



parallel tools platform

<http://eclipse.org/ptp>

A New and Improved
Eclipse Parallel Tools Platform

Advancing the Development of Scientific Applications

Jay Alameda, NCSA
alameda@illinois.edu

Jeff Overbey, NCSA
overbey2@illinois.edu

 **UCAR** Software Engineering Assembly
University Corporation for Atmospheric Research

February 24, 2012

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

Based on slides by
Greg Watson, Beth Tibbitts, and others

Tutorial Outline

Time (Tentative)	Module	Topics
8:30-9:00	1. Eclipse & PTP Installation	✦ Installation of Eclipse and PTP (can start early as people arrive)
9:00-9:30	2. Introduction & Overview	✦ Eclipse architecture & organization overview
9:30-10:30	3. Developing with Eclipse	✦ Eclipse basics; Creating a new project from CVS; Local, remote, and synchronized projects ✦ Editing C files; MPI Features; Building w/ Makefile
10:30-10:45	BREAK	
10:45-11:45	3. Developing with Eclipse (continued)	Continue from before the break... ✦ Resource Managers and launching a parallel app ✦ Fortran, Refactoring, other Advanced Features
11:45-12:00	4. Wrap-up	✦ NCSA HPC Workbench, Other Tools, website, mailing lists, future features

About the Tutorial Installation

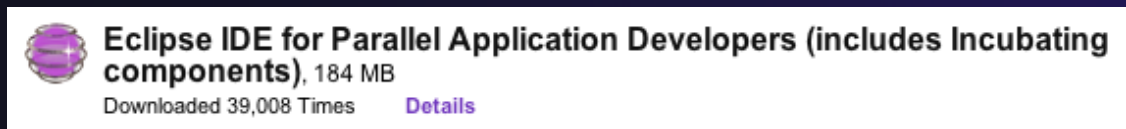
- ★ This tutorial assumes you have Eclipse and PTP pre-installed on your laptop
- ★ If you already have Eclipse installed, go directly to “Starting Eclipse”, slide 5
- ★ If you don’t have Eclipse installed, you will need to follow the handouts so that you can catch up with the rest of the class
- ★ Note: up-to-date info on installing PTP and its pre-reqs is available from the release notes:
 - ★ http://wiki.eclipse.org/PTP/release_notes/5.0
 - ★ This information may supersede these slides

System Prerequisites

- ★ Local system (running Eclipse)
 - ★ Linux (just about any version)
 - ★ Mac OS X (10.5/Leopard or later)
 - ★ Windows (XP or later)
- ★ 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 Java SE 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 (3.7) is also known as “Indigo”
- ✦ Eclipse is available in a number of different packages for different kinds of development
 - ✦ <http://eclipse.org/downloads>
- ✦ With Indigo, there is a new package directly relevant for HPC:
 - ✦ Eclipse IDE for Parallel Application Developers
 - ✦ **This is recommended for all new installs**



“Parallel Package”

- ✦ Can also add PTP to an existing Eclipse installation



Eclipse Installation

- ✦ Download the “Eclipse IDE for Parallel Application Developers” package
 - ✦ <http://download.eclipse.org>
- ✦ Make sure you match the architecture with that of your laptop
- ✦ If your machine is Linux or Mac OS X, untar the file
 - ✦ On Mac OS X you can just double-click in the Finder
- ✦ If your machine is Windows, unzip the file
- ✦ 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 launch from a terminal window instead (like Linux)

✦ **Windows**

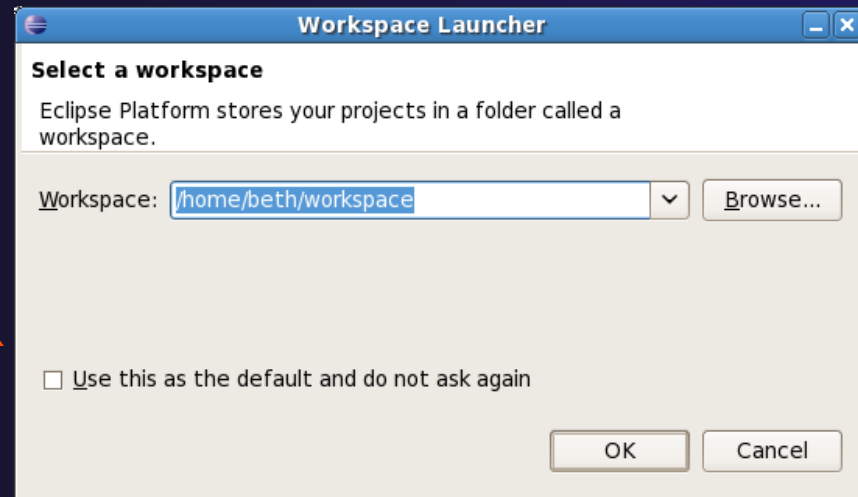
- ✦ Open the **eclipse** folder
- ✦ Double-click on the **eclipse** executable



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

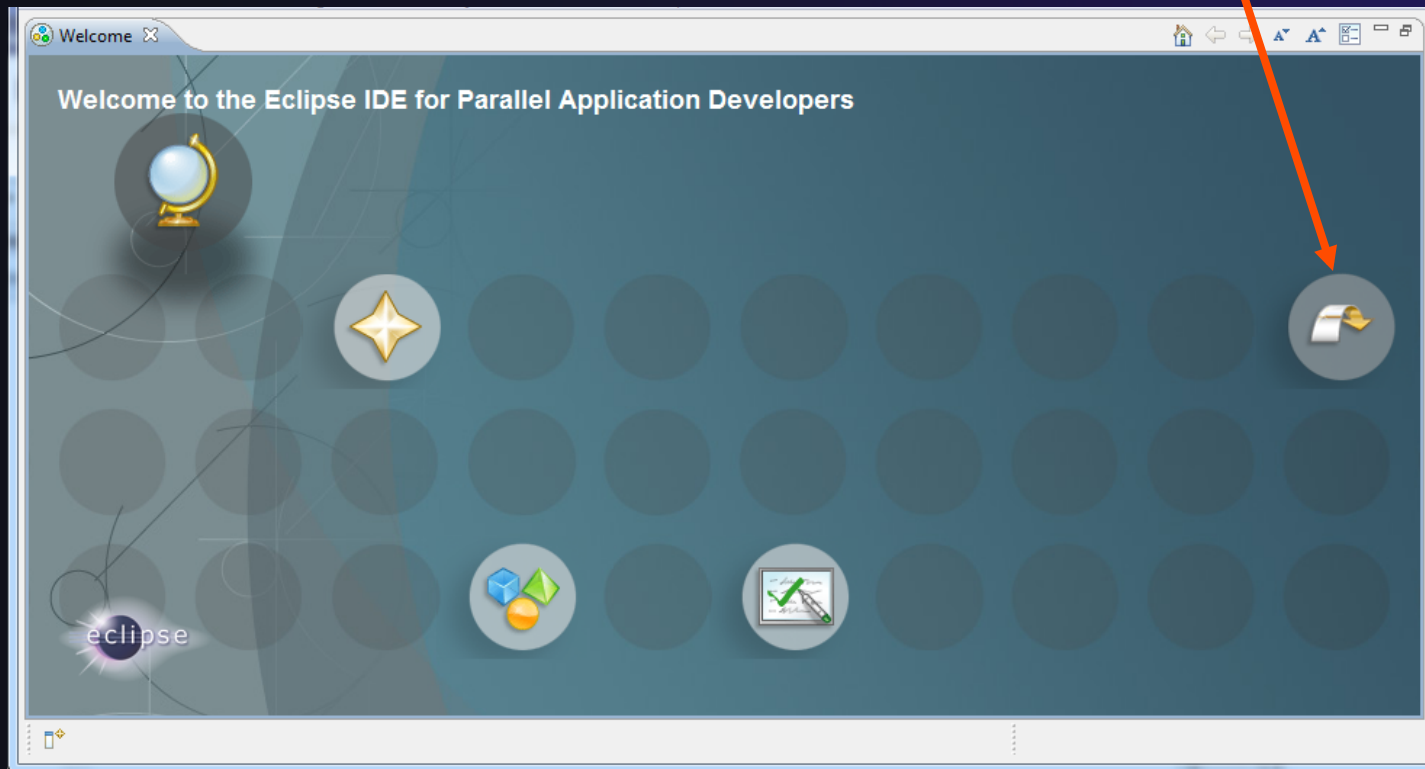
The prompt can be turned off



Eclipse Welcome Page



- ★ Displayed when Eclipse is run for the first time
- Select “Go to the workbench”

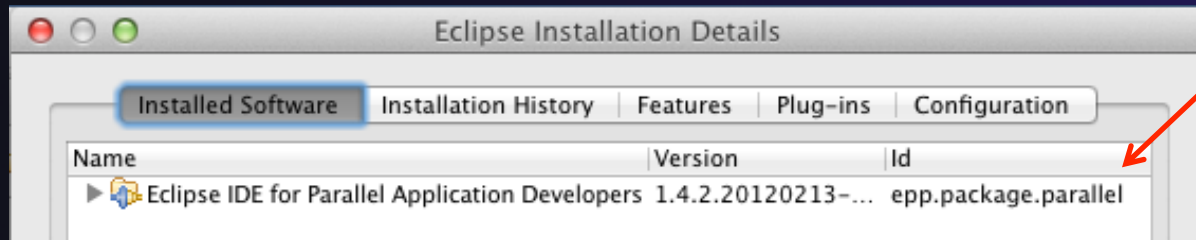




Check Installation Details

- ✦ To confirm you have installed OK
 - ✦ Mac: **Eclipse>About Eclipse**
 - ✦ Others: **Help>About**
- ✦ Choose **Installation Details**
- ✦ Confirm you have the following installed software

Differs depending on base download



- ✦ Close the dialog: **Close, OK**

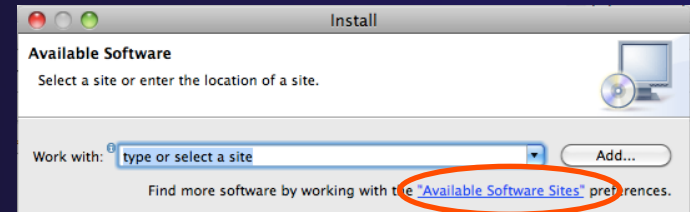
Checking for PTP Updates

- ✦ From time-to-time there may be newer PTP releases than the Indigo release
 - ✦ Indigo and “Parallel package” updates are released only in Sept and February
- ✦ PTP maintains its own update site with the most recent release
 - ✦ Bug fix releases can be more frequent than Indigo’s and what is within the parallel package
- ✦ **You must enable the PTP-specific update site before the updates will be found**



Updating PTP

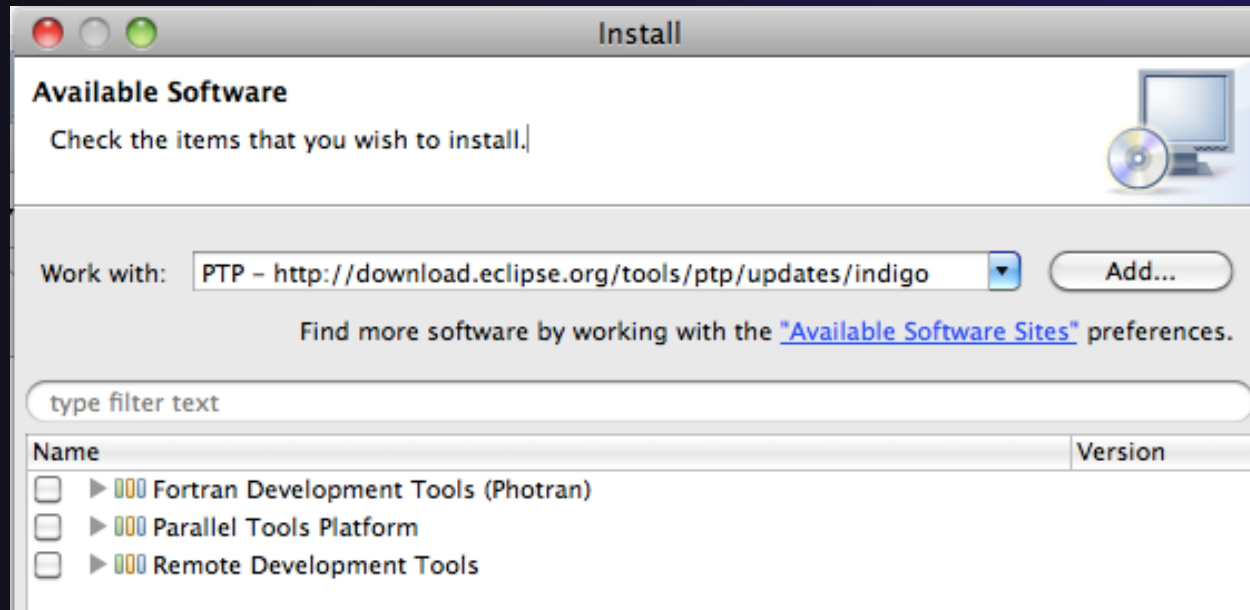
- ✦ Enable PTP-specific update site
 - ✦ **Help>Install New Software...**
 - ✦ Click **Available Software Sites** link
 - ✦ Ensure this checkbox is selected for the PTP site:
<http://download.eclipse.org/tools/ptp/updates/indigo>
 - ✦ Choose **OK**
 - ✦ Choose **Cancel** (to return to Eclipse workbench)
- ✦ Now select **Help>Check for updates**
 - ✦ If you see “No updates were found” ...
 - ✦ It's only because there are no updates in the “Eclipse IDE for Parallel Application Developers”
 - ✦ We will update the PTP within it





Updating PTP (2)

- ★ We will get the PTP release that is more recent than what is currently (Nov. 2011) within the parallel package
- ★ Now select **Help>Install New Software...**
 - ★ In the **Work With:** dropdown box, select the PTP update site you confirmed already:

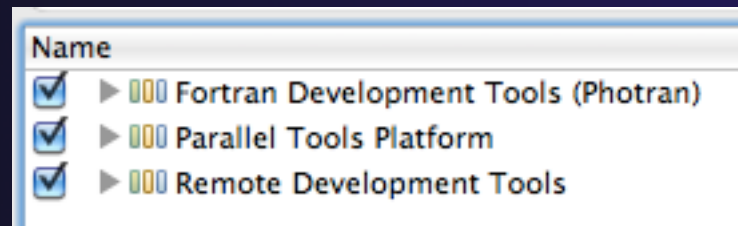




Updating PTP (3)

★ Quick and dirty:

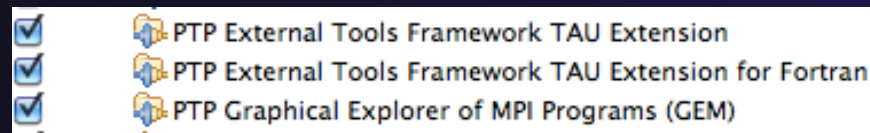
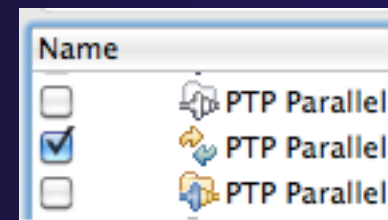
- ★ Check everything - which updates existing features and adds a few more



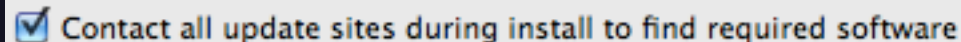
★ Detailed:

- ★ Open each feature and check the ones you want to update
- ★ Icons indicate: Grey plug: already installed and up to date
Double arrow: can be updated
Color plug: Not installed yet

Note: For this tutorial, install GEM and TAU



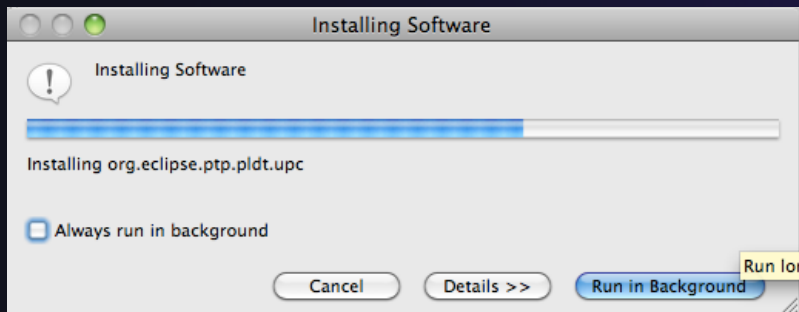
Note: if conference network is slow, consider unchecking:



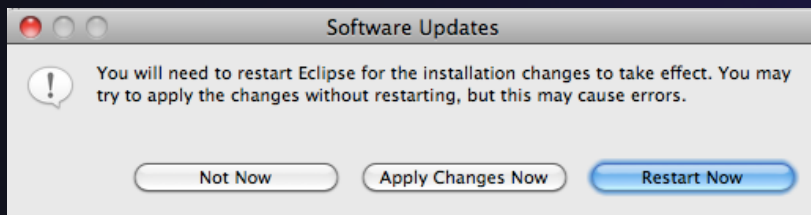


Updating PTP (4)

- ★ Select **Next** to continue updating PTP
- ★ Select **Next** to confirm features to install
- ★ Accept the License agreement and select **Finish**



- ★ Select **Restart Now** when prompted



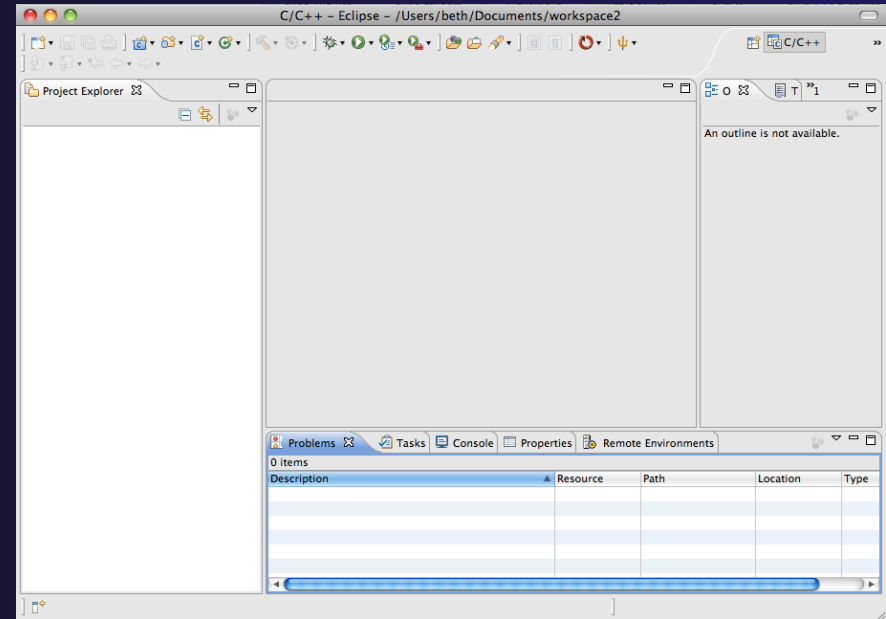
Wait for installation to finish

If conference network is too slow, we have this cached on USB



Restart after Install

- ✦ If any top-level features are installed... Welcome page informs you of new features installed
- ✦ We only updated PTP, so we land back at C/C++ Perspective
... Ready to go!



- ✦ **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

New and Improved Features

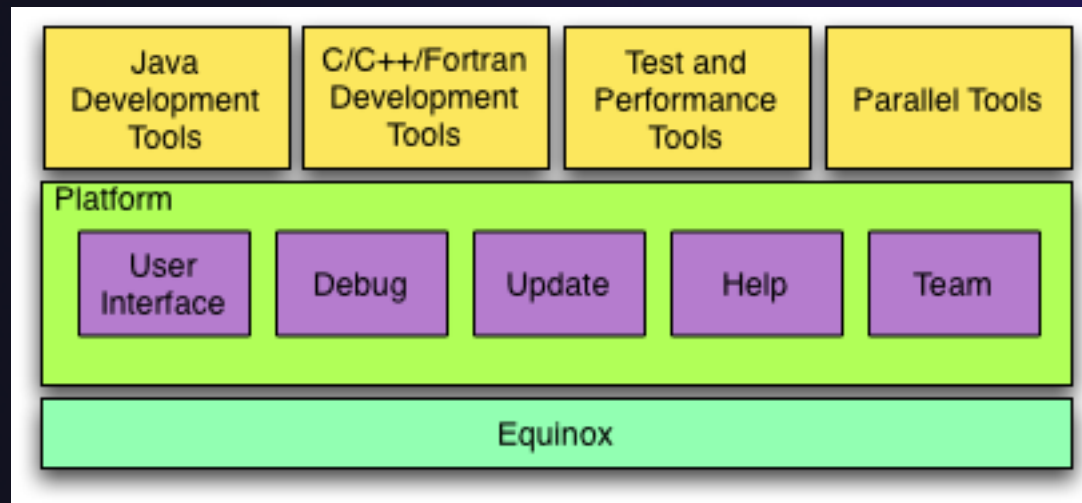
- ★ More flexible projects
 - ★ Synchronized projects overcome many problems of remote projects
 - ★ Allows development when “off-line”
 - ★ Works with non-C/C++ projects
- ★ More customizable resource managers
 - ★ Resource managers can now be added by users
 - ★ Able to have site-specific configurations
 - ★ Interactive launch using job schedulers now supported

New and Improved Features (2)

- ★ Scalable system/job monitoring
 - ★ New perspective allows monitoring of systems of virtually any size
 - ★ View shows location of jobs on cluster
 - ★ Active and inactive jobs views
- ★ Remote support for performance tools
 - ★ External Tools Framework has been extended to support remote systems
 - ★ Performance tools such as TAU can now launch and collect data from remote systems

What is Eclipse?

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

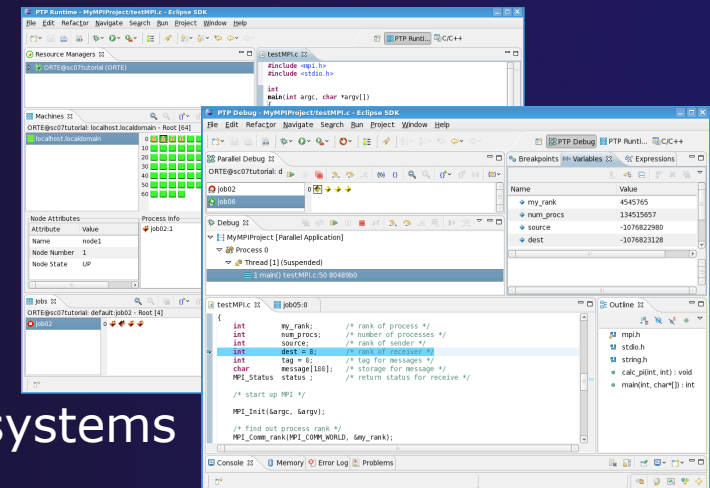


Eclipse Features

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

Parallel Tools Platform (PTP)

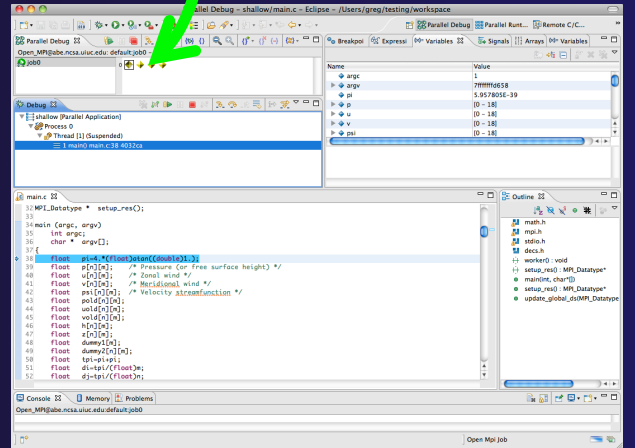
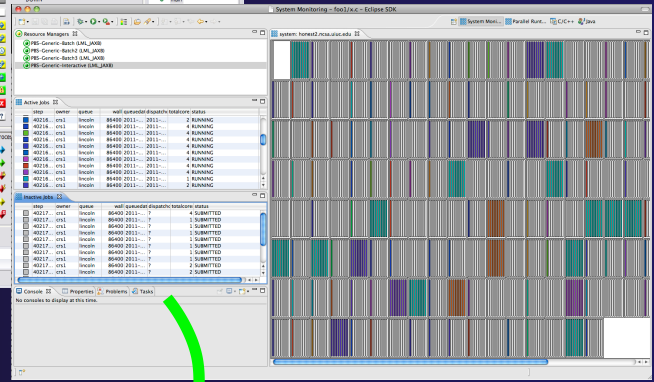
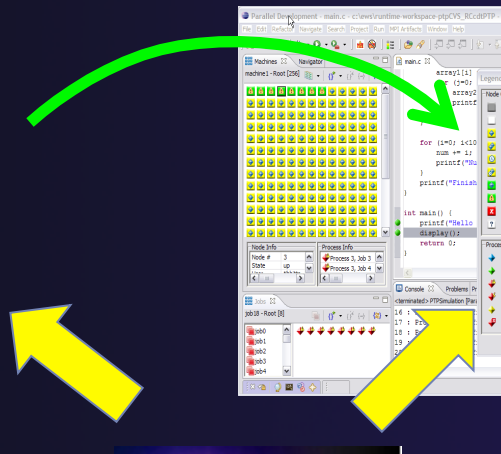
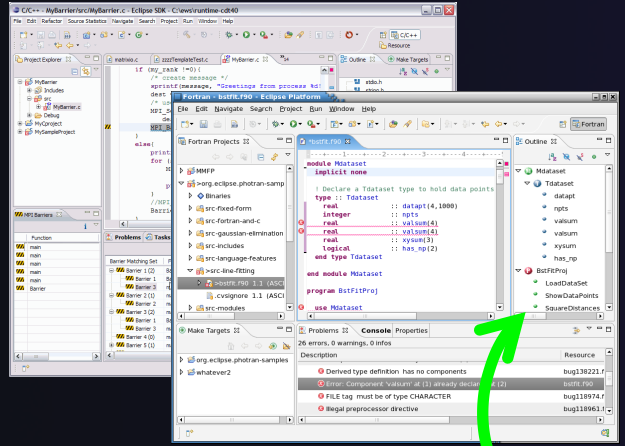
- ★ The Parallel Tools Platform aims to provide a highly integrated environment specifically designed for parallel application development
- ★ Features include:
 - ★ An integrated development environment (IDE) that supports a wide range of parallel architectures and runtime systems
 - ★ A scalable parallel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



Eclipse PTP Family of Tools

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

Launching & Monitoring



Performance Tuning
(TAU, PPW, ...)

