



parallel tools platform

<http://eclipse.org/ptp>

# Developing Scientific Applications Using Eclipse and the Parallel Tools Platform

Greg Watson, IBM  
g.watson@computer.org

Jay Alameda, NCSA  
jalameda@ncsa.uiuc.edu

Beth Tibbitts, IBM  
tibbitts@us.ibm.com

Jeff Overbey, UIUC  
overbey2@illinois.edu

July 7, 2010

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 Blue Waters sustained petascale computing project, which is supported by the National Science Foundation under award number OCI 07-25070.

# Eclipse Parallel Tools Platform (PTP)

- ★ Material adapted from SC09 tutorial
- ★ Tutorial source modules indicated in next slide
- ★ All PTP tutorials posted to
  - ★ <http://wiki.eclipse.org/ptp>
- ★ Primary changes: restrict scope to functions that work remotely
- ★ Eclipse 3.6 (Helios) and PTP 4.0 released June 23, 2010
- ★ First coordinated release of Eclipse PTP!
- ★ First release with significant improvement of usability of remote capabilities

# Topics covered today

- ★ Overview
- ★ C/C++/MPI development
- ★ Remote build
- ★ Remote execution
- ★ Remote debugging
- ★ Other features and wrap-up
- ★ More slides than have time (certainly!)
- ★ Not all machines are supported equally well (unfortunately!)
- ★ Please don't hesitate to ask questions – I'd rather have you not get stuck and not get as far, rather than getting stuck early..

# Out of band setup

- ★ Please see your machine's instructions at
  - ★ <https://wiki.ncsa.illinois.edu/display/AAG/AASG+Projects>
- ★ Need to
  - ★ Copy mpi code to your home directory
  - ★ Modify environment to have suitable version of java and MPI
- ★ Verification of correct out of band install
  - ★ `ls -la ~/mpi`
  - ★ `Java -version`
  - ★ `Which mpicc`
- ★ Machine paths at
  - ★ <https://wiki.ncsa.illinois.edu/display/AAG/AASG+Projects>

# Module 3: Working with C/C++ , MPI and Remote Machines

## ★ Objective

- ★ Learn how to use Eclipse to develop C programs
- ★ Learn how to develop, build and launch, and debug an MPI program on a remote parallel machine

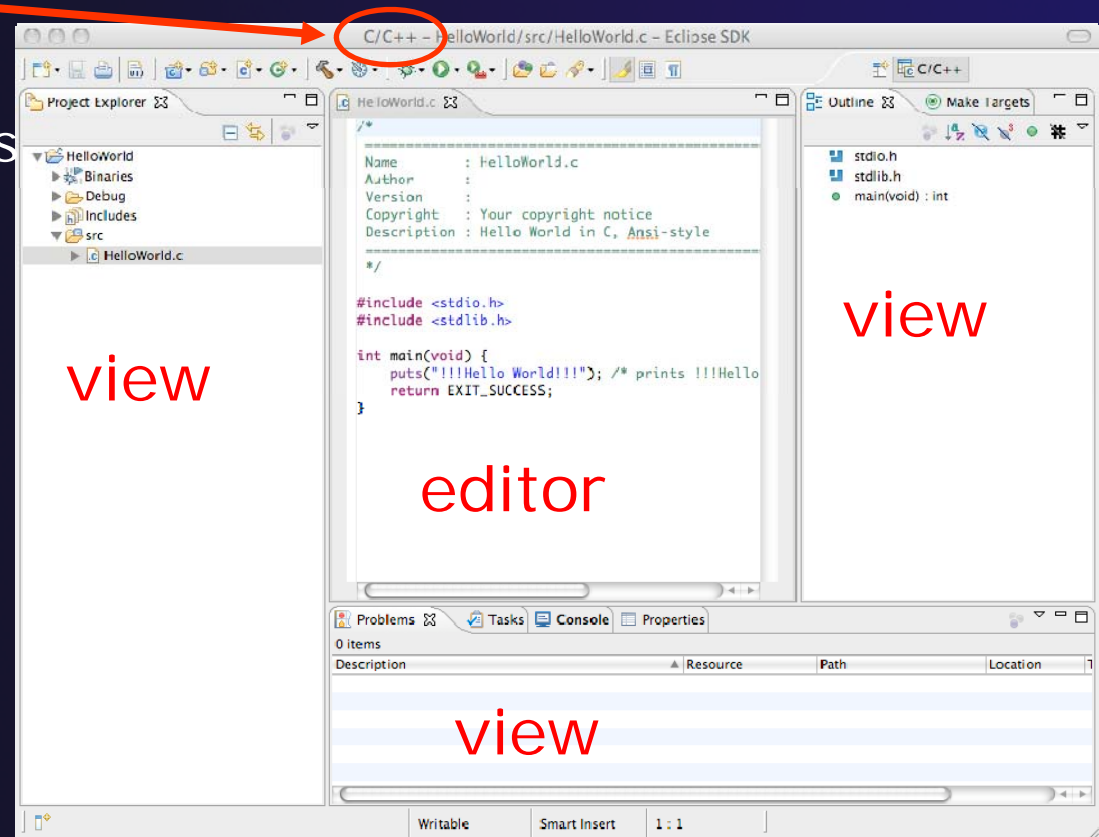
## ★ Contents

- ★ Brief introduction to the C/C++ Development Tools (CDT)
- ★ Create a simple application
- ★ Remote project setup
- ★ Working with resource managers
- ★ Launching a parallel application

# Workbench

- ✦ A Workbench contains perspectives
- ✦ A Perspective contains views and editors

