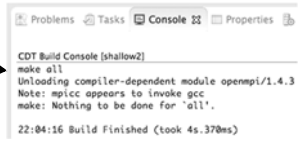
Build Problems

- 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

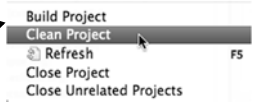


Forcing a Rebuild

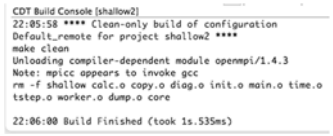
- If no changes have been made, `make` doesn't think a build is needed e.g. if you only change the Makefile



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



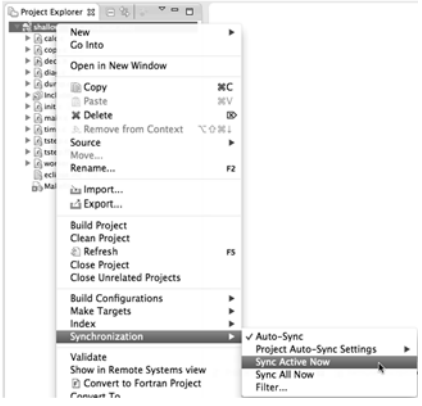
- Build console will display results



- Rebuild project by clicking on build button again 

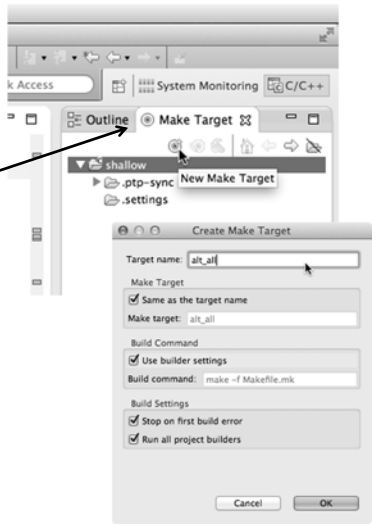
Forcing a Resync

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



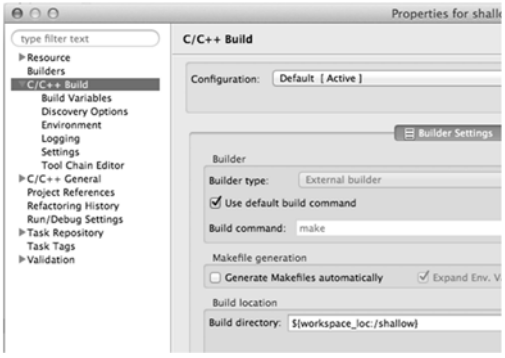
Creating Make Targets

- 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



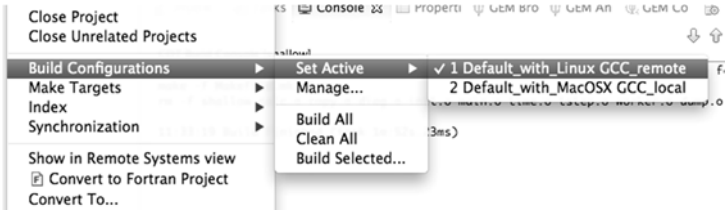
Build Configuration

- The build configuration is specified in the project properties
- Open the properties by right-clicking on the project name in the **Project Explorer** view and selecting **Properties** (bottom of the context menu list)
- **C/C++ Build**
  - Configure the build command
  - Default is “make” but this can be changed to anything
- **C/C++ Build > Settings**
  - Binary and Error parser selection
  - Tool Chain settings (managed projects only)
- **C/C++ Build > Environment**
  - Modify/add environment variables passed to build
- **C/C++ Build > Logging**
  - Enable/disable build logging



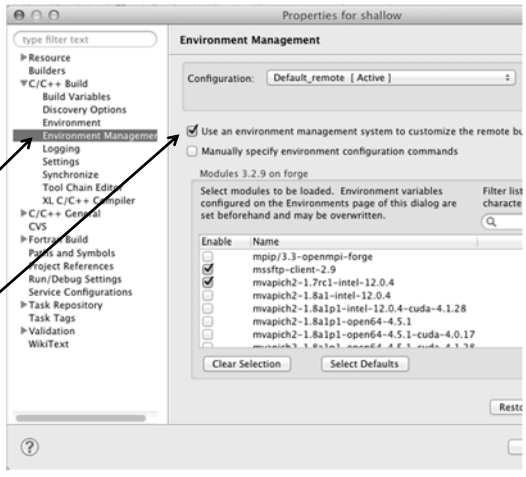
Selecting Build Configuration

- Multiple build configurations may be available
  - Remote and local build configuration
  - Build configurations for different architectures
- The active build configuration is set from the **Build Configurations** project context menu
  - Right click on project, then select the build configuration from the **Build Configurations > Set Active** menu



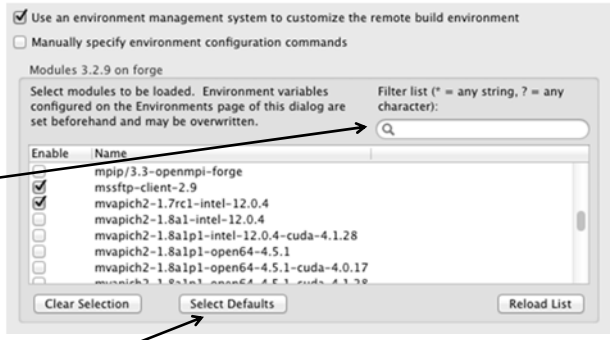
Configuring Build Modules

- If the remote system has Modules installed, a custom set of modules can be configured for building C/C++ projects
- In the project properties, navigate to **C/C++ Build > Environment Management**
- Check **Use an environment management system to customize the remote build environment**



Configuring Build Modules (2)

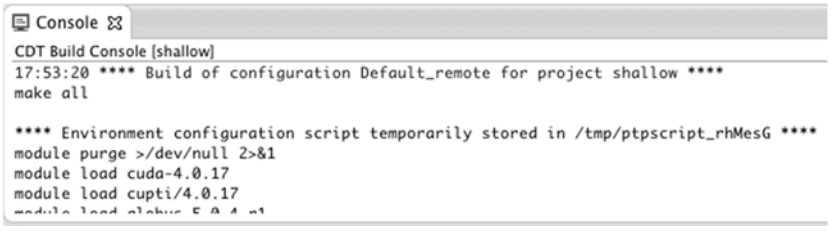
- Select modules from the list
- Use the **Filter list** field to quickly find modules with a given name



- Click **Select Defaults** to check only those modules that are present in a new Bash login shell

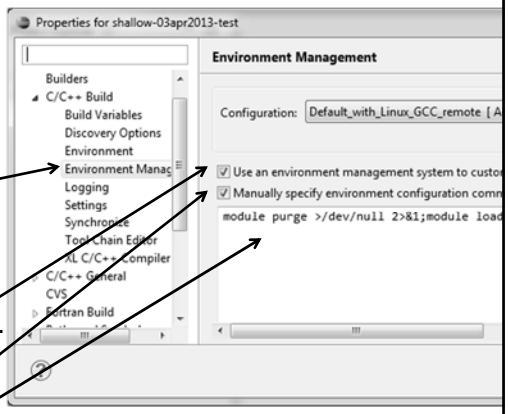
Configuring Build Modules (3)

- To 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



Configuring Build Modules: SEA

- Right click on Project, select project properties
  - C/C++ Build, Environment Management
  - Select “Use an environment management system..
  - Select “Manually specify environment…”



Add string and then hit **OK**

module purge >/dev/null 2>&1;module load intel/12.1.5;module load ncarbinlibs/1.0;module load ncarcompilers/1.0;module load ncarenv/1.0;module load netcdf/4.2;module load workshop;module load tau

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

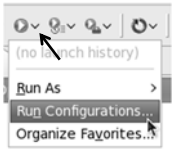
1. Open the Makefile in Eclipse. Note the line starting with "tags:" – this defines a make target named **tags**.
2. Open the Outline view while the Makefile is open. What icon is used to denote make targets in the Outline?
3. Right-click the **tags** entry in the Outline view. Add a Make Target for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

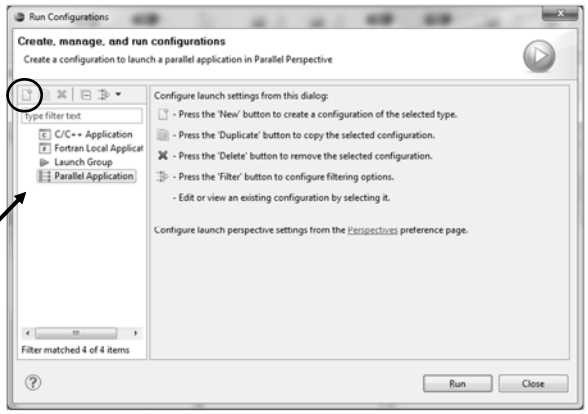
- Objective
  - Learn how to run an MPI program on a remote system
- Contents
  - Creating a run configuration
  - Configuring the application run
  - Monitoring the system and jobs
  - Controlling jobs
  - Obtaining job output

Creating a Run Configuration



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

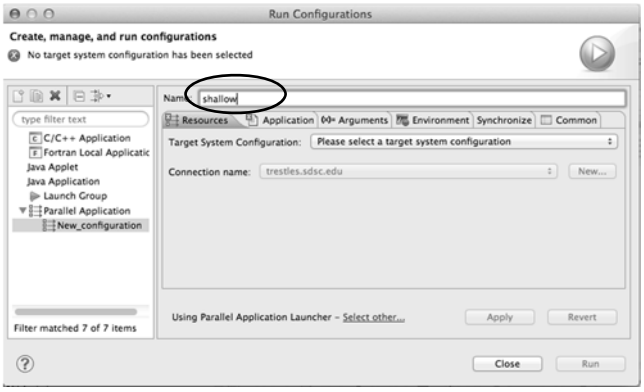
  Or, just double-click on **Parallel Application** to create a new one



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.

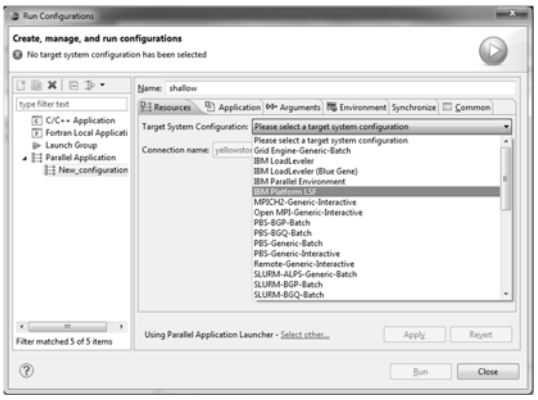
Set Run Configuration Name

- Enter a name for this run configuration
  - E.g. "shallow"
- This allows you to easily re-run the same application



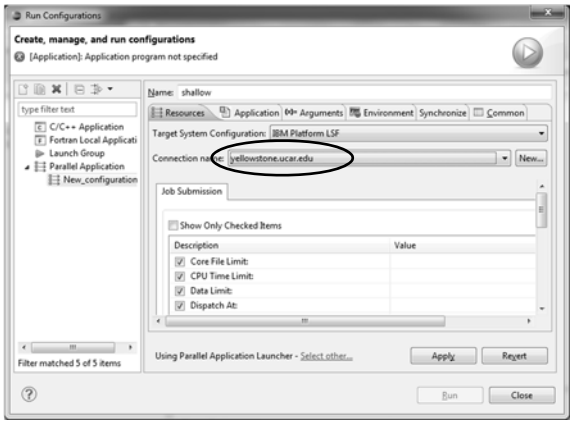
Configuring the Target System

- In **Resources** tab, select a **Target System Configuration** that corresponds to your target system
  - The tutorial instructor will indicate what Target System Configuration to select
  - SEA: use IBM Platform LSF
- 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



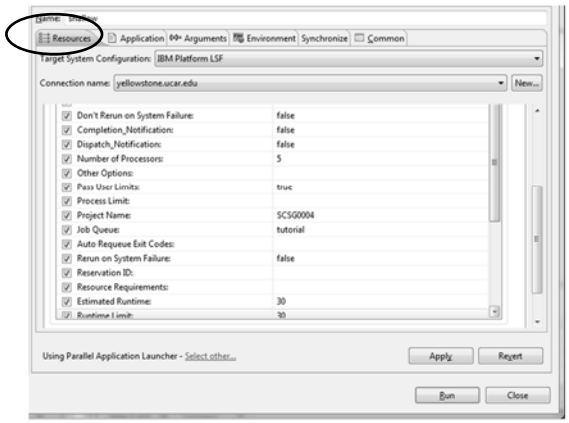
Configure the Connection

- 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
- SEA: Select the connection you already have to yellowstone.ucar.edu



Resources Tab

- The content of the **Resources** tab will vary depending on the target system configuration selected
- This example shows the IBM Platform LSF configuration
- For the SEA workshop, we will indicate the settings that need to be set on the next slide



LSF Settings for SEA Tutorial

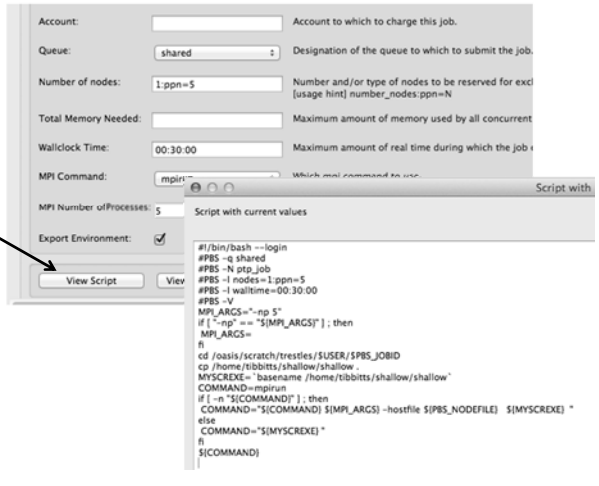
Description	Value
Exclusive Execution	false
Job is Resizable	false
Don't Rerun on system failure	True
Completion_Notification	false
Dispatch_Notification	false
Number of Processors	5
Pass User Limits	True
Project Name	SCSG004
Job Queue	tutorial
Rerun on System Failure	false
Estimated Runtime	30
Runtime Limit:	30
STDERR Path (Append)	/glade/u/home/jalameda/shallow-03apr2013-test/
STDOUT Path (Append)	/glade/u/home/jalameda/shallow-03apr2013-test/
MPI Command	mpirun.lsf

Pull down menu

Your project directory

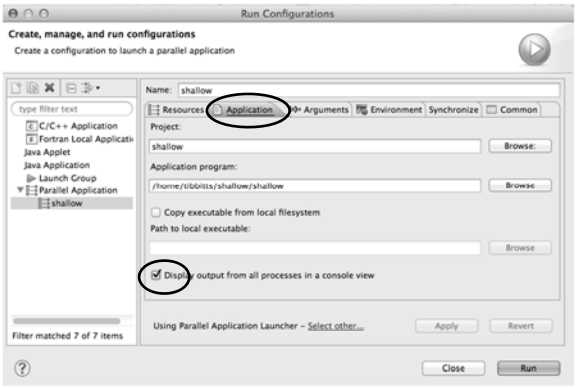
Viewing the Job Script

- 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



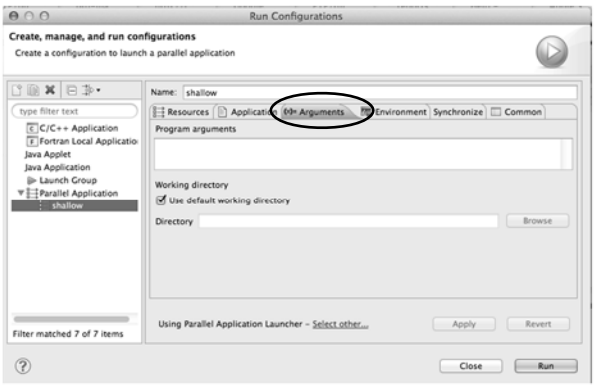
Application Tab

- 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**



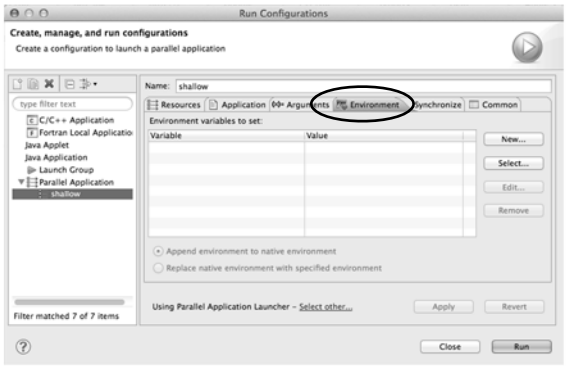
Arguments Tab (Optional)

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



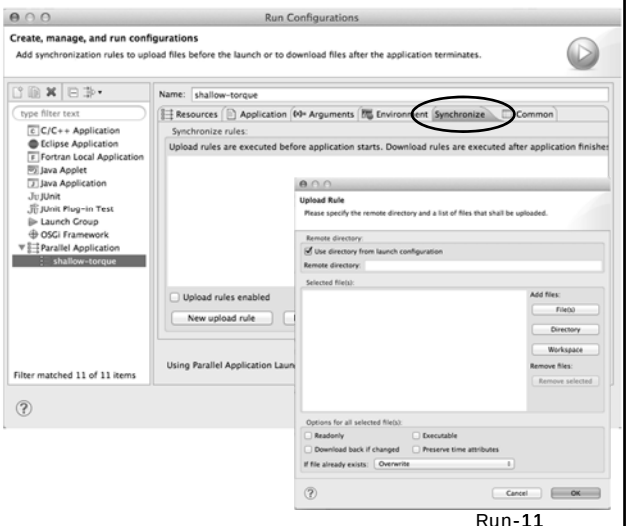
Environment Tab (Optional)