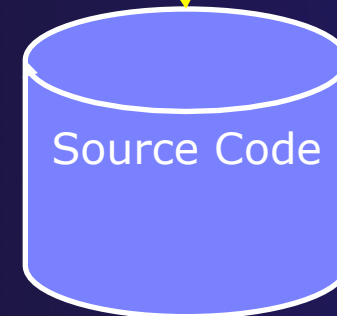
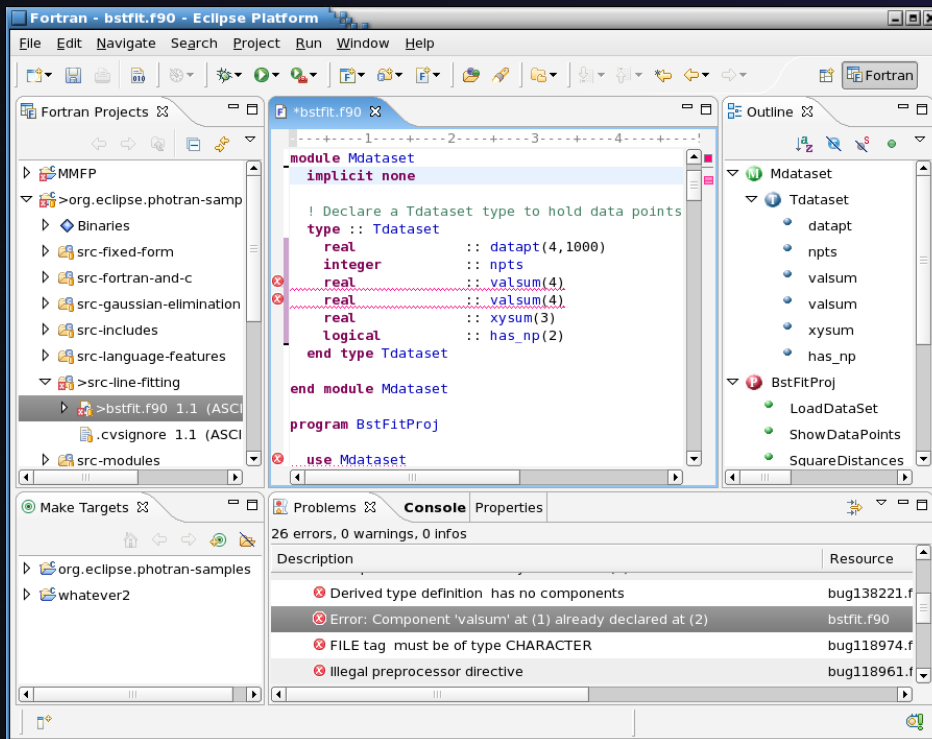
Parallel Debugging

Module 2

2-6

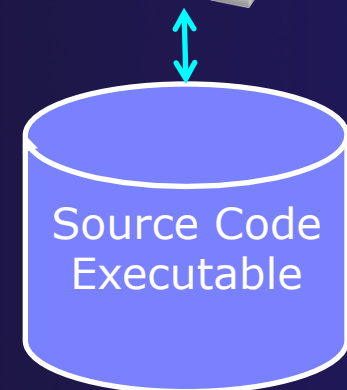
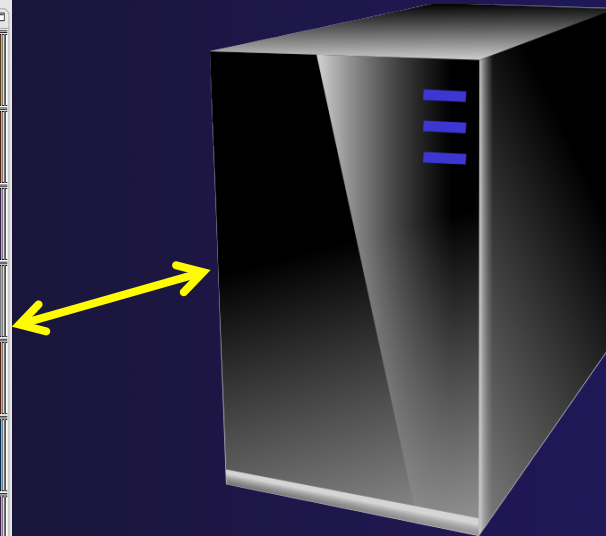
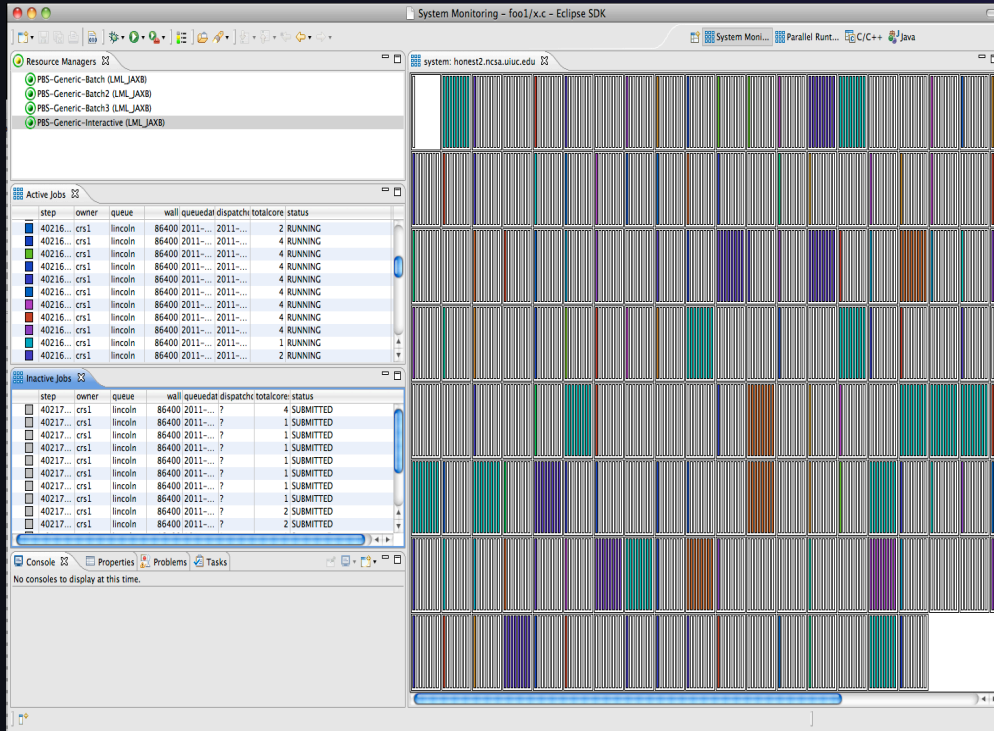
How Eclipse is Used

Editing/Compiling

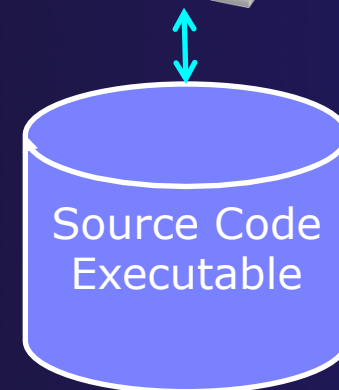
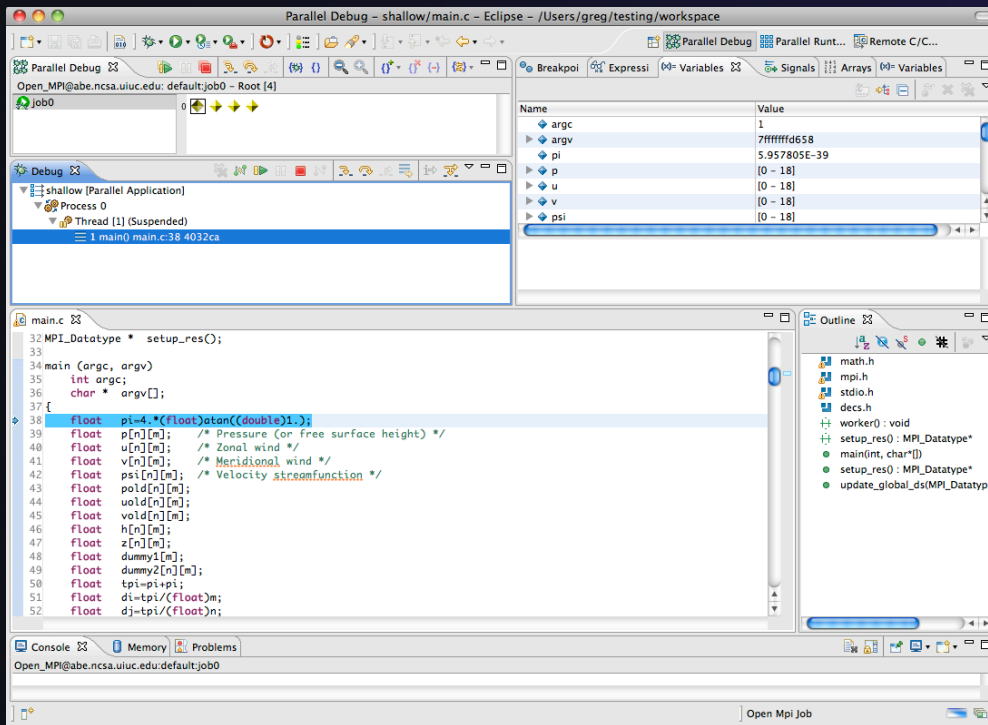


How Eclipse is Used

Launching/Monitoring

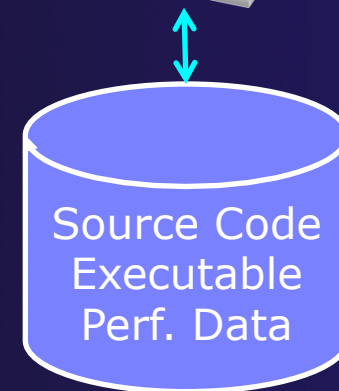
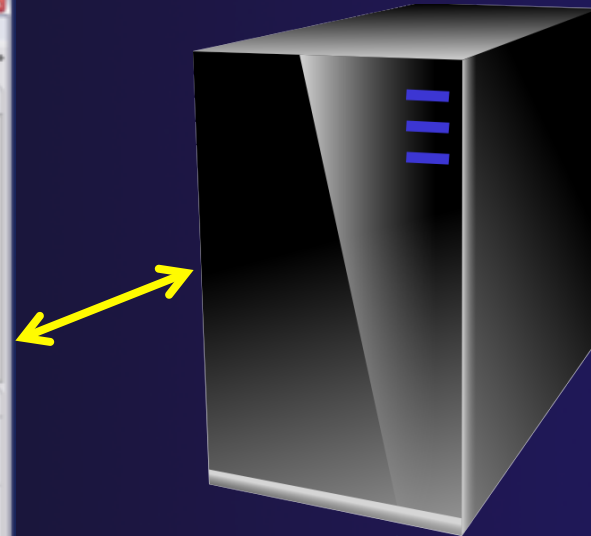
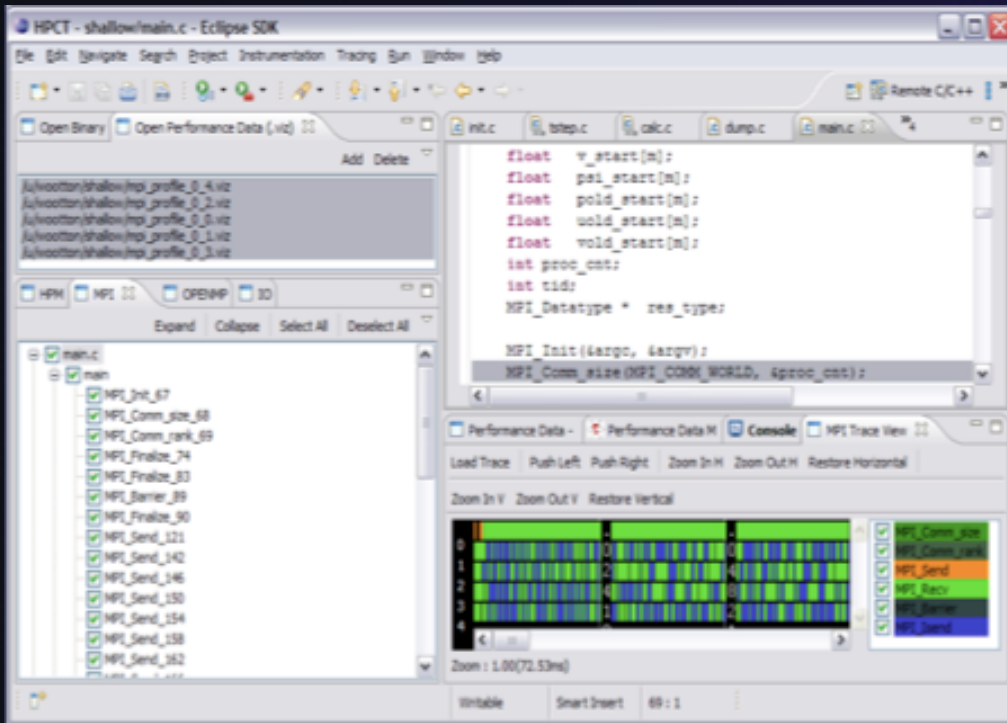


How Eclipse is Used Debugging



How Eclipse is Used

Performance Tuning



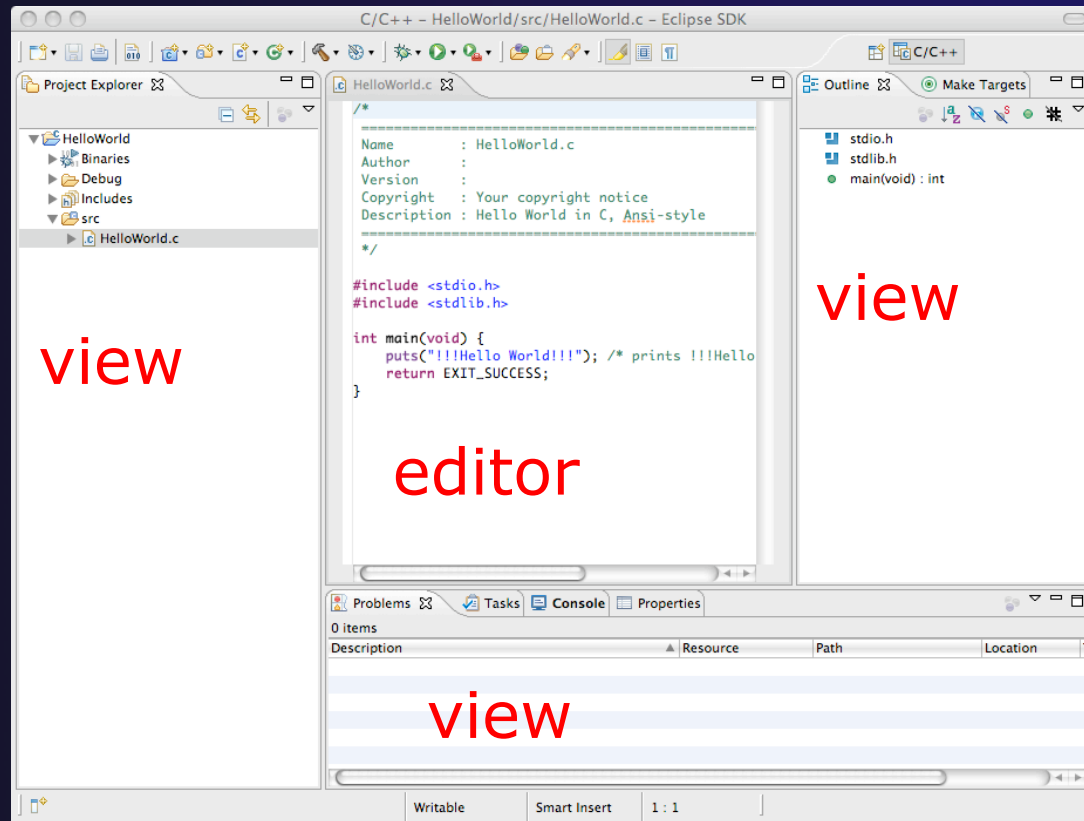
Contents

- ★ Basic Eclipse Features (3-2)
- ★ [Projects In Eclipse](#) (3-13)
- ★ [Editor Features](#) (3-24)
- ★ [Team Features](#) (3-34)
- ★ [MPI Features](#) (3-40)
- ★ [Synchronizing the Project](#) (3-56)
- ★ [Building the Project](#) (3-62)
- ★ [Running: Resource Manager Configuration](#) (3-69)
- ★ [Running: Launching a Job](#)(3-82)
- ★ [Advanced Features: Searching](#) (3-90)
- ★ [Fortran Specifics](#) (3-99)
- ★ [Advanced editing: Code Templates](#) (3-108)
- ★ [Refactoring and Transformation](#) (3-113)

Basic Eclipse Features

Eclipse Basics

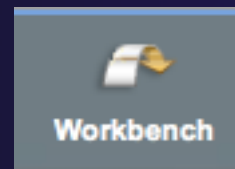
- ✦ A *workbench* contains the menus, toolbars, editors and views that make up the main Eclipse window
- ✦ The workbench represents the desktop development environment
 - ✦ Contains a set of tools for resource mgmt
 - ✦ Provides a common way of navigating through the resources
- ✦ Multiple workbenches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



perspective

Perspectives

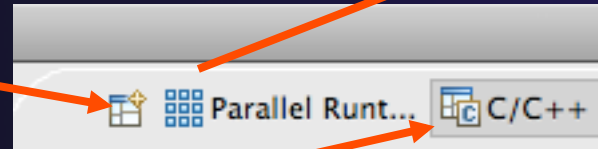
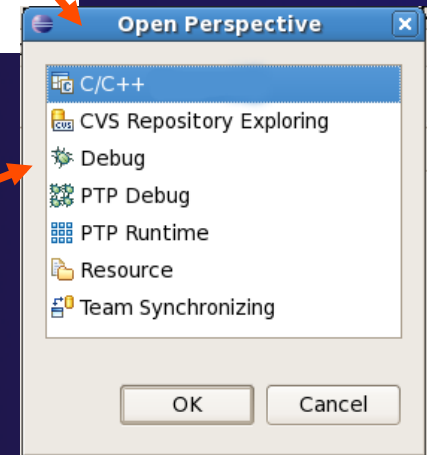
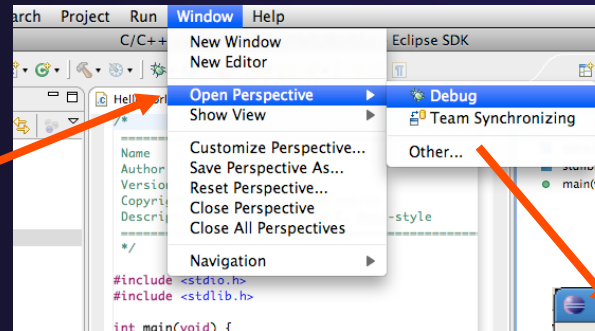
- ✦ Perspectives define the layout of views and editors in the workbench
- ✦ They are *task oriented*, i.e. they contain specific views for doing certain tasks:
 - ✦ There is a **Resource Perspective** for manipulating resources
 - ✦ **C/C++ Perspective** for manipulating compiled code
 - ✦ **Debug Perspective** for debugging applications
- ✦ 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

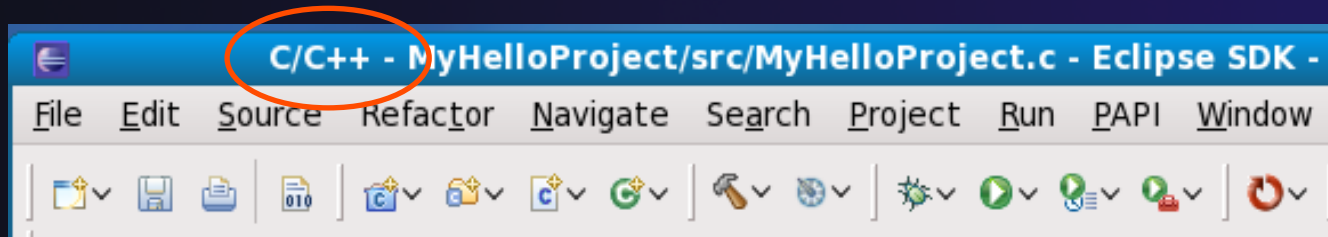


★ Switch to the C/C++ Perspective



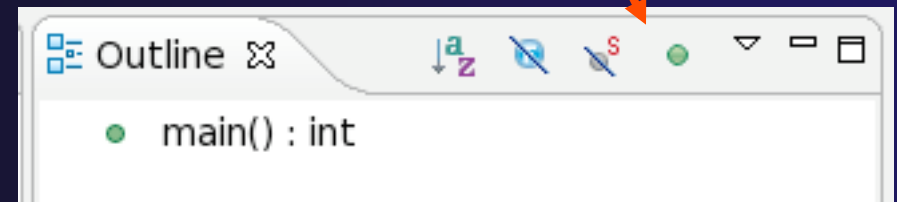
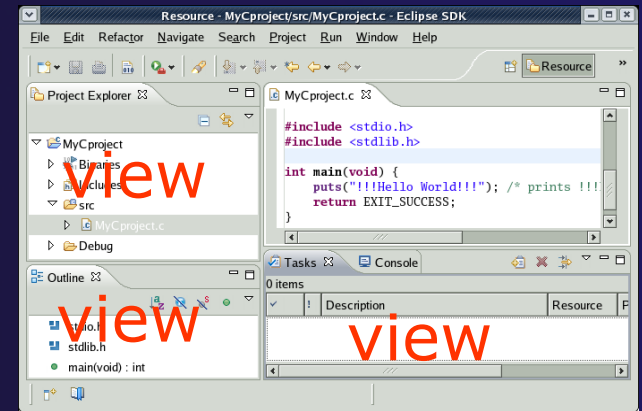
Which Perspective?

- ✦ Which Perspective am in in?
See Title Bar



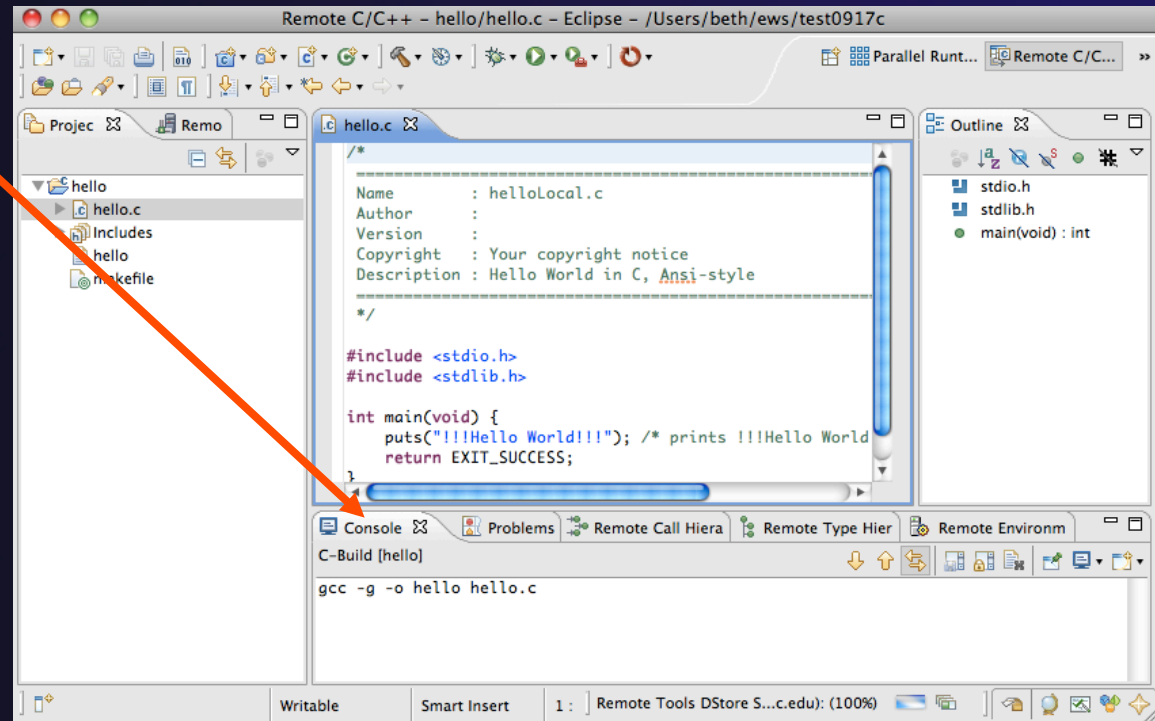
Views

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



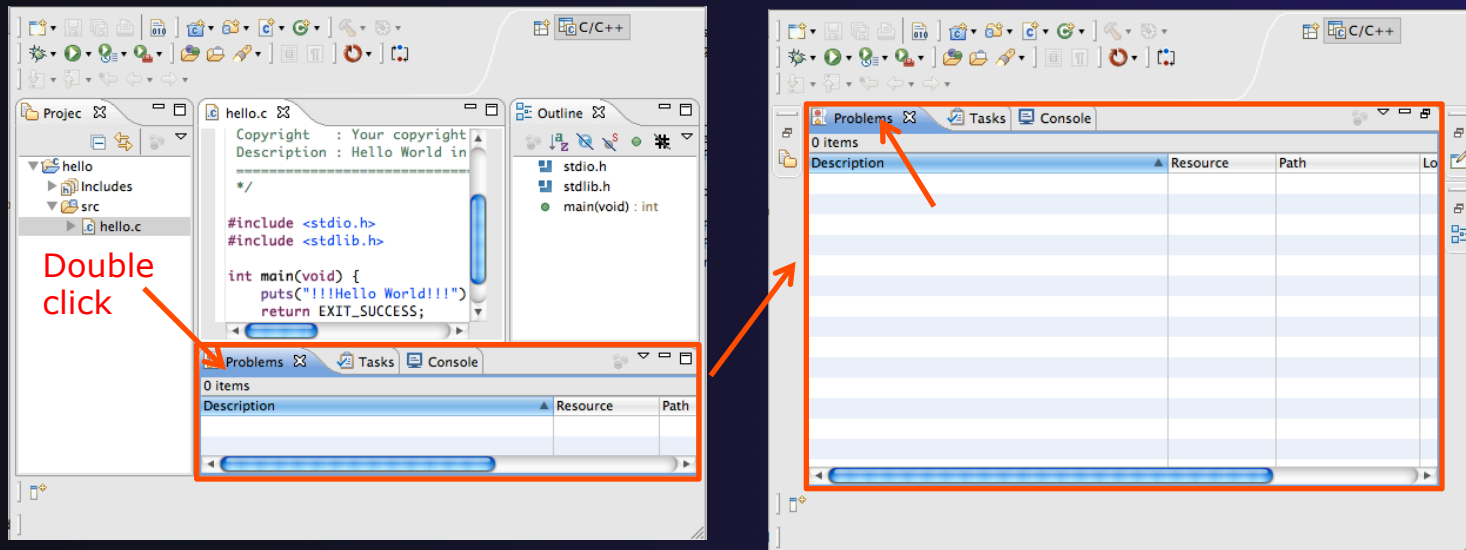
Stacked Views

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



Expand a View

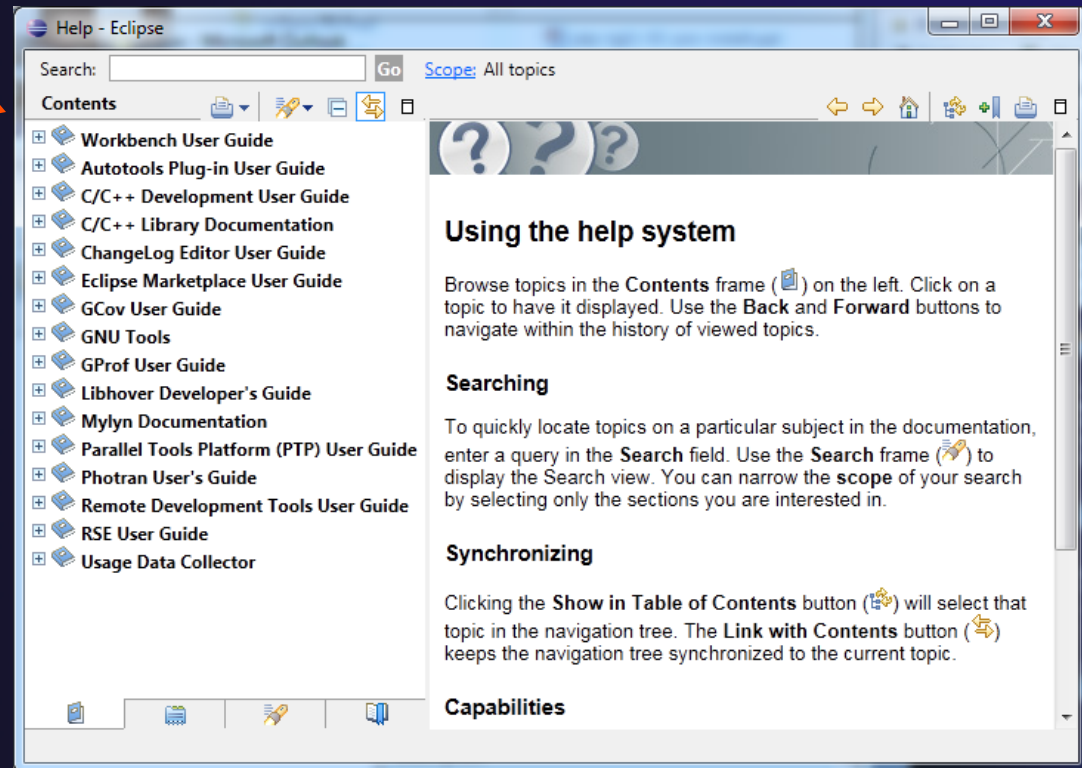
- ★ Double-click on a view/editor's tab to fill the workbench with its content;
- ★ Repeat to return to original size