- ✦ 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



# Perspectives

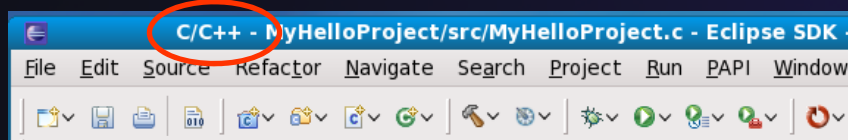
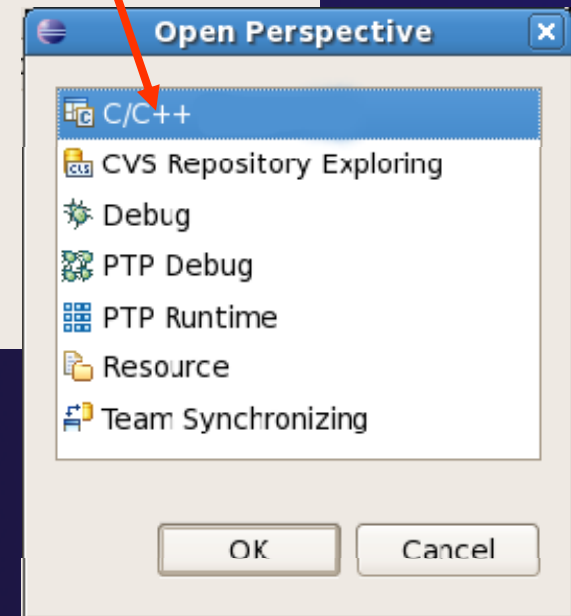
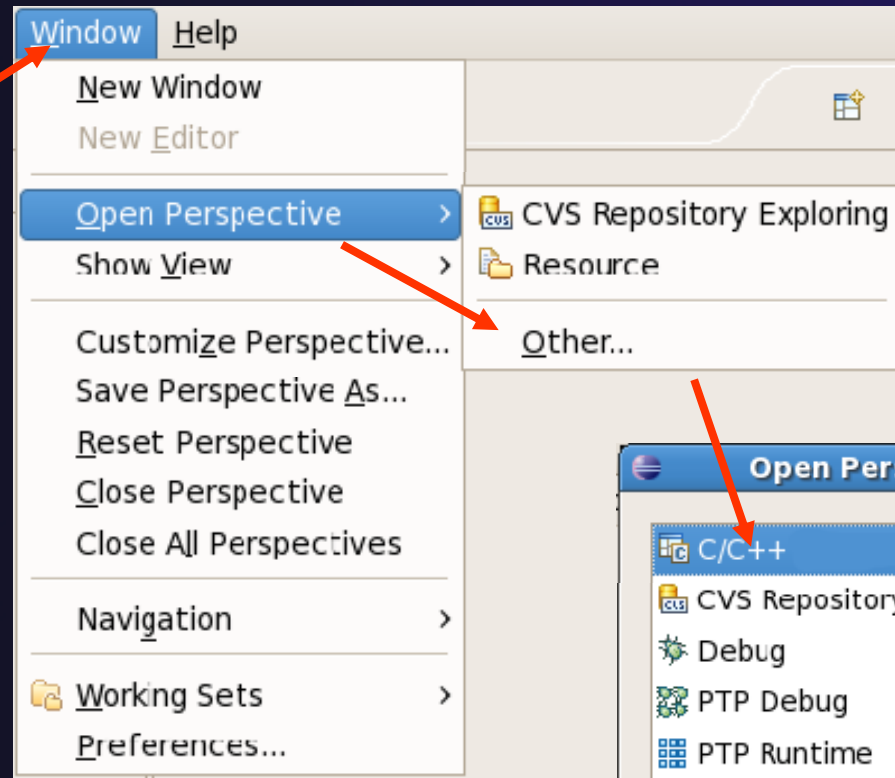
- ★ Perspectives define the layout of views in the Workbench
- ★ They are task oriented, i.e. they contain specific views for doing certain tasks:
  - ★ There is a Resource Perspective for manipulating resources
  - ★ C/C++ Perspective for manipulating compiled code
  - ★ Debug Perspective for debugging applications
- ★ You can easily switch between perspectives

# Switch to C/C++ Perspective



★ Only needed if you're not already in the perspective

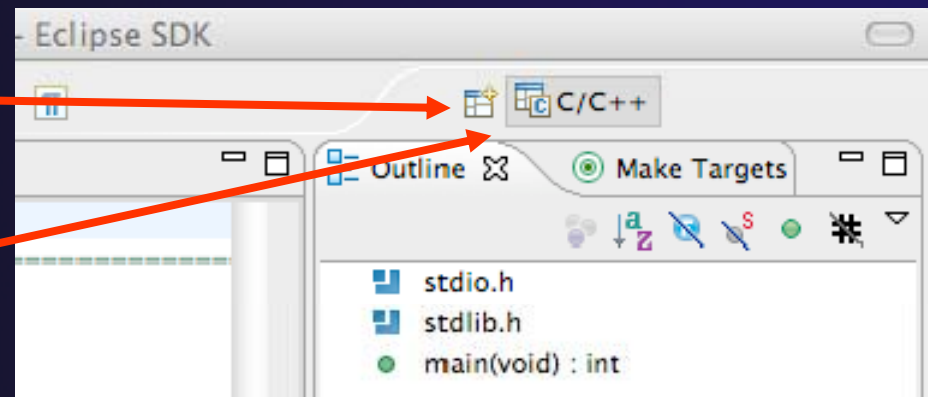
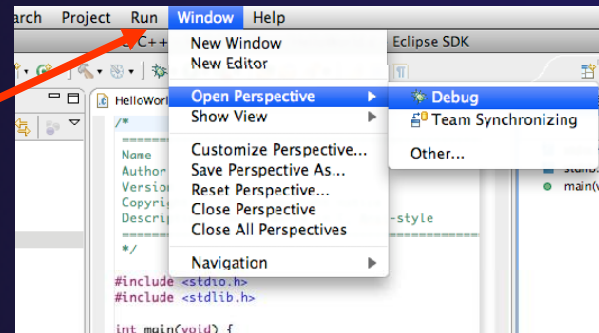
★ What Perspective am I in?  
See Title Bar