- 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 in a separate module



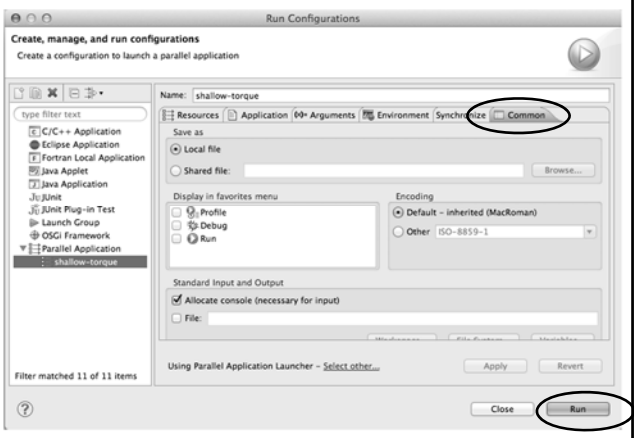
Synchronize Tab (Optional)

- 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



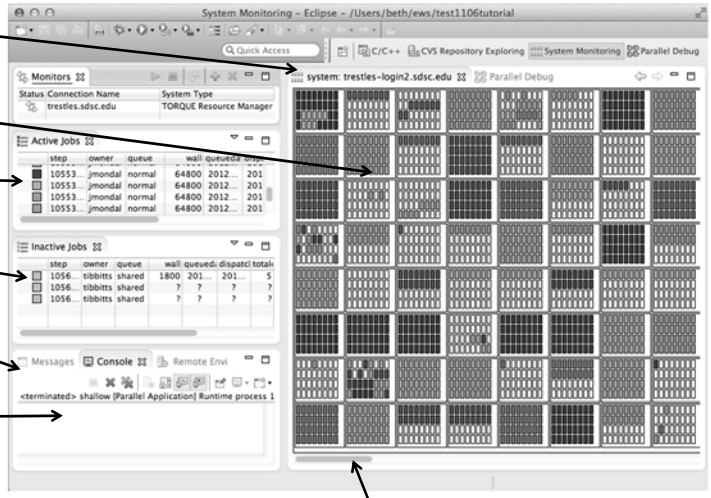
Run

- 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

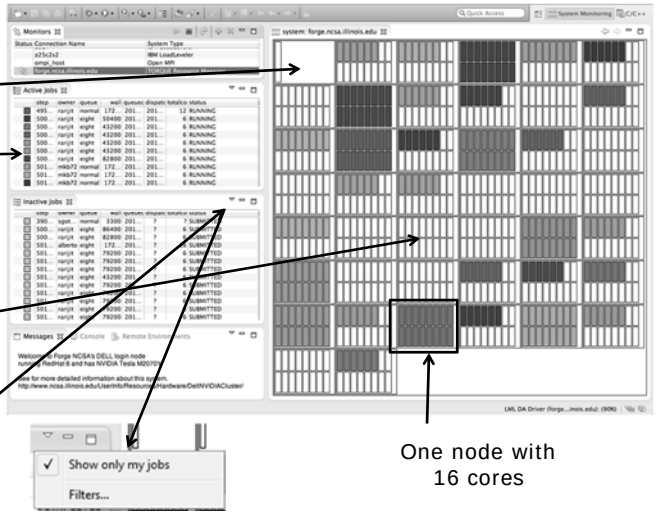


Moving views

- 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

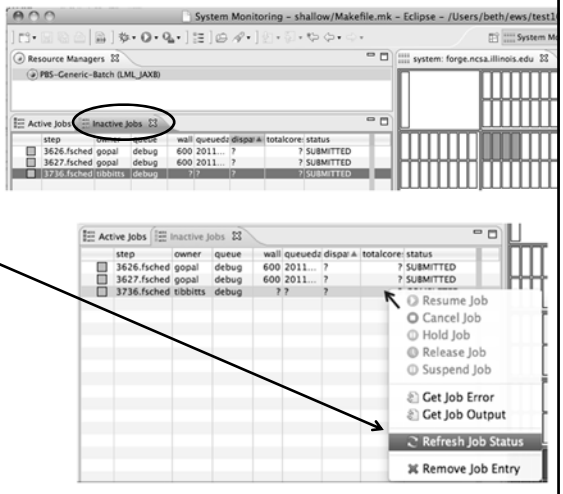
System Monitoring

- **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
- Pull down to filter jobs, or select your own



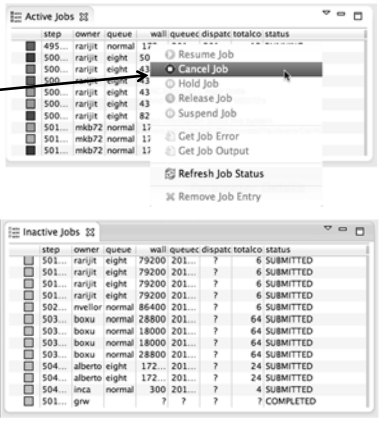
Job Monitoring

- 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



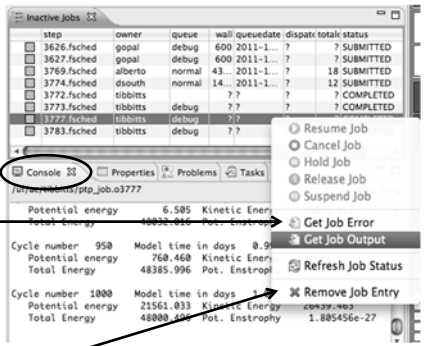
Controlling Jobs

- 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



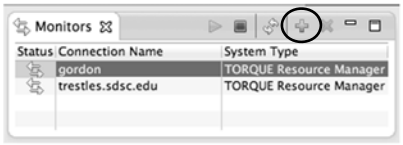
Obtaining Job Output

- 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
  - SEA: output/error will be in project directory, force synchronization to see
- 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

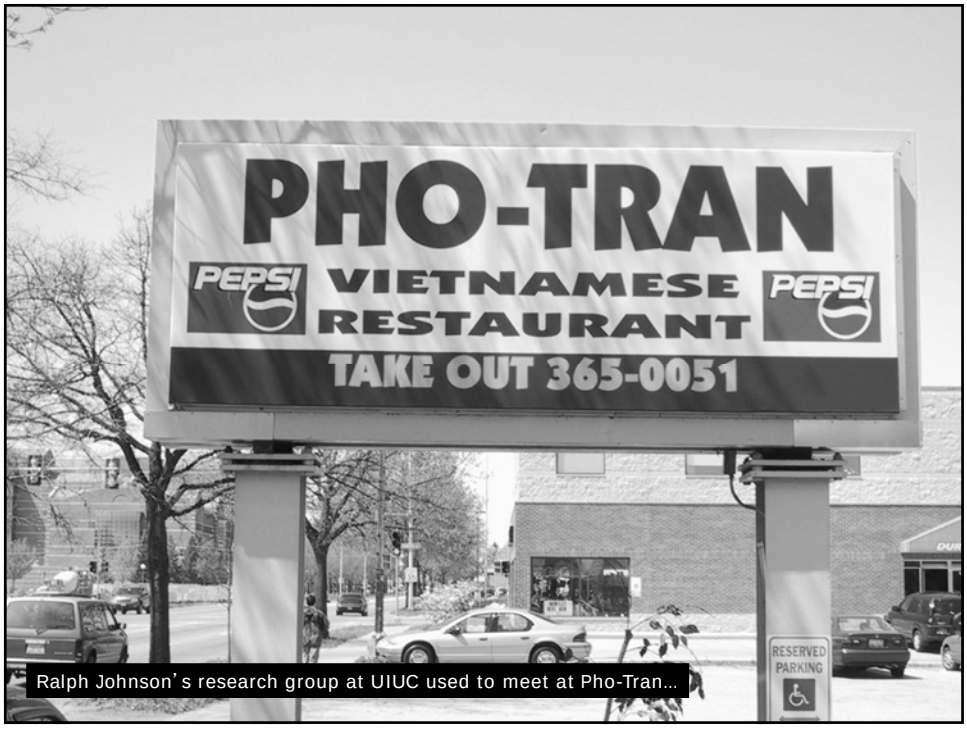
Double click new monitor to start

Exercise

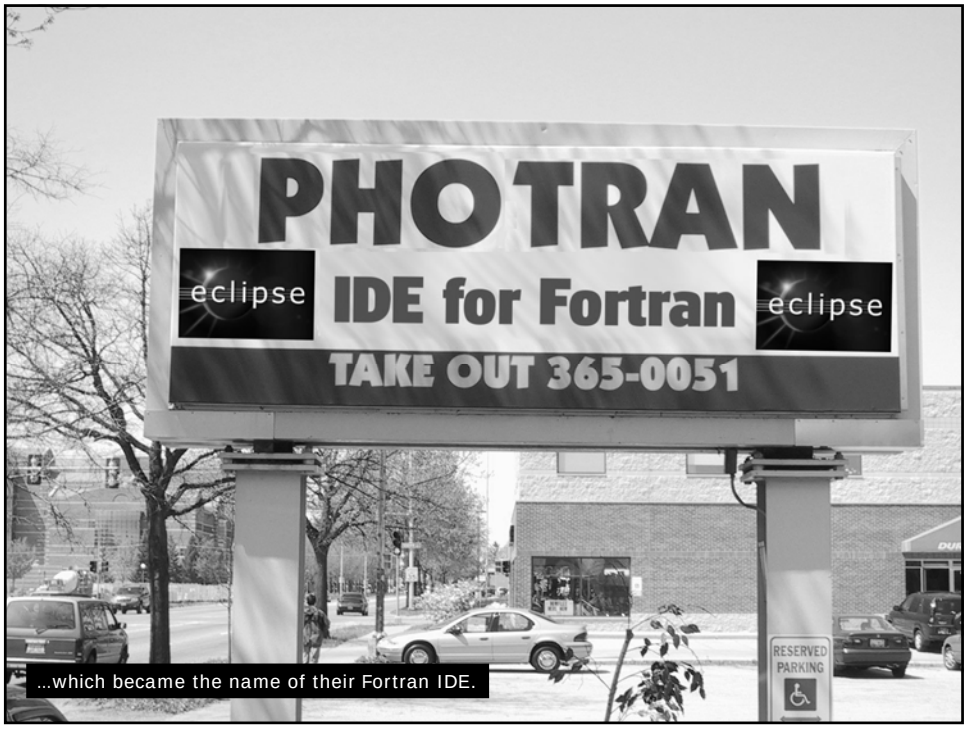
1. Start with your 'shallow' project
2. Create a run configuration
3. Complete the Resources tab
4. Select the executable in the Application tab
5. Submit the job
6. Check the job is visible in the Inactive Jobs view, moves to the Active Jobs view when it starts running (although it may be too quick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



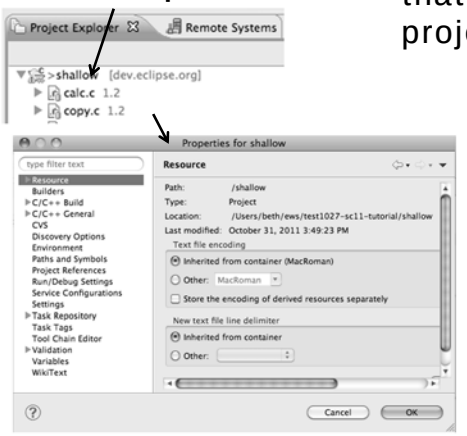
Ralph Johnson's research group at UIUC used to meet at Pho-Tran...



Configuring Fortran Projects

Project Properties

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

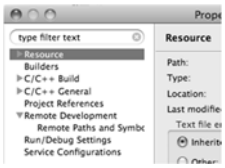


- *Project properties* are settings that can be changed for each project
  - Contrast with *workspace preferences,* which are the same regardless of what project is being edited
    - e.g., editor colors
    - Set in **Window▶ Preferences** (on Mac, **Eclipse▶ Preferences**)
    - Careful! Dialog is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

Converting to a Fortran Project

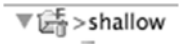
- Are there categories labeled **Fortran General** and **Fortran Build** in the project properties?

No Fortran categories →




Do this once

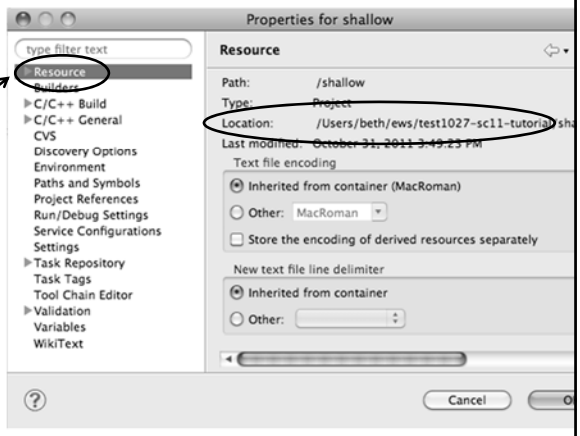
- If not, the project is not a Fortran Project
  - Switch to the Fortran Perspective
  - In the Fortran Projects view, right-click on the project, and click **Convert to Fortran Project**
  - Don't worry; it's still a C/C++ project, too



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

Project Location

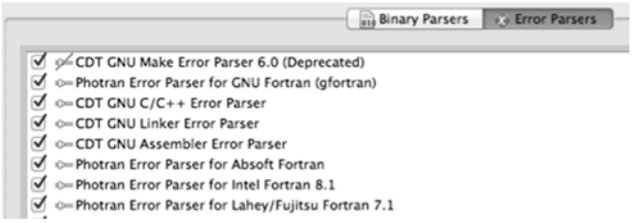
- How to tell where a project resides?
- In the project properties dialog, select the **Resource** category



Error Parsers

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


Do this once



Fortran Source Form Settings

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

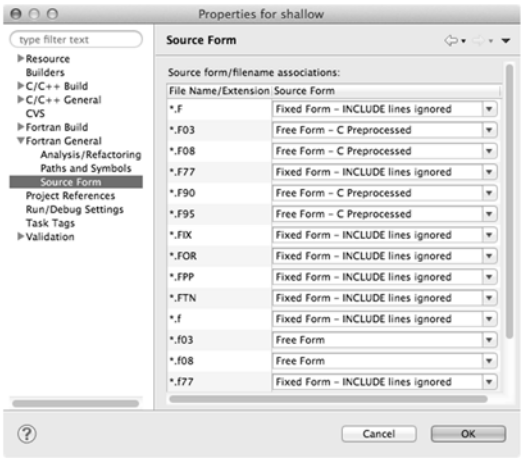
Fixed form:	.f	.fix	.for	.fpp	.ftn	.f77
Free form:	.f08	.f03	.f95	.f90		< unpreprocessed
	.F08	.F03	.F95	.F90		< preprocessed

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

Fortran Source Form Settings

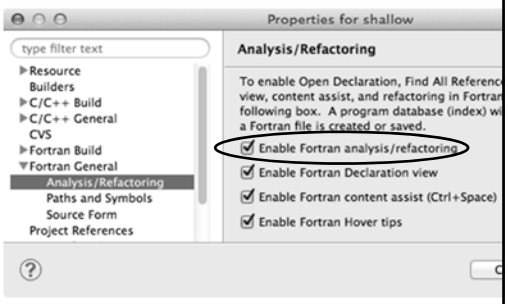
Do this once

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



Enabling Fortran Advanced Features

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



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

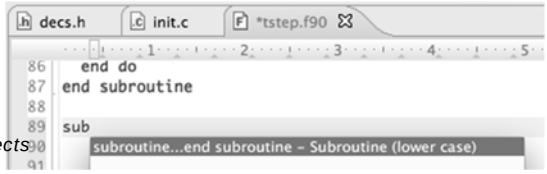
Advanced Editing

Code Templates

Code Templates

(C/C++ and Fortran)

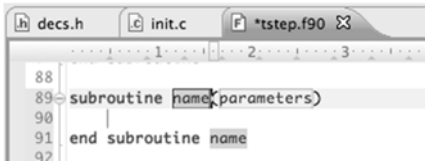
- Auto-complete common code patterns
  - For loops/do loops, if constructs, etc.
  - Also MPI code templates
- Included with content assist proposals (when **Ctrl-Space** is pressed)
  - E.g., after the last line in tstep.f90, type "sub" and press **Ctrl-Space**
  - Press **Enter** to insert the template