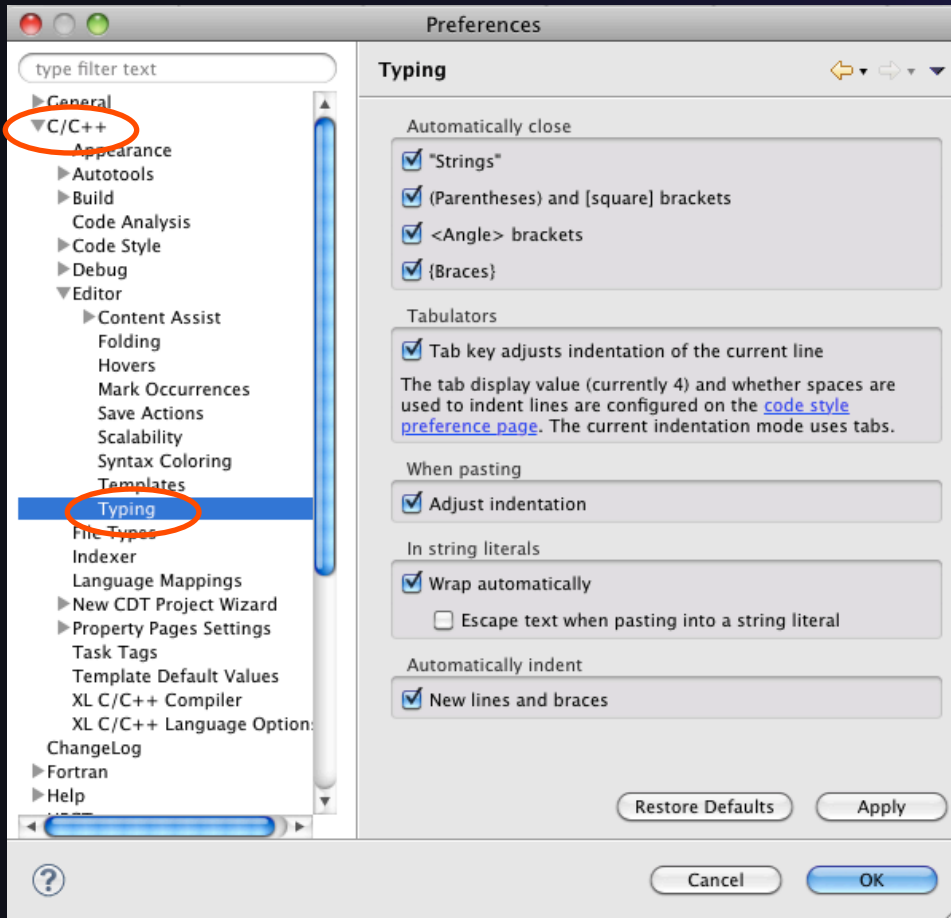
- ★ Window > Reset Perspective returns everything to original positions

Help

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

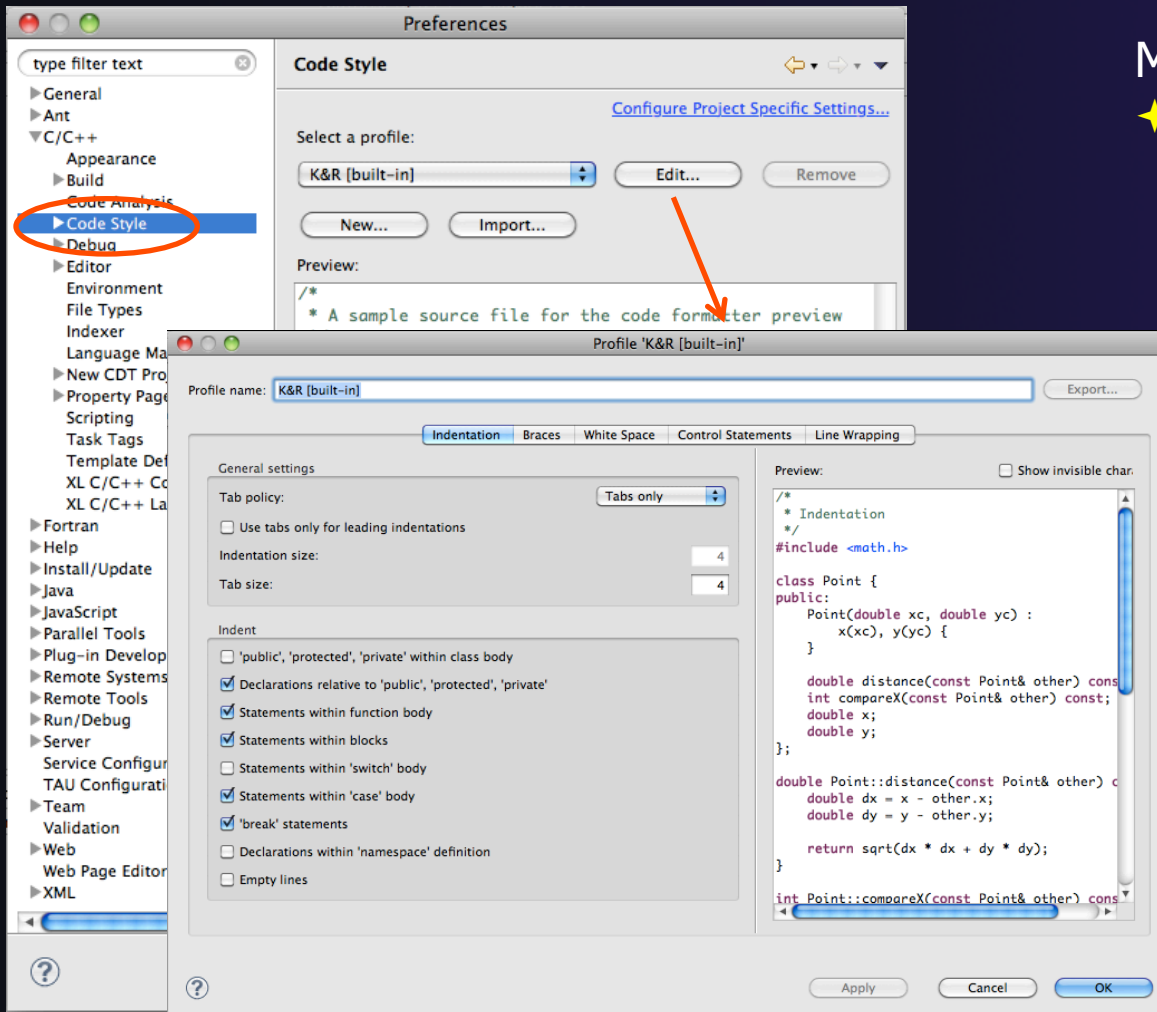


Eclipse Preferences



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

Preferences Example



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

Projects In Eclipse

Project Types

★ Local

- ★ Source is located on local machine, builds happen locally

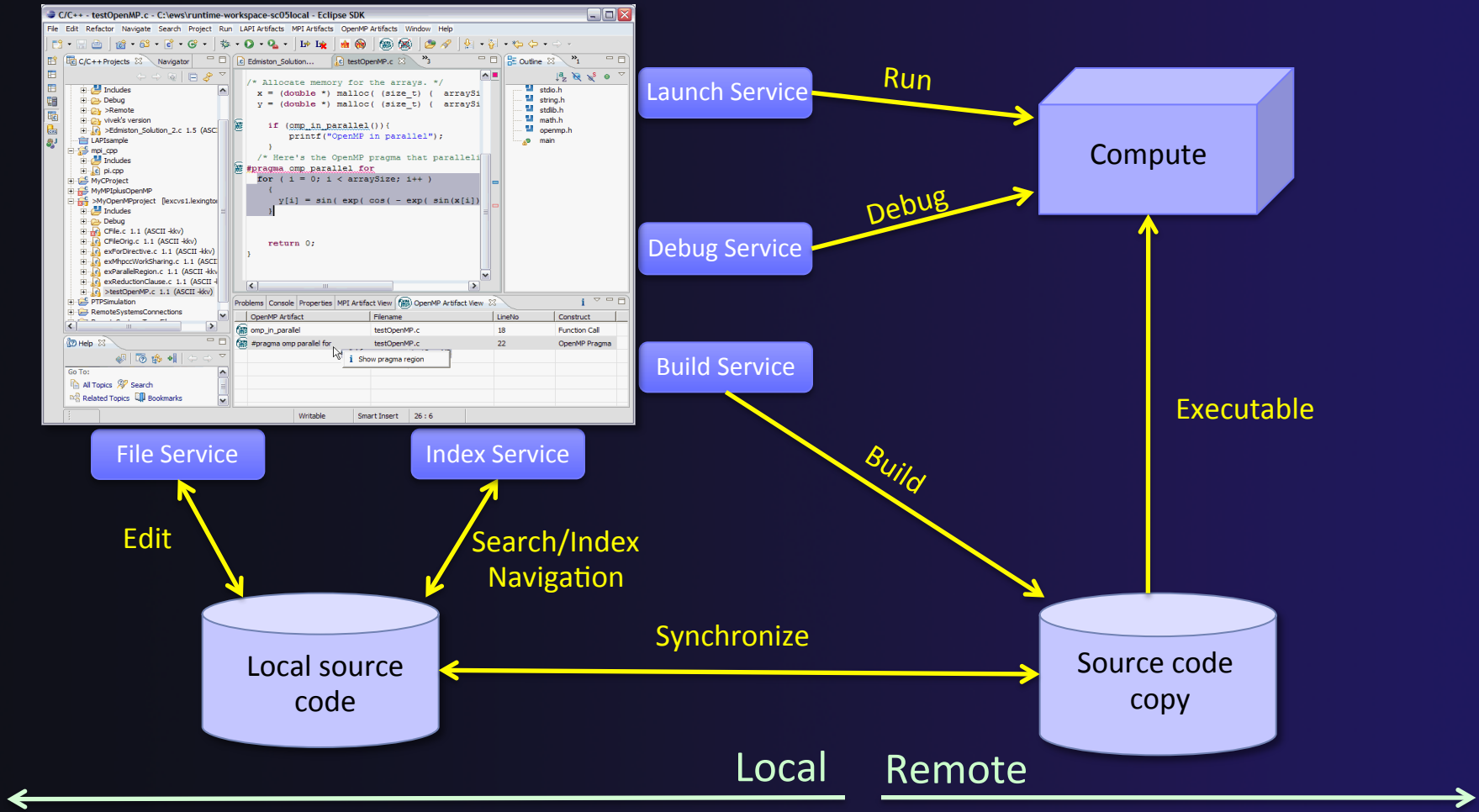
★ Synchronized

- ★ Source is local, then synchronized with remote machine(s)
- ★ Building and launching happens remotely (can also happen locally)

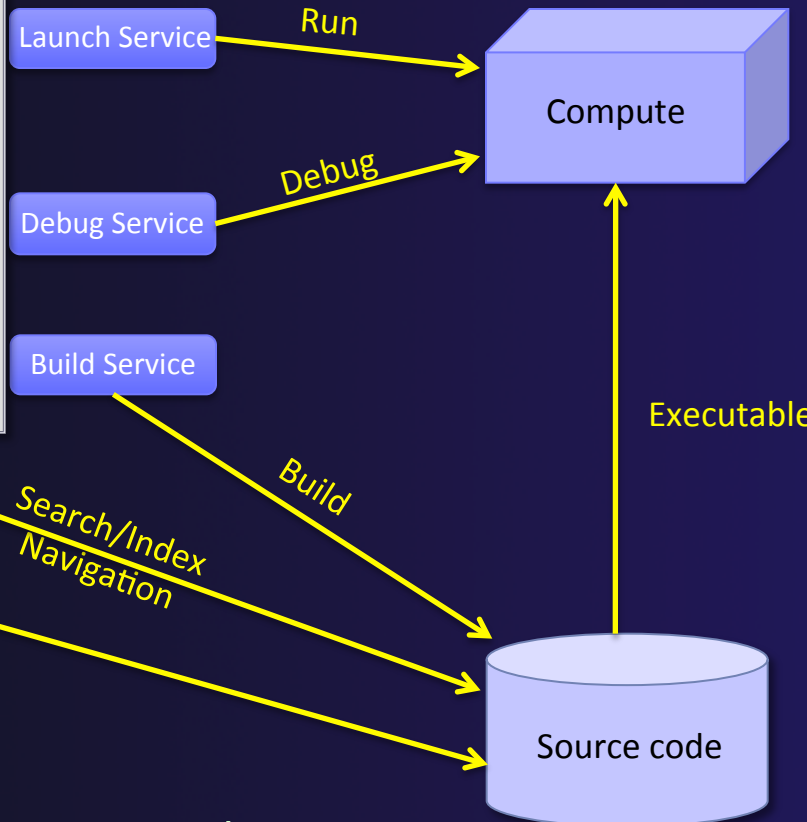
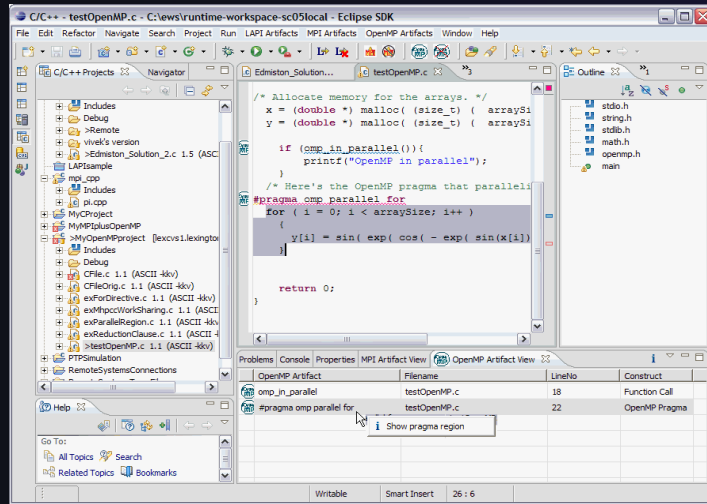
★ Remote

- ★ Source is located on remote machine(s), build and launch takes place on remote machine(s)

Synchronized Projects



Remote Projects



Local Remote



C, C++, and Fortran Projects

Build types

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

Parallel programs can be run on local machine or on a remote system

- ★ MPI (or other runtime) needs to be installed
- ★ An application built locally probably can't be run on a remote machine unless their architectures are the same

Checking out the project

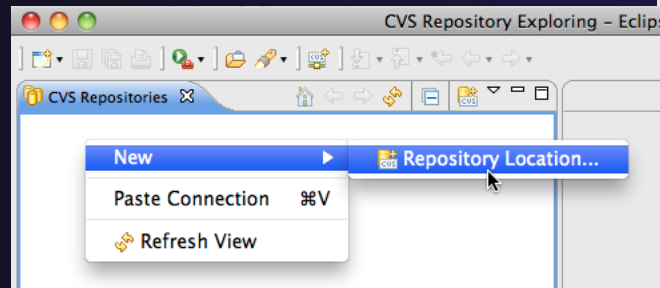
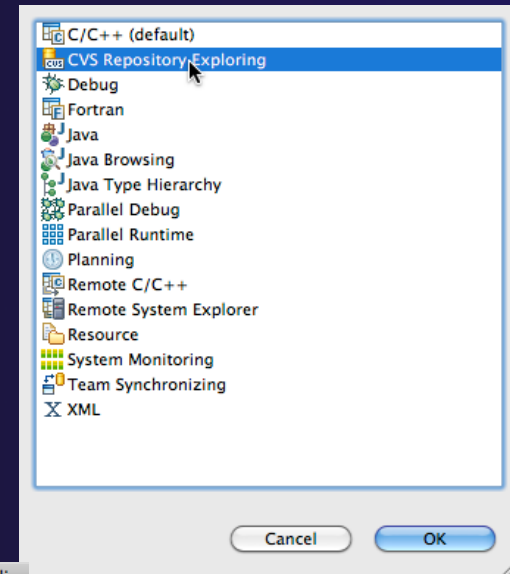
Using a Source Code Repository Introduction to Team Features



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

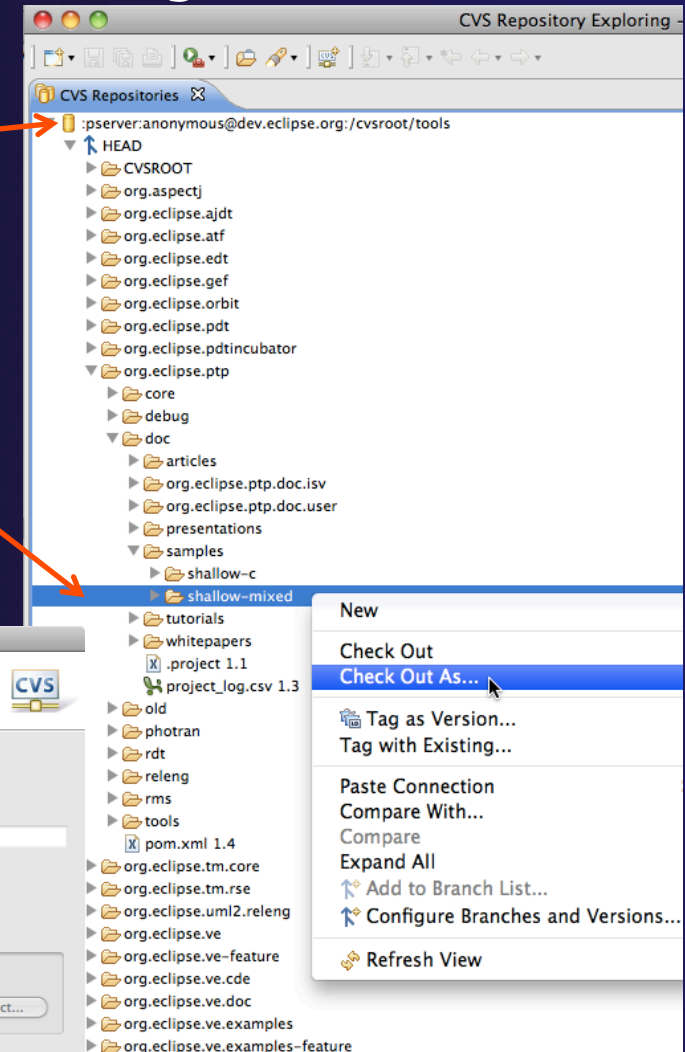
- ★ Enter **Host:** dev.eclipse.org
- ★ **Repository path:** /cvsroot/tools
- ➔ ★ For anonymous access:
 - ★ **User:** anonymous
 - ★ No password is required
 - ★ **Connection type:** pserver (default)
- ★ For authorized access:
 - ★ **User:** your userid
 - ★ **Password:** your password
 - ★ **Connection type:** change to **extssh**
- ★ Select **Finish**

The screenshot shows the 'Add CVS Repository' dialog box in Eclipse IDE. The dialog is titled 'Add CVS Repository' and has a subtitle 'Add a new CVS Repository to the CVS Repositories view'. It features a CVS logo in the top right corner. The dialog is divided into several sections: 'Location' with fields for 'Host' (dev.eclipse.org) and 'Repository path' (/cvsroot/tools); 'Authentication' with a 'User' dropdown (anonymous) and a 'Password' field; 'Connection' with a 'Connection type' dropdown (pserver) and radio buttons for 'Use default port' (selected) and 'Use port:'. There are also checkboxes for 'Validate connection on finish' (checked) and 'Save password (could trigger secure storage login)'. A link for 'Configure connection preferences...' is provided. At the bottom, there are 'Cancel' and 'Finish' buttons.



Checking out the Project

- ✦ Expand the repository location
- ✦ Expand **HEAD**
- ✦ Expand **org.eclipse.ptp, doc, and samples**
- ✦ Right click on **shallow-mixed** and select **Check Out As...**
- ✦ On **Check Out As** dialog, select **Finish**



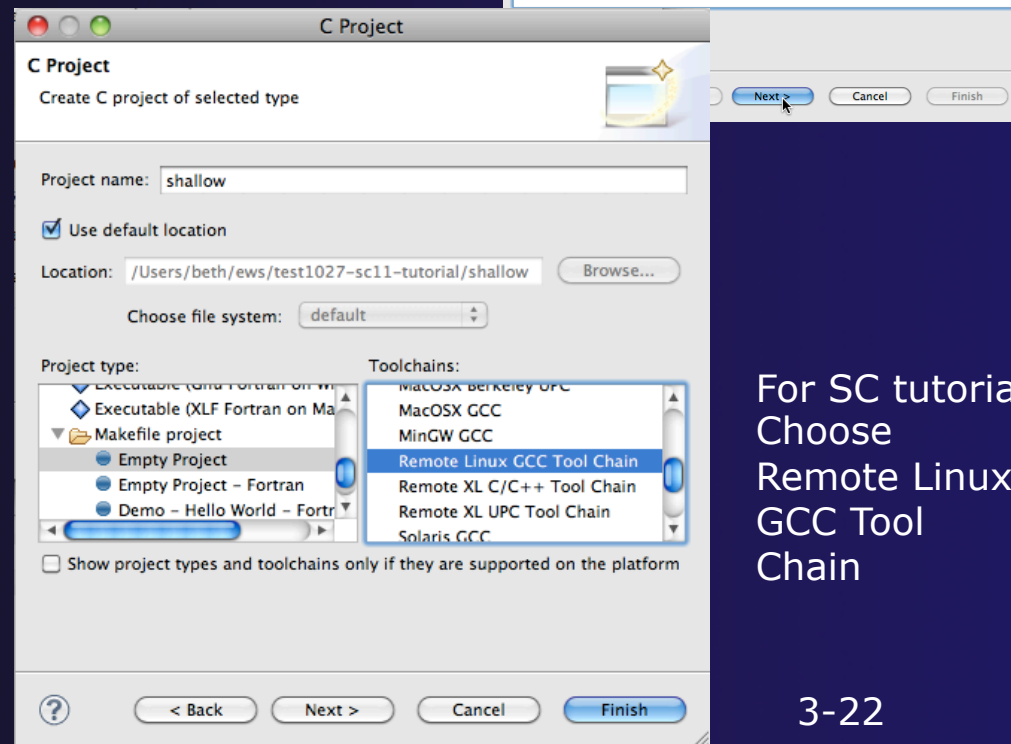
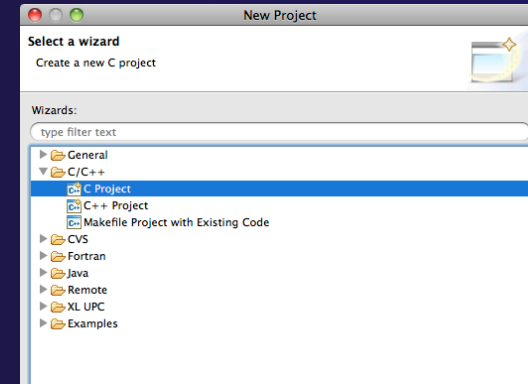
The default of “Check out as a project configured using the New Project Wizard” is what we want



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 **C/C++**
- ✦ Select **C Project** and click on **Next>**
- ✦ Enter 'shallow' as **Project Name**
- ✦ Under **Project type**, expand **Makefile project** - scroll to the bottom
- ✦ Select **Empty Project**
- ✦ Select a toolchain that matches your system from **Toolchains**
 - ✦ Since we will build/run this on the remote system, choose an appropriate toolchain
 - ✦ You may need to uncheck "Show project types and toolchains only if they are supported on the platform"
- ✦ Click on **Finish**

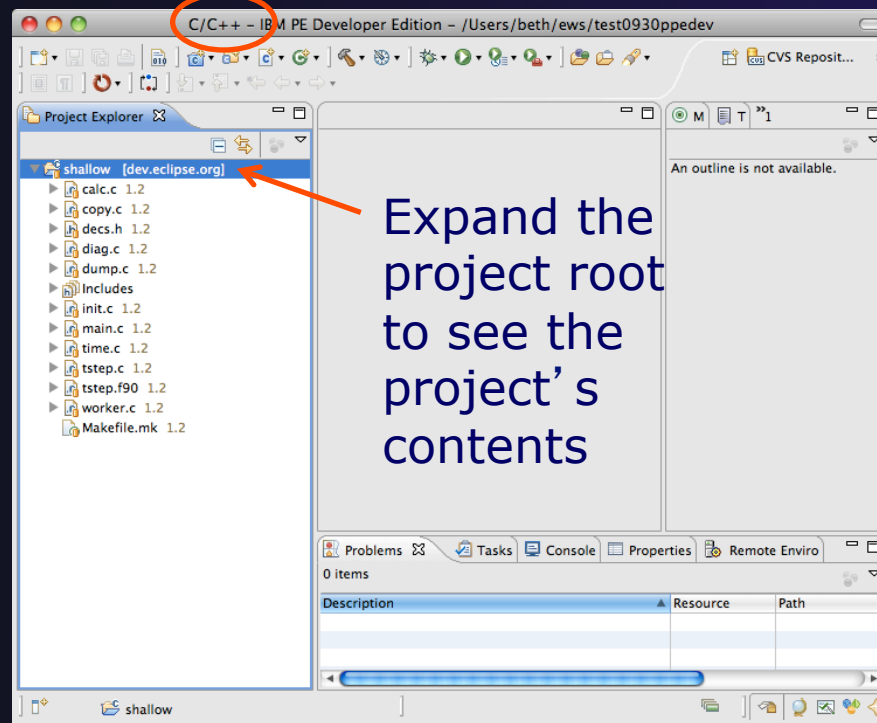
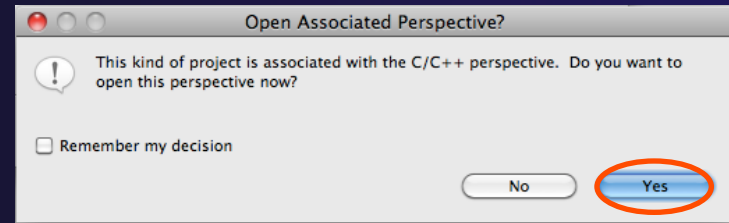


For SC tutorial
Choose
Remote Linux
GCC Tool
Chain



C/C++ Perspective

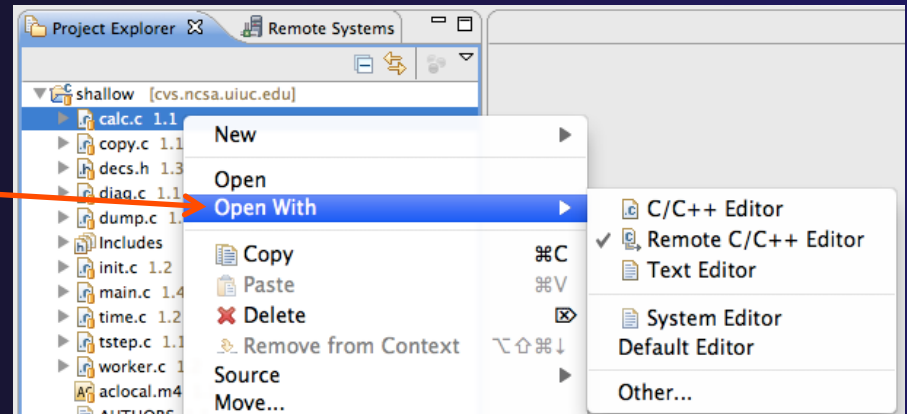
- ✦ Switch to the C/C++ Perspective when Prompted
- ✦ You should now see the “shallow” project in your workspace