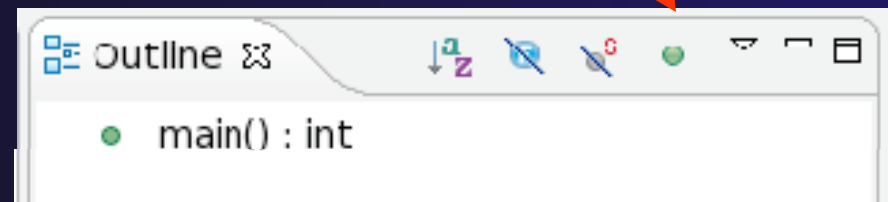
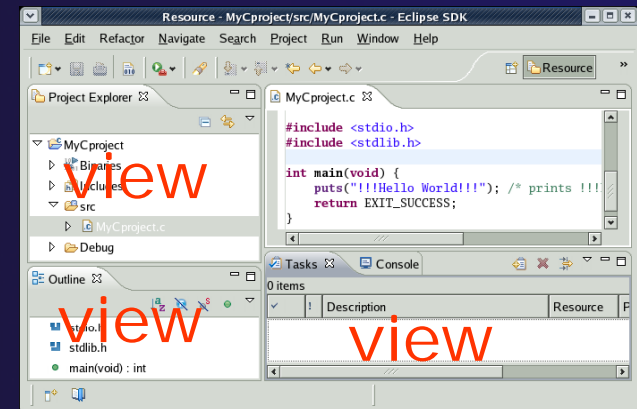
# Switching Perspectives

- ★ You can switch Perspectives by:
  - ★ Choosing the **Window**►**Open Perspective** menu option
  - ★ Clicking on the **Open Perspective** button
  - ★ Clicking on a perspective shortcut button