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

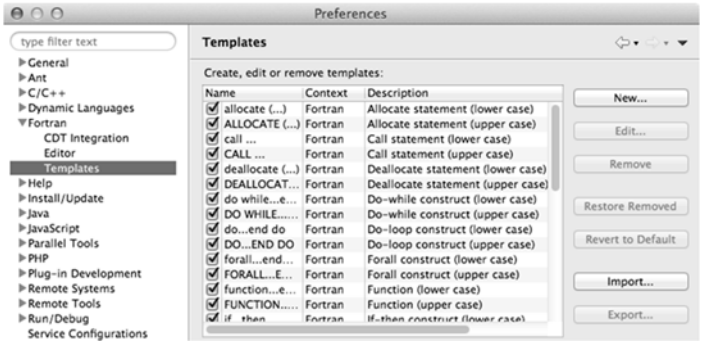
Exercise

- 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**



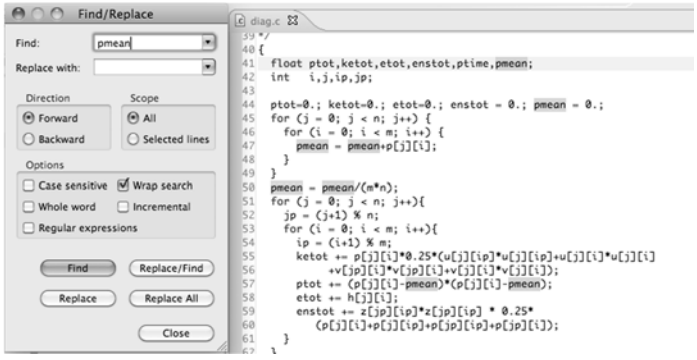
- Can import/export templates to XML files

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

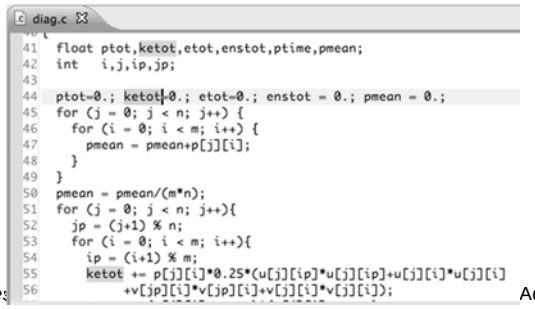
- Simple Find within editor buffer
- Ctrl-F (Mac: Command-F)



Mark Occurrences

(C/C++ Only)

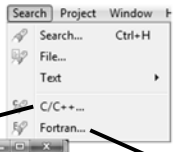
- Double-click on a variable in the CDT editor
- All occurrences in the source file are highlighted to make locating the variable easier
- Alt-shift-O to turn off (Mac: Alt-Cmd-O)



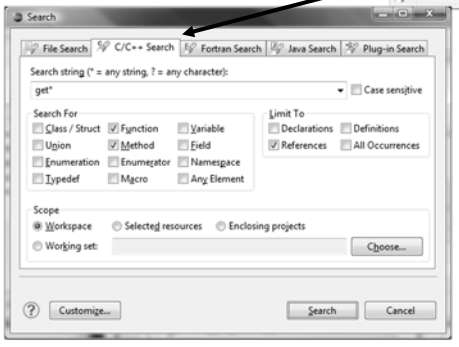
Language-Based Searching

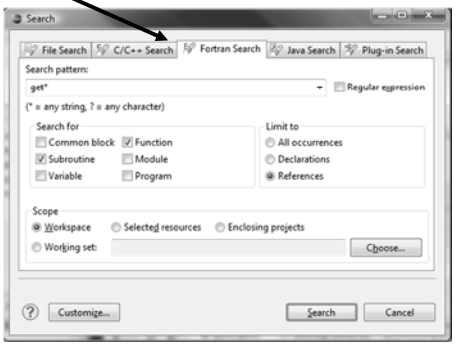
(C/C++ and Fortran)

- "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



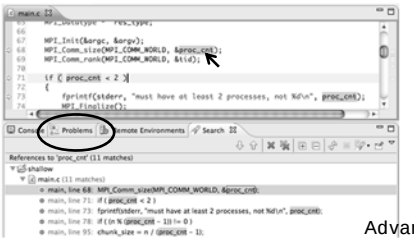


Find References

(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**

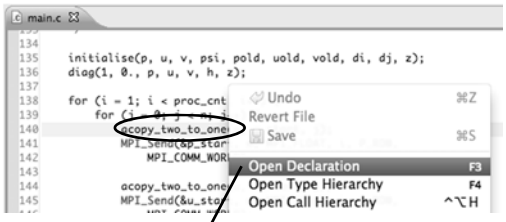
  - **Search** view shows matches



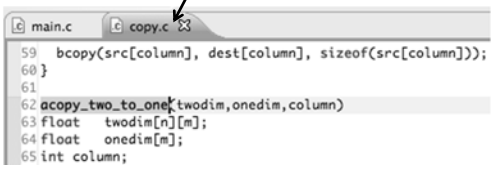
Open Declaration

(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



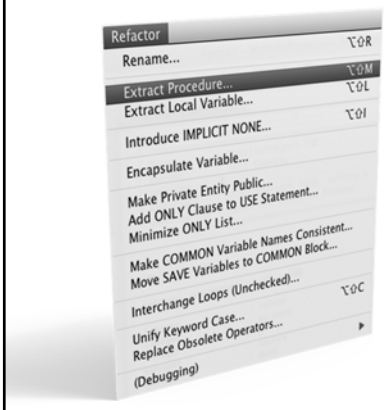
Search – Try It!

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)



- Refactoring is the research motivation for Photran @ Illinois
  - Illinois is a leader in refactoring research
  - "Refactoring" was coined in our group (Opdyke & Johnson, 1990)
  - We had the first dissertation… (Opdyke, 1992)
  - …and built the first refactoring tool… (Roberts, Brant, & Johnson, 1997)
  - …and first supported the C preprocessor (Garrido, 2005)
  - Photran's agenda: refactorings for HPC, language evolution, refactoring framework
- Photran 7.0: 31 refactorings

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**)

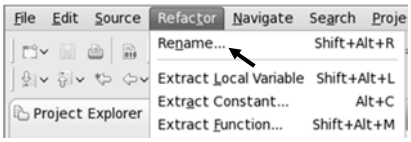
✔						
✖	**Free Form, Preprocessed:**	.F08	.F03	.F95	.F90	
✖	**Fixed Form:**	.f	.fix	.for	.fpp	.ftn	.f77

- 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):

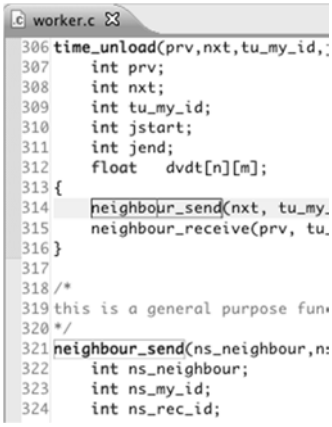
Refactoring	Uses	Percentage
Rename	179,871	74.8%
Extract Local Variable	13,523	5.6%
Move	13,208	5.5%
Extract Method	10,581	4.4%
Change Method Signature	4,764	2.0%
Inline	4,102	1.7%
Extract Constant	3,363	1.4%
(16 Other Refactorings)	10,924	4.5%

- 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)

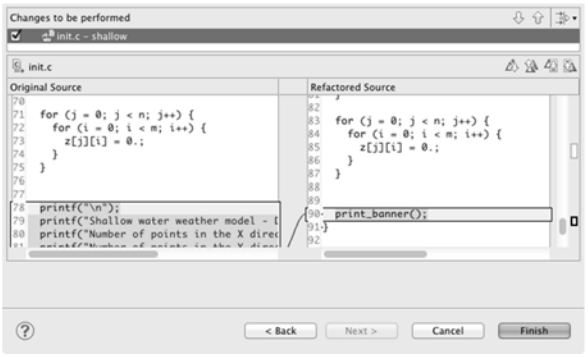
- Position the caret over an identifier.
- Press **Ctrl-1** (**Command-1** on Mac).
- Enter a new name. Changes are propagated within the file as you type.



Extract Function Refactoring

(also available in Fortran - "Extract Procedure")

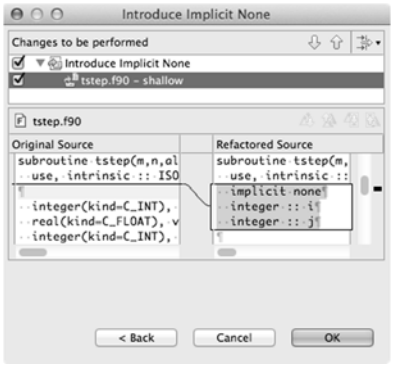
- Moves statements into a new function, replacing the statements with a call to that function
- Local variables are passed as arguments



- 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

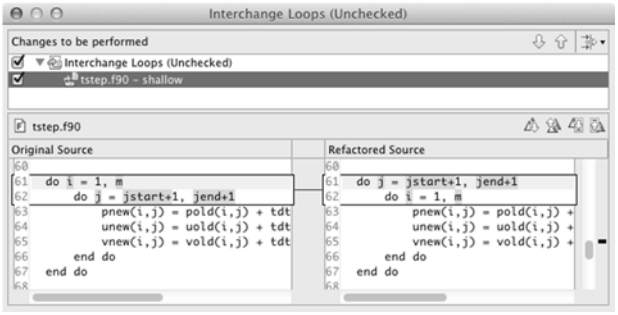


- Introduce in a single file by opening the file and selecting **Refactor▸Coding Style▸ Introduce IMPLICIT NONE…**
- Introduce in multiple files by selecting them in the Fortran Projects view, right-clicking on the selection, and choosing **Refactor▸Coding Style▸Introduce IMPLICIT NONE…**

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

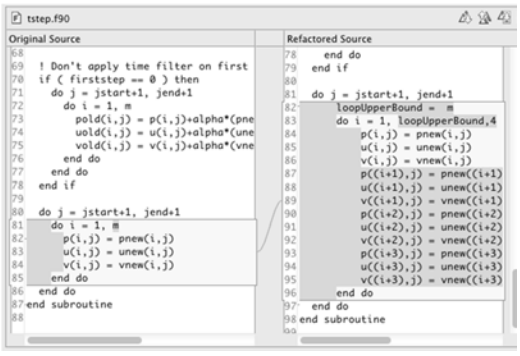
(Fortran only)

- **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…

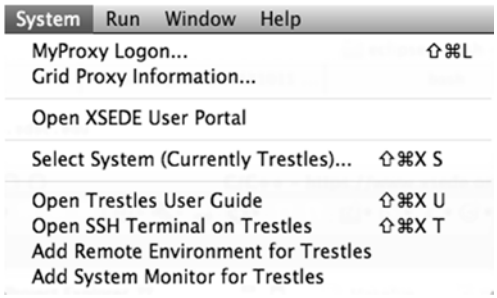
1. In init.c, extract the `printf` statements at the bottom of the file into a new function called `print_banner`
2. In worker.c, change the spellings of `neighbour_send` and `neighbour_receive` to American English
3. In tstep.f90, make the (Fortran) `tstep` subroutine IMPLICIT NONE

NCSA/XSEDE Features

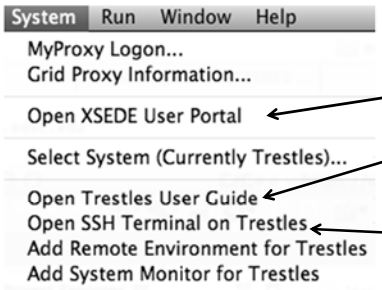
- Objectives
  - Install NCSA's GSI auth and XSEDE support plug-ins
  - Become familiar with the System menu
- Contents
  - Capabilities
  - Installation
- Prerequisites
  - (none)

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



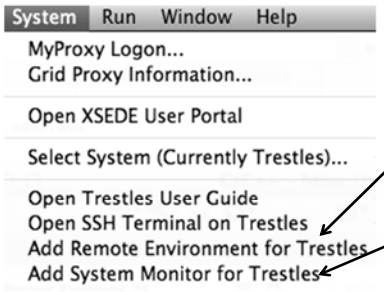
System Menu



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

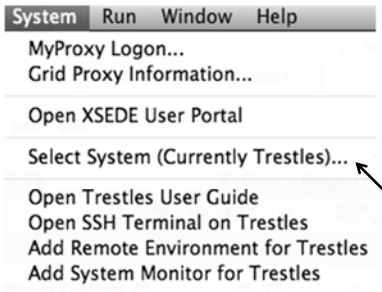
Eclipse-integrated SSH terminals are provided by the Remote System Explorer (RSE), one of the features that is included in the Eclipse for Parallel Application Developers package.

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

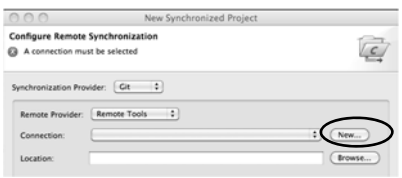
   `http://forecaster.ncsa.uiuc.edu/updates/juno`

   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

Configuring SSH Tunnel

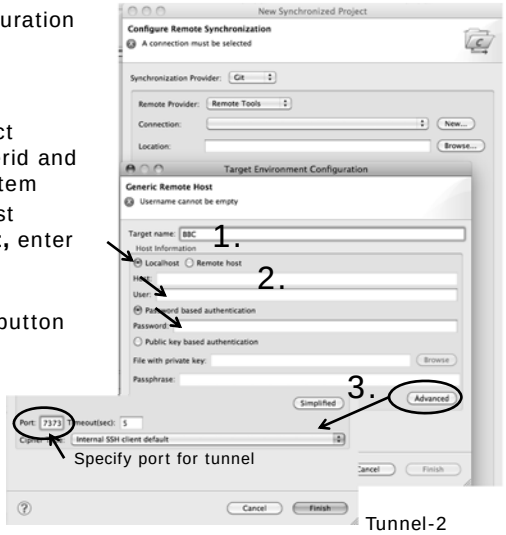
Configure the Synchronized Project - SSH tunnel (1)

- 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**



Parallel Debugging

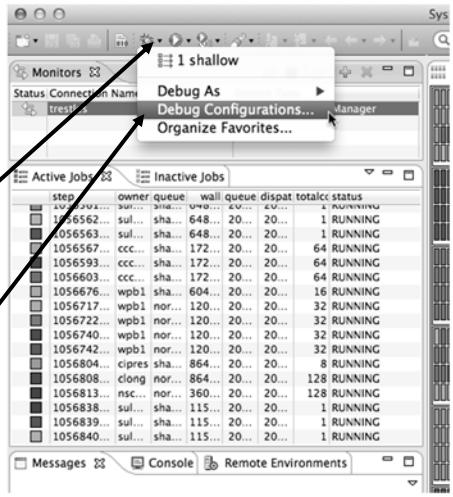
- Objective
  - Learn the basics of debugging parallel programs
- Contents
  - Launching a debug session
  - The Parallel Debug Perspective
  - Controlling sets of processes
  - Controlling individual processes
  - Parallel Breakpoints
  - Terminating processes

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*

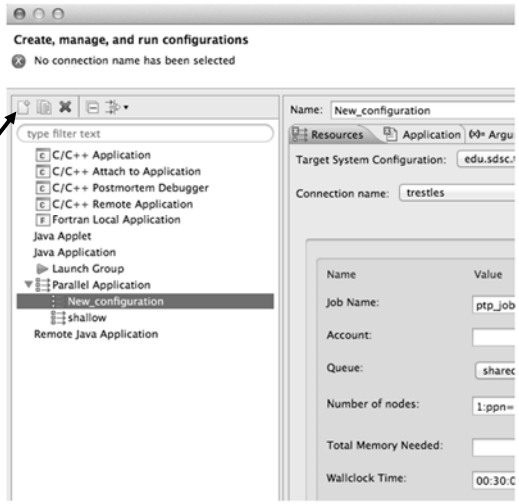
Create a Debug Configuration

- 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



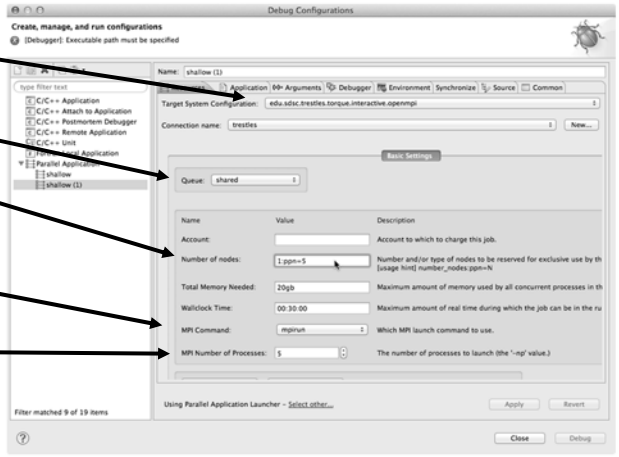
Create a New Configuration

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



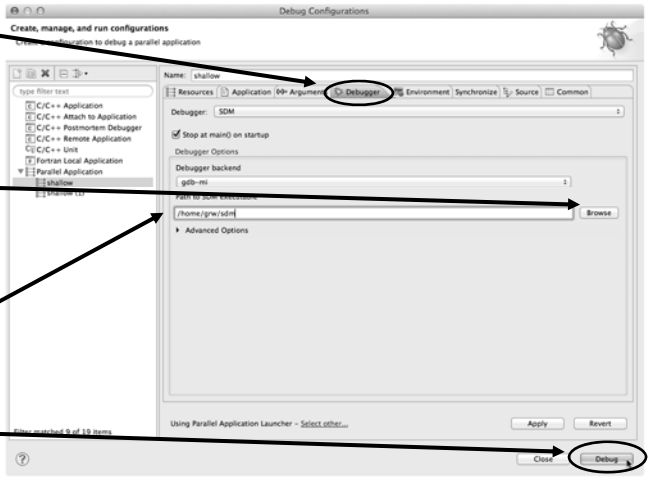
Configure the Resources Tab

- 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 the Debug Tab

- Select **Debugger** tab
- Choose “gdb-mi” for the **Debugger backend**
- Click **Browse** and select “sdm “in your home directory
- Click **Ok**
- Make sure the debugger path is correct
- Click on **Debug** to launch the program