Editor Features

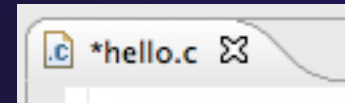
Editors

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



Saving File in Editor

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

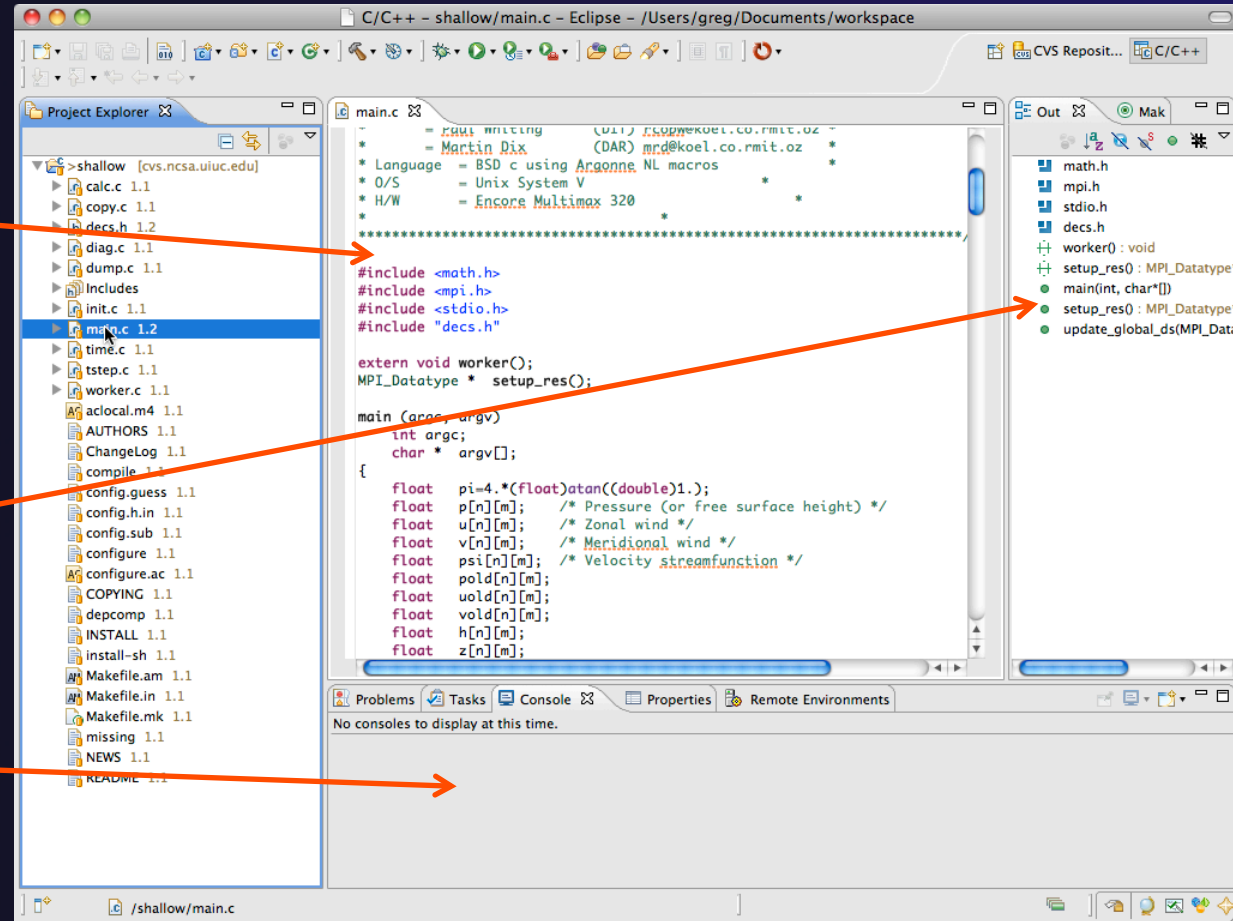


- ★ Save the changes by using Command/Ctrl-S or **File>Save**
- ★ Undo last change using **Command/Ctrl Z**



Editor and Outline View

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

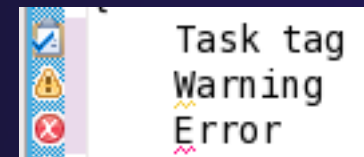


Source Code Editors & Markers

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

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

Icons:



Code Analysis (Codan)

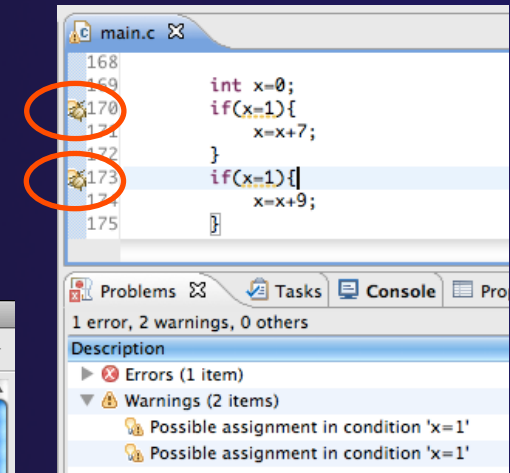
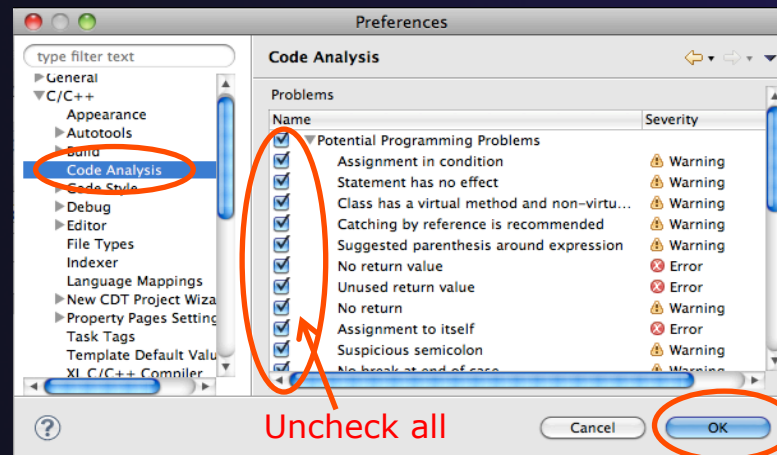
- ★ If you see bug icons in the editor marker bar, they are likely suggestions from Codan
- ★ Code checkers can flag possible errors, even if code is technically correct
- ★ To turn them off, use Preferences

Window > Preferences or Mac: Eclipse > Preferences

C/C++ > Code Analysis

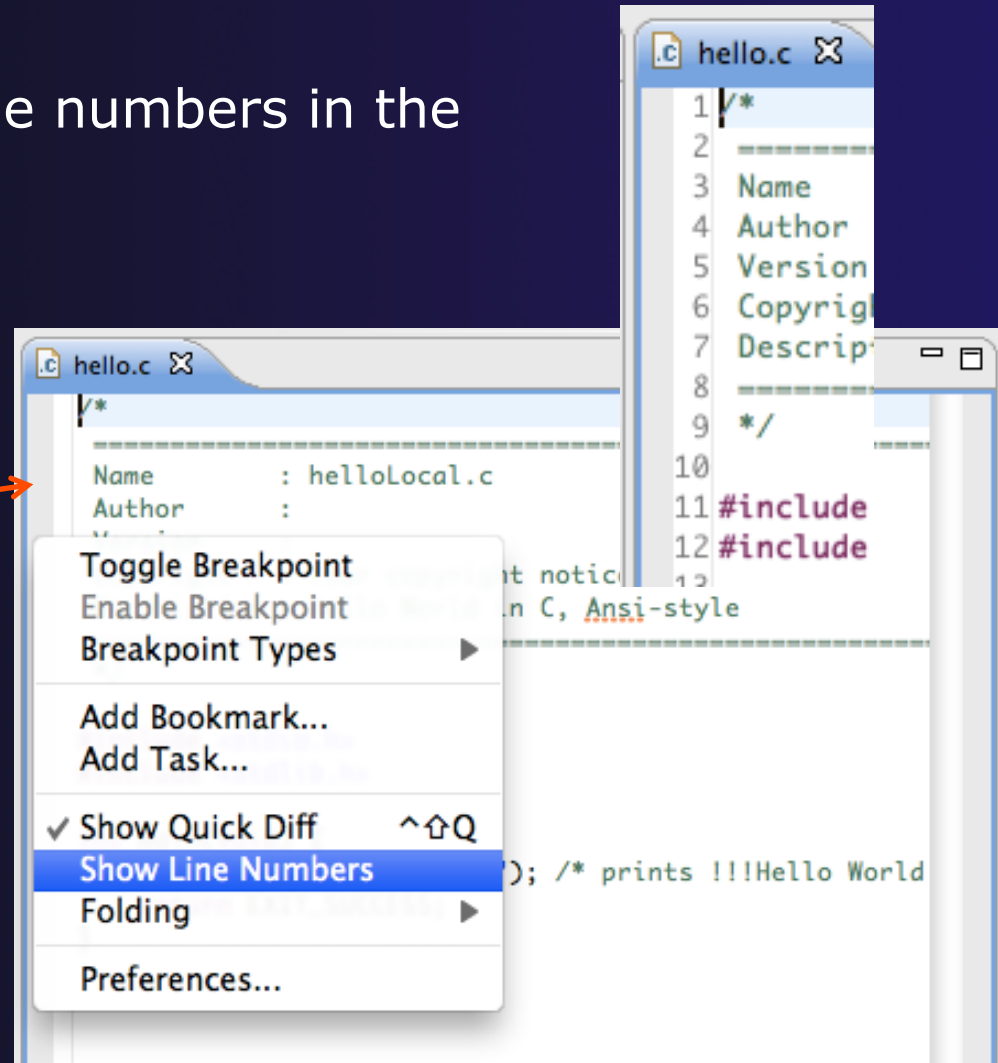
and uncheck
all problems

- ★ Select OK to close Preferences
- ★ To remove icons: Rightmouse on Project > Run C/C++ Code Analysis



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
 - ★ Click on **Show Line Numbers**





Navigating to Other Files

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

```

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

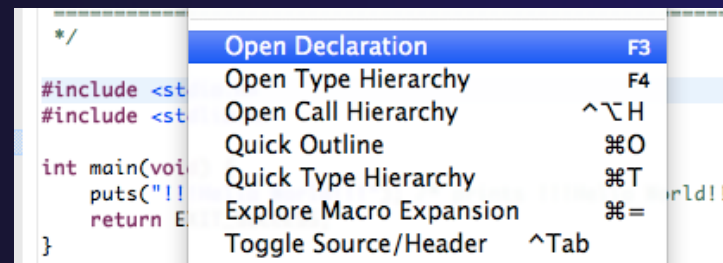
```

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

```

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

```



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



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

```

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

```

Press '^Space' to show Template Propos

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

Hit ctrl-space again
for code templates

```

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

```

```

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

```

More info on code templates later



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
```

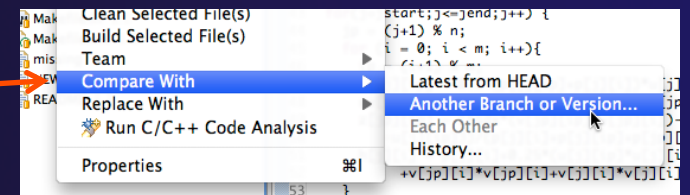
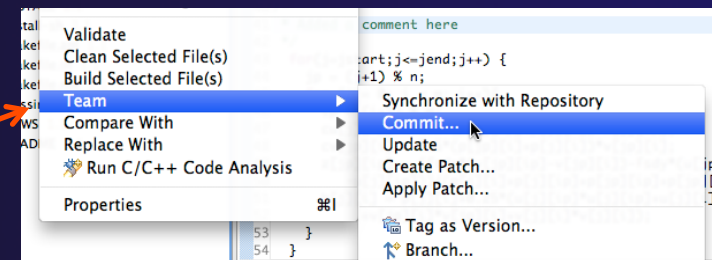
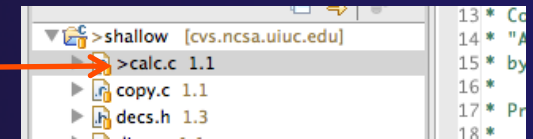
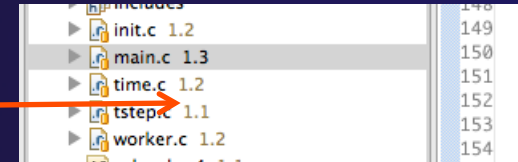
Team Features

“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

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)





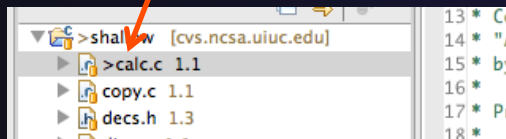
File Modification

- ✦ 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         }

```





View Changes

- ★ Right-click on “calc.c” and select **Compare With>Latest from HEAD**
- ★ 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

```

C Compare
  Translation Unit
    calcuvzh
    calcuvzh
    cu
    cv
    fsdx
    fsdy

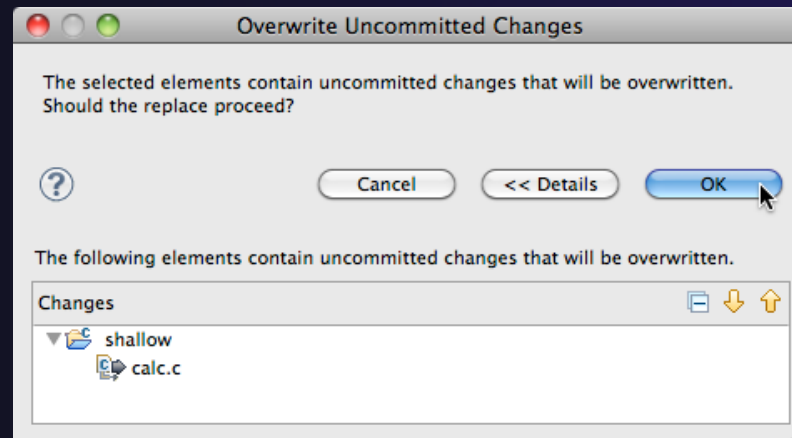
C Compare Viewer
  Local File 1.1
32 float v[n][m];
33 float cu[n][m];
34 float cv[n][m];
35 float h[n][m];
36 float z[n][m];
37 float fsdx, fsdy;
38 {
39   int i,j,ip,jp;
40 /*
41  * Added a comment here
42 */
43   for(j=jstart;j<=jend;j++) {
44     jp = (j+1) % n;
45     for (i = 0; i < m; i++){
46       ip = (i+1) % m;
47       cu[j][ip] = 0.5*(p[j][ip]+p[j][i])*u[j][
48       cv[jp][i] = 0.5*(p[jp][i]+p[j][i])*v[jp]
49       z[jp][ip] = (fsdx*(v[jp][ip]-v[jp][i])-f
50
  Remote File 1.1
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   for(j=jstart;j<=jend;j++) {
42     jp = (j+1) % n;
43     for (i = 0; i < m; i++){
44       ip = (i+1) % m;
45       cu[j][ip] = 0.5*(p[j][ip]+p[j][i])*u
46       cv[jp][i] = 0.5*(p[jp][i]+p[j][i])*v
47       z[jp][ip] = (fsdx*(v[jp][ip]-v[jp][i]
49

```




Revert To The Latest Version

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



MPI Features

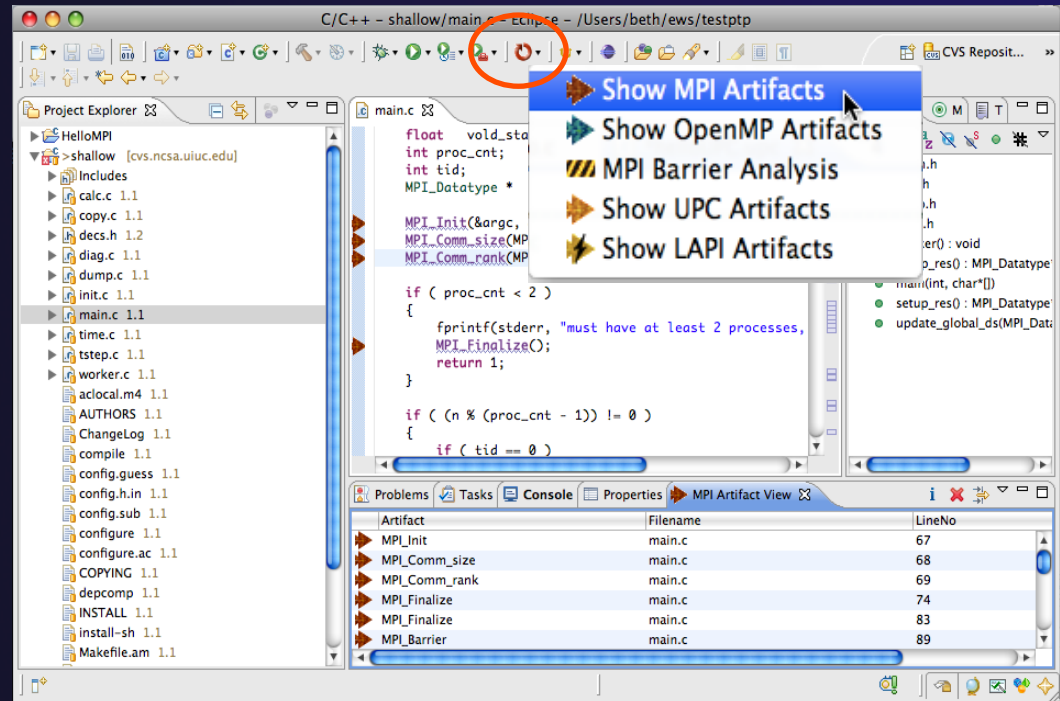
MPI-Specific Features

- ★ PTP's Parallel Language Development Tools (PLDT) has several features specifically for developing MPI code
 - ★ Show MPI Artifacts
 - ★ Code completion
 - ★ Context Sensitive Help for MPI
 - ★ Hover Help
 - ★ MPI Templates in the editor
 - ★ MPI Barrier Analysis

Show MPI Artifacts




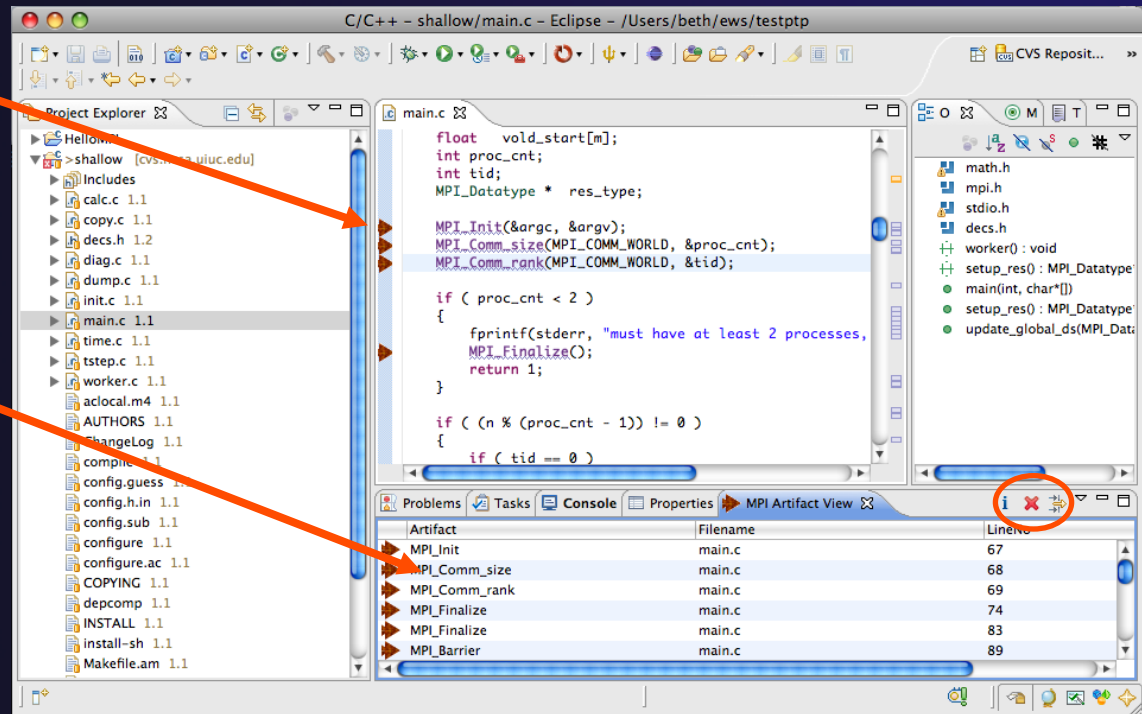
- ✦ In Project Explorer, select a project, folder, or a single source file
 - ✦ The analysis will be run on the selected resources
- ✦ Select **Show MPI Artifacts**
- ✦ Run the analysis by clicking on drop-down menu next to the analysis button
- ✦ Works on local and remote files



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 



Artifact	Filename	LineNo
MPI_Init	main.c	67
MPI_Comm_size	main.c	68
MPI_Comm_rank	main.c	69
MPI_Finalize	main.c	74
MPI_Finalize	main.c	83
MPI_Barrier	main.c	89



MPI Editor Features

```

float vold_start[m];
int proc_cnt;
int tid;
MPI_Datatype * res_type;

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

MPI_B

```

- MPI_Barrier(MPI_Comm) int
- MPI_Bcast(void*, int, MPI_Datatype, int, MPI_Comm)
- MPI_Bsend(void*, int, MPI_Datatype, int, int, MPI_Comm)
- MPI_Bsend_init(void*, int, MPI_Datatype, int, MPI_Comm)
- MPI_Buffer_attach(void*, int) int
- MPI_Buffer_detach(void*, int*) int
- MPI_Barrier(MPI_Comm comm) : int
- MPI_Bcast(void * buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm)
- MPI_Bsend(void * buf, int count, MPI_Datatype datatype, int dest, MPI_Comm comm)
- MPI_Bsend_init(void * buf, int count, MPI_Datatype datatype, int dest, MPI_Comm comm)

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

```

float vold_start[m];
int proc_cnt;
int tid;
MPI_Datatype * res_type;

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

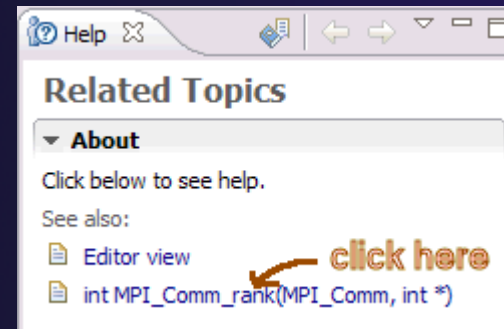
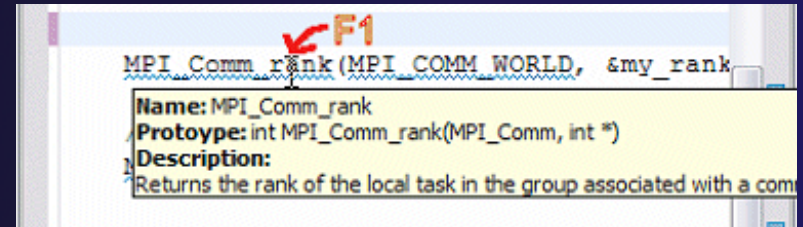
```

Name: MPI_Comm_rank
Prototype: int MPI_Comm_rank(MPI_Comm, int *)
Description: Returns the rank of the local task in the group associated with a communicator.

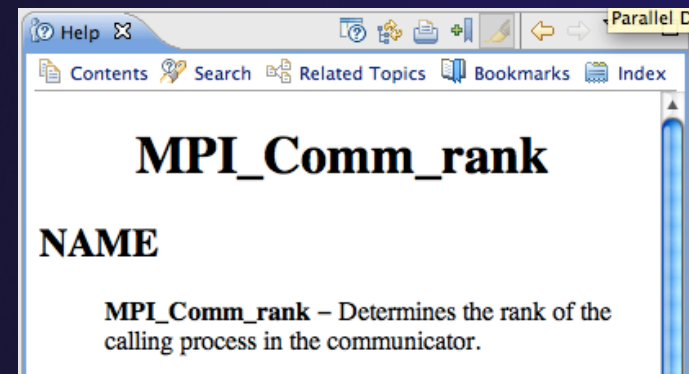
- ★ Hover over MPI API
- ★ Displays the function prototype and a description

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

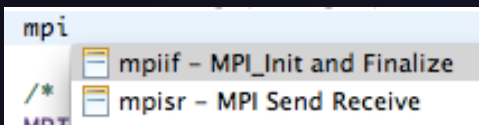
✦ Example:
MPI send-receive

✦ 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

Local files only

The screenshot displays the Eclipse IDE interface for a C++ project named 'MyBarrier'. The main editor shows the source code of 'MyBarrier.c', which includes MPI barrier synchronization. The 'Barrier Matches' view at the bottom left shows a table of barrier synchronization points:

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

The 'Barrier Errors' view at the bottom right shows a summary of synchronization errors:

- Error
- Path 1 (1 barrier(s))
- Path 2 (0 barrier(s))
- Error
- Loop (dynamic number of barriers)

Verify barrier synchronization in C/ MPI programs

Interprocedural static analysis outputs:

✦ For verified programs, lists barrier statements that synchronize together (match)

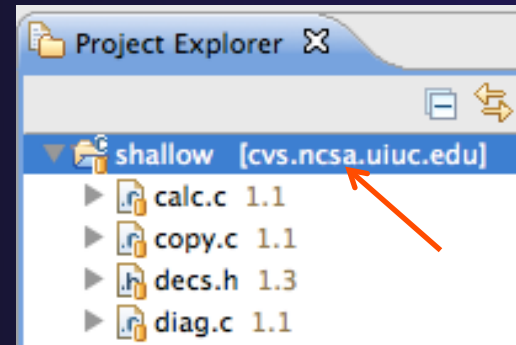
✦ For synchronization errors, reports counter example that illustrates and explains the error



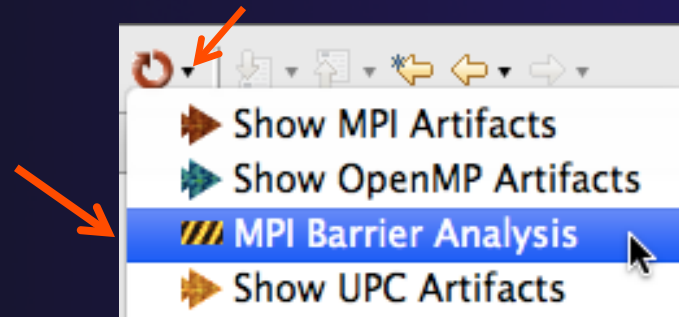
MPI Barrier Analysis – Try it

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 – Try It (2)

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

The screenshot shows the Eclipse IDE interface. The Project Explorer on the left lists files in the 'shallow' project, including 'main.c 1.4'. The main editor window displays the code for 'main.c', with lines 89 and 90 highlighted in blue, indicating MPI Barrier matches. The code is as follows:

```

83     MPI_Finalize();
84     return 1;
85 }
86
87 if (tid != 0) {
88     worker();
89     MPI_Barrier(MPI_COMM_WORLD);
90     MPI_Finalize();
91 } else {
92
93     /* master process */
94
95     duplication = 1 / (ProcessCount - 1);

```

The bottom panel shows the 'MPI Barrier Matches' table:

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



MPI Barrier Analysis - views

The screenshot displays three panels from the MPI Barrier Analysis tool:

- MPI Barriers view:** A table listing barriers found in the code.

Function
main
main
main
main
main
Barrier
- Barrier Matches view:** A table showing groups of barriers that match together.

Barrier Matching Set	Function	Filename	LineNo
Barrier 1 (2)	Barrier	MyBarrier.c	8
Barrier 1	Barrier	MyBarrier.c	8
Barrier 3	main	MyBarrier.c	41
Barrier 2 (1)	main	MyBarrier.c	31
Barrier 2	main	MyBarrier.c	31
Barrier 3 (2)	main	MyBarrier.c	41
Barrier 1	Barrier	MyBarrier.c	8
Barrier 3	main	MyBarrier.c	41
Barrier 4 (0)	main	MyBarrier.c	57
Barrier 5 (1)	main	MyBarrier.c	62
- Barrier Errors view:** A table showing error messages.

Barrier Matching Set	Function
Error	main
Path 1 (1 barrier(s))	
Path 2 (0 barrier(s))	
Error	main
Loop (dynamic number of barriers)	

Three orange arrows point from the text below to the respective views in the screenshot.