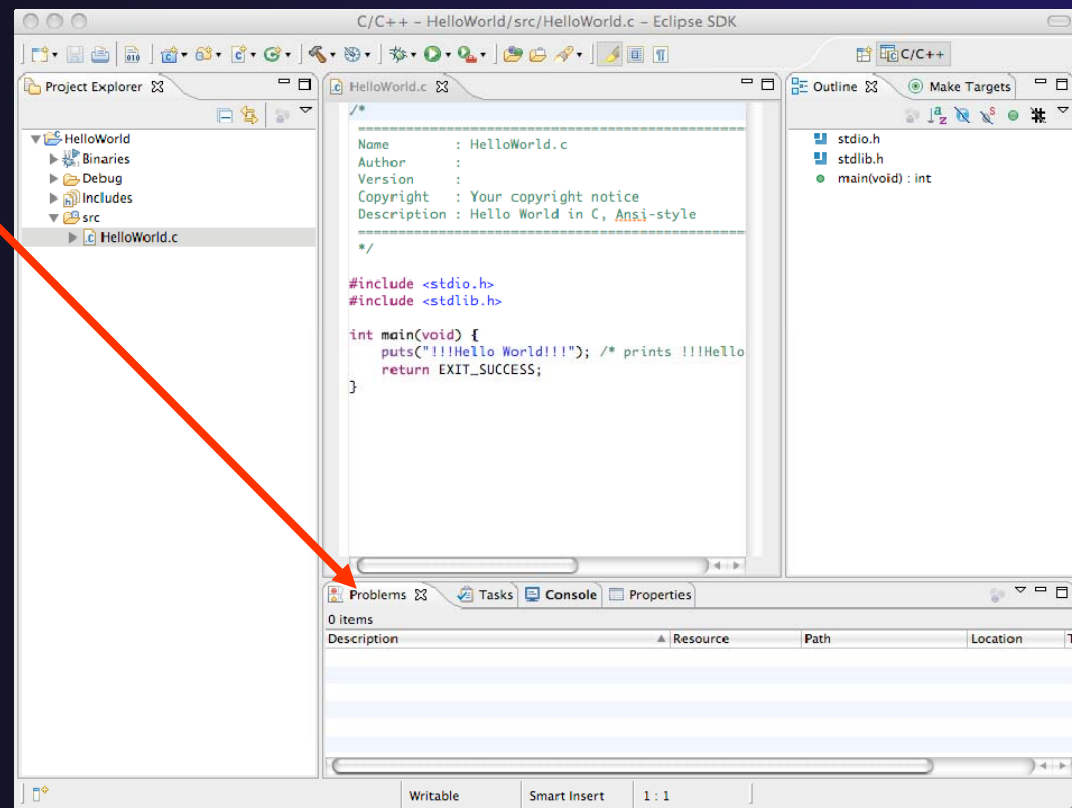
# Views

- ✦ 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



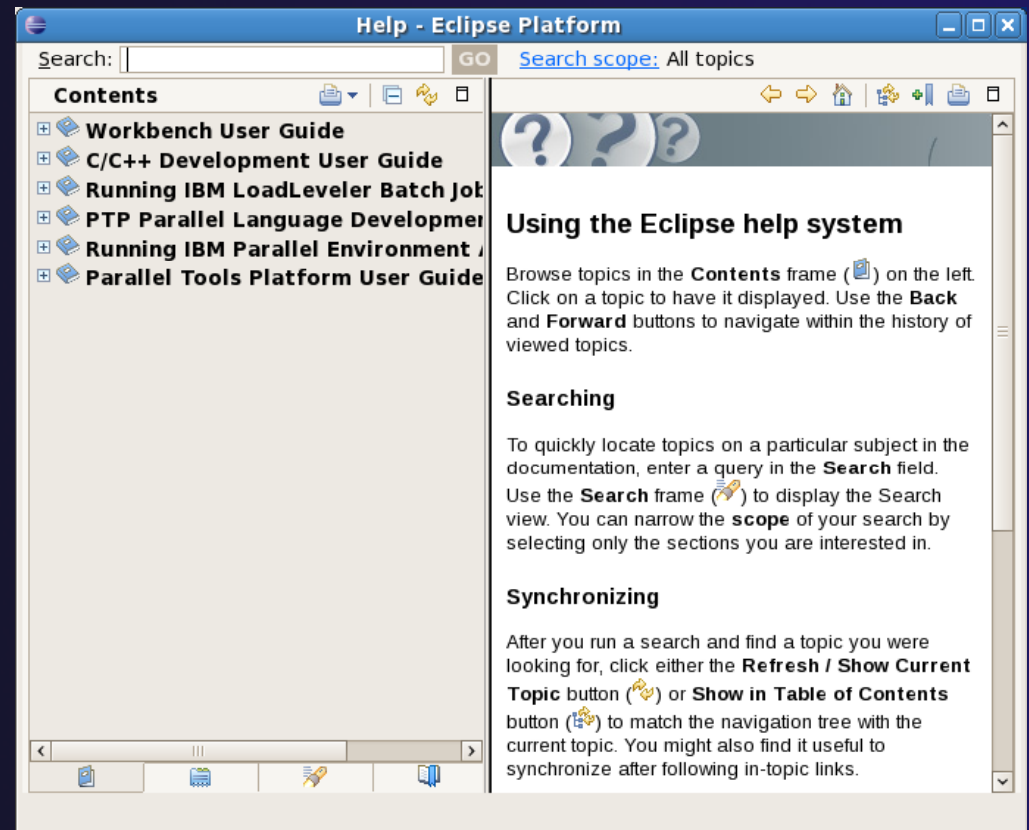
# Stacked Views

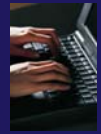
- ★ Stacked views appear as tabs
- ★ Selecting a tab brings that view to the foreground



# Help

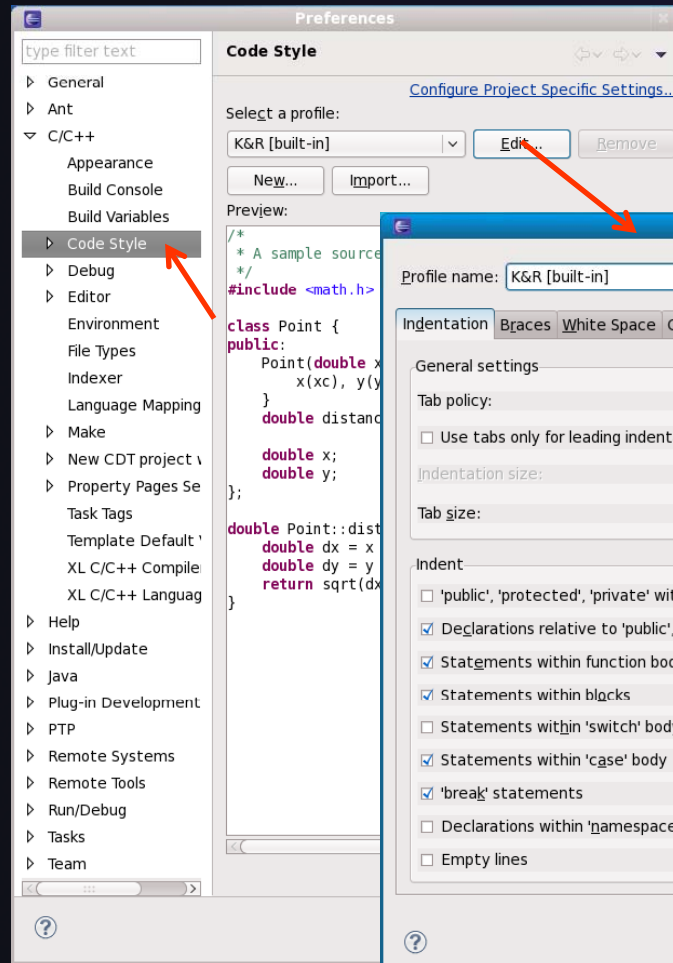
- ★ Access help
  - ★ Help ► Help Contents
  - ★ Help ► Search
  - ★ Help ► Dynamic Help
- ★ **Help Contents** provides detailed help on different Eclipse features
- ★ **Search** allows you to search for help locally, or using Google or the Eclipse web site
- ★ **Dynamic Help** shows help related to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