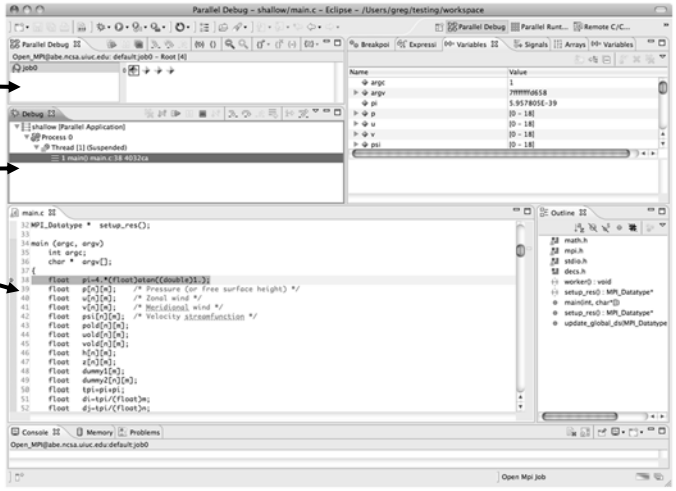


Exercise

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

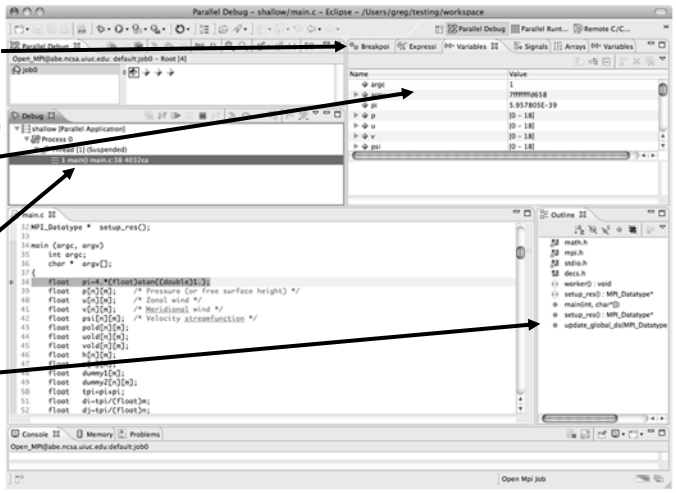
The Parallel Debug Perspective (1)

- **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



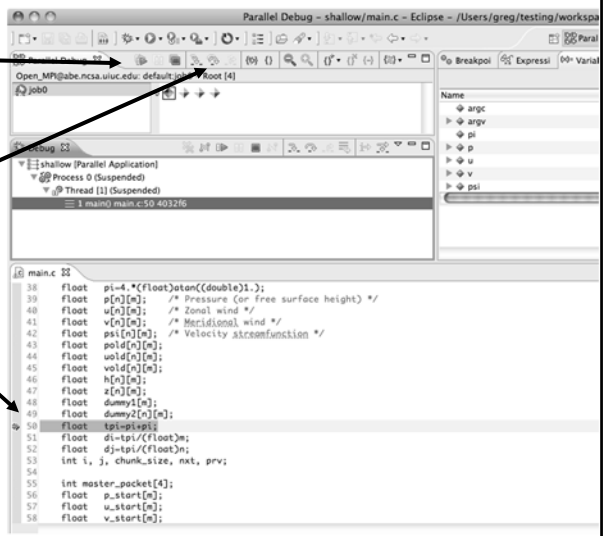
The Parallel Debug Perspective (2)

- **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



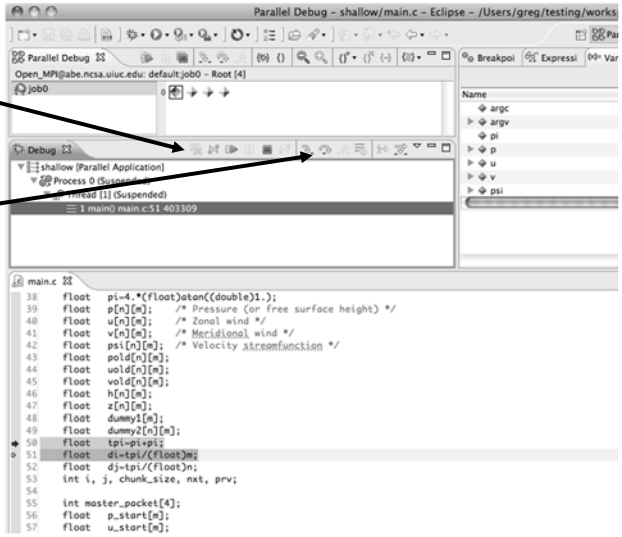
Stepping All Processes

- The buttons in the **Parallel Debug View** control groups of processes
- Click on the **Step Over** button
- Observe that all process icons change to green, then back to yellow
- Notice that the current line marker has moved to the next source line



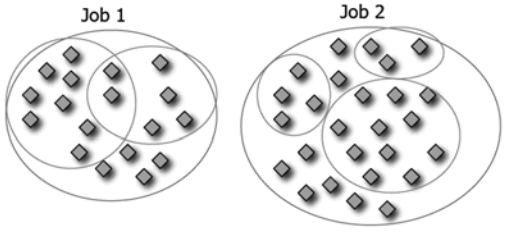
Stepping An Individual Process

- The buttons in the **Debug view** are used to control an individual process, in this case process 0
- Click the **Step Over** button
- You will now see two current line markers, the first shows the position of process 0, the second shows the positions of processes 1-3



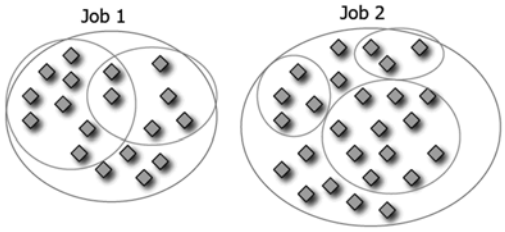
Process Sets (1)

- 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



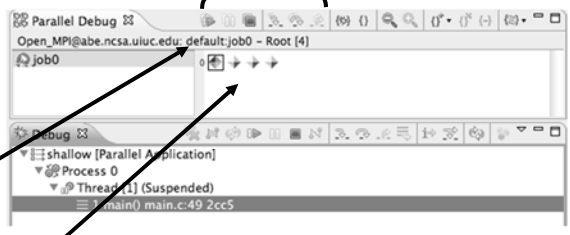
Process Sets (2)

- 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

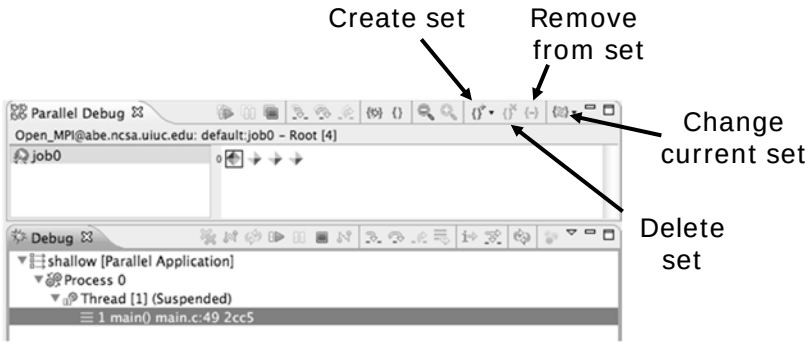
- Debug operations on the **Parallel Debug view** toolbar always apply to the current set:
  - Resume, suspend, stop, step into, step over, step return
- The current process set is listed next to job name along with number of processes in the set
- The processes in process set are visible in right hand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



Root set = all processes

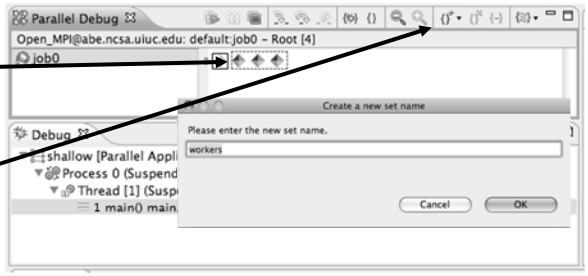
Managing Process Sets

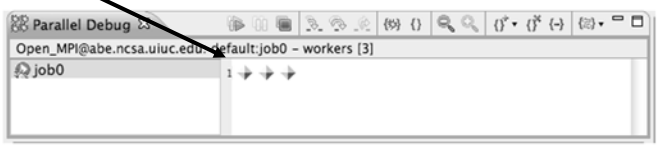
- The remaining icons in the toolbar of the **Parallel Debug view** allow you to create, modify, and delete process sets, and to change the current process set



Creating A New Process Set

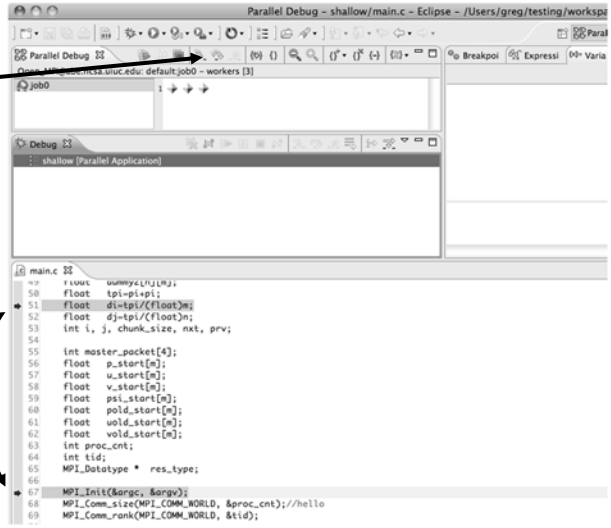
- Select the processes you want in the set by clicking and dragging, in this case, the last three
- Click on the **Create Set** button
- Enter a name for the set, in this case **workers**, and click **OK**
- You will see the view change to display only the selected processes





Stepping Using New Process Set

- With the **workers** set active, click the **Step Over** button
- You will see only the first current line marker move
- Step a couple more times
- You should see two line markers, one for the single master process, and one for the 3 worker processes

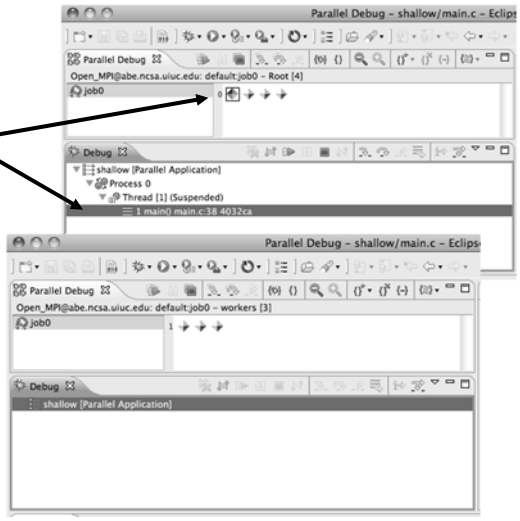


Process Registration

- 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)

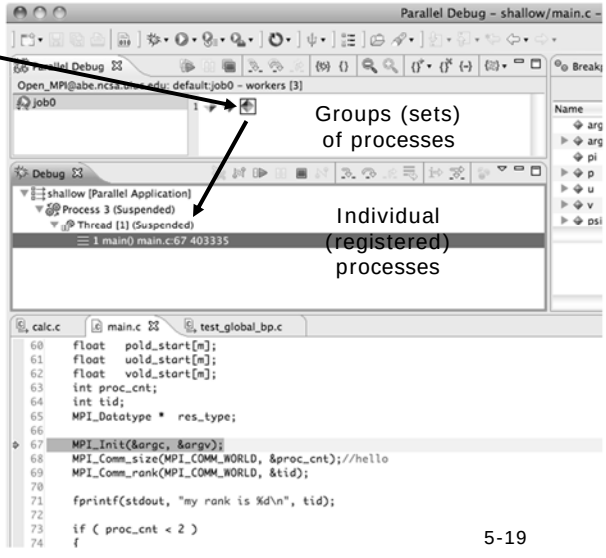
- By default, process 0 was registered when the debug session was launched
- Registered processes are surrounded by a box and shown in the Debug view



- 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



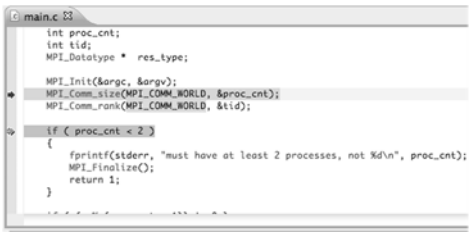
- To un-register a process, double-click on the process icon or select a number of processes and click on the **unregister** button

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

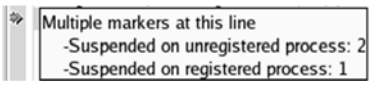
- The highlight color depends on the processes suspended at that line:
  - **Blue:** All registered process(es)
  - **Orange:** All unregistered process(es)
  - **Green:** Registered or unregistered process with no source line (e.g. suspended in a library routine)
- The marker depends on the type of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

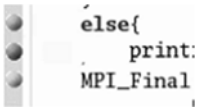


Exercise

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)

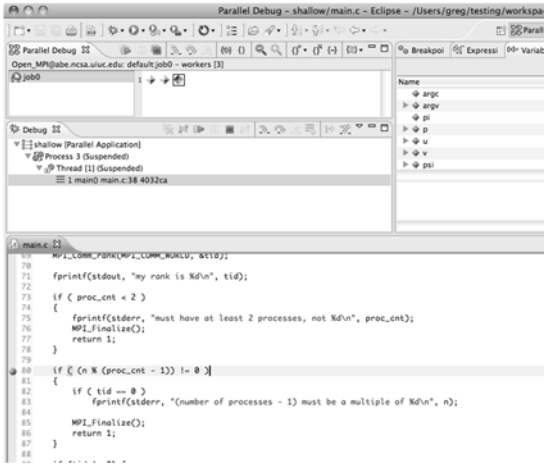
Breakpoints

- 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



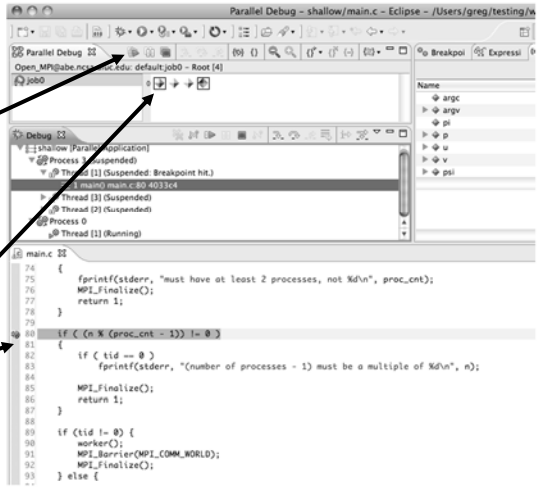
Creating A Breakpoint

- 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



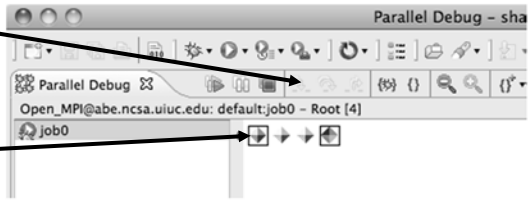
Hitting the Breakpoint

- 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 1-3 are suspended on the breakpoint

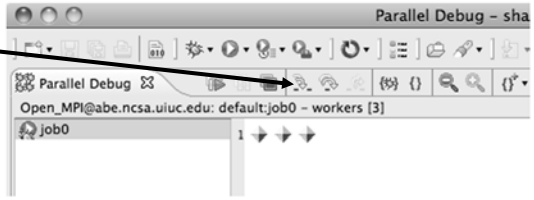


More On Stepping

- 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

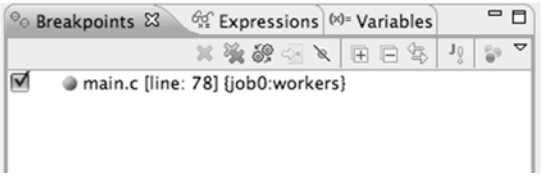


- Switch to the set of suspended processes (the **workers** set)
- You will now see the **Step** buttons become enabled