MPI Barriers view

Simply lists the barriers

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

Barrier Matches view

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

Barrier Errors view

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



Barrier Errors

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

```

120     prv = worker[PREV];
121     nxt = worker[NEXT];
122     jstart = worker[JSTART];
123     jend = worker[JEND];
124
125     MPI_B
126     /*
127     recei
128     */
129
130     for C
131     M
132

```

Autocomplete suggestions:

- MPI_Barrier(MPI_Comm) int Blocks each task until
- MPI_Bcast(void*, int, MPI_Datatype, int, MPI_
- MPI_Bsend(void*, int, MPI_Datatype, int, int,
- MPI_Bsend_init(void*, int, MPI_Datatype, int,
- MPI_Buffer_attach(void*, int) int
- MPI_Buffer_detach(void*, int*) int

```

124
125     MPI_Barrier(MPI_COMM_WORLD);
126

```

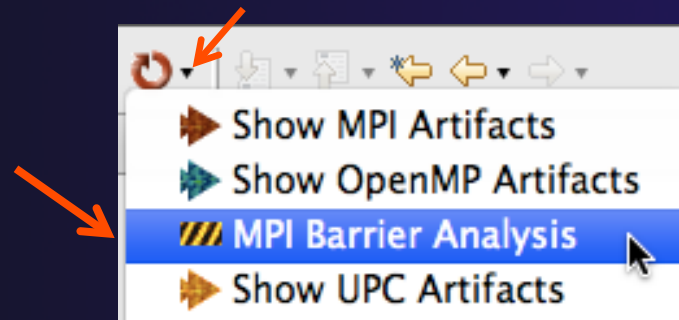


Barrier Errors (2)

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



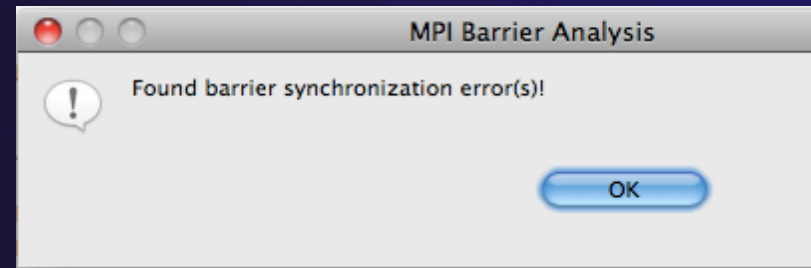
- ★ Run barrier analysis on shallow project again
 - ★ Select shallow project in Project Explorer first





Barrier Errors (3)

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



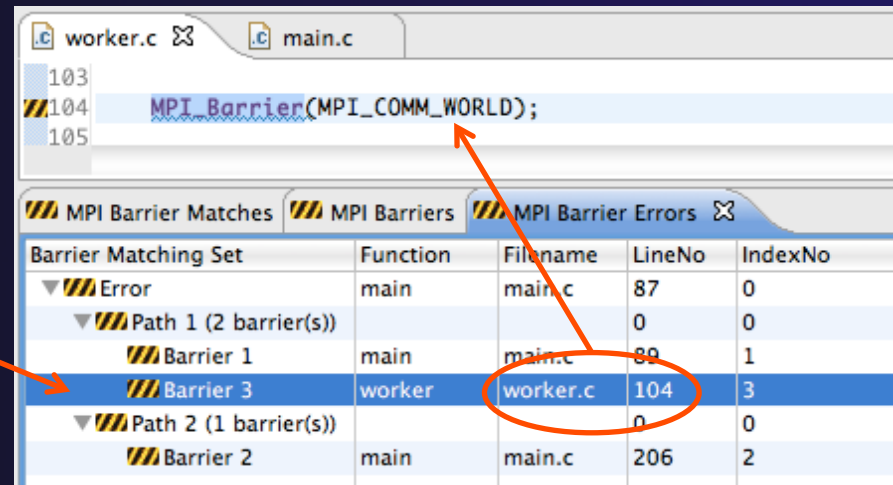
Barrier Matching Set	Function	Filename	LineNo	IndexNo
▼ Error	main	main.c	87	0
▼ Path 1 (2 barrier(s))			0	0
Barrier 1	main	main.c	89	1
Barrier 3	worker	worker.c	125	3
▼ Path 2 (1 barrier(s))			0	0
Barrier 2	main	main.c	206	2

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

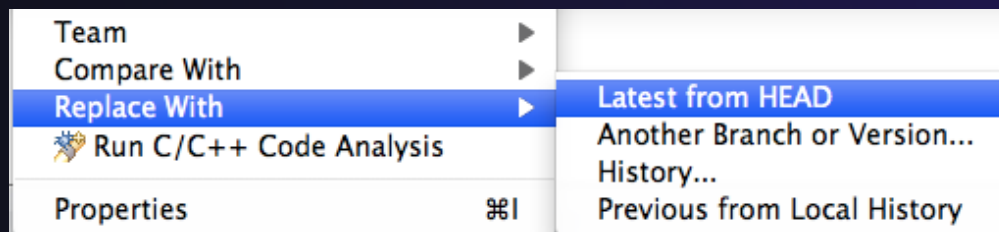


Fix Barrier Error

- ★ Fix the Barrier Error before continuing
- ★ Double-click on the barrier in worker.c to quickly navigate to it



- ★ Remove the line and save the file
-or-
Right mouse on worker.c in Project Explorer and do **Replace With > Latest from HEAD**





Remove Barrier Markers

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

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

Synchronizing the Project

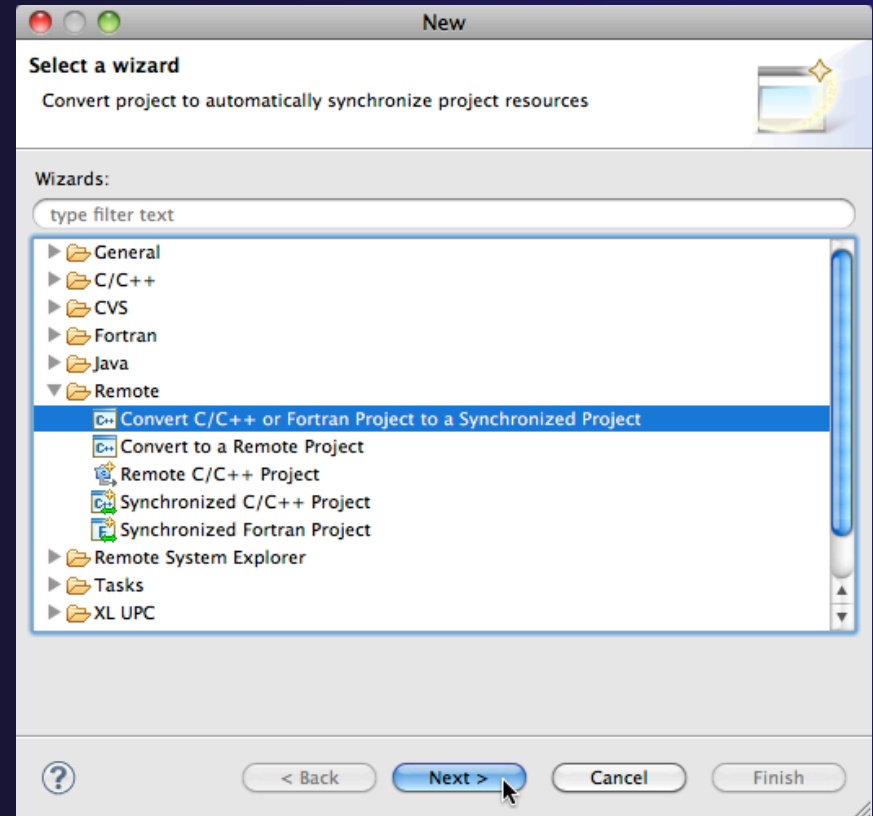
Synchronizing the Project

- ✦ Because we will be running on a remote system, we must also build on that system
- ✦ Source files must be available to build
- ✦ We will use a synchronized project to do this
 - ✦ Only needs to be done once for each project
 - ✦ A synchronized project could have been created initially
- ✦ Files are synchronized automatically when they are saved
- ✦ A full synchronize is also performed prior to a build



Converting To Synchronized

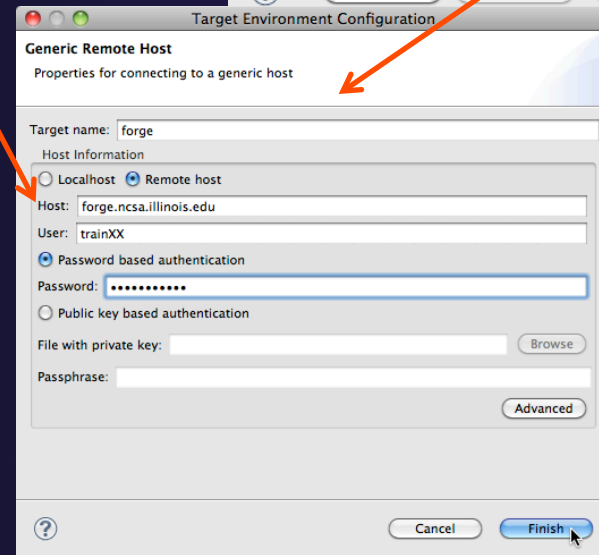
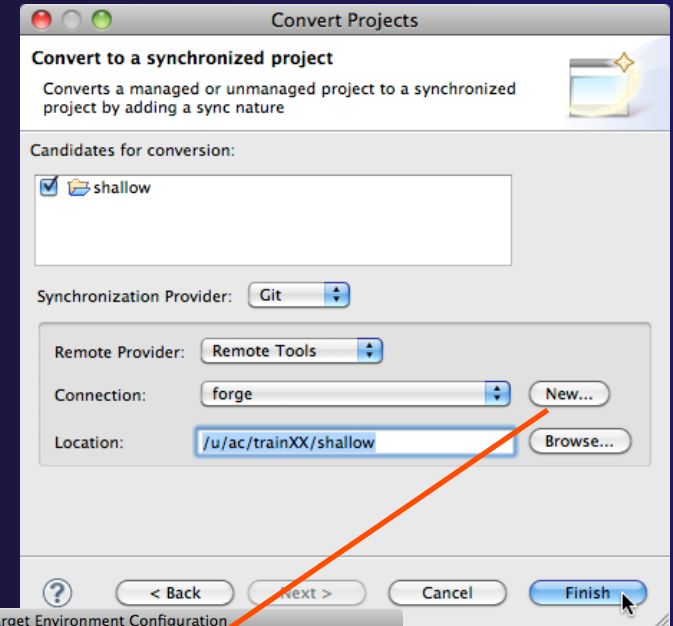
- ★ Select **File>New>Other...**
- ★ Open the Remote folder
- ★ Select **Convert C/C++ or Fortran Project to a Synchronized Project**
- ★ Click **Next>**





Convert Projects Wizard

- ✦ Select checkbox next to “shallow”
- ✦ For **Connection:**, click on **New...**
Enter as directed:
 - ✦ **Target name**
 - ✦ **Host** name of remote system
 - ✦ **User id** and **Password**
- ✦ Click **Finish** to close it
- ✦ Back in the **Convert Projects** dialog,
- ✦ Enter a directory name in the **Location** field; select **Browse...**
 - ✦ Sample: /u/ac/trainXX/shallow
 - ✦ Project files will be copied under this directory
- ✦ Click **Finish**






Synchronized Project

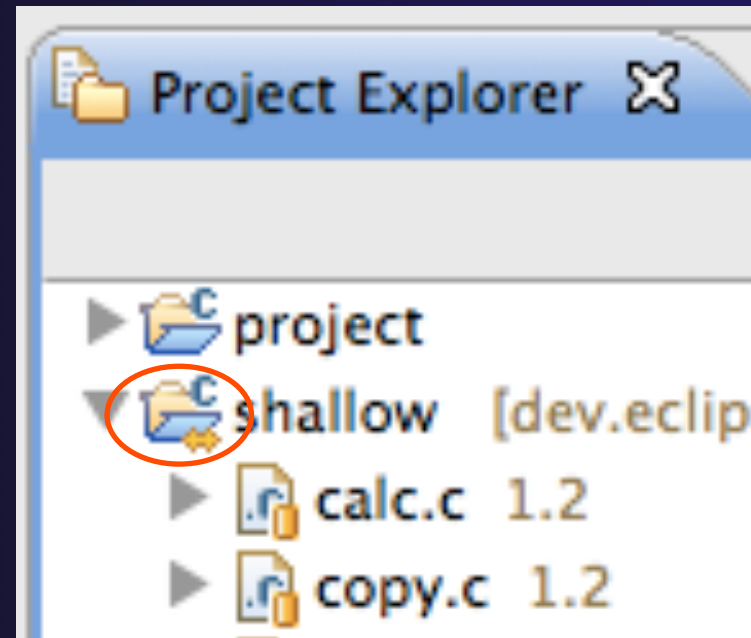
- ★ Back in the Project Explorer, decorator on project icon indicates synchronized project
- ★ Double-+ icon

- ★ Before sync

▼  shallow [dev.eclipse.org]

- ★ After sync

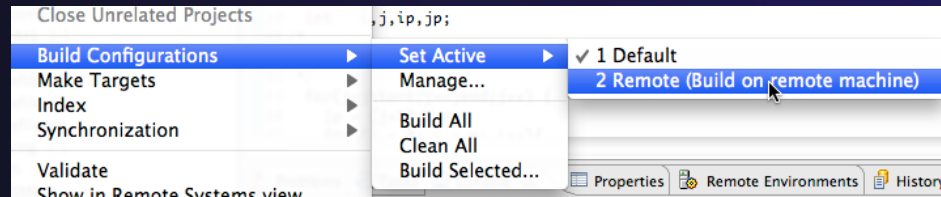
▼  shallow [dev.eclipse.org]



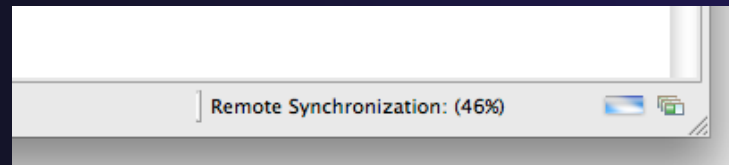


Set Active Build Configuration

- ★ The “Active” build configuration determines which system will be used for both synchronizing and building
- ★ Right-click on the project and select **Build Configurations>Set Active>Remote (Build on remote machine)**



- ★ The project should synchronize immediately

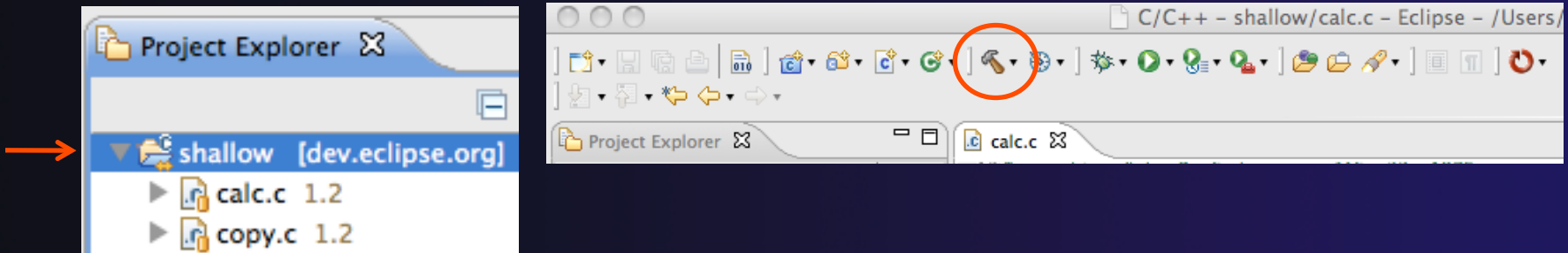


Building the Project



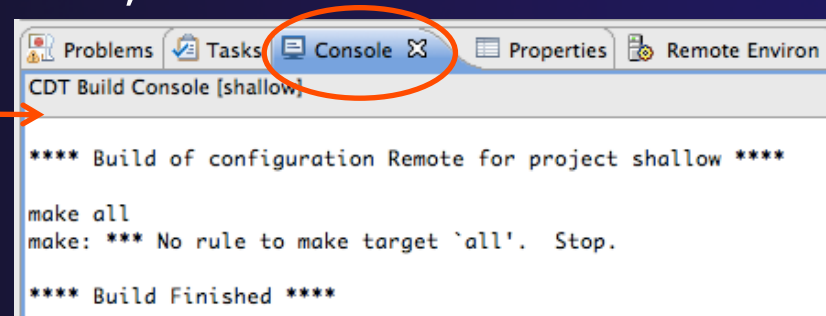
Building the Project

- ★ Select the project in Project Explorer, click on the  hammer button to run the build



- ★ By default, the Build Configuration assumes there is a Makefile (or makefile) for the project
- ★ In this case, there is no Makefile, so the build will fail.
See Console:

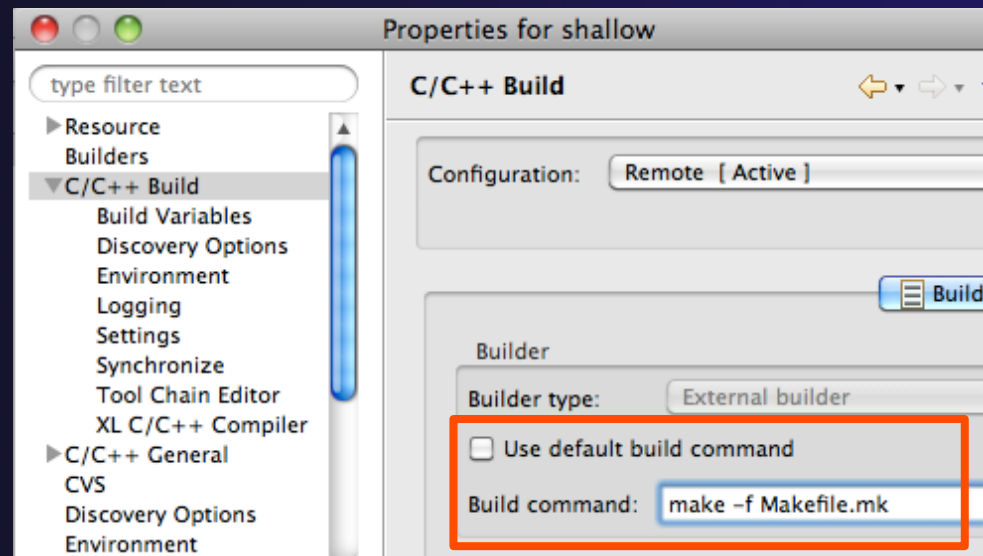
- ★ We'll see how to change it if the build command is different from 'make -f Makefile'



Fixing the Build Command: Editing Project Properties




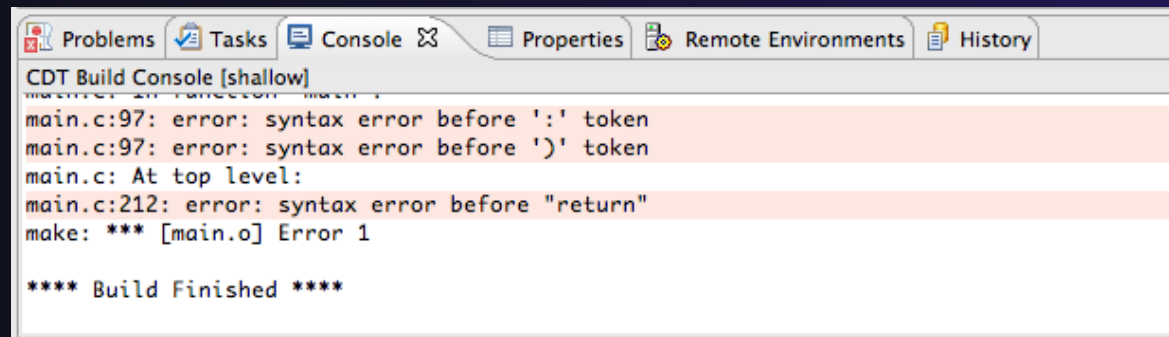
- ★ The build command is specified in the project properties
- ★ Open the properties by right-clicking on “shallow” and selecting **Properties** (bottom of the context menu list)
- ★ Click on **C/C++ Build**
- ★ Uncheck **Use default build command**
- ★ Enter “make -f Makefile.mk” in the **Build Command** field
- ★ Click **OK** to close project properties dialog





Re-Building the Project

- ★ Click on the  button again to run the build
- ★ Build output will be shown in the **Console** view

A screenshot of the CDT Build Console window. The window title is "CDT Build Console [shallow]". The console output shows several error messages: "main.c:97: error: syntax error before ':' token", "main.c:97: error: syntax error before ')' token", "main.c: At top level:", and "main.c:212: error: syntax error before 'return'". The output concludes with "make: *** [main.o] Error 1" and "**** Build Finished ****". The console view is part of an IDE interface with tabs for Problems, Tasks, Console, Properties, Remote Environments, and History.

```
CDT Build Console [shallow]
main.c:97: error: syntax error before ':' token
main.c:97: error: syntax error before ')' token
main.c: At top level:
main.c:212: error: syntax error before "return"
make: *** [main.o] Error 1

**** Build Finished ****
```

- ★ Exact output depends on your compiler



Build Problems

The screenshot shows the Eclipse IDE with a C/C++ project named 'shallow'. The main editor displays the file 'main.c' with the following code snippet:

```

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

```

The Problems view at the bottom shows the following error list:

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


★ Build problems will be shown in a variety of ways

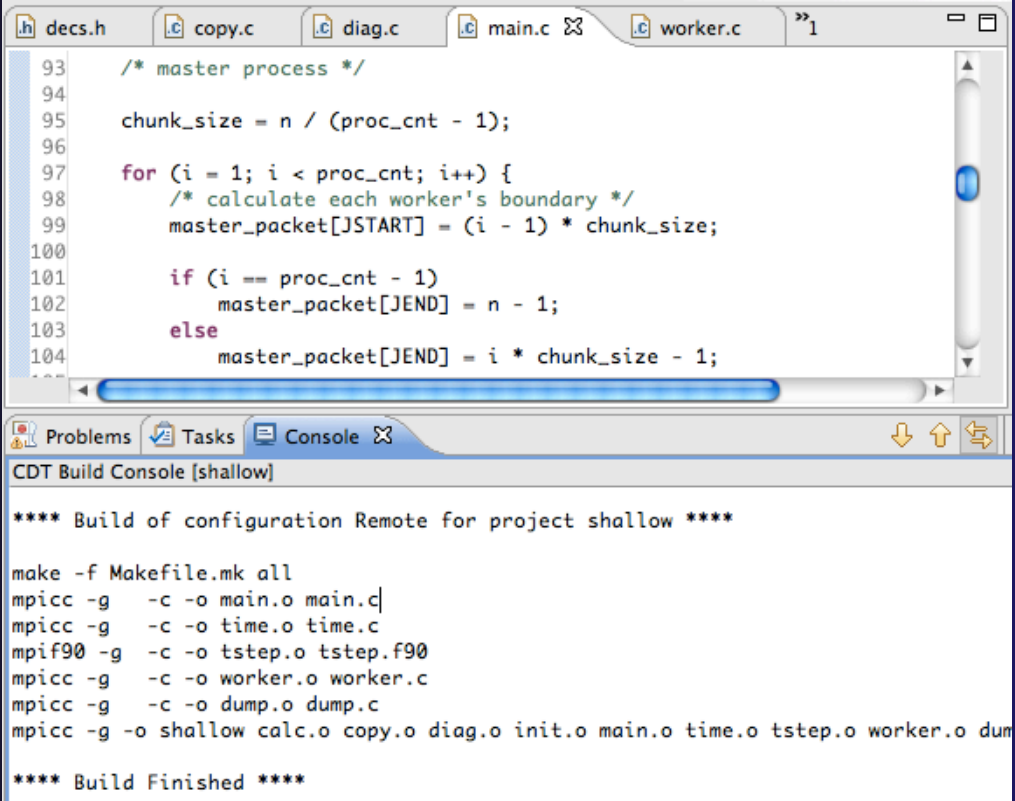
- ★ Marker on file
- ★ Marker on editor line
- ★ Line is highlighted
- ★ Marker on overview ruler
- ★ Listed in the **Problems view**

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



Fix Build Problems

- ✦ Fix errors by changing ‘:’ to ‘;’ on line 97
- ✦ Save the file
- ✦ Rebuild by pressing build button 
- ✦ Error markers have been removed
- ✦ Check console for correct build output



The screenshot shows an IDE with several tabs: decs.h, copy.c, diag.c, main.c, and worker.c. The main.c tab is active, showing the following code:

```
93  /* master process */
94
95  chunk_size = n / (proc_cnt - 1);
96
97  for (i = 1; i < proc_cnt; i++) {
98      /* calculate each worker's boundary */
99      master_packet[JSTART] = (i - 1) * chunk_size;
100
101      if (i == proc_cnt - 1)
102          master_packet[JEND] = n - 1;
103      else
104          master_packet[JEND] = i * chunk_size - 1;
```

The console window at the bottom shows the build output for a project named 'shallow':

```
**** Build of configuration Remote for project shallow ****

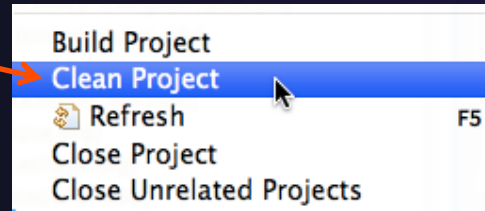
make -f Makefile.mk all
mpicc -g -c -o main.o main.c]
mpicc -g -c -o time.o time.c
mpif90 -g -c -o tstep.o tstep.f90
mpicc -g -c -o worker.o worker.c
mpicc -g -c -o dump.o dump.c
mpicc -g -o shallow calc.o copy.o diag.o init.o main.o time.o tstep.o worker.o dum

**** Build Finished ****
```



Forcing a Rebuild

- ✦ If no changes have been made, make doesn't think a build is needed
- ✦ In Project Explorer, Rightmouse on project, select **Clean Project**



A screenshot of the CDT Build Console window. The title bar shows 'Problems', 'Tasks', and 'Console'. The console output shows: 'CDT Build Console [shallow]', 'make -f Makefile.mk all', and 'make: Nothing to be done for `all`.'. An orange arrow points from the text 'make doesn't think a build is needed' in the list above to this console output.

- ✦ See console

A screenshot of the CDT Build Console window. The title bar shows 'Problems', 'Tasks', and 'Console'. The console output shows: 'CDT Build Console [shallow]', '**** Clean-only build of configurat', 'make -f Makefile.mk clean', and 'rm -f shallow calc.o copy.o diag.o'. An orange arrow points from the text 'See console' in the list above to this console output.

- ✦ Then rebuild 

Running the Program

Resource Managers

Running the Program

- ✦ Creating a resource manager
- ✦ Starting the resource manager
- ✦ Creating a run configuration
- ✦ Running (launching) the application
- ✦ Viewing the application run



Do this
once

Much of the following setup is configuration that you only need to do once: This icon will remind you.

Resource Managers

- ✦ PTP uses the term “resource manager” to refer to any subsystem that controls the resources required for launching a parallel job.
- ✦ Examples:
 - ✦ Batch scheduler (e.g. LoadLeveler, PBS, SLURM)
 - ✦ Interactive execution (e.g. Open MPI, MPICH2, etc.)
- ✦ Each resource manager controls one target system
- ✦ Resource Managers can be local or remote

Monitoring/Runtime Perspectives

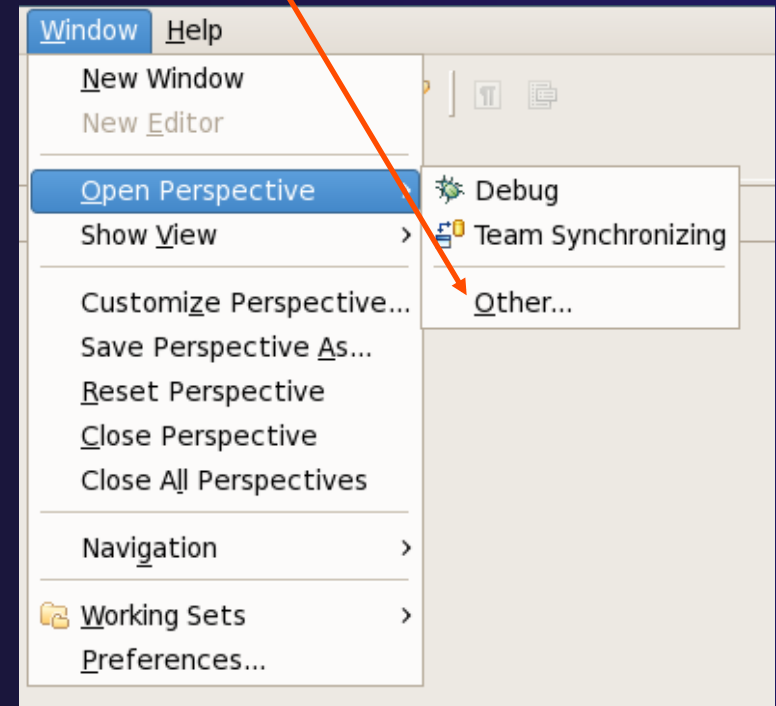
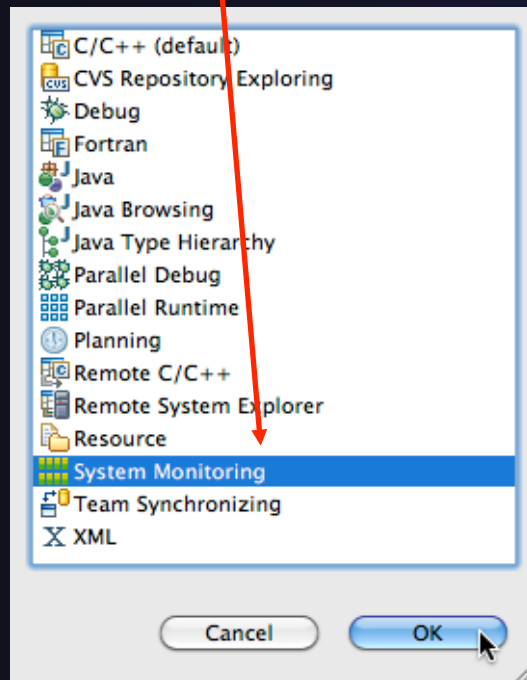
- ★ Parallel Runtime Perspective
 - ★ Used for legacy PTP Resource Managers
- ★ System Monitoring Perspective
 - ★ Used for newer Configurable Resource Managers (since PTP 5.0)
- ★ Which one is used?