# Preferences

- ★ Eclipse Preferences allow customization of almost everything



- ★ Open **Window ▶ Preferences...**

- ★ C/C++ preferences allow many options

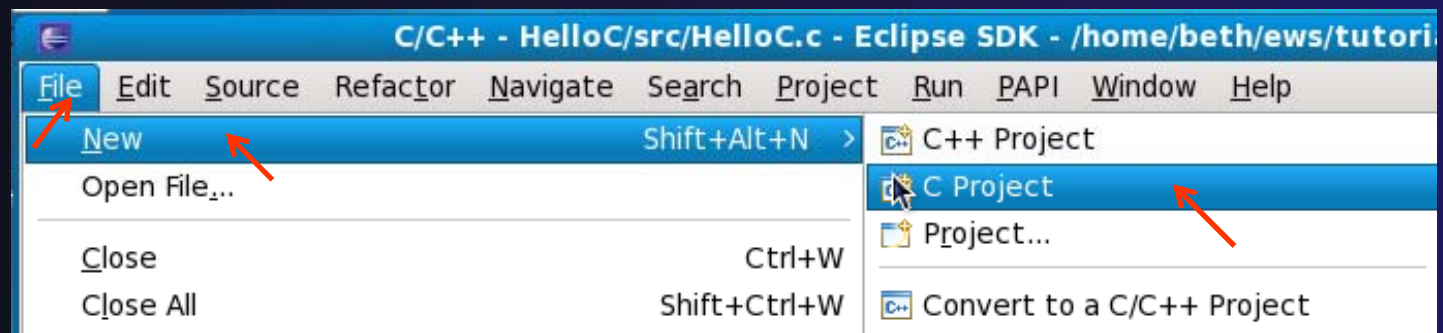
- ★ Code formatting settings ("Code Style") shown here



# Creating a C/C++ Application

Steps:

- ✦ Create a new C project
- ✦ Edit source code
- ✦ Save and build



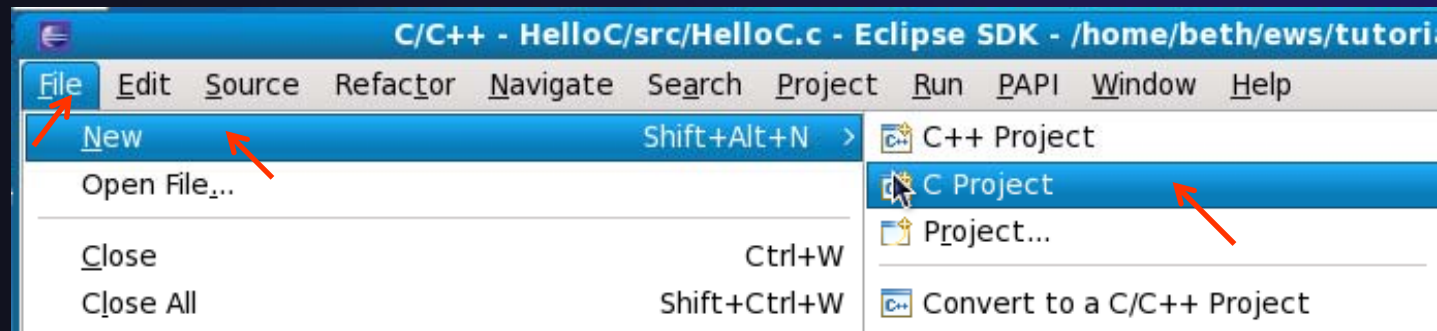


# Creating a C/C++ Application

Steps:

- ✦ Create a new C project
- ✦ Edit source code
- ✦ Save and build

Note: the next several slides will illustrate some local development concepts. While local development is not the focus of this mini-tutorial, we will use what we learn to help with remote development