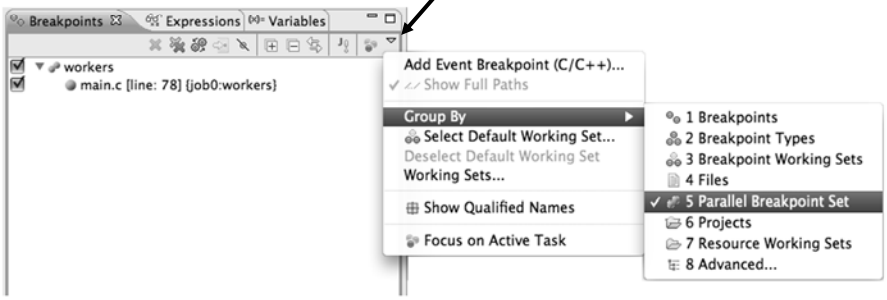
Breakpoint Information

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



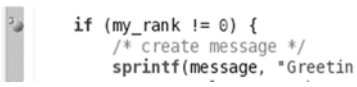
Breakpoints View

- 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

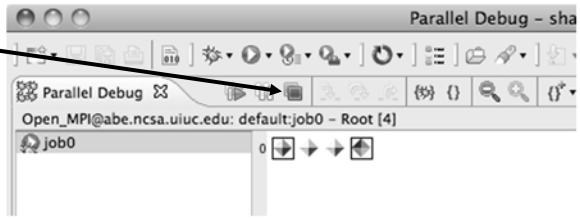


Exercise

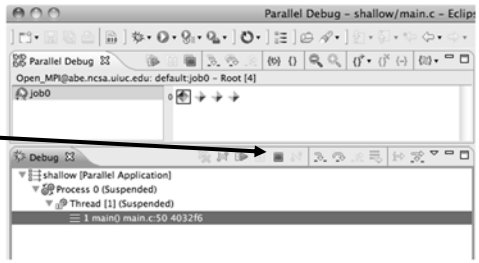
1. Select the “worker” process set
2. Create a breakpoint by double-clicking on right hand bar at line 88 (worker function)
3. Hover over breakpoint to see properties
4. Switch to “root” process set
5. Observer breakpoint color changes to blue
6. Resume all processes
7. Observe “worker” processes at 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

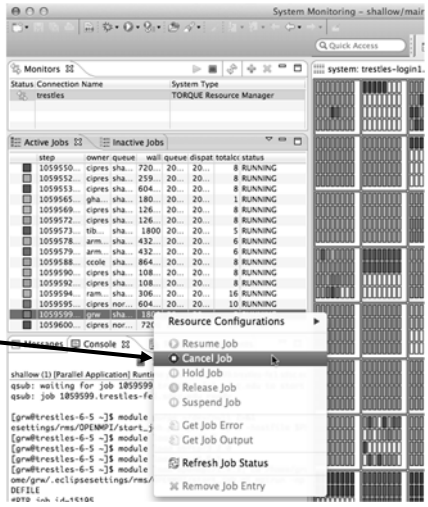


- You can also use the terminate icon in the **Debug** view to terminate the currently selected process



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

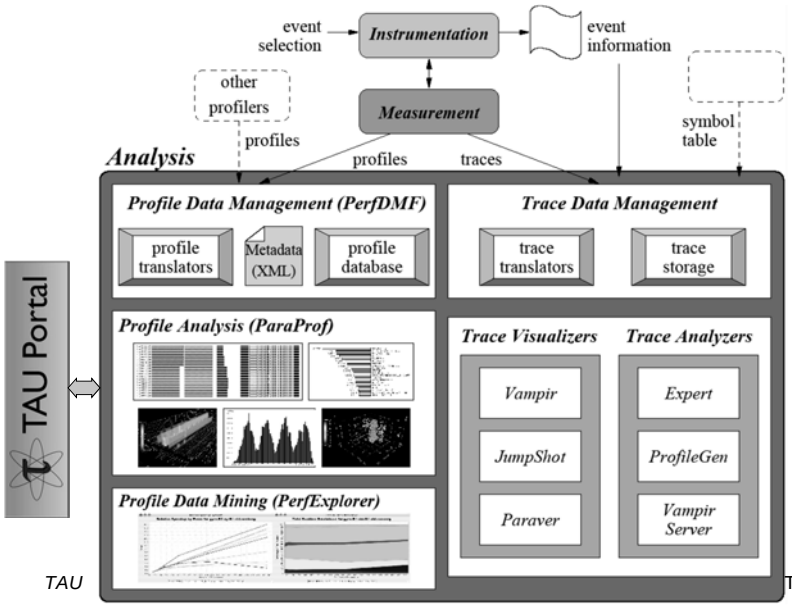
Performance Tuning and Analysis Tools - TAU

- 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

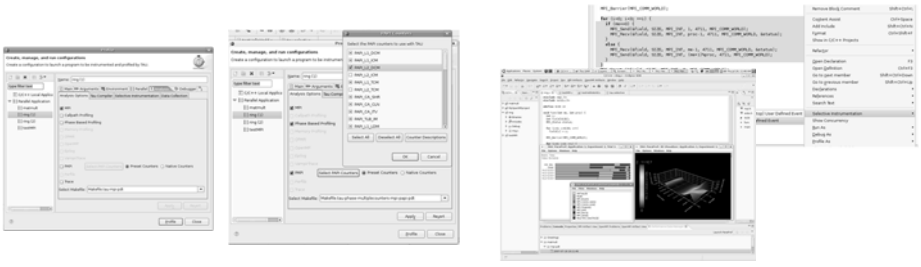
TAU Performance System Architecture



PTP TAU plug-ins

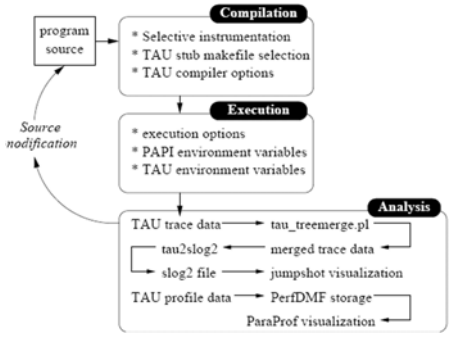
http://www.cs.uoregon.edu/research/tau

- 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



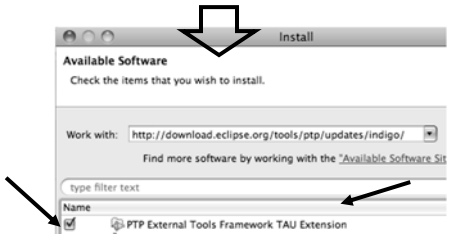
TAU Integration with PTP

- 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.tgz
  - tar -zxf tautools.tgz
  - cd tautools-2.22b
  - ./configure
  - Set path as shown (launch eclipse from this environment)
  - Run 'taudb_configure --create-default'
  - 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 yellowstone run the following module commands:

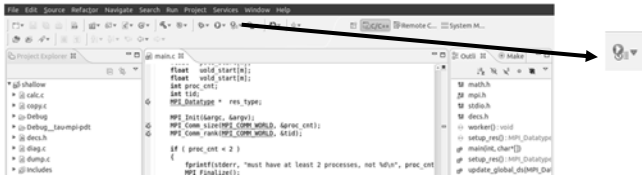
  module load workshop; module load tau
- Include 'eclipse.inc' in the makefile*
  - Create an empty eclipse.inc file in the same directory as the makefile
  - Place 'include eclipse.inc' in the makefile after regular compiler definitions
  - ETFw will modify eclipse.inc to set CC/CXX/FC variables

Selective Instrumentation

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

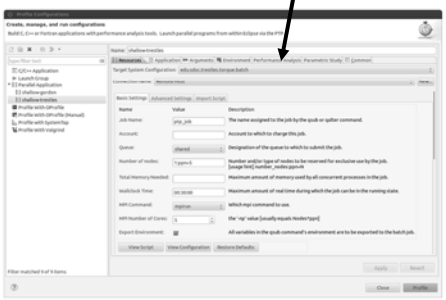


- From the dropdown menu select 'Profile configurations…'

Select Configuration

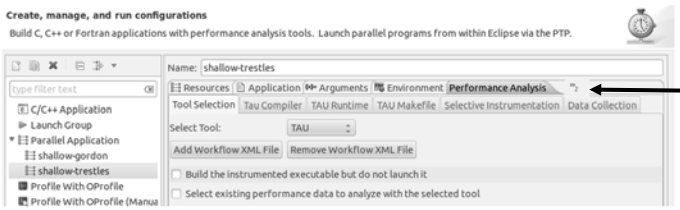
- Select the shallow configuration prepared earlier
- The Resource and Application configuration tabs require little or no modification
  - We are using the same resource manager and Torque settings
  - Since we are using a makefile project the application will be rebuilt in and run from the previously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

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



Select Tool/Workflow

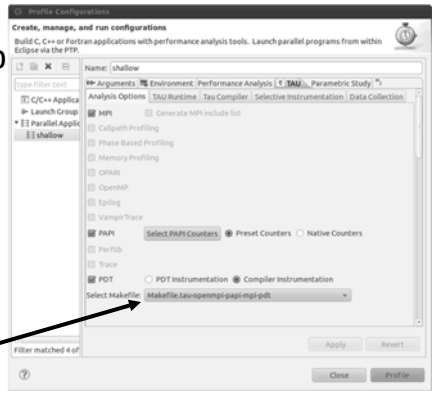
- Select the **Performance Analysis** tab and choose the TAU tool set in the 'Select Tool' dropdown box
  - Other tools may be available, either installed as plug-ins or loaded from workflow definition XML files
  - Configuration sub-panes appear depending on the selected tool



Tabs may be hidden if the window is too small

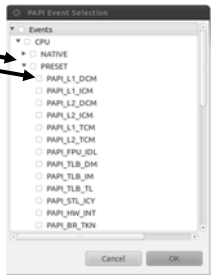
Select TAU Configuration

- 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-icpc-papi-mpi-pdt



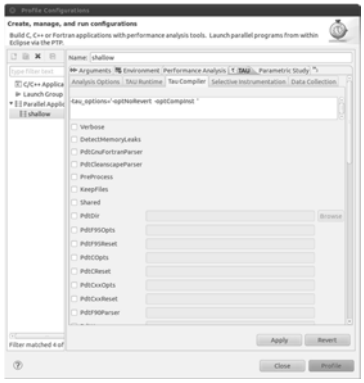
Choose PAPI Hardware Counters

- 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**



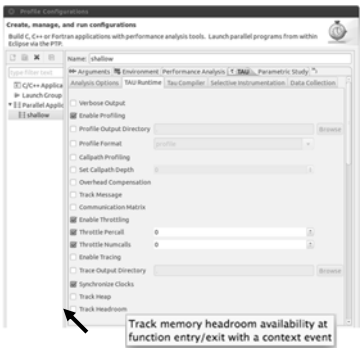
Compiler Options

- 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

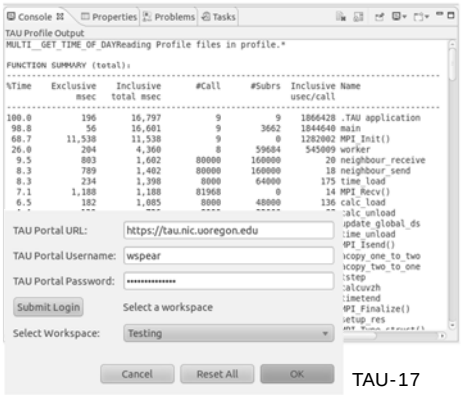
- TAU Runtime options
  - Set environment variables used by TAU
  - Control data collection behavior
  - Verbose provides debugging info
  - Callpath shows call stack placement of events
  - Throttling reduces overhead
  - Tracing generates execution timelines



Hover help

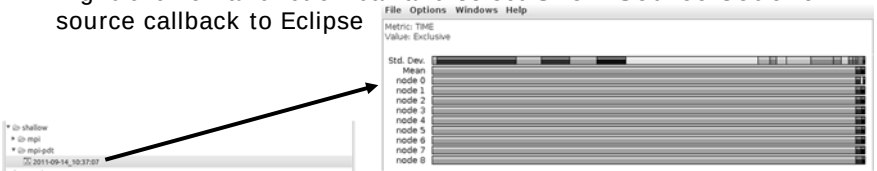
Working with Profiles

- 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



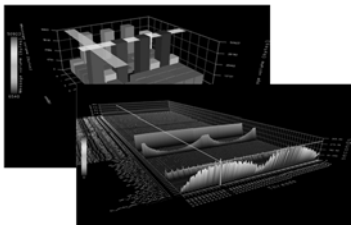
Launch TAU Analysis

- 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

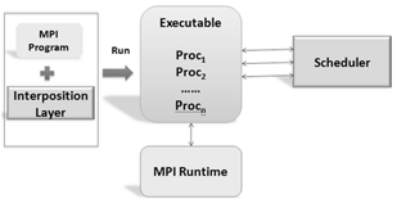
GEM

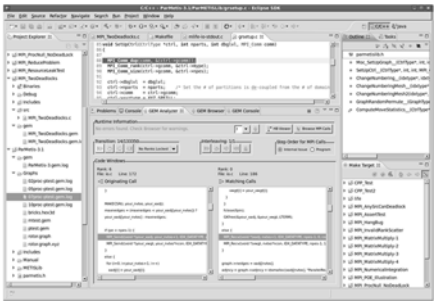
Graphical Explorer of MPI Programs

GEM
Graphical Explorer of MPI Programs

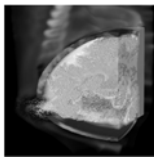
- Dynamic verification for MPI C/C++ that detects:
  - Deadlocks
  - MPI object leaks (e.g. communicators, requests, datatypes)
  - Functionally irrelevant barriers
  - Local assertion violations
  - MPI Send/Recv Type Mismatches
- Offers rigorous coverage guarantees
  - Complete nondeterministic coverage for MPI (MPI_ANY_SOURCE)
  - Determines relevant interleavings, replaying as necessary
  - Examines communication / synchronization behaviors

GEM - Overview





- Front-end for In-situ Partial Order (ISP) developed at University of Utah
- Contributes "push-button" C/C++ MPI verification and analysis to the development cycle
- Automatically instruments and runs user code, displaying post verification results
- Variety of views & tools to facilitate debugging and MPI runtime understanding



(Image courtesy of Steve Parker, U of Utah)

GEM – Views & Tools

Analyzer View

Highlights bugs, and facilitates post-verification review / debugging

Browser View

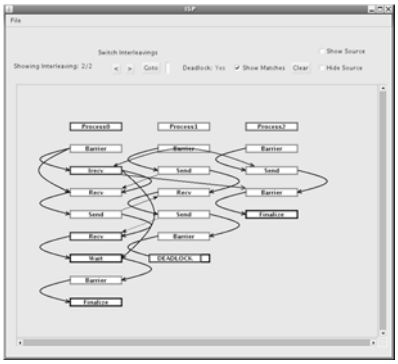
Groups and localizes MPI problems. Maps errors to source code in Eclipse editor

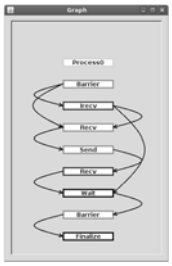
GEM – Views & Tools (cont.)

Happens-Before Viewer

Shows required orderings and communication matches

(currently an external tool – not supported in remote and synchronized projects)





Plugins for GEM

GEM can be extended to import and analyze MPI projects from the popular MPI programming book:

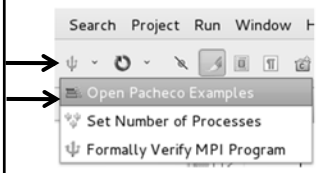
"Parallel Programming with MPI"

by Peter Pacheco

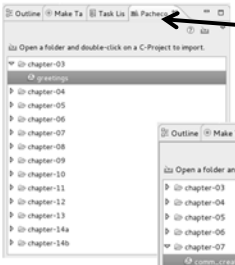
- Pacheco plugin was developed at University of Utah (not part of PTP) with full permission by the book's author
- ***An excellent way to learn MPI using PTP/GEM***
- Update Site: **http://www.cs.utah.edu/formal_verification/Pacheco/**
- **Prerequisite**: PTP 6.0 with GEM installed

Pacheco Plugin

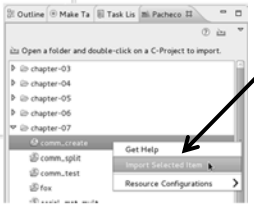
Ties in to GEM toolbar button

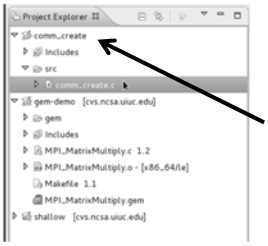


Provides a New Eclipse View with programming examples from book



Automatically import examples into workspace as Eclipse C projects





Build, work with and do analysis on any of these example projects using PTP and GEM

GEM PTP Installation

- This tutorial assumes that the GEM plugin for PTP is installed – they are not included by default in the "Eclipse for Parallel Application Developers" package
- GEM should already be installed for the tutorial, but the installation section shows how to install GEM from the PTP update site – be sure **GEM** was selected

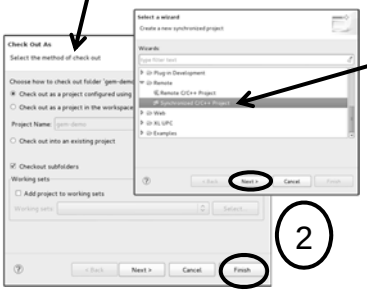
To confirm:
- Help>Install New Software…
- Select the link: "What is already installed" at the bottom of the dialog
- You should see the GEM there

GEM

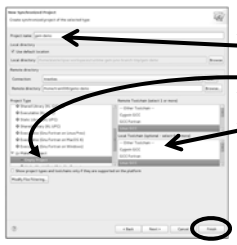
Hands-on Section

GEM Hands-on (0) Checkout GEM Project

- (1) Check out GEM project "gem-demo" from CVS repo. This is the same repository as the **shallow** project.
- (2) Check out using New Project Wizard