Resource Manager	System Monitoring	Parallel Runtime
IBM LoadLeveler		✓
IBM Parallel Env		✓
MPICH2		✓
Open MPI		✓
PBS-Batch-Generic	✓	
PBS-Batch-Interactive	✓	
Remote Launch		✓
SLURM		✓



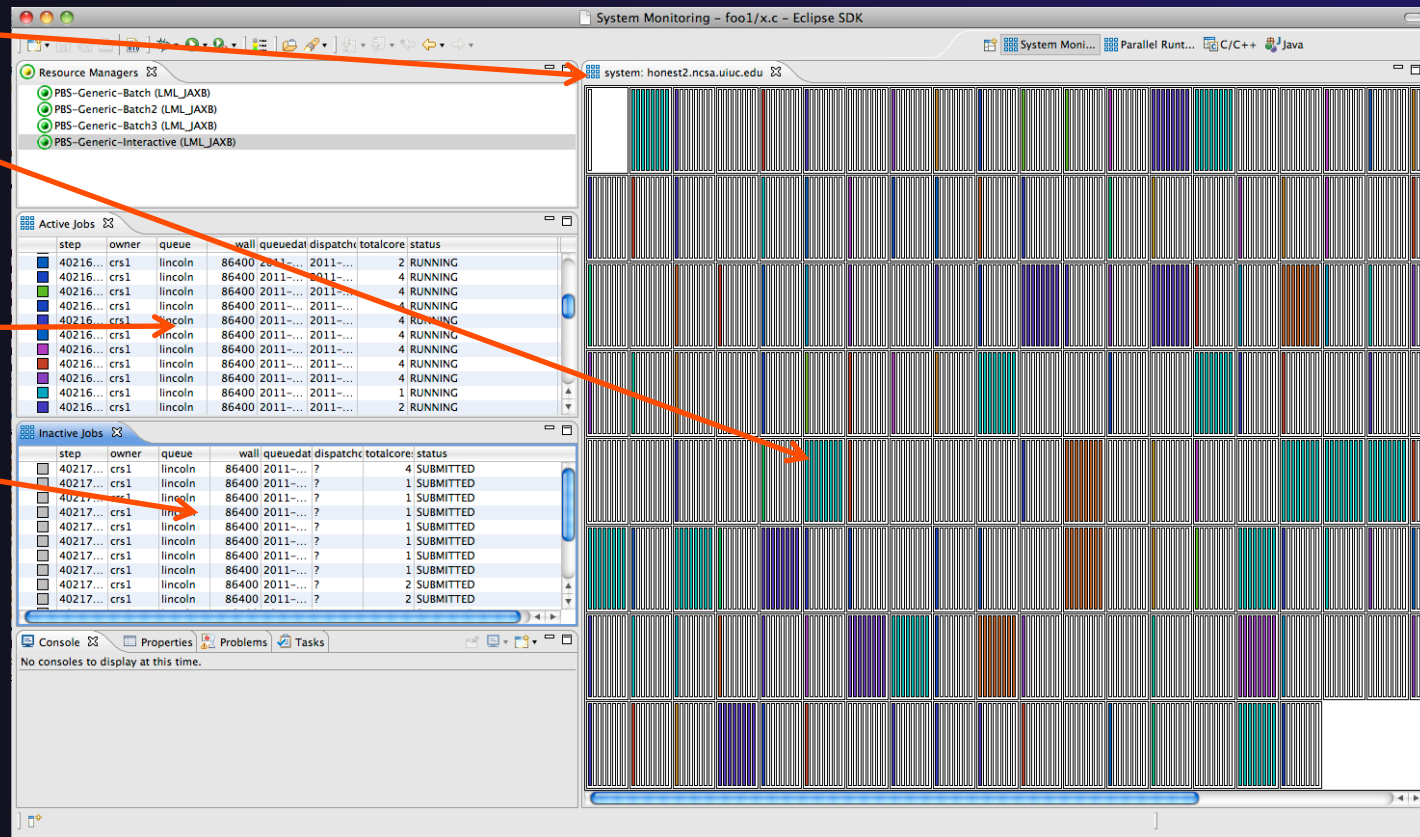
Preparing to Launch

- ✦ Setting up a resource manager is done in the System Monitoring perspective
 - ✦ (For PTP 5.0, this applies to PBS)
- ✦ Select **Window>Open Perspective>Other...**
- ✦ Choose **System Monitoring** and click **OK**



System Monitoring Perspective

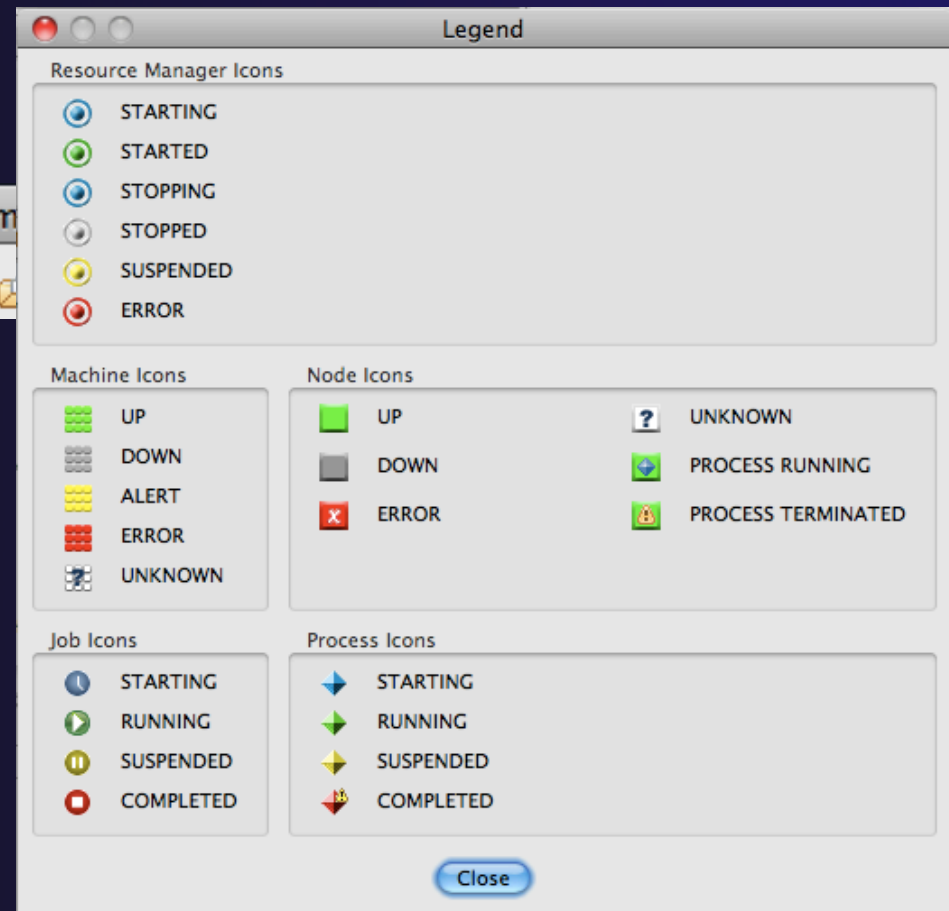
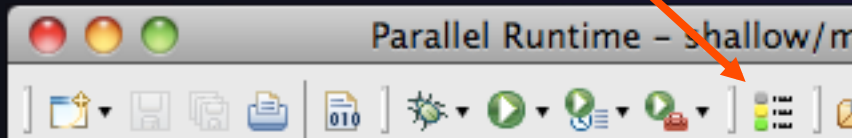
- ✦ System view
- ✦ Jobs running on system
- ✦ Active jobs
- ✦ Inactive jobs





About PTP Icons

- ★ Open using legend icon in toolbar



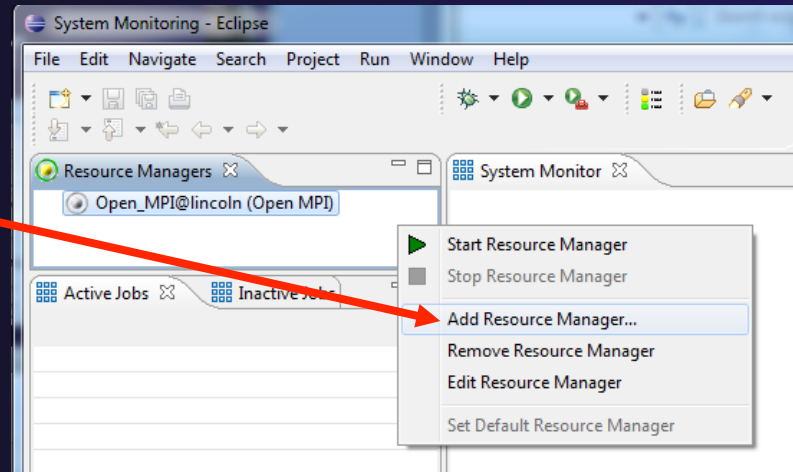


Configuring Job Scheduler

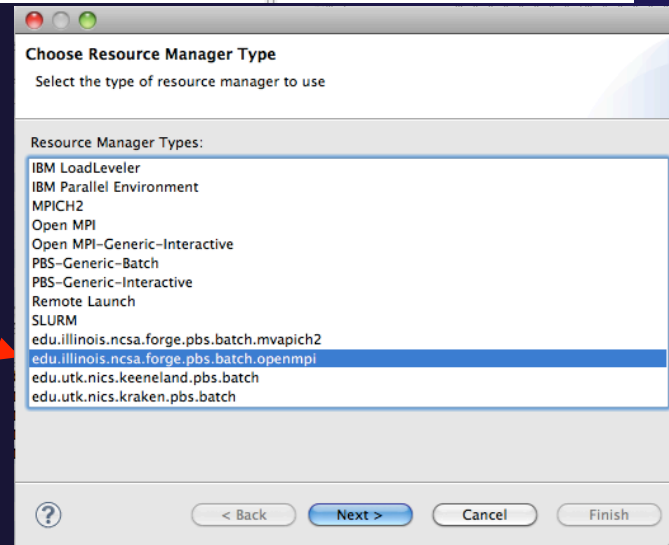


Do this once

- ★ Right-click in Resource Managers view and select **Add Resource Manager**



- ★ Choose Resource Manager Type:
edu.illinois.ncsa.forge
.pbs.batch.openmpi



- ★ Select **Next>**



Configure Control Connection

- ✦ Choose **Remote Tools** for **Remote service provider**
- ✦ Choose the remote connection you made previously
- ✦ Click **Next>**



Do this
once

Control Connection configuration
Enter connection information

Remote service provider: Remote Tools

Connection name: forge

▶ Advanced Options

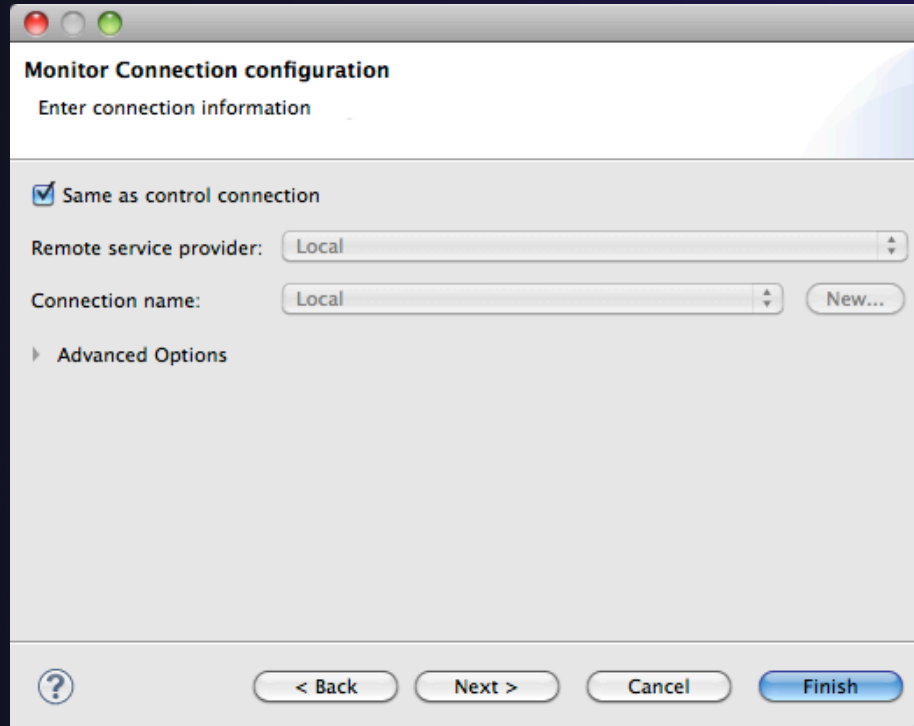
Configure Monitor Connection



- ★ Keep default Monitor Connection (same as Control Connection), click **Next**



Do this
once



The screenshot shows a dialog box titled "Monitor Connection configuration" with the subtitle "Enter connection information". It features a checked checkbox labeled "Same as control connection". Below this, there are two dropdown menus: "Remote service provider:" set to "Local" and "Connection name:" set to "Local". A "New..." button is located to the right of the "Connection name:" dropdown. An "Advanced Options" section is collapsed. At the bottom, there are four buttons: a help icon (?), "< Back", "Next >", "Cancel", and "Finish".



Common Configuration

- ✦ Keep default name
- ✦ Can automatically start Resource Manager (leave unselected today)
- ✦ Click **Finish**



Do this once

Common Resource Manager Configuration
Change any settings for the resource manager

Name and description

Use default name and description:

Name:

Description:

Startup

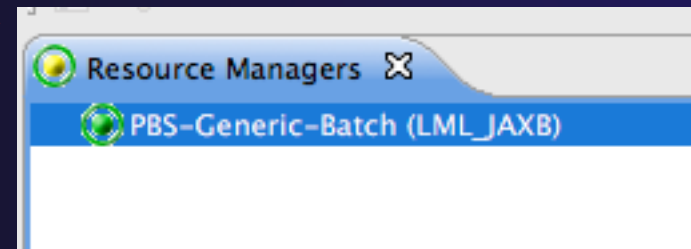
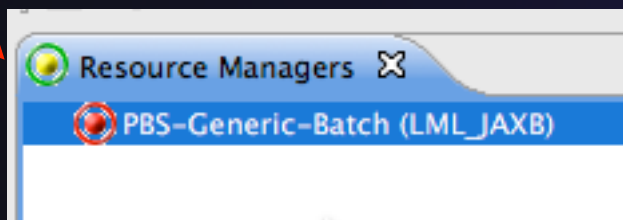
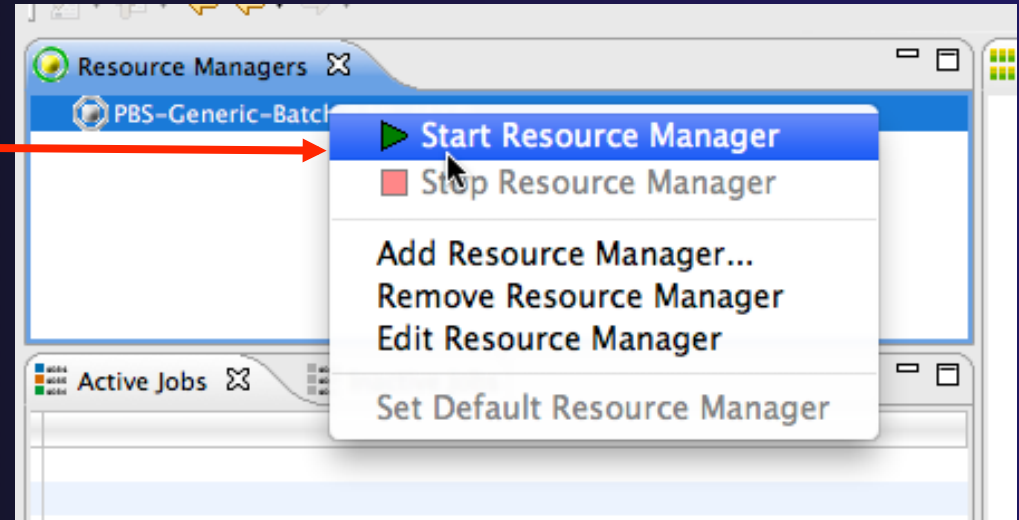
Automatically start resource manager when Eclipse starts

? < Back Next > Cancel Finish



Starting the Resource Manager

- ★ Right click on new resource manager and select **Start resource manager**
- ★ If everything is ok, you should see the resource manager change to **green**
- ★ If something goes wrong, it will change to **red**





System Monitoring

forge.ncsa.illinois.edu

System Monitoring - shallow/Makefile.mk - Eclipse - /Users/beth/ews/test1027-sc11-tutorial

Resource Managers: PBS- Generic-Batch (LML_JAXB)

Select Resource Manager

step	owner	queue	wall	queued	dispatc	totalcores	status
3716.fsched	alberto	normal	43200	201...	2011...	18	RUNNING
3718.fsched	dsouth	normal	14400	201...	2011...	6	RUNNING
3719.fsched	dsouth	normal	14400	201...	2011...	4	RUNNING
3720.fsched	alberto	normal	43200	201...	2011...	18	RUNNING
3722.fsched	dhguo	normal	12600	201...	2011...	1	RUNNING
3723.fsched	cheatham	normal	43200	201...	2011...	16	RUNNING
3724.fsched	cheatham	normal	43200	201...	2011...	16	RUNNING
3725.fsched	cheatham	normal	43200	201...	2011...	16	RUNNING
3726.fsched	cheatham	normal	43200	201...	2011...	16	RUNNING
3727.fsched	cheatham	normal	43200	201...	2011...	16	RUNNING
3728.fsched	cheatham	normal	43200	201...	2011...	16	RUNNING
3729.fsched	cheatham	normal	43200	201...	2011...	16	RUNNING
3730.fsched	cheatham	normal	43200	201...	2011...	16	RUNNING
3731.fsched	cheatham	normal	43200	201...	2011...	16	RUNNING

Console: CDT Build Console [shallow]

```
**** Build of configuration Remote for project shallow ****
make -f Makefile.mk all
mpicc -g -c -o calc.o calc.c
mpicc -g -c -o copy.o copy.c
```

One node with 16 cores

- ★ **System** view, with abstraction of nodes for selected Resource Manager
- ★ Active and inactive jobs
- ★ Hover over node in **System** view to see job running on node in **Active Jobs** view
- ★ Hold mouse button down on a job in **Active Jobs** view to see where it is running in **System** view

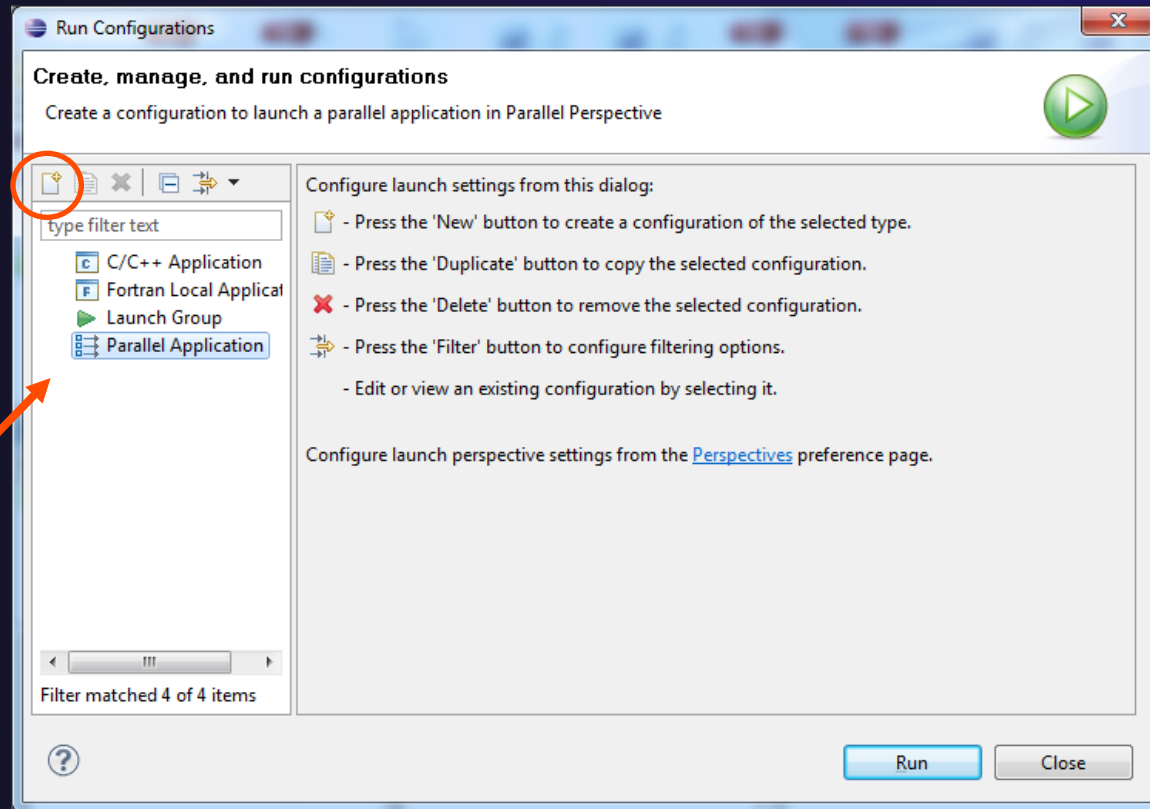
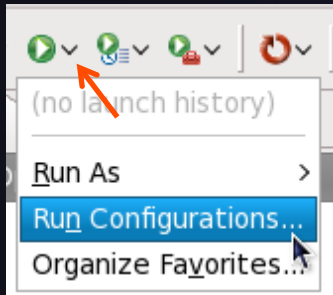
Running the Program (Launching a Job)




Do this once



Create 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

Depending on which flavor of Eclipse you installed, you might have more choices in Application types

Note: we sometimes interchange the terms "Run Configuration" and "Launch Configuration"

Complete the Resources Tab



Do this once



- ✦ Enter a name for this run configuration, e.g. “shallow-pbs-batch”
- ✦ In **Resources** tab, select the PBS resource manager you just created (edu.illinois.ncsa.forge....)
- ✦ Select the destination queue – **debug**
- ✦ The **MPI Command** field allows this job to be run as an MPI job
 - ✦ Choose **mpirun**
- ✦ Enter the resources needed to run this job
 - ✦ Use 1 nodes, 4 cores (MPI tasks)

The screenshot shows the 'Run Configurations' dialog box with the 'Resources' tab selected. The configuration is for a job named 'shallow-pbs-batch' using the 'edu.illinois.ncsa.forge.pbs.batch.openmpi' resource manager. The 'Queue' is set to 'debug', the 'Number of nodes' is 1, and the 'MPI Command' is 'mpirun'. The 'MPI Number of Cores' is set to 4. The 'Export Environment' checkbox is checked.

Name	Value	Description
Job Name:	ptp_job	The name assigned to the job by the qsub or qalter command.
Account:		Account to which to charge this job.
Queue:	debug	Designation of the queue to which to submit the job.
Number of nodes:	1	Number and/or type of nodes to be reserved for exclusive use.
Total Memory Needed:		Maximum amount of memory used by all concurrent processes.
Wallclock Time:	00:05:00	Maximum amount of real time during which the job can be run.
MPI Command:	mpirun	Which mpi command to use.
MPI Number of Cores:	4	the '-np' value
Export Environment:	<input checked="" type="checkbox"/>	All variables in the qsub command's environment are to be exported.

Buttons at the bottom: View Script, View Configuration, Restore Defaults, Apply, Revert, Close, Run.

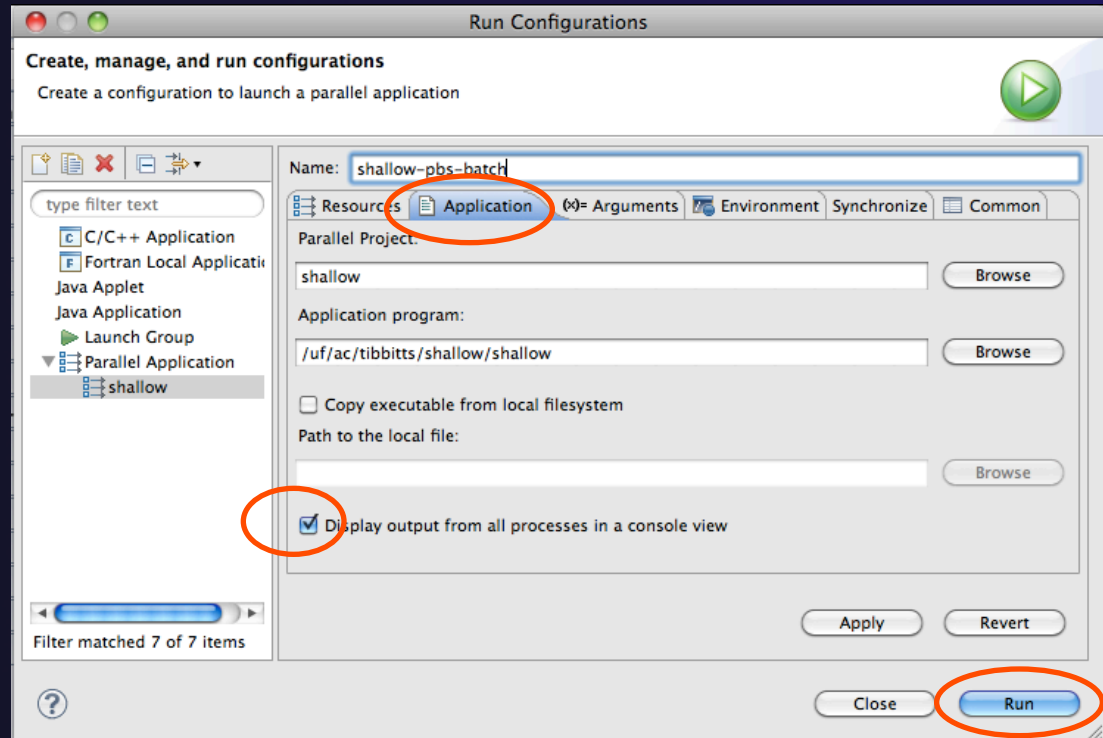
Complete the Application Tab



Do this once



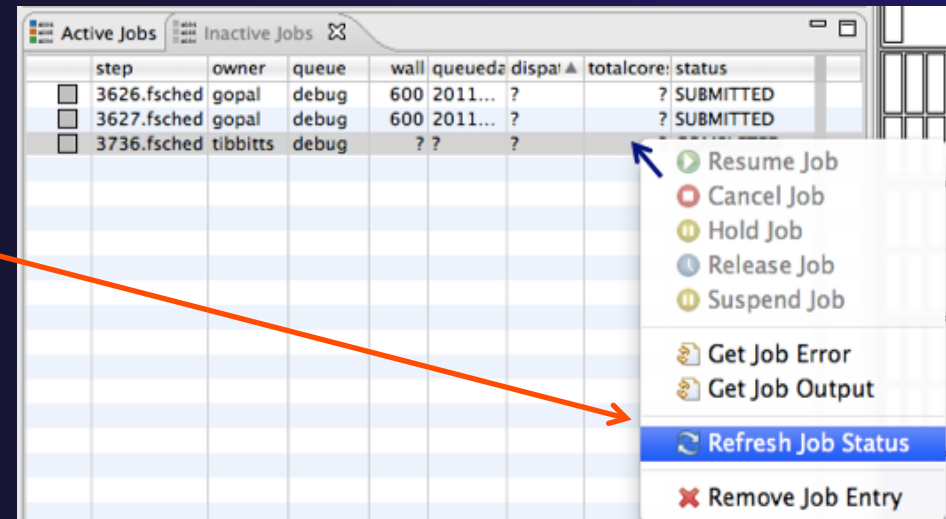
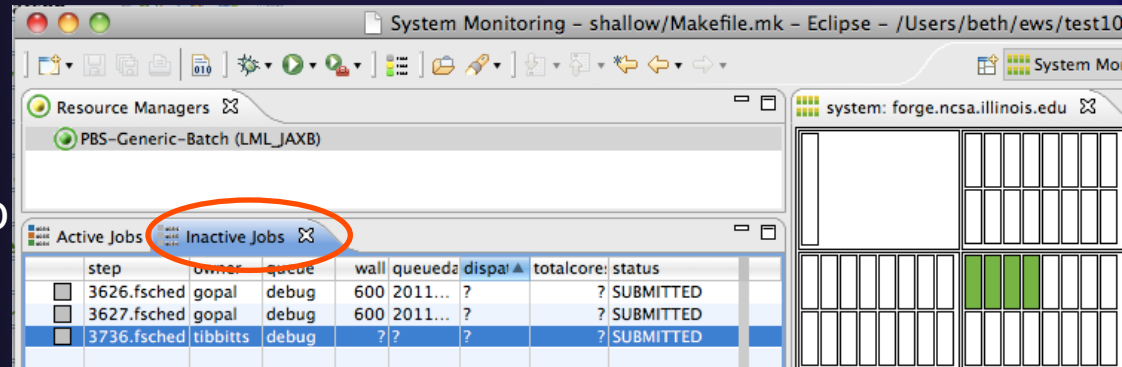
- ★ Select the **Application** tab
- ★ Choose the **Application program** by clicking the **Browse** button and locating the executable on the remote machine
 - ★ Use the same “shallow” executable
- ★ Select **Display output from all processes in a console view**
- ★ Click **Run** to submit the application to the job scheduler