# New C Project Wizard

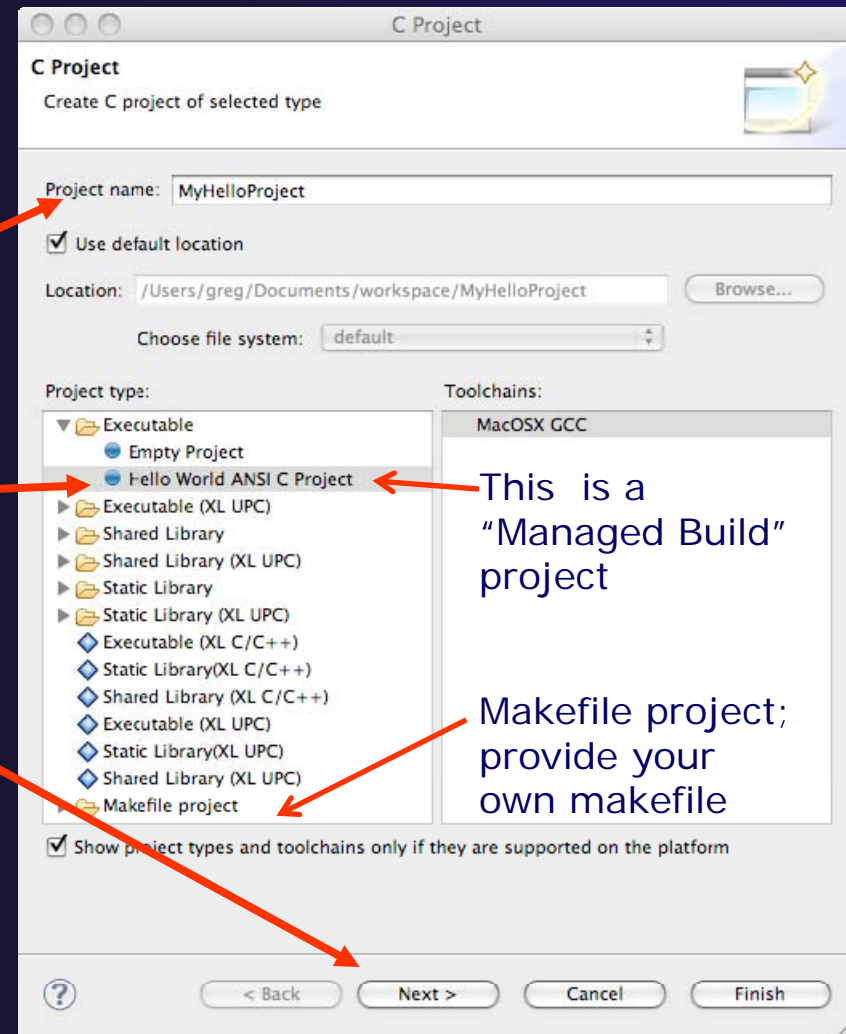
Create a new C project

★ **File ▶ New ▶ C Project**  
(see prev. slide)

★ Name the project  
'MyHelloProject'

★ Under Project types, under Executable, select **Hello World ANSI C Project**  
(no makefile req'd) and hit **Next**

★ On **Basic Settings** page, fill in information for your new project (**Author name** etc.) and hit **Finish**