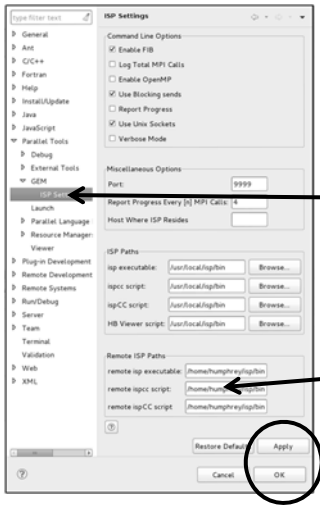


- Create a new Synchronized C Project



- (3) Name the project:
  - **"gem-demo"**
- **Empty Makefile Project**
- **Linux GCC** toolchain for both **remote** and **local**

GEM Hands-on (1)
Configure GEM Preferences



- For the tutorial, you will only need to set:

  "**Remote ISP Paths**"

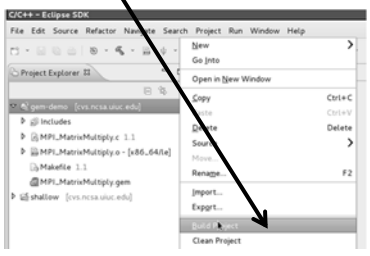
- Window→Preferences→Parallel Tools→GEM→…
- Use the following for all remote paths (ignore local):

  /home/humphrey/isp/bin

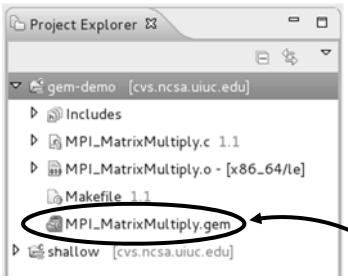
- Leave everything else at defaults
- Click Apply and OK

GEM Hands-on (2)
Build GEM Project

- Right click on **gem-demo**
- Select **"Build Project"**



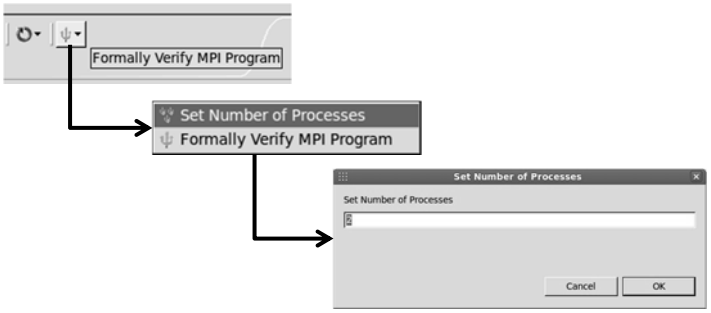
NOTE: The build automatically works here because the GEM compiler wrapper path is in the Makefile for the gem-demo project



- Instrumented executable (**.gem** extension) is visible after build (output in console)
- Force a project sync if this is not visible

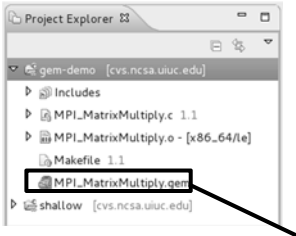
GEM Hands-on (3)
Set Number of Processes

- Locate the trident Icon on the Eclipse toolbar
- From pull-down menu, set number of processes to **2**



GEM Hands-on (4)

- Locate file the profiled binary **MPI_MatrixMultiply.gem** in the Project Explorer view

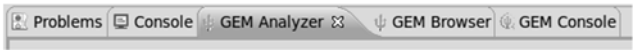


- Right click on: **MPI_MatrixMultiply.gem**
- Select:
  - GEM -> Formally Verify Profiled MPI Program

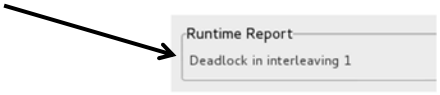
Finally, wait for analysis. This may take several seconds and you will see output in the **GEM Console View**

GEM Hands-on (5)

- Notice that three new Eclipse views have opened:
  - *See next slide if your views are not positioned correctly*



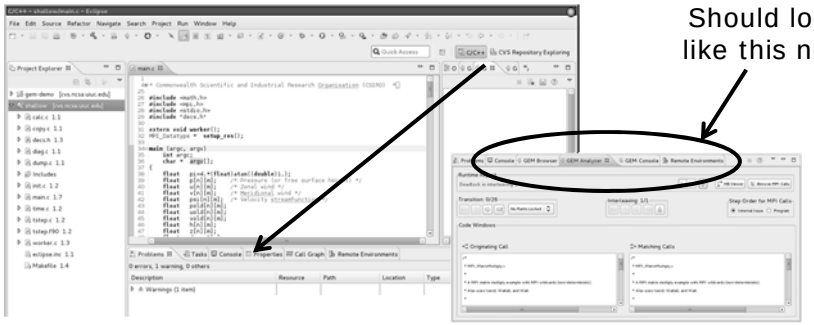
- In the **GEM Analyzer View** (in focus by default), notice there was a deadlock detected. No subsequent errors or warning will be flagged until this is fixed



- We can localize and fix this issue using:
  - **GEM Analyzer View** or the **GEM Browser View**
  - In this example, we will use the **Analyzer View** and the transition navigation buttons

* *FIX GEM VIEWS* *

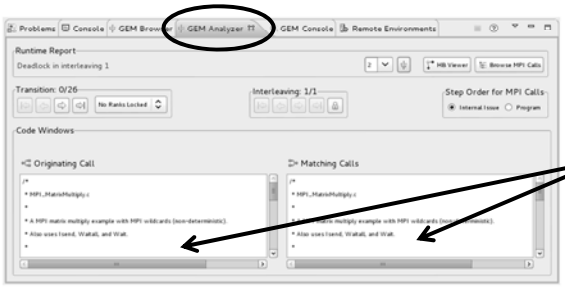
(NOTE: you may not need to do this)



Simply drag GEM views to position them underneath the Eclipse Editor

GEM Hands-on (6)

- Double Click on the **Gem Analyzer View** tab
  - This makes the view fill the workbench window
- Fix the deadlock using the **GEM Analyzer View**

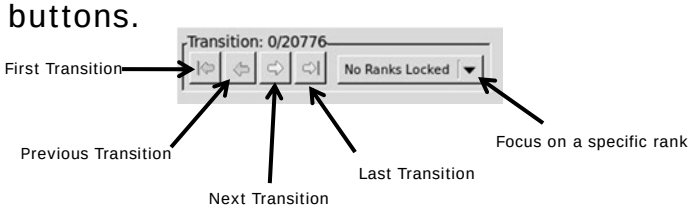


- The GEM Analyzer View should look similar to this figure
- **MPI_MatrixMultiply.c** is open in both code view windows. This is where we will trace execution as and MPI calls (P2P and collective matches) can be examined as they happened at runtime

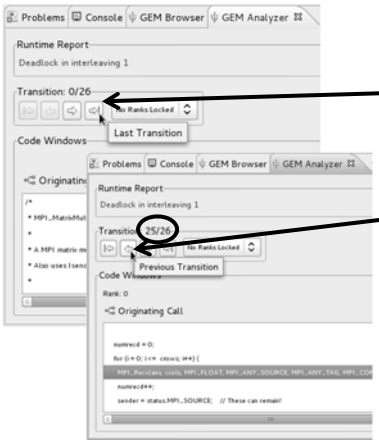
GEM Hands-on (7)

- Under **Step Order for MPI Calls**, select **Internal Issue Order**. This is the order in which GEM issues the intercepted MPI calls to the MPI runtime

- Using the **Transition Group**, we will step through the single interleaving discovered by GEM using the navigation buttons.



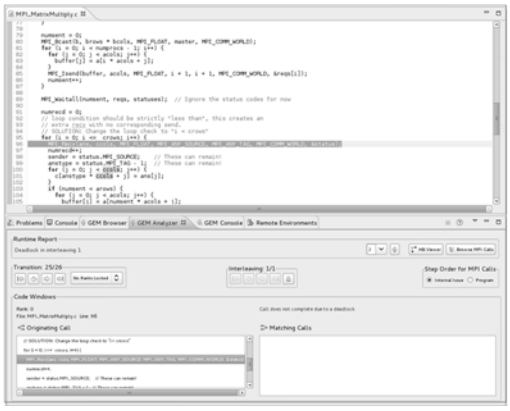
GEM Hands-on (8)



- *Using navigation buttons:*
- Navigate to last transition using **last transition** button
- Then single step backward to transition 25/26 using **previous transition** button
- Looking at **24/25**, we see it's the last call to actually complete before deadlocking, so **25/26** is the problem

We are essentially looking at calls as issued to the MPI runtime and reverse debugging to point of deadlock

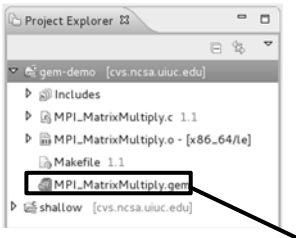
GEM Hands-on (9)



- Double click on the highlighted recv (25/26).
- This takes you to line in the Eclipse editor
- **Notice the comment:** this loop creates an extra recv that will never have a matching send
- Fix as suggested in the comments

GEM Hands-on (10)

- Rebuild the gem-demo project



- Right click on:
  MPI_MatrixMultiply.gem
- Select:
  - GEM -> Formally Verify Profiled MPI Program

Finally, wait for analysis. This may take several seconds and you will see output in the **GEM Console View**

GEM Hands-on (11)

- Notice again, the three GEM views:
  - GEM Analyzer, GEM Browser, GEM Console

- In the **GEM Analyzer View** (in focus by default), notice there were no MPI errors, e.g. deadlock, assertion violation, but warnings were issued

- Examine these warnings in the **GEM Browser View**

GEM Hands-on (12)

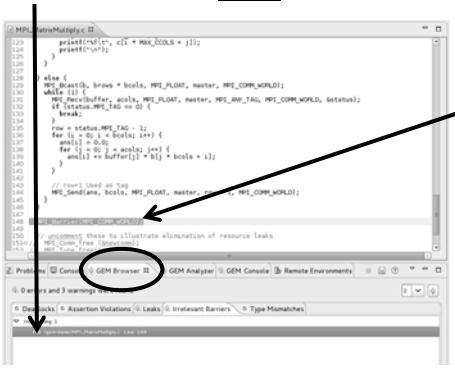
- Click on the tab for the **GEM Browser View**

- Notice GEM has found two types of warnings:
  - Functionally Irrelevant Barriers (unnecessary synchronization)
  - Resource Leaks (allocated MPI data types and communicators not freed)

- GEM maps these warnings (and errors) to the offending line of source code within the Eclipse editor

GEM Hands-on (13)

- Click on **Irrelevant Barriers** tab
- Expand the **Interleaving 1** tree
- Click on the **FIB** line



- Notice you are taken to the corresponding line in the Eclipse Editor
- This **MPI_Barrier()** call is "functionally irrelevant" and can be safely removed without changing program behavior
- We'll investigate this further in the **exercise section** at the end of this section

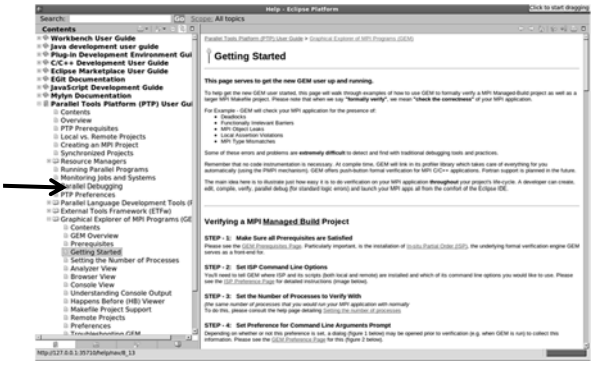
GEM Hands-on (14)

NOTE

Fixing these warnings is left as an exercise at the end of this section

GEM Hands-on (15)

- Briefly examine the GEM help plugin
- GEM Help also walks through the Managed-Build, PI-C example created by the new MPI project wizard
  - Help -> Help Contents -> PTP -> GEM



Reference Slides

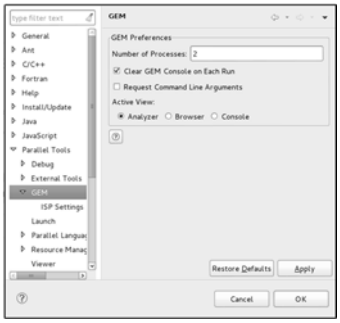
- **NOTE:** The following slides are not part of the presentation or hands-on section
- They are meant to be used as further reference and to provide more detailed information on:
  - GEM setup, configuration and preference pages
  - Using GEM views
  - Help contribution
  - Troubleshooting
  - GEM Success stories

ISP Installation

- **ISP itself must be installed prior to using GEM**
  - Download ISP at http://www.cs.utah.edu/fv/ISP
- Make sure libtool, automake and autoconf are installed.
- Just untar isp-0.3.1.tar.gz into a tmp directory:
  - Configure and install
    - ./configure –prefix=/my/preferred/install/location
    - make
    - make install
      - This installs binaries and necessary scripts

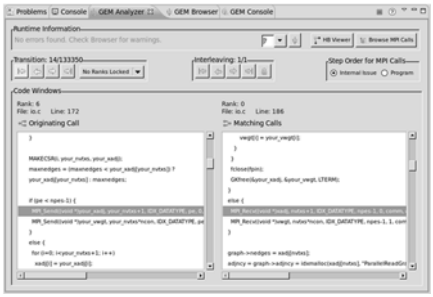
GEM Preferences

Set preferences for GEM and its underlying verification tool, ISP



GEM Analyzer View

- Reports program errors, and runtime statistics
- Debug-style source code stepping of interleavings
  - Point-to-point / Collective Operation matches
  - Internal Issue Order / Program Order views
  - Rank Lock feature – focus on a particular process
- Also controls:
  - Call Browser
  - Happens Before Viewer launch
  - Re-launching of GEM

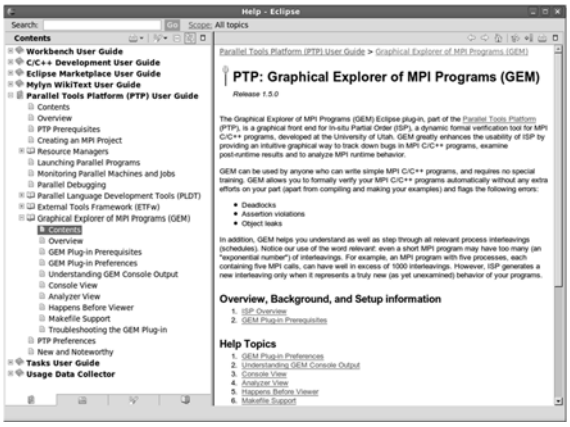


GEM Browser View

- Tabbed browsing for each type of MPI error/warning
- Each error/warning mapped to offending line of source code in Eclipse editor
- One click to visit the Eclipse editor, to examine:
  - Calls involved in deadlock
  - Irrelevant barriers
  - MPI Object Leaks sites
  - MPI type mismatches
  - Local Assertion Violations

GEM – Help Plugin

Extensive how-to sections, graphical aids and trouble shooting section



GEM/ISP Success Stories

- Umpire Tests
  - http://www.cs.utah.edu/fv/ISP-Tests
  - Documents bugs missed by tests, caught by ISP
- MADRE (EuroPVM/MPI 2007)
  - Previously documented deadlock detected
- N-Body Simulation Code
  - Previously unknown resource leak caught during EuroPVM/MPI 2009 tutorial !
- Large Case Studies
  - ParMETIS, MPI-BLAST, IRS (Sequoia Benchmark), and a few SPEC-MPI benchmarks could be handled
- Full Tutorial including LiveDVD ISO available
  - Visit http://www.cs.utah.edu/fv/GEM

Exercise (1) – Fix MPI Warnings

1. Comment out the MPI_Barrier call on line 152 of **MPI_MatrixMultiply.c**
2. Uncomment lines 155 & 156 of the same source file. These last two lines are the corresponding “free” for the MPI communicator that was created and also the MPI datatype that was committed
3. Build the GEM project again to incorporate these changes into the profiled executable
4. Re-run GEM and check that there are in fact no errors or warnings

Exercise (2) – Multiple Interleavings

1. Re-run GEM on the gem-demo project using "3" processes
2. How many Interleavings does GEM report?
3. Use the Interleaving navigation buttons to explore this. You can reference GEM help for the documentation on Interleaving navigation buttons, or simply click the help icon in the **GEM Analyzer View** to quickly navigate to specific help with the **GEM Analyzer View**

Gcov and gprof support in linux tools

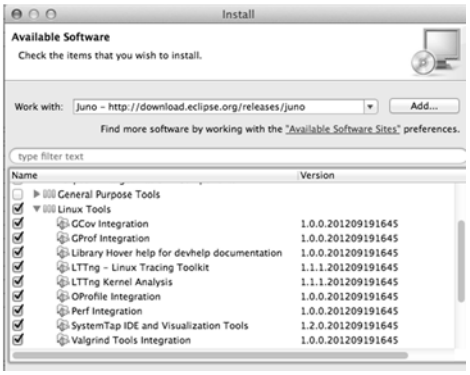
- Objective
  - Learn how to use Eclipse-based interfaces to GNU tools Gcov and Gprof
- Contents
  - Build with "-pg" for gprof profiling
  - Build with "-ftest-coverage –fprofile-arcs" for gcov
  - Run gcov to determine code coverage – which parts of your program are logically getting exercised
  -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