Job Monitoring

- Job initially appears in “Inactive Jobs”, then in “Active Jobs”, then returns to Inactive on completion
- This short-running program may not run long enough to appear in “Active Jobs”
- Status refreshes automatically every 60 sec
Or force refresh with menu
- After status = COMPLETED, Can view output or error by right clicking on job, selecting appropriate output





Job Output

- After status = COMPLETED, Can view output or error by right clicking on job, selecting appropriate output
- Output/Error info shows in Console View

step	owner	queue	wall	queuedate	dispatch	total	status
3626.fsched	gopal	debug	600	2011-1...	?	?	SUBMITTED
3627.fsched	gopal	debug	600	2011-1...	?	?	SUBMITTED
3769.fsched	alberto	normal	43...	2011-1...	?	18	SUBMITTED
3774.fsched	dsouth	normal	14...	2011-1...	?	12	SUBMITTED
3772.fsched	tibbitts	?	??	?	?	?	COMPLETED
3773.fsched	tibbitts	debug	??	??	?	?	COMPLETED
3777.fsched	tibbitts	debug	??	??	?	?	COMPLETED
3783.fsched	tibbitts	debug	??	??	?	?	COMPLETED

Console View

```

/ur/ae/tibbitts/ptp_job.o3777

Potential energy      6.505  Kinetic Energy
Total Energy         48032.016  Pot. Enstrophy

Cycle number  950  Model time in days  0.9
Potential energy      760.460  Kinetic Energy
Total Energy         48385.996  Pot. Enstrophy

Cycle number  1000  Model time in days  1.0
Potential energy     21561.033  Kinetic Energy
Total Energy         48000.496  Pot. Enstrophy
  
```

Building before Run



Do this once

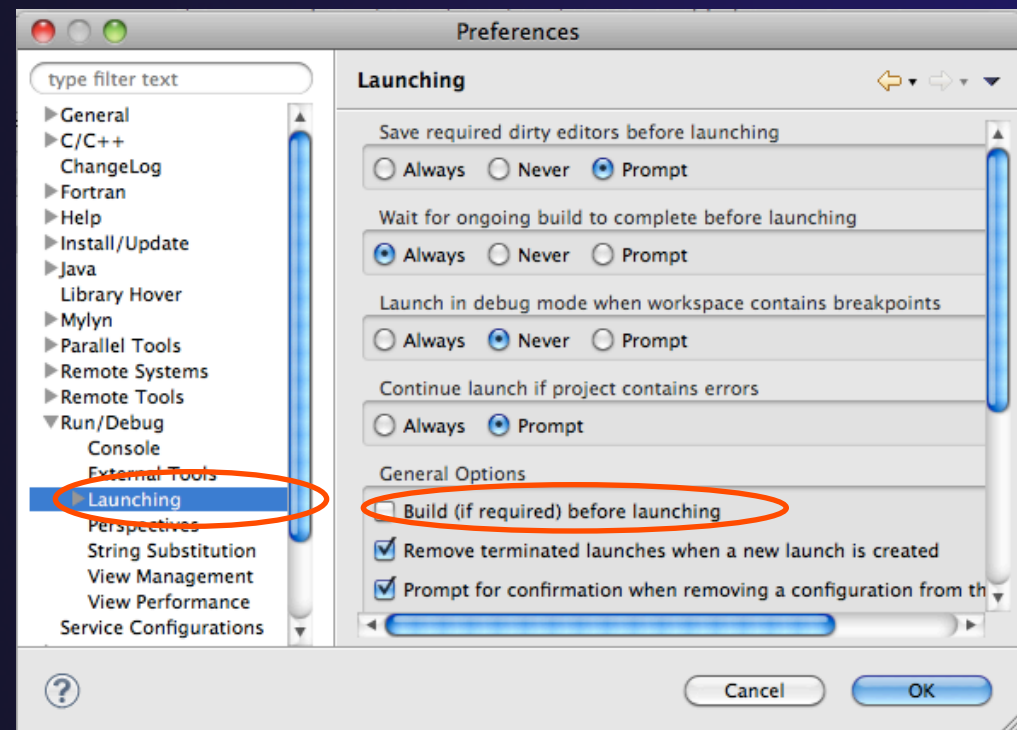
★ If projects build prior to launch, you can turn it off.

★ Go into **Preferences>Run/Debug** and click on **Launching**.

★ Uncheck "**Build (if required) before launching**"

★ Should be set by default now

To bring up **Preferences** dialog, use Window>Preferences or Mac: Eclipse>Preferences





Exercise

- ✦ Start with your 'shallow' project
- ✦ Create and start Resource Manager
- ✦ Build; Run shallow
- ✦ See results
- ✦ Change something
 - ✦ Change m and n in decs.h
- ✦ Rebuild and re-run

Advanced Features

Searching
Fortran
Refactoring

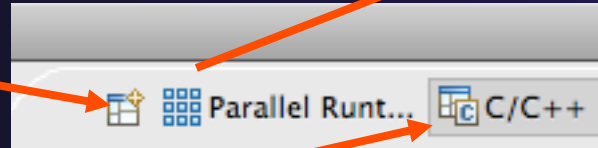
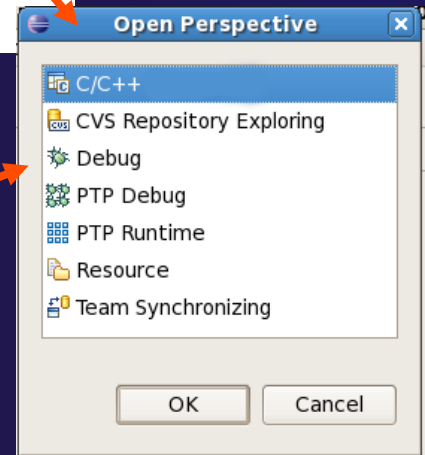
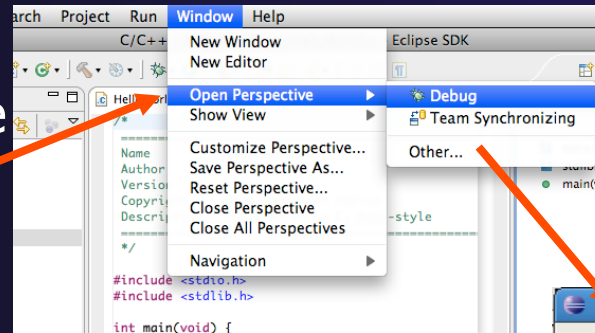
Searching



Switching Perspectives

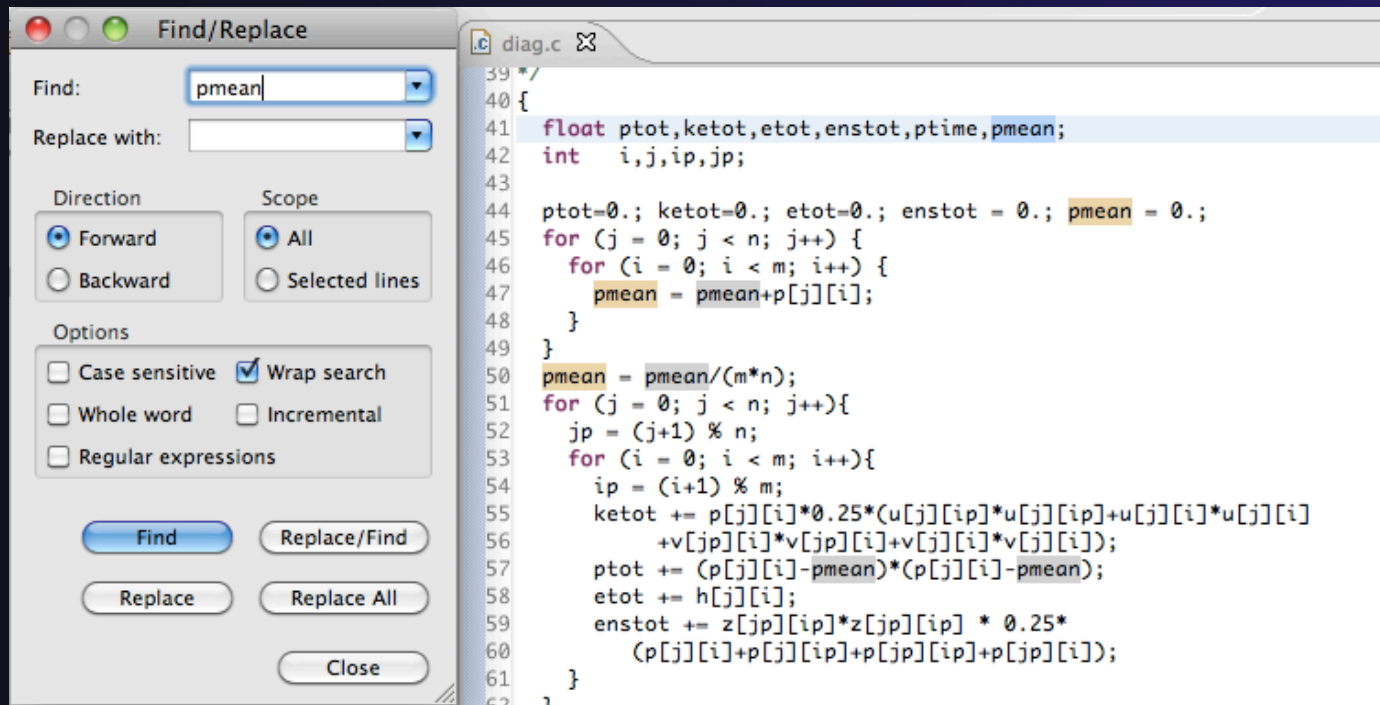
✦ Switch to C/C++ Perspective one of three ways:

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



Find/Replace within Editor

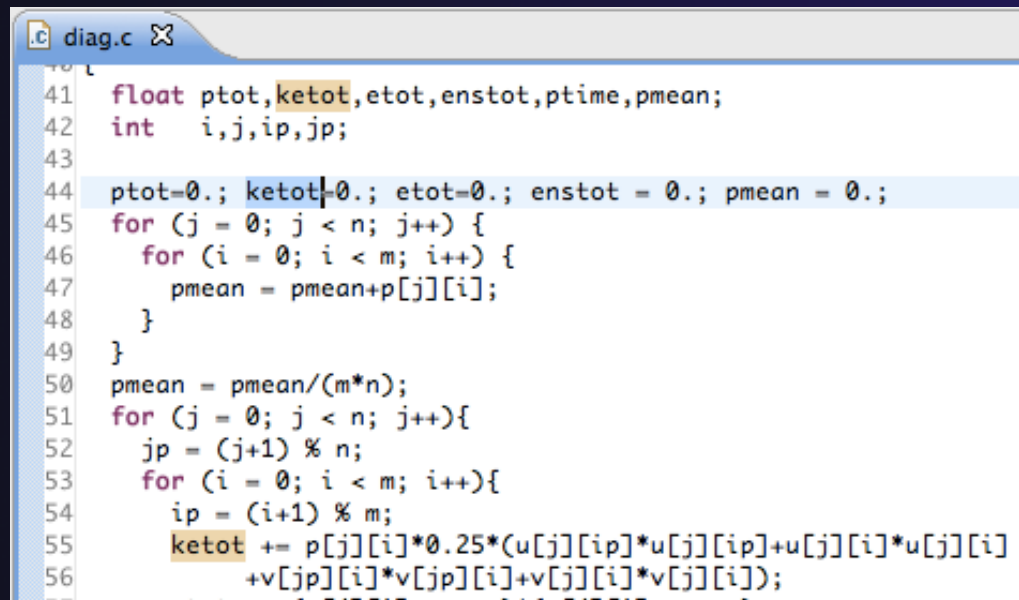
- ✦ Simple Find within editor buffer
- ✦ Ctrl-F (Mac: Command-F)



Mark Occurrences

(C/C++ Only)

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

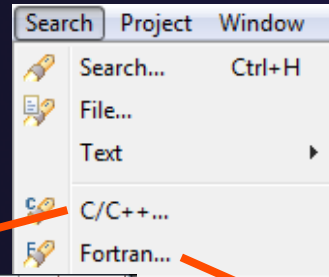


```
diag.c
41 float ptot, ketot, etot, enstot, ptime, pmean;
42 int i, j, ip, jp;
43
44 ptot=0.; ketot=0.; etot=0.; enstot = 0.; pmean = 0.;
45 for (j = 0; j < n; j++) {
46     for (i = 0; i < m; i++) {
47         pmean = pmean+p[j][i];
48     }
49 }
50 pmean = pmean/(m*n);
51 for (j = 0; j < n; j++){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

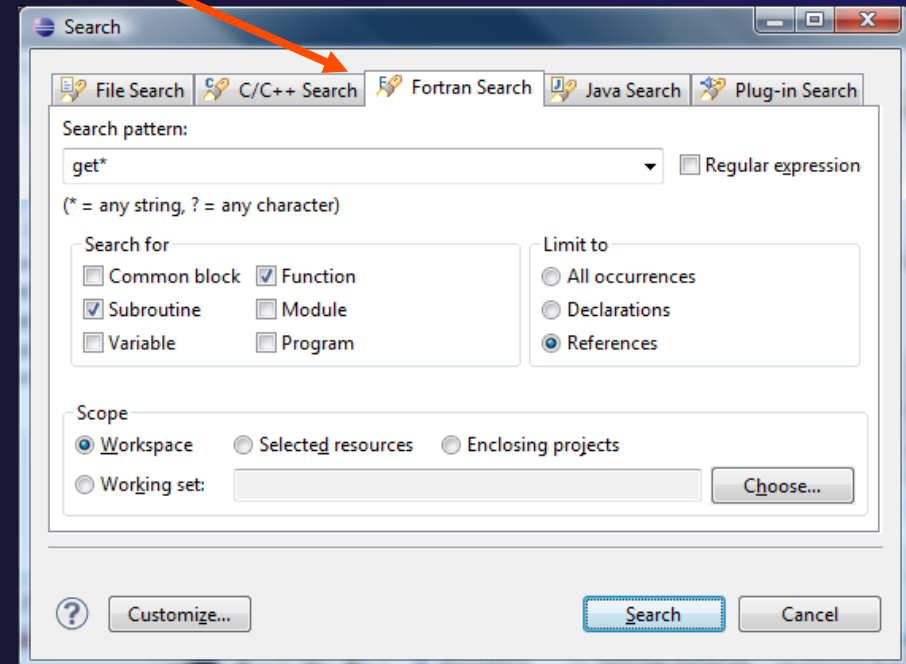
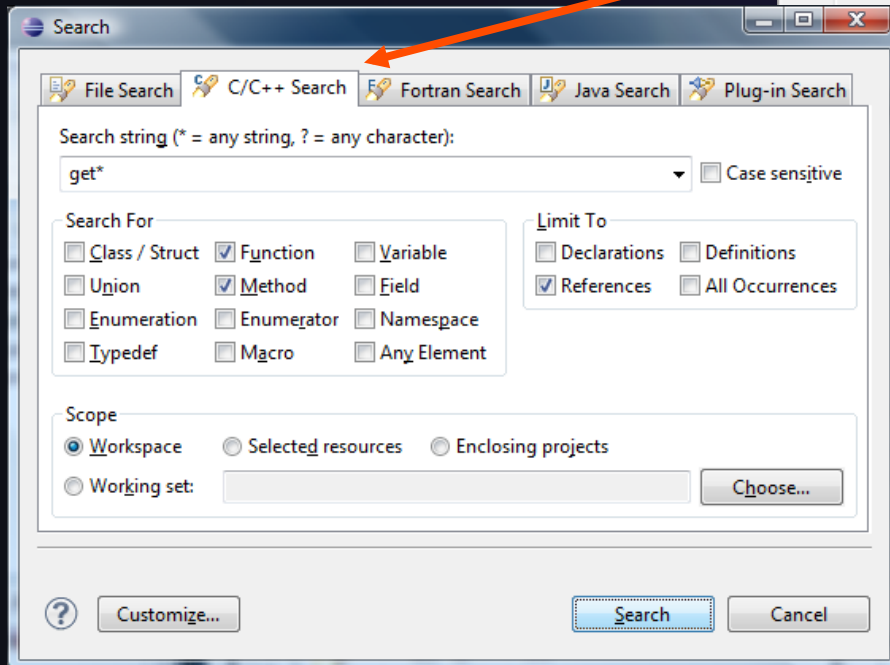

Language-Based Searching

(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



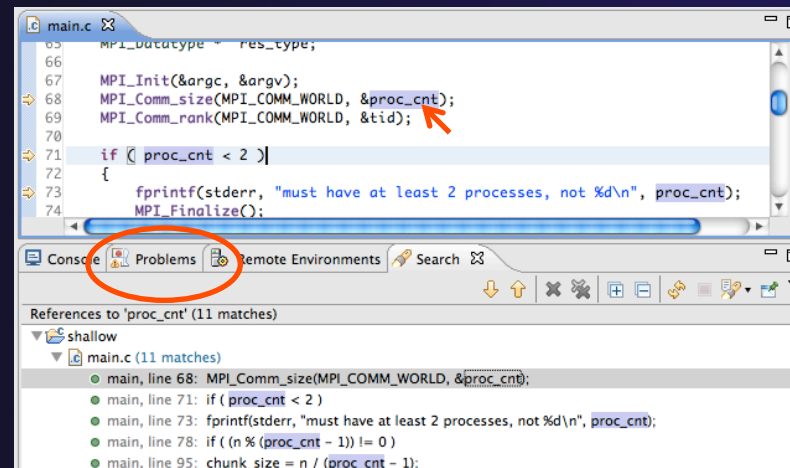
Find References

(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

```

134
135 initialise(p, u, v, psi, pold, uold, vold, di, dj, z);
136 diag(1, 0., p, u, v, h, z);
137
138 for (i = 1; i < proc_cnt; i++)
139     for (j = 0; j < n; j++)
140         acopy_two_to_one(i, j, p, u, v, h, z);
141     MPI_Send(&star, 1, MPI_COMM_WORLD, j, 0);
142
143
144     acopy_two_to_one(i, j, p, u, v, h, z);
145     MPI_Send(&star, 1, MPI_COMM_WORLD, j, 0);
146
  
```

Goes to its declaration in copy.c

```

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



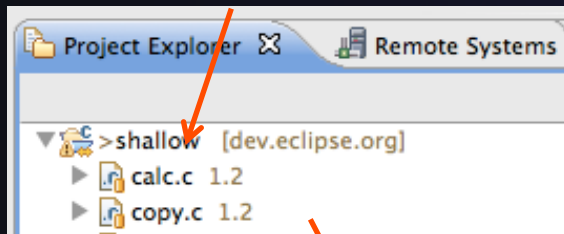
Search – Try It!

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

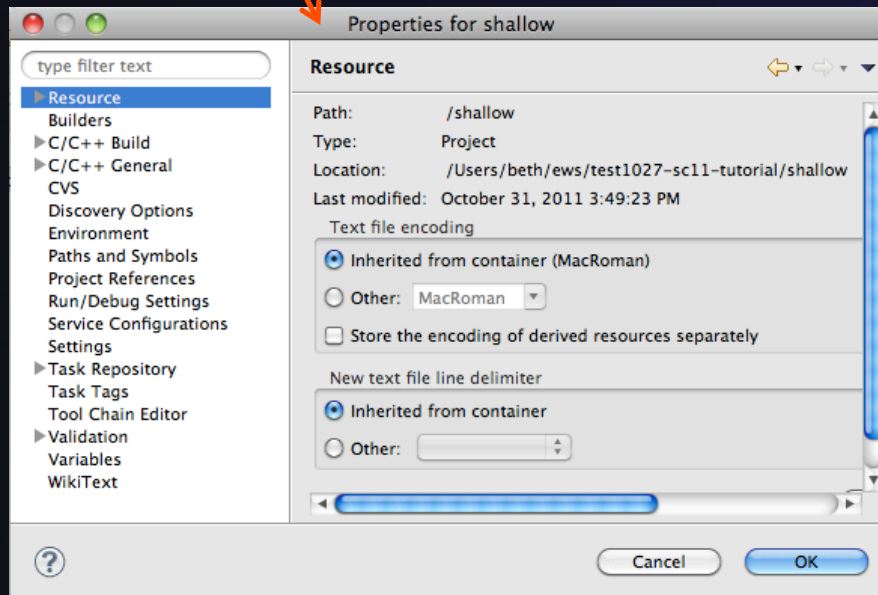
Fortran Specifics

Project Properties

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



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

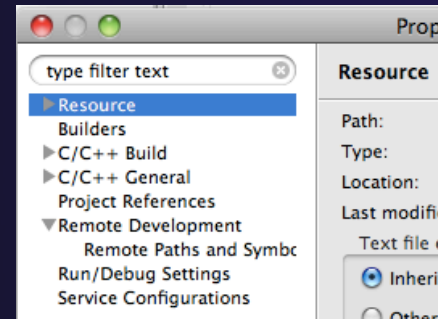


- ★ Contrast with *workspace preferences*, which are the same regardless of what project is being edited
 - ★ e.g., editor colors
 - ★ Set in **Window ► Preferences** (on Mac, **Eclipse ► Preferences**)
 - ★ Careful! Dialog is very similar

Converting to a Fortran Project

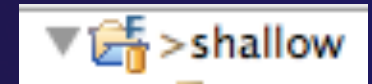
- ★ Are there categories labeled **Fortran General** and **Fortran Build** in the project properties?

No Fortran categories →



Do this once

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

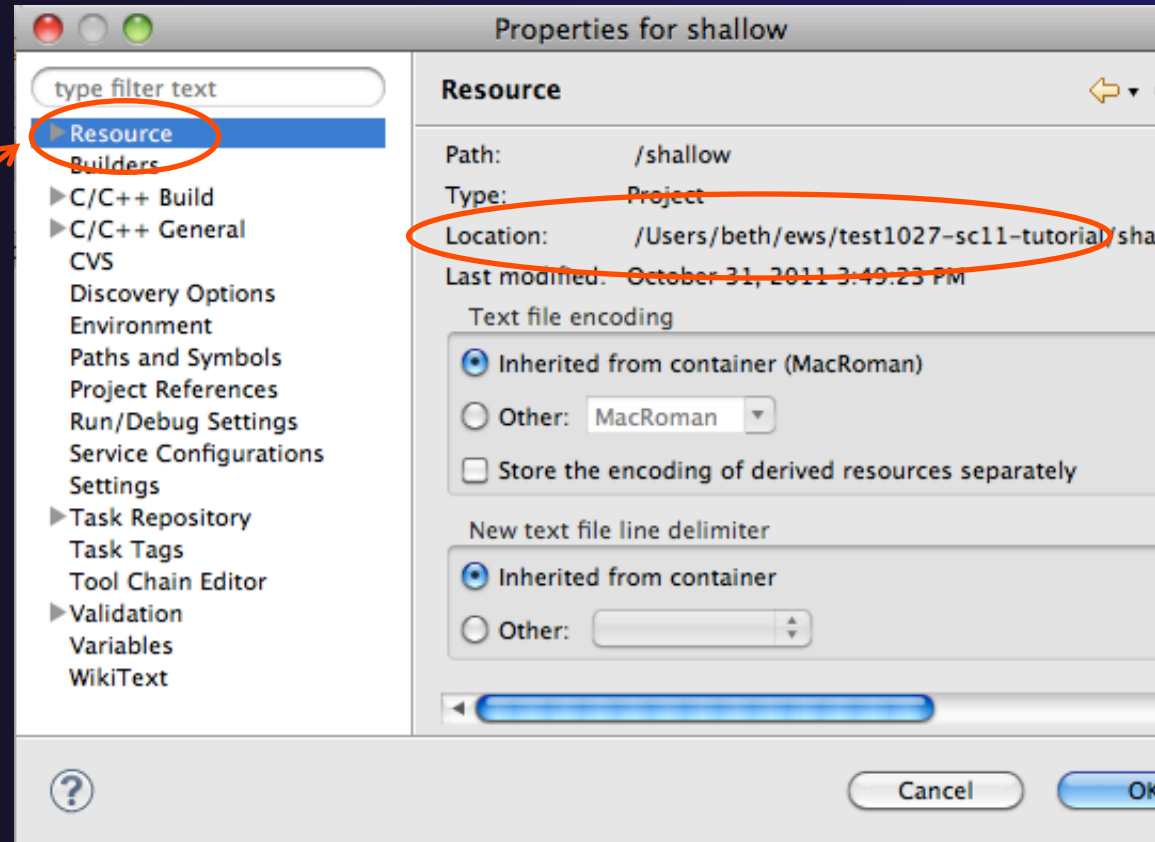


- ★ *Every* Fortran project is also a C/C++ Project

Project Location

★ How to tell where a project resides?