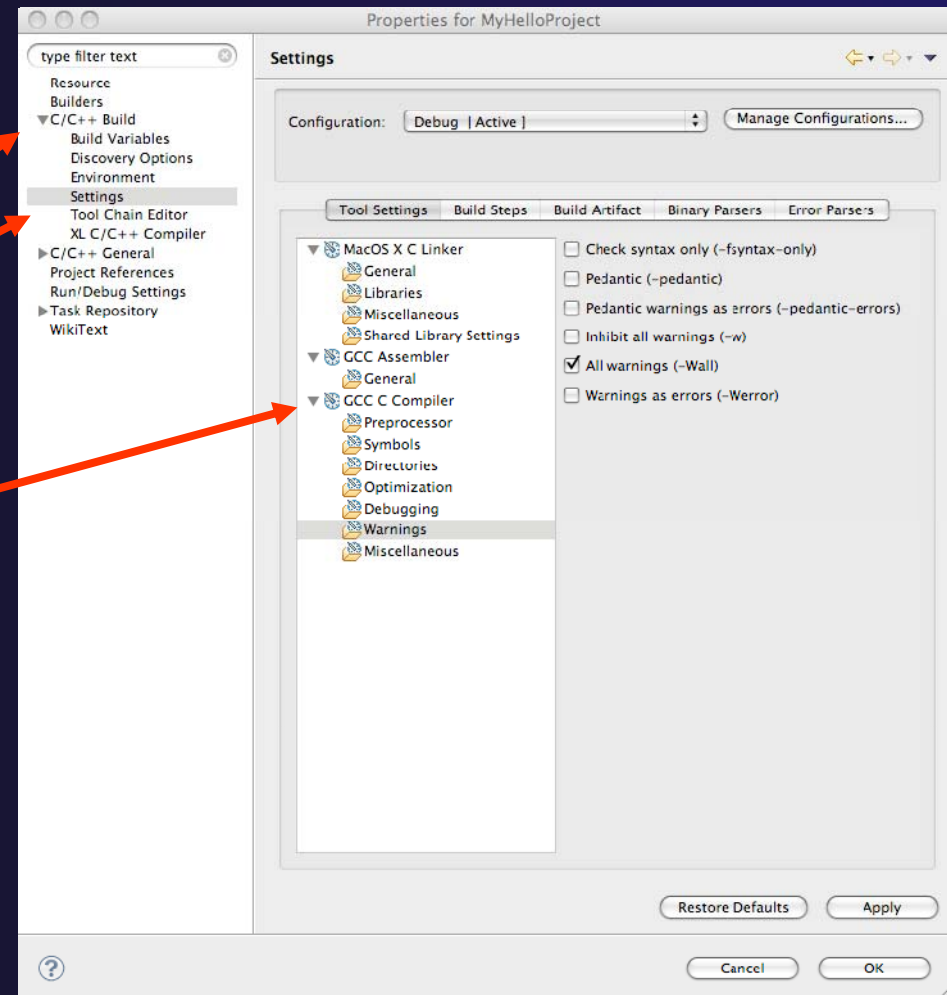




# Changing the C/C++ Build Settings Manually



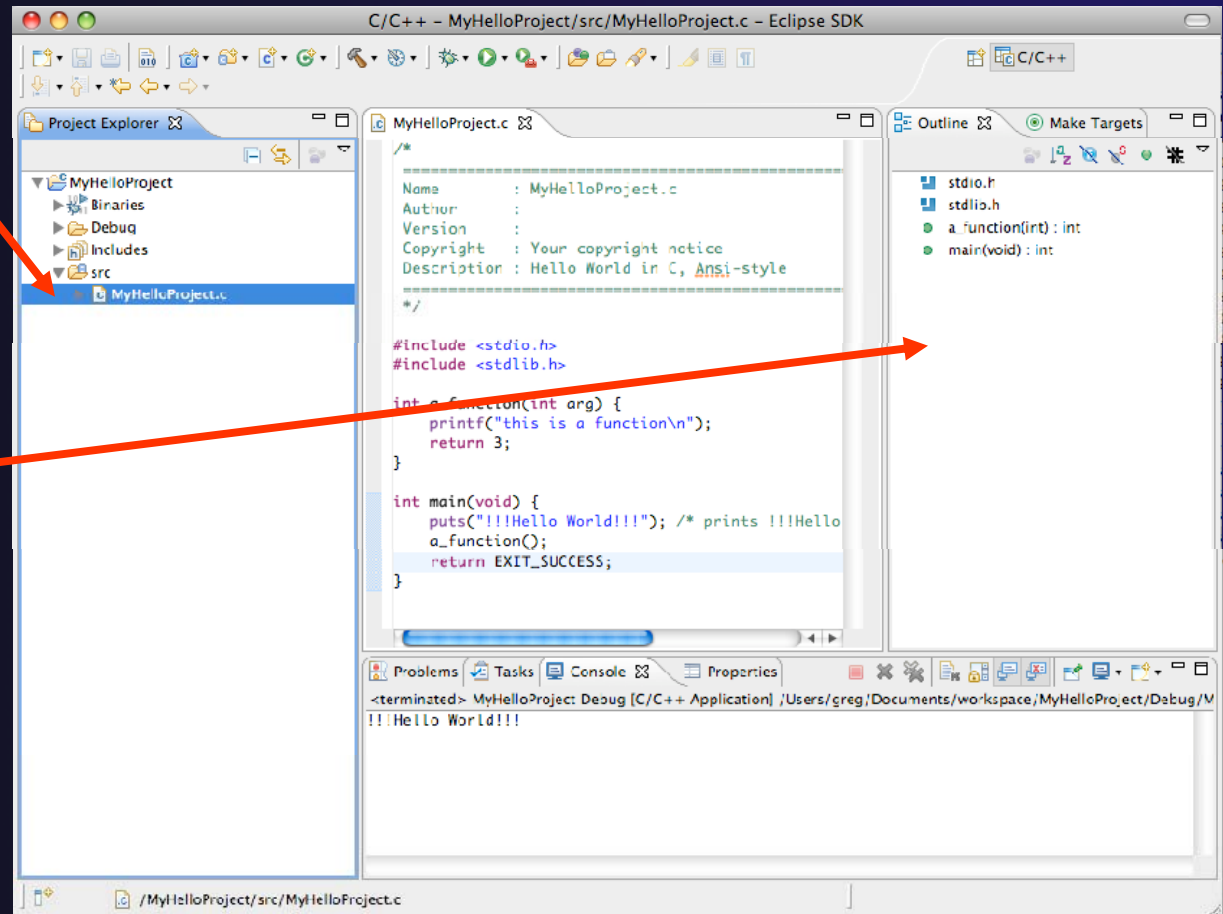
- ✦ Open the project properties by right-mouse clicking on project and select **Properties**
- ✦ Open **C/C++ Build**
- ✦ Select **Settings**
- ✦ Select **C Compiler** to change compiler settings
- ✦ Select **C Linker** to change linker settings
- ✦ It's also possible to change compiler/linker arguments
- ✦ Hit **OK** to close



# Editor and Outline View

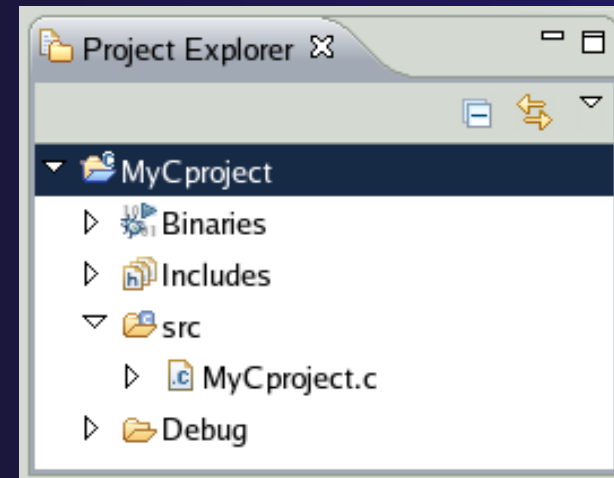


- ★ Double-click on source file in the **Project Explorer** to open C editor
- ★ Outline view is shown for file in editor
- ★ We'll describe the editor in the next few slides...



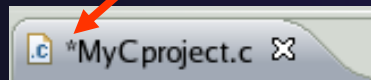
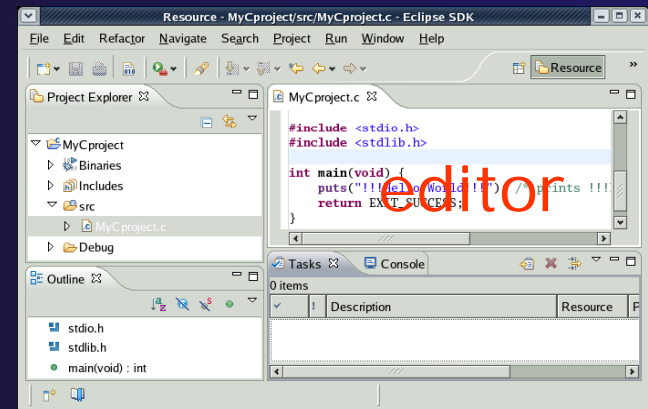
# Project Explorer View

- ★ Represents user's data
- ★ It is a set of user defined resources
  - ★ Files
  - ★ Folders
  - ★ Projects
    - ★ Collections of files and folders
    - ★ Plus meta-data
- ★ Resources are visible in the Project Explorer View



# 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
  - ✦ 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
- ✦ When you change a resource, an asterisk on the editor's title bar indicates unsaved changes
- ✦ How to Save