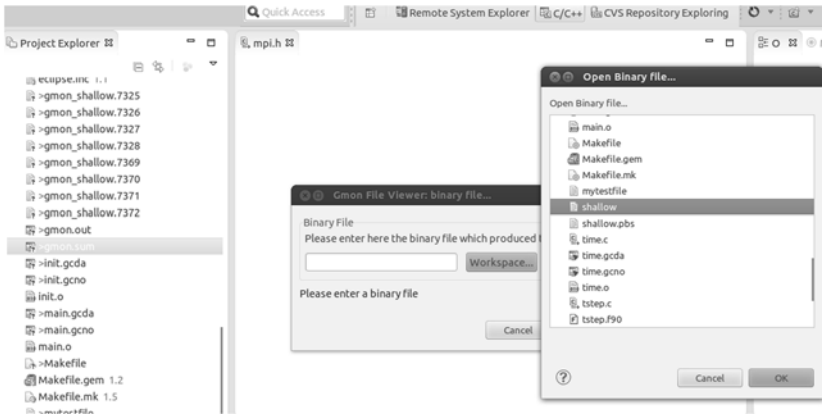
- Some of the Linux Tools are included with **Eclipse for Parallel Application Developers** package
- Everything you need for this tutorial is in the package
- To install manually:
  - Help > Install New Software
  - In **Work With:** Select Juno update site
  - Under **Linux Tools**, Select the tools 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 linuxtools 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

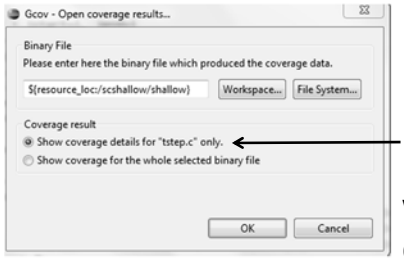
Linux Tools – click to view

- 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)

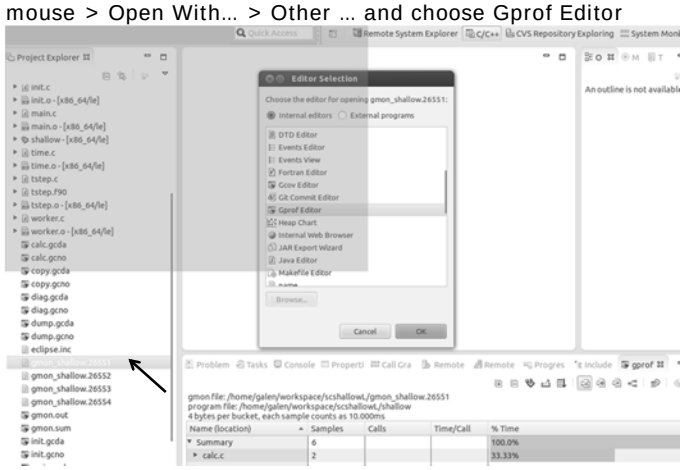


Windows only. Mac/Linux can show coverage for whole file

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

gprof

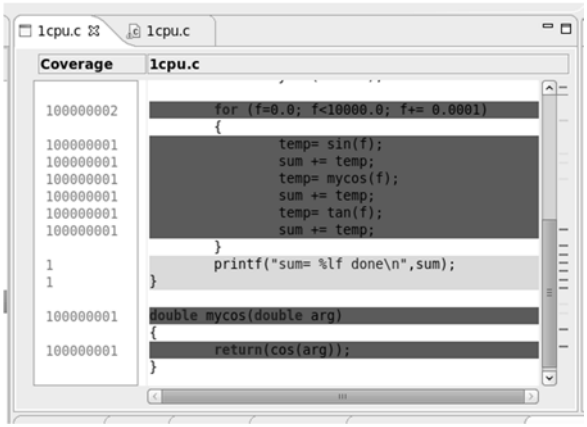
gmon file: /home/arnoldg/workspace/linux_tools_demo/gmon.out
program file: /home/arnoldg/workspace/linux_tools_demo/1cpu
4 bytes per bucket, each sample counts as 10.000ms

Name (location)	Samples	Calls	Time/Call	%Time
Summary	131			100.0%
1cpu.c	131			100.0%
main	0	0		0.0%
mycos	19	100000001	1ns	14.5%
mycos (1cpu.c:140)	5			3.82%
0x40094c	5			3.82%
mycos (1cpu.c:30)	14			10.69%
0x400930	7			5.34%
0x400938	6			4.58%
0x40093c	1			0.76%
work	112	1	1.120s	85.5%
work (1cpu.c:17)	4			3.05%
work (1cpu.c:19)	9			6.87%
work (1cpu.c:20)	36			27.48%
work (1cpu.c:21)	7			5.34%

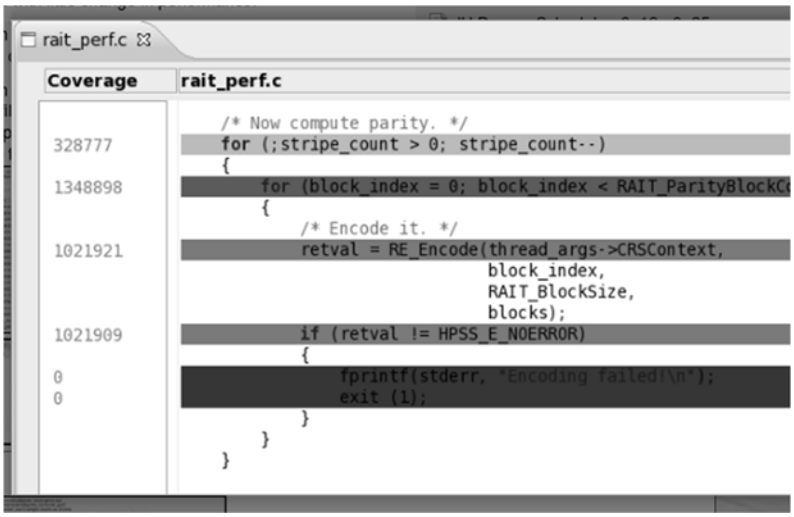
/u/ncsa/arnoldg/c/1cpu.c:140

Run code, inspect gcov display

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

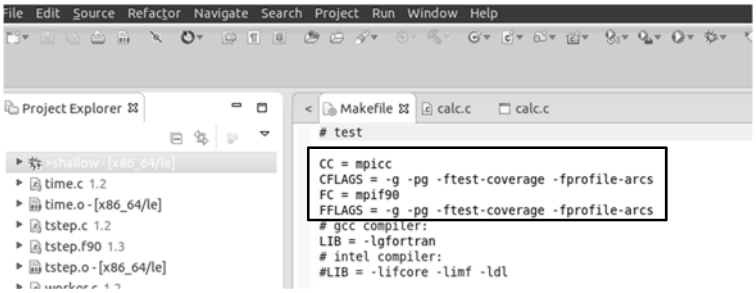


Gcov with a production code, unexecuted region



gprof with shallow project, MPI

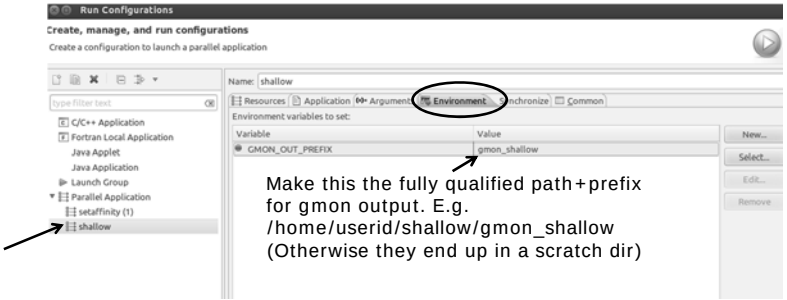
Add compiler flags to Makefile



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)

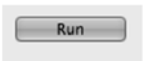
Modify Run Configuration
Run > Run Configurations … to modify existing Run Configuration



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



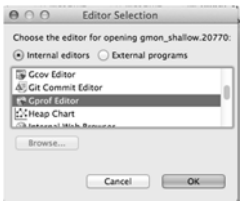
- See build results in Console
- See new files in Project Explorer (You may need to force a Sync)

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**

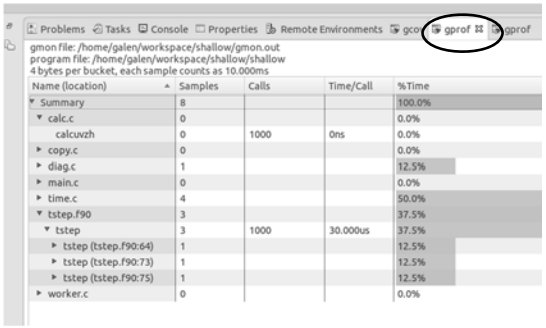


gprof with shallow project, 1 rank (2)

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.