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

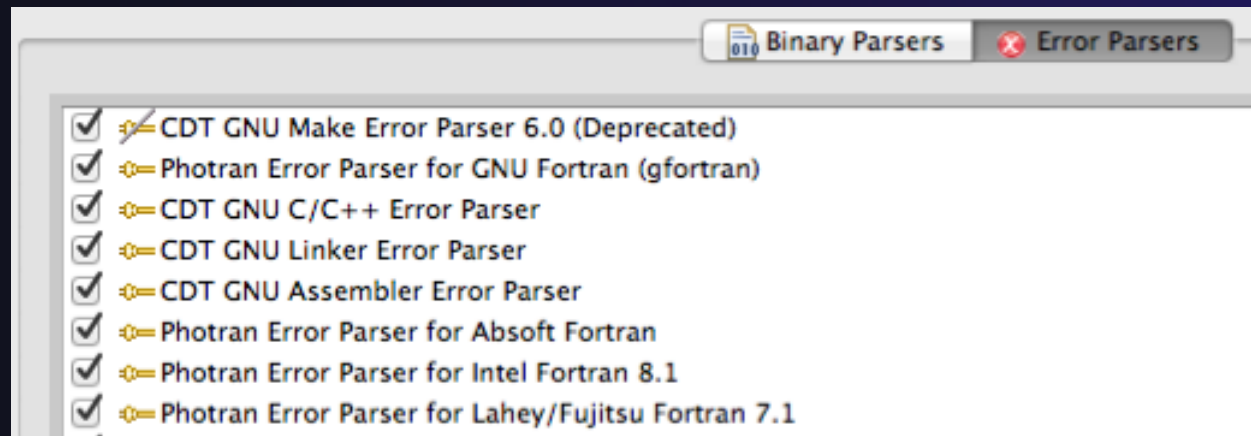


Error Parsers

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



Do this
once



Fortran Source Form Settings

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

- ★ Source form determined by filename extension
- ★ Defaults are similar to most Fortran compilers:

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

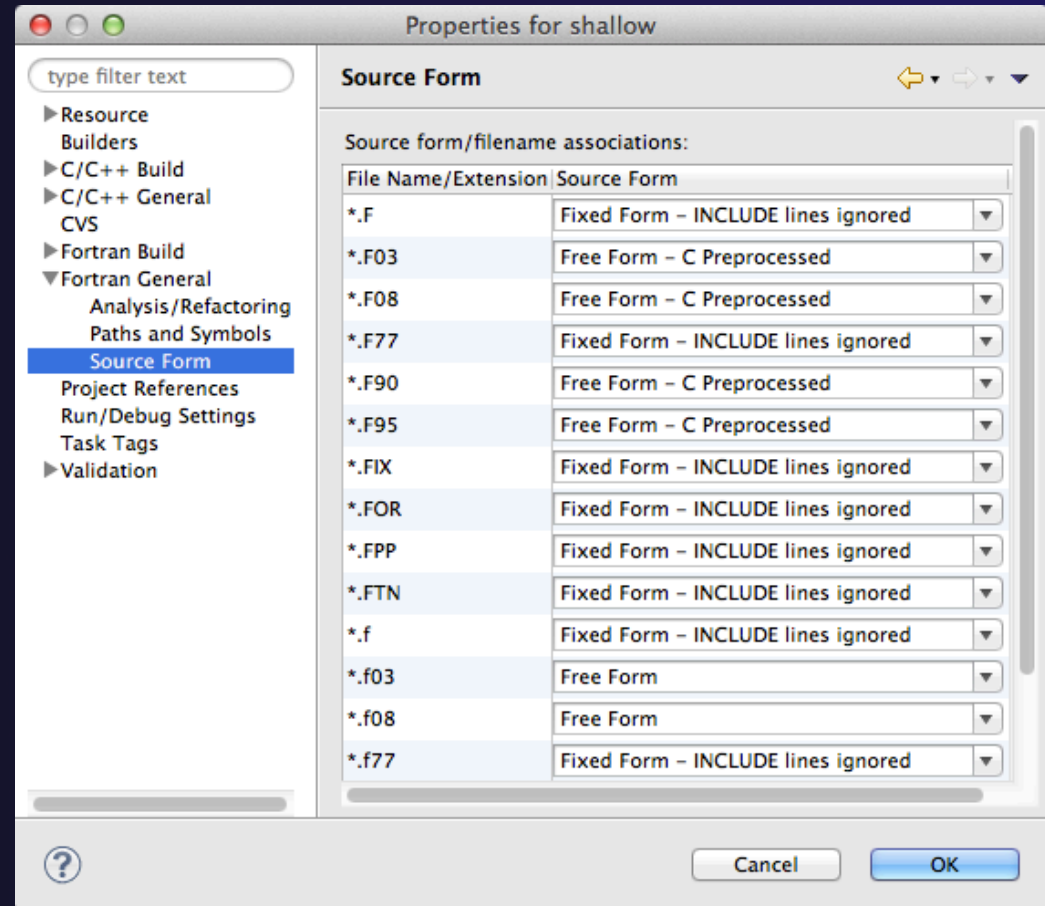
- ★ Many features *will not work* if filename extensions are associated with the wrong source form (outline view, content assist, search, refactorings, etc.)

Fortran Source Form Settings



Do this
once

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

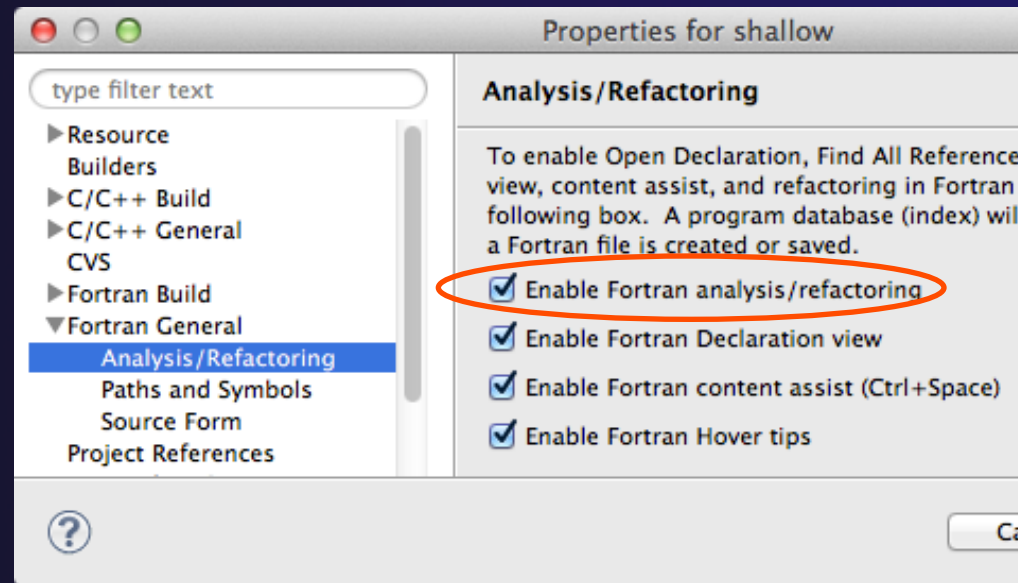


Enabling Fortran Advanced Features

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



Do this once





Project Properties – Try It!

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

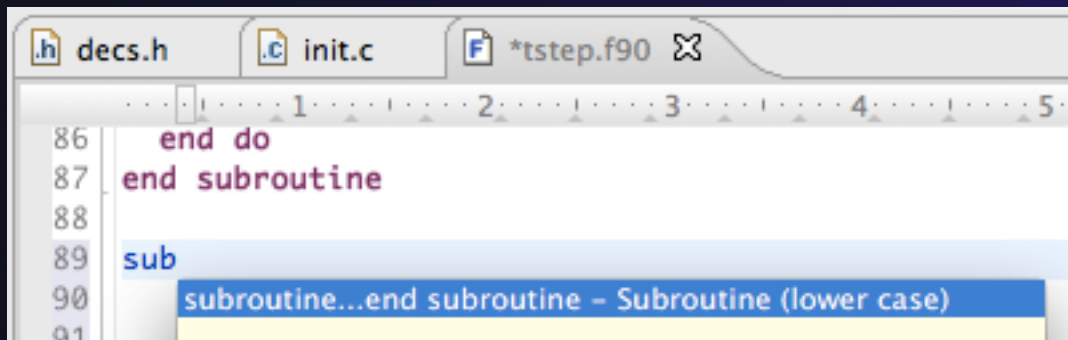
Advanced Editing

Code Templates

Code Templates

(C/C++ and Fortran)

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



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

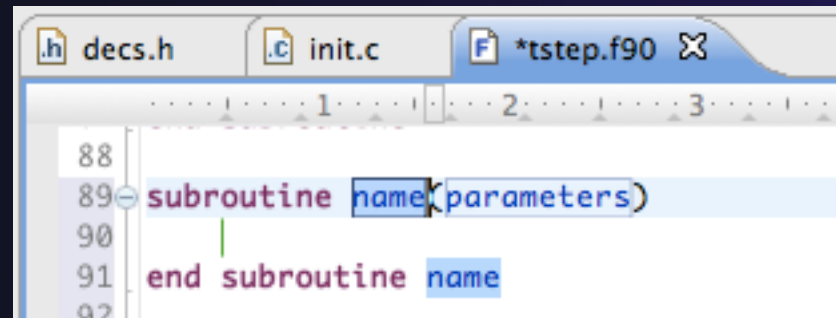
```
86   end do
87   end subroutine
88
89   sub
90   subroutine...end subroutine - Subroutine (lower case)
91
```

A dropdown menu is visible below the 'sub' keyword on line 89, showing the suggestion 'subroutine...end subroutine - Subroutine (lower case)'.

Code Templates (2)

(C/C++ and Fortran)

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



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

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

The words 'name' and 'parameters' in line 89, and 'name' in line 91, are highlighted in blue, indicating they are completion fields.

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



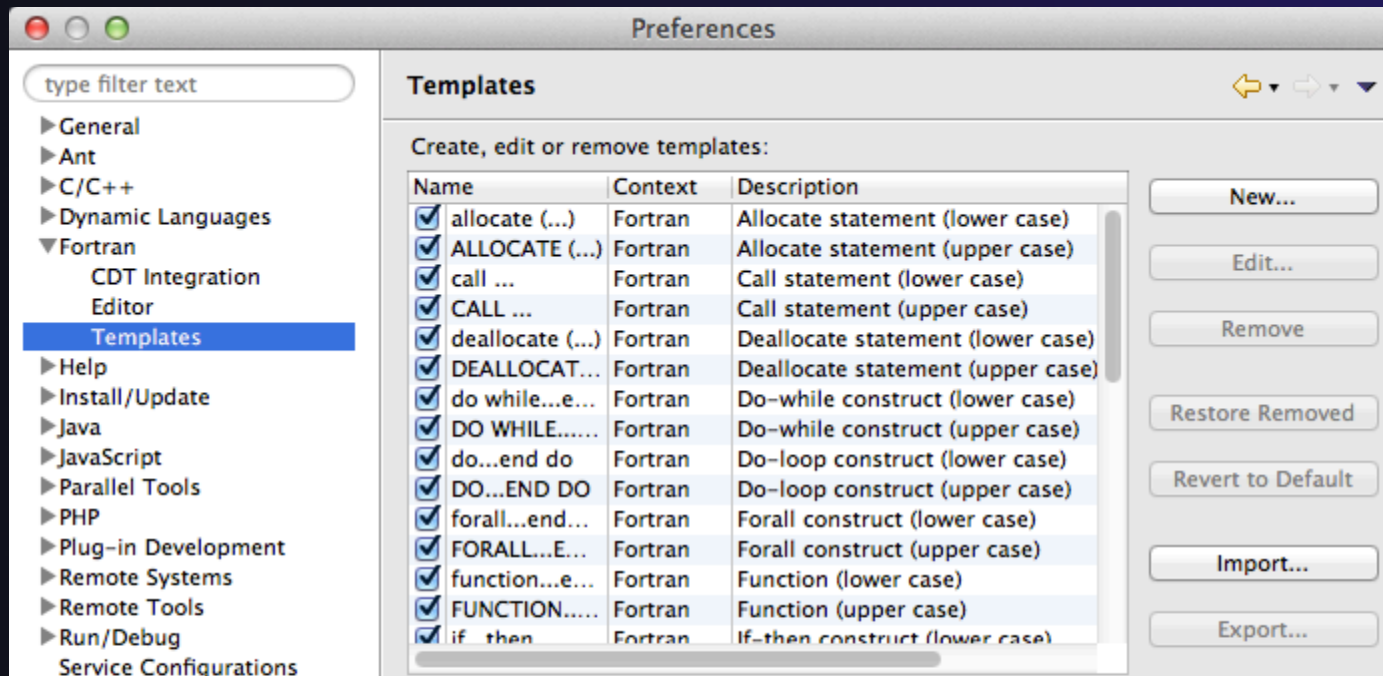
Advanced Editing – Try It!

- ✦ 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**

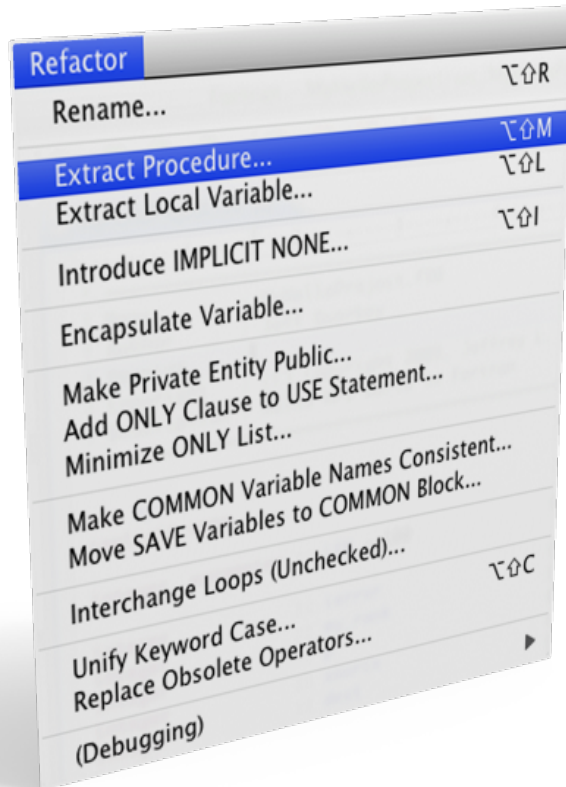


- ★ Can import/export templates to XML files

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

- ★ Refactor menu will be empty if

- ★ Refactoring not enabled in project properties

(recall from earlier that it is enabled in

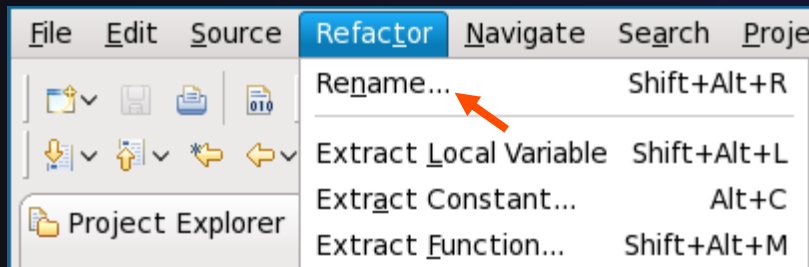
Project Properties: Fortran General ▶ Analysis/Refactoring)

- ★ The file in the active editor is fixed form
 - ★ The file in the active editor is preprocessed

Rename Refactoring

(also available in Fortran)

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



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

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

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

Rename in File

(C/C++ Only)

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

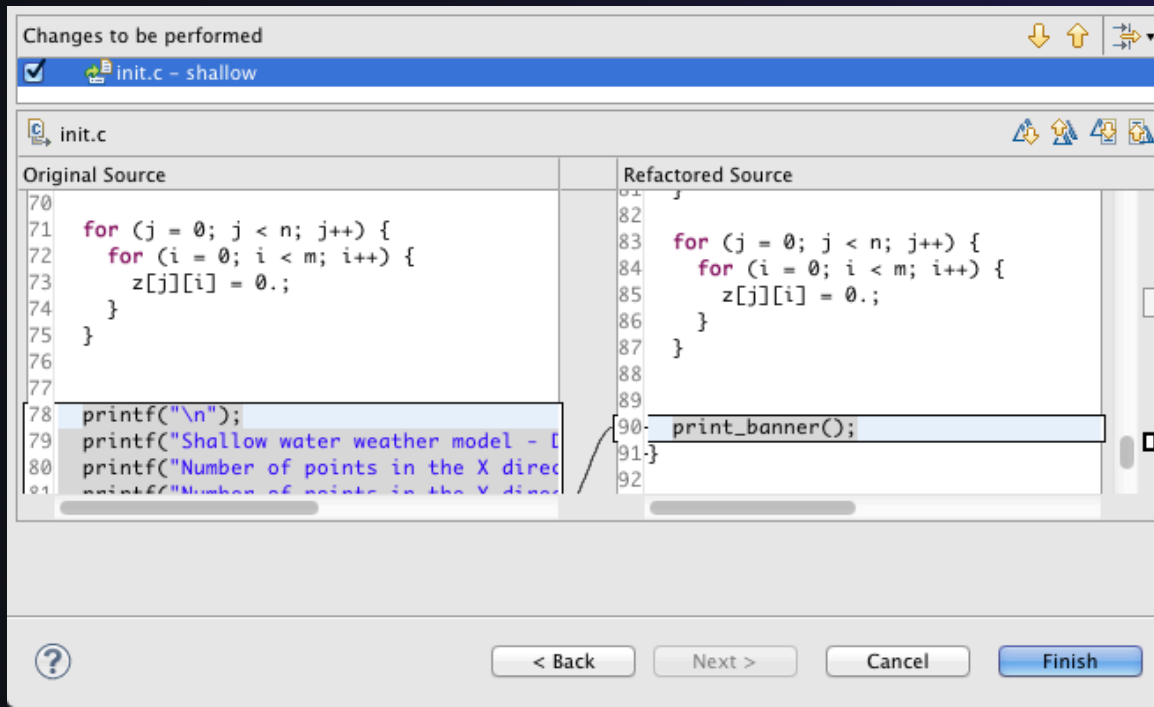


```
worker.c X
306 time_unload(prv,nxt,tu_my_id,
307     int prv;
308     int nxt;
309     int tu_my_id;
310     int jstart;
311     int jend;
312     float  dvdt[n][m];
313 {
314     neighbour_send(nxt, tu_my.
315     neighbour_receive(prv, tu.
316 }
317
318 /*
319 this is a general purpose fun
320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

Extract Function Refactoring

(also available in Fortran - "Extract Procedure")

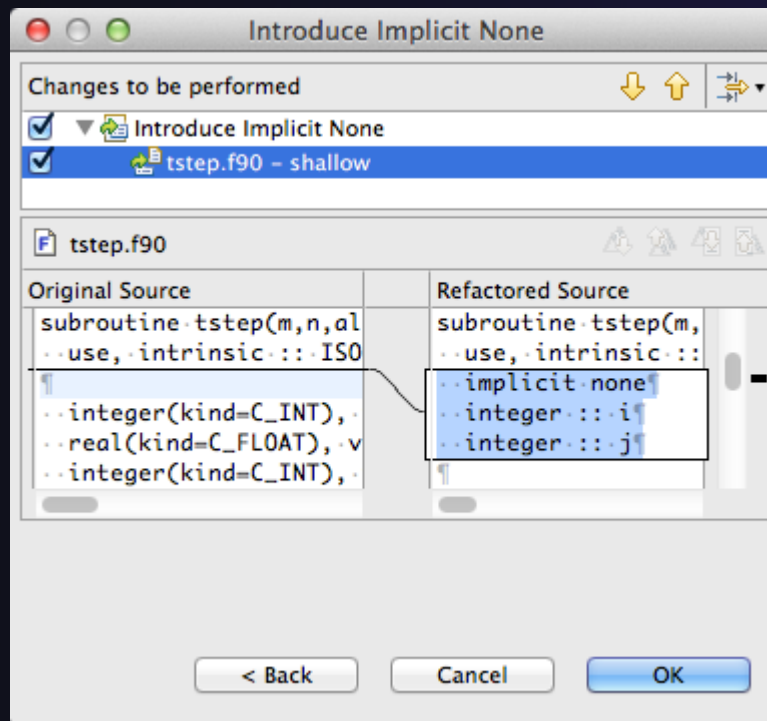
- ✦ Moves statements into a new function, replacing the statements with a call to that function
- ✦ Local variables are passed as arguments



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

Introduce IMPLICIT NONE Refactoring

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

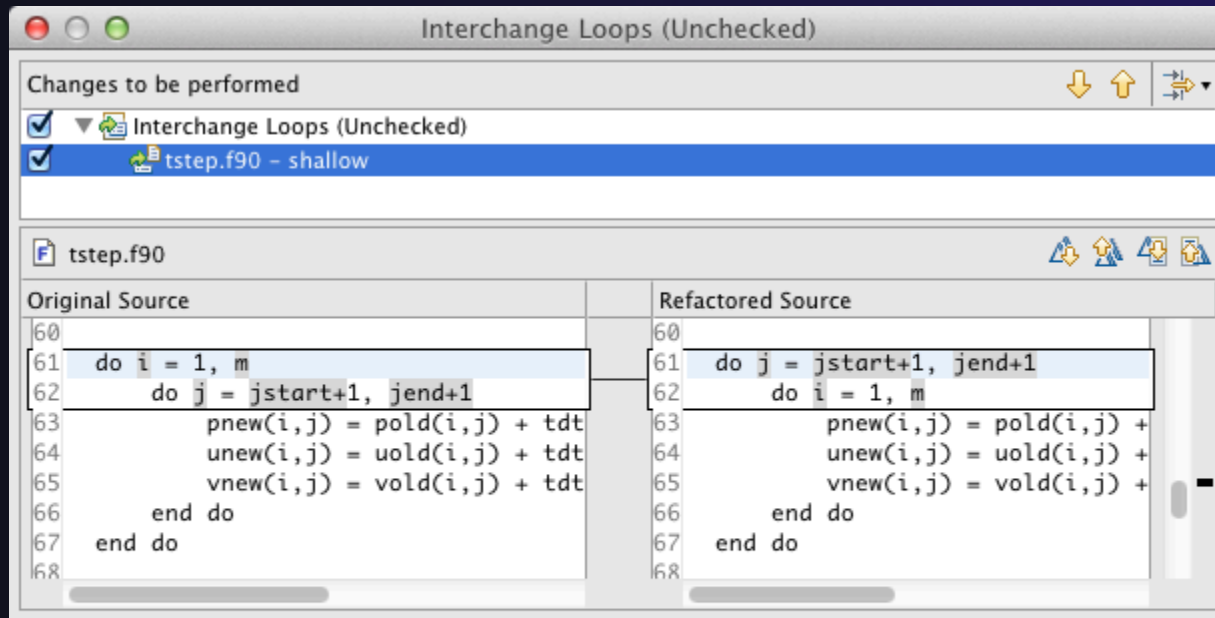


- ✦ Introduce in a single file by opening the file and selecting **Refactor ▶ Coding Style ▶ Introduce IMPLICIT NONE...**
- ✦ Introduce in multiple files by selecting them in the Project Explorer view, right-clicking on the selection, and choosing **Refactor ▶ Coding Style ▶ Introduce IMPLICIT NONE...**

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



Unroll 4x

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

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



Refactoring & Transformation – Try It!

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 Blue Waters HPC Workbench

- ★ Tools for NCSA Blue Waters
 - ★ <http://www.ncsa.illinois.edu/BlueWaters/>
 - ★ Sustained Petaflop system
- ★ Based on Eclipse and PTP
- ★ Includes some related tools
 - ★ Performance tools
 - ★ Workflow tools (<https://wiki.ncsa.uiuc.edu/display/MRD+Public+Space+Home+Page>)
- ★ Part of the enhanced computational environment described at:
<http://www.ncsa.illinois.edu/BlueWaters/ece.html>



NSF SI2 Workbench for High Performance Computing

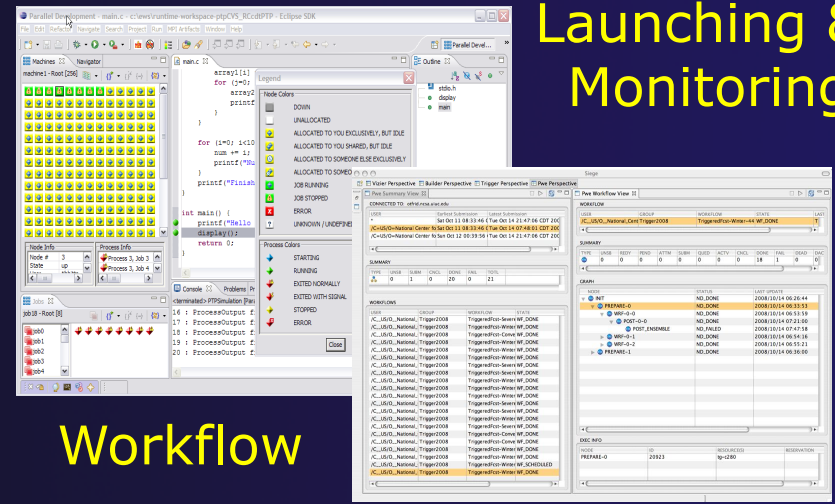
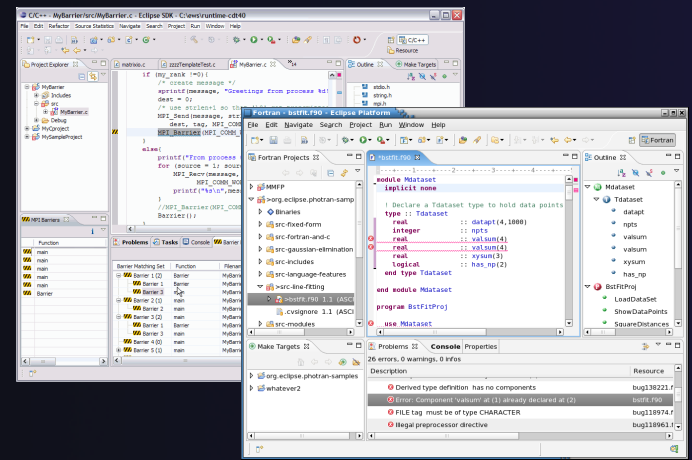
- ★ “SI2-SSI Productive and Accessible Development Workbench for HPC Applications”, which is supported by the National Science Foundation under award number OCI 1047956
- ★ Produce a productive and accessible development workbench using Eclipse PTP
- ★ Key Components
 - ★ Determining Requirements, Ensuring Impact
 - ★ Make improvements to Eclipse PTP
 - ★ Engineering Process
 - ★ Metrics
 - ★ Outreach/Training/Education

NCSA HPC Workbench

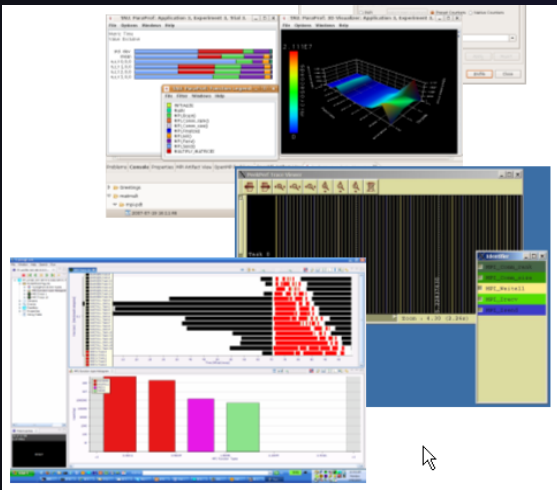
PTP

Launching & Monitoring

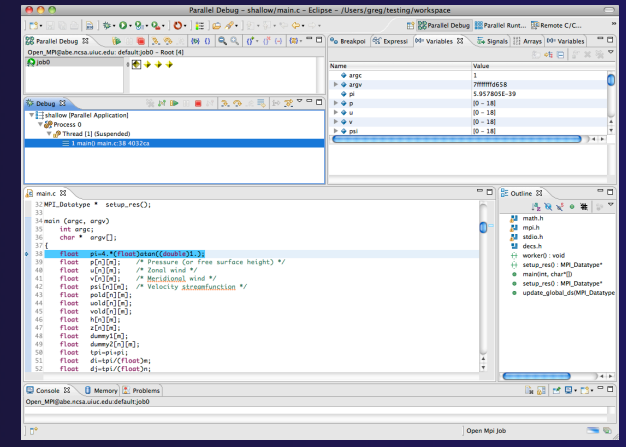
Coding & Analysis (C/C++, Fortran)



Workflow



Performance Tuning



Parallel Debugger

Planned PTP Future Work

- ★ Scalability improvements
 - ★ UI to support 1M processes
 - ★ Very large application support
- ★ Usability improvements
 - ★ New wizard to improve setup experience
 - ★ Ability to share configuration information
- ★ Resource Managers
 - ★ More implementations of configurable resource managers
- ★ Synchronized project improvements
 - ★ Conversion wizard
 - ★ Resolving merge conflicts

Useful Eclipse Tools

- ✦ Linux Tools (autotools, valgrind, Oprofile, Gprof)
 - ✦ <http://eclipse.org/linuxtools>
- ✦ Python
 - ✦ <http://pydev.org>
- ✦ Ruby
 - ✦ <http://www.apтана.com/products/radrails>
- ✦ Perl
 - ✦ <http://www.epic-ide.org>
- ✦ Git
 - ✦ <http://www.eclipse.org/egit>
- ✦ VI bindings
 - ✦ Vrappер (open source) - <http://vrappер.sourceforge.net>
 - ✦ viPlugin (commercial) - <http://www.viplugin.com>

Online Information

- ★ Information about PTP
 - ★ Main web site for downloads, documentation, etc.
 - ★ <http://eclipse.org/ptp>
 - ★ Wiki for designs, planning, meetings, etc.
 - ★ <http://wiki.eclipse.org/PTP>
 - ★ Articles and other documents
 - ★ <http://wiki.eclipse.org/PTP/articles>

- ★ Information about Photran
 - ★ Main web site for downloads, documentation, etc.
 - ★ <http://eclipse.org/photran>
 - ★ User's manuals
 - ★ <http://wiki.eclipse.org/PTP/photran/documentation>

Mailing Lists

★ PTP Mailing lists

- ★ Major announcements (new releases, etc.) - low volume
 - ★ <http://dev.eclipse.org/mailman/listinfo/ptp-announce>
- ★ User discussion and queries - medium volume
 - ★ <http://dev.eclipse.org/mailman/listinfo/ptp-user>
- ★ Developer discussions - high volume
 - ★ <http://dev.eclipse.org/mailman/listinfo/ptp-dev>

★ Photran Mailing lists

- ★ User discussion and queries
 - ★ <http://dev.eclipse.org/mailman/listinfo/photran>
- ★ Developer discussions –
 - ★ Also on ptp-dev list (see above)

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 will only succeed with your participation!