Gprof with shallow project, summary

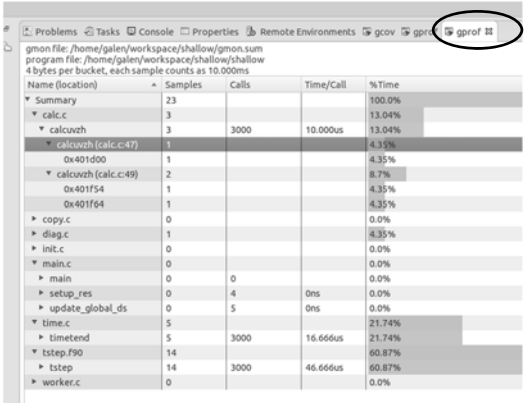
Aggregate the gmon output – invoke from a terminal:

```
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

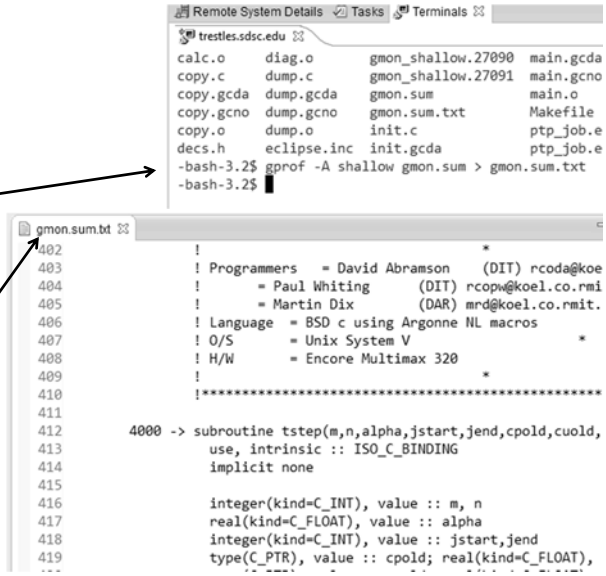


Gprof viewer does not currently work well on Windows

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

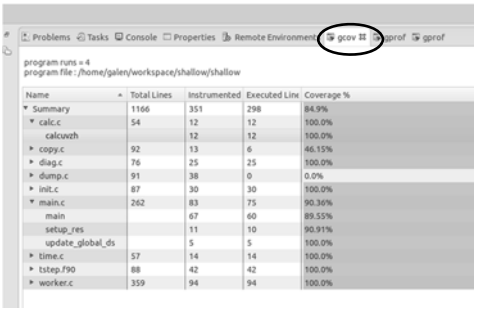
The linuxtools team knows about the issue with Juno and they're working on it.



Gcov with shallow project

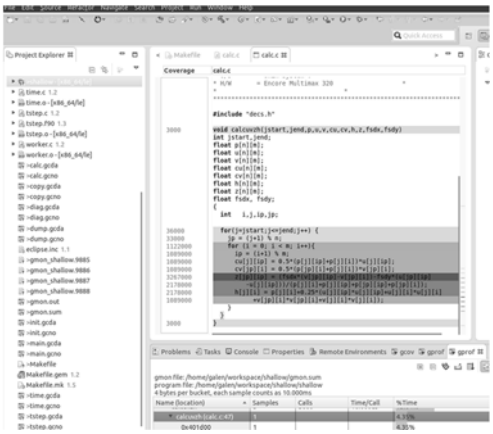
The gcov view is simlar to the gprof view but keep in mind that you're looking at code coverage and not necessarily performance or timing information (though there is a relationship...code not executed is performing quite well !). Also note that multiple executions will accumulate values in the gcov 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



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)?

WRF & Eclipse

- Objective
  - To learn how to work with the Weather Research and Forecasting (WRF) code with Eclipse
- Contents
  - Import WRF from local directory and make into synchronized project
  - Checkout WRF from CVS repository, and make into synchronized project
  - Interactive configure script execution
  - Driving makefile from Eclipse

WRF Scenarios

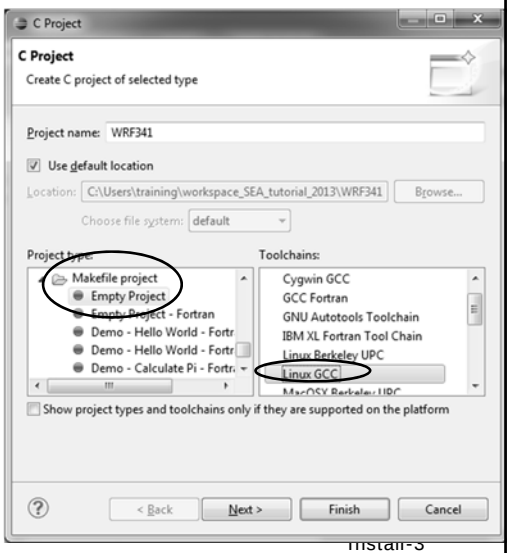
- Case A: User downloads tarball from WRF download site
- Case B: User checks out WRF from source repository
- Quirks
  - WRF uses a user-interactive configure script to get things going
  - But, can drive make after configure script is initially run

Things to watch for

- Beware of “helpful” archive programs deciding what is ascii, binary when extracting WRF from its archive
  - Especially on windows
- Execute bits do not seem to be set on scripts
  - Trying to understand the source of this issue
- One scenario not shown (but very possible) is to move the archive to the remote machine, extract, and then synchronize the source from the remote machine
- Binary filters for synchronization could be very important

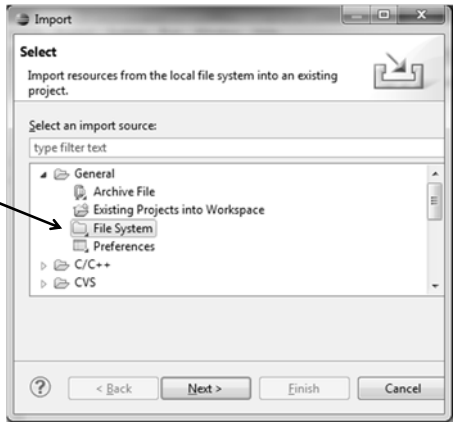
Case A: WRF on local filesystem

- Assuming you have the source download unpacked on your local filesystem…
- Create new, empty project
- File>New>C Project
  - Makefile project, empty project
  - Linux GCC
  - Project name
- Finish



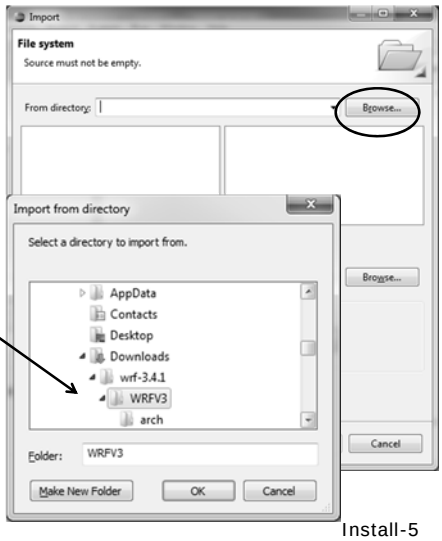
Import WRF

- Import WRF from local filesystem
  - Right click on project name, Import
  - General, Filesystem



Filesystem Browser

- Select Browse to look at your local filesystem
- Navigate to source directory

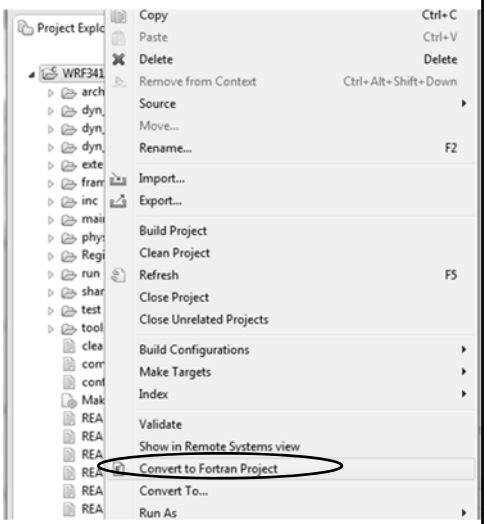


Select source directory

- Selecting directory automatically selects entire contents of directory

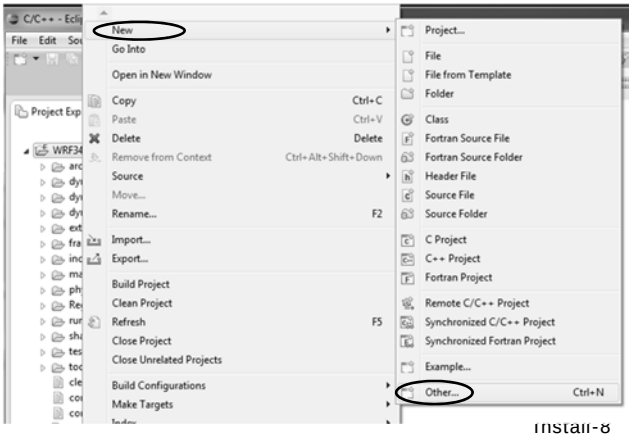
Add Fortran Flavor to project

- Fortran projects are derivatives of C/C++ projects
- Can add Fortran Flavor easily
  - Right click on project, select “Convert to Fortran Project”



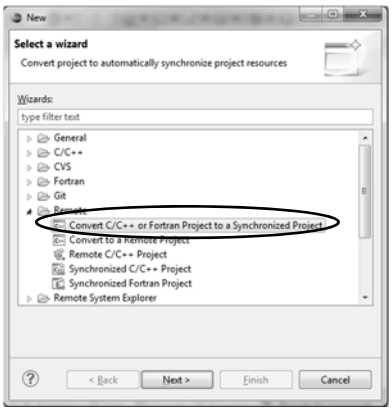
Convert to a synchronized project

- Right click on project name, New>Other



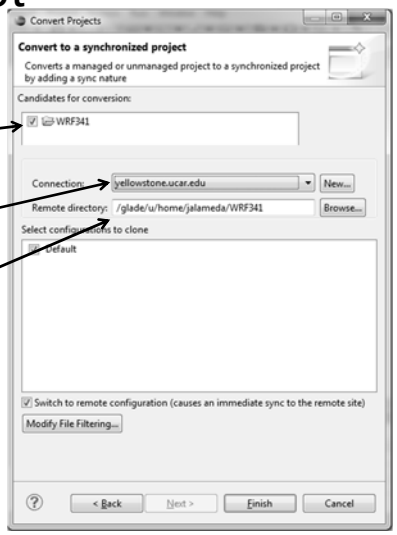
Select conversion wizard…

- Open Remote, then "Convert C/C++ or Fortran Project to a Synchronized Project"



Converting to synchronized project

- Can only convert one project at a time
  - New project is eligible
- Select open connection to yellowstone
- Destination directory should autofill
- Select Finish



Case A complete

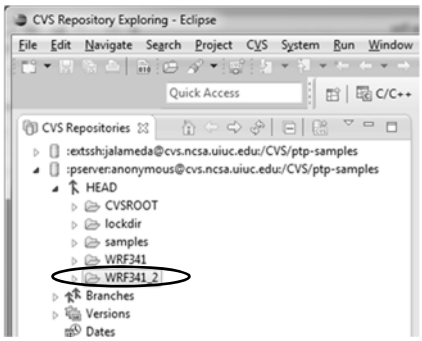
- We have successfully imported source from our filesystem, and pushed it to yellowstone
- Next, we'll try checking out the code from CVS

Case B: Checking WRF out from CVS

- Notes
  - I checked WRF into CVS from Eclipse
    - If you check in WRF in from a synchronized project, cannot “remake” the checked out code into a synchronized project
  - Hence, the code in the repository is not from a synchronized project
    - But it is an Eclipse project, which changes things a bit from the earlier tutorial exercises

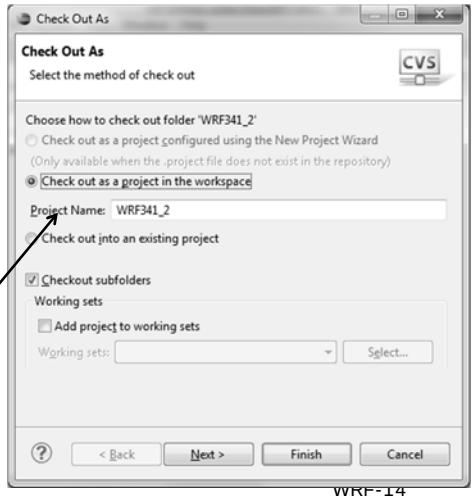
Back to CVS repository exploring

- Change to the CVS Repository Exploring Perspective
- Using previous pserver connection, Open Head, select WRF341_2



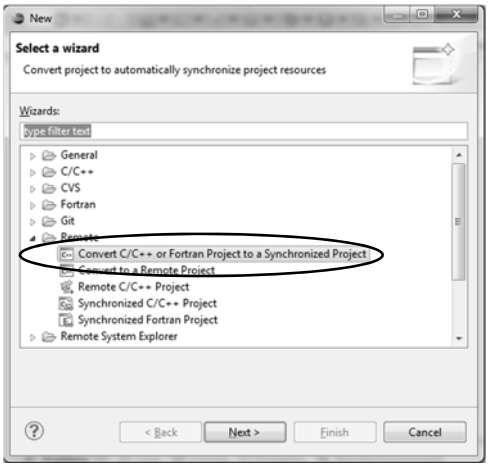
Check out as

- Right click on WRF341_2, select "Check out as…"
  - Note that since this project is already an Eclipse project, you can only just check out the project
  - Can rename on the way (change project name)



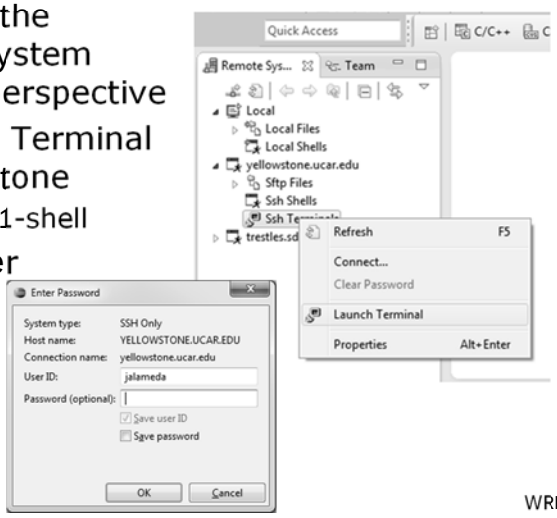
Convert to synchronized project

- File>New>Other>
- Remote, "Convert C/C++ or Fortran Project to a Synchronized Project"



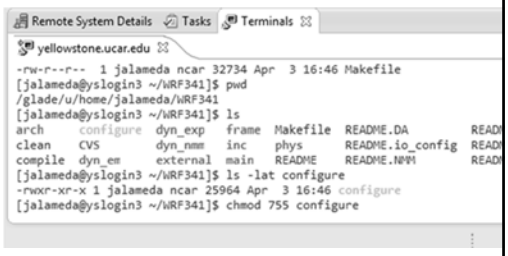
Interactive Configure

- Switch to the Remote System Explorer perspective
- Open SSH Terminal to yellowstone
  - Ptp-03-01-shell
- Don't enter password
- Yubikey...



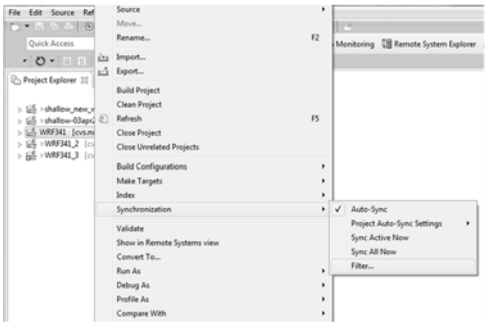
Check permission of configure

- Cd to WRF directory on yellowstone
- Chmod if necessary:
- Run configure, answering questions appropriately
- After configure, you will have a Makefile that you can configure Eclipse to drive

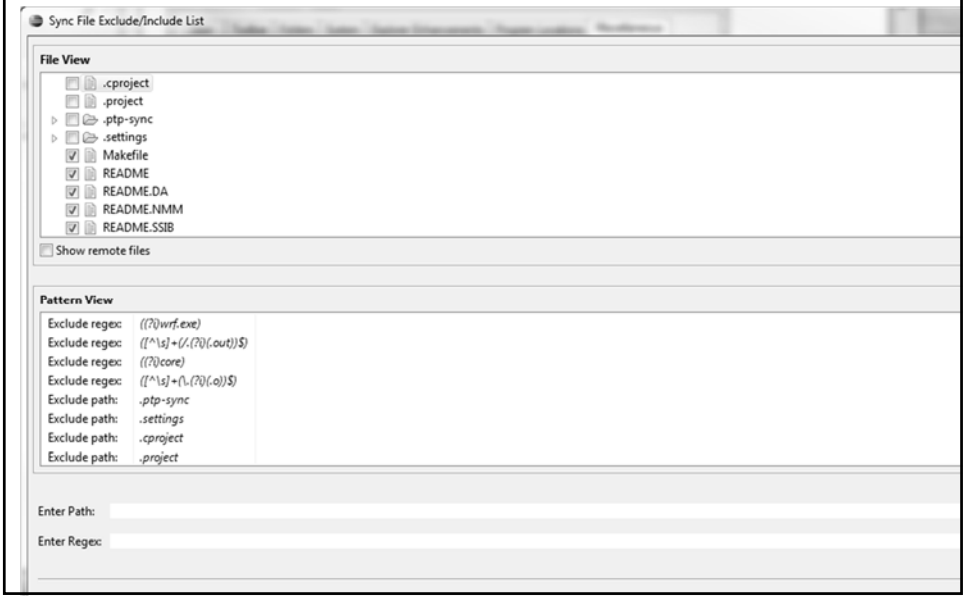


Synchronization Filtering

- Git (used in synchronized projects) does not like large, binary files
- Suggest to establish filters on the project, to prevent performance issues
- Right click on project, Syncronize, Filter



Filters using java regex



Java Regex reference

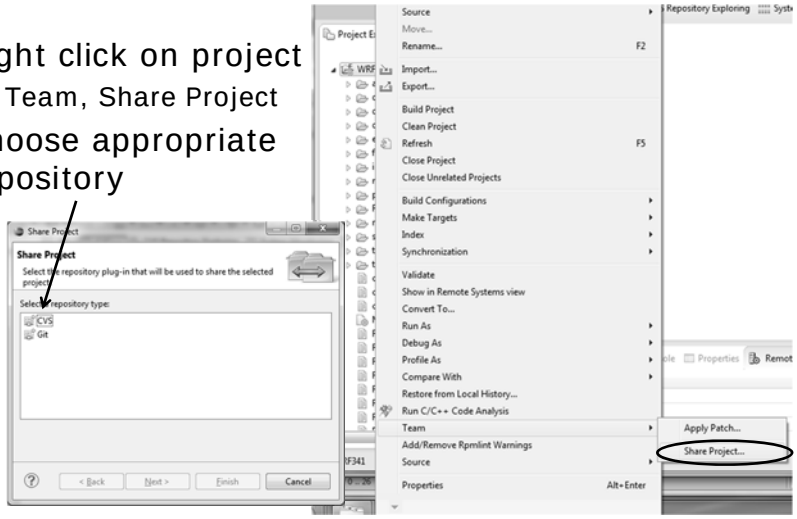
- If not familiar with Java regex, this helped me:
- http://www.mkyong.com/regular-expressions/10-java-regular-expression-examples-you-should-know/

Exercise

- Create a WRF synchronized project
- Set up some synchronization filters for your synchonized project
- Run configure script
- Make WRF from Eclipse
- Make run configuration for WRF

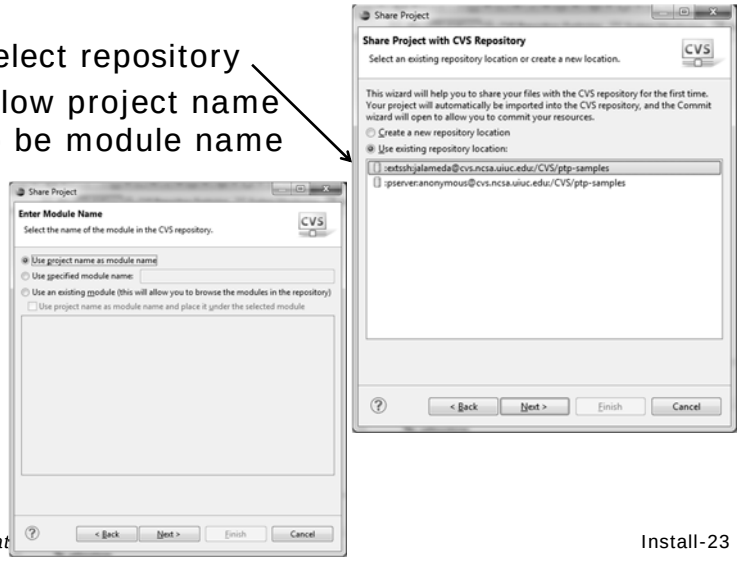
Sharing source code via CVS

- Right click on project
  - Team, Share Project
- Choose appropriate repository



Select appropriate repository

- Select repository
- Allow project name to be module name

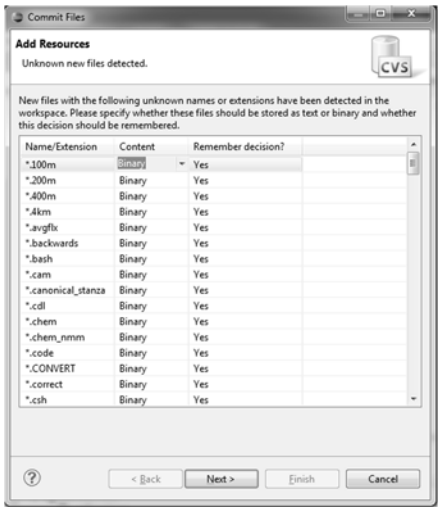


Share resources

- And launch the commit wizard

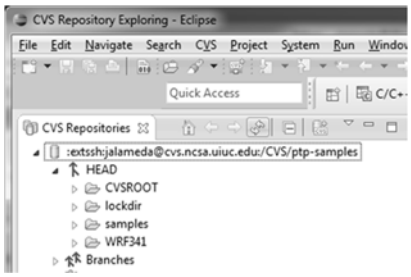
File types

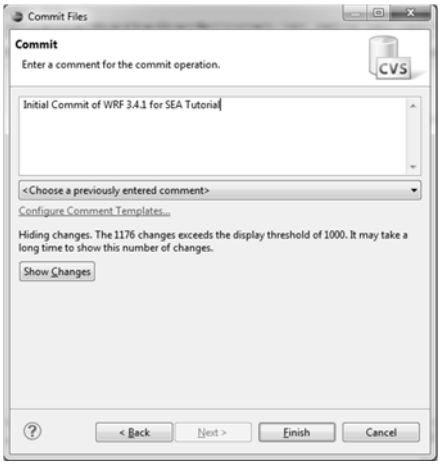
- Note that can classify unknown file types (and make persistent)



Add commit comment

- Commit comment
- Finish, and the commit is done





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

- PTP Tutorial @ XSEDE13
  - July 22, 2013; San Diego, CA
  - https://www.xsede.org/web/xsede13
- Please complete feedback forms
  - Our paper feedback form (last 2 pages of handout)
- Your feedback is valuable!

Thanks for attending
We hope you found it useful

SEA2013: April 4-5, 2013 A Unified Environment for the Development and Performance Tuning of Parallel Scientific Applications
Tutorial Feedback

Thank you for attending our tutorial/workshop. As a part of an on-going effort to improve the utility and usability of the tools we are developing, we are asking attendees to answer the following questions. Note that all answers are anonymous. *Comments are welcome anywhere.*

1. Eclipse - how useful does Eclipse in general seem to you? (CIRCLE ONE)

1	2	3	4	5
very useful		Neutral		very useless

2. PTP: How useful does the functionality represented by this tool seem to you?

1	2	3	4	5
very useful		Neutral		very useless

Would you expect to increase your productivity by using this tool? If so, how much of an increase in productivity do you think you may achieve by using this? Why?

_____

3. Usability: How **usable** do you think the tools are with their current interface?

1	2	3	4	5
very usable		Neutral		very **un**usable

Was there anything in particular that you found confusing or counterintuitive? If so, what?

_____

Do you have any suggestions for improving the **user interface** or additional functionality? Please describe:

_____

4. Other tools?

What other tools would you like to see developed that would have a large impact on *your own productivity?*

_____

What other tools would you like to see developed that would have a large impact on the *performance of the applications* you work on?

_____

5. Tutorial presentation

How useful was the **content** of the workshop to your current position?

1	2	3	4	5
very useful		neutral		very useless

How well **presented** was the information?

1	2	3	4	5
excellent	good	satisfactory	not so good	unacceptable

How well did the **instructors** handle questions?

1	2	3	4	5
excellent	good	satisfactory	not so good	unacceptable

SEA2013: April 4-5, 2013 A Unified Environment for the Development and Performance Tuning of Parallel Scientific Applications
Tutorial Feedback

Is there other material you would have liked to have had presented? If so, what?

_____

Is there material you think should be deleted? If so, what?

_____

6. Your Background:

How many years experience do you have in the IT industry? _____ In HPC? _____

List languages in which you have written significant programs: _____
Which is predominant currently? Why?

_____

Do you currently use an IDE (Integrated Development Environment, e.g. Eclipse)? If not, why?

If you've used Eclipse in the past, what has it helped you with? Has it helped you accomplish any science results?

If you have not used Eclipse in the past, what do you see that it could help you with? Do you think it may help you achieve any new science results faster, perhaps by helping make your development more efficient?

_____
_____

What is your current job role?

_____

Can you describe the major application(s) you are working on now?

_____

Can you describe what you see as the major bottlenecks to increased productivity where you work?

_____
_____

Are there any other relevant comments you want to share?

_____

(Optional) Name: _____Email:_____

Thank you for your feedback!