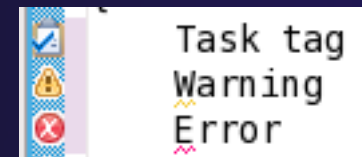


# Source Code Editors

- ★ 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;
}
```

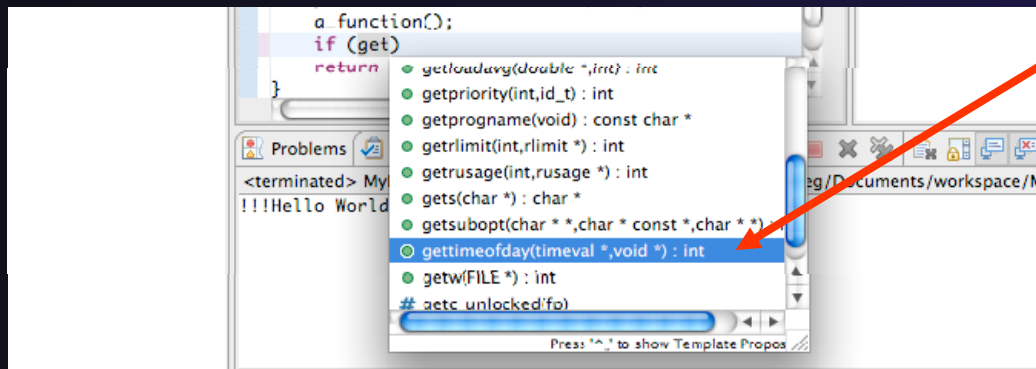
Icons:



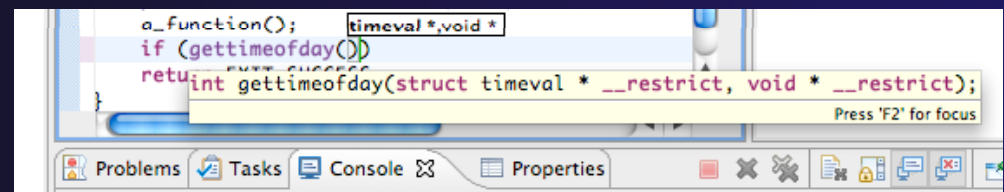


# Content Assist

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




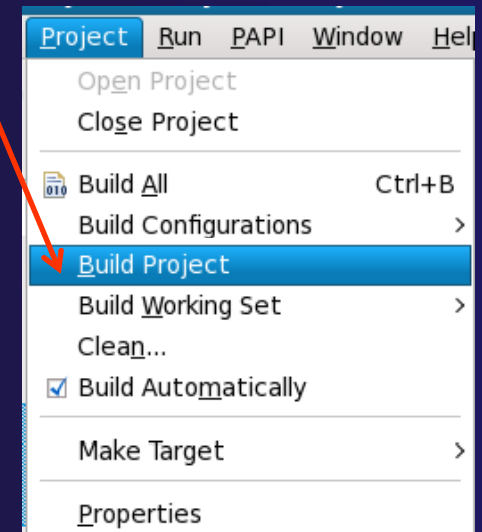
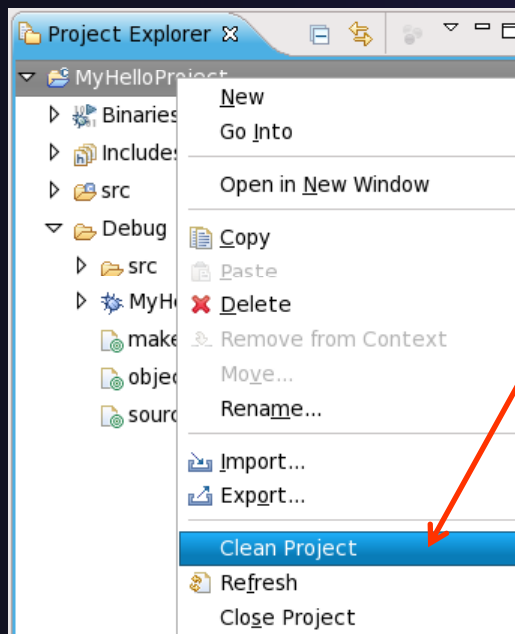
- ✦ Hover over a program element in the source file to see additional information





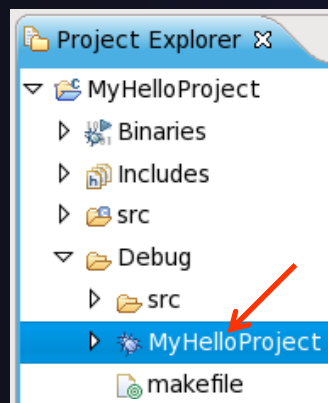
# Build

- ✦ Your program should build when created.
- ✦ To rebuild, many ways include: 
  - ✦ Select project, Hit hammer icon in toolbar
  - ✦ Select project, Project ► Build Project
  - ✦ Right mouse on project, Clean Project



## Build (2)

- ★ See the results of the build in the Console View
- ★ Executable should be in Debug folder:



```
C-Build [MyHelloProject]

**** Build of configuration Debug for project MyHelloProject ****

make all
Building file: ../src/MyHelloProject.c
Invoking: GCC C Compiler
gcc -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"src/MyHelloProject.d" -
MT"src/MyHelloProject.d" -o"src/MyHelloProject.o" "../src/MyHelloProject.c"
Finished building: ../src/MyHelloProject.c

Building target: MyHelloProject
Invoking: GCC C Linker
gcc -o"MyHelloProject" ../src/MyHelloProject.o
Finished building target: MyHelloProject
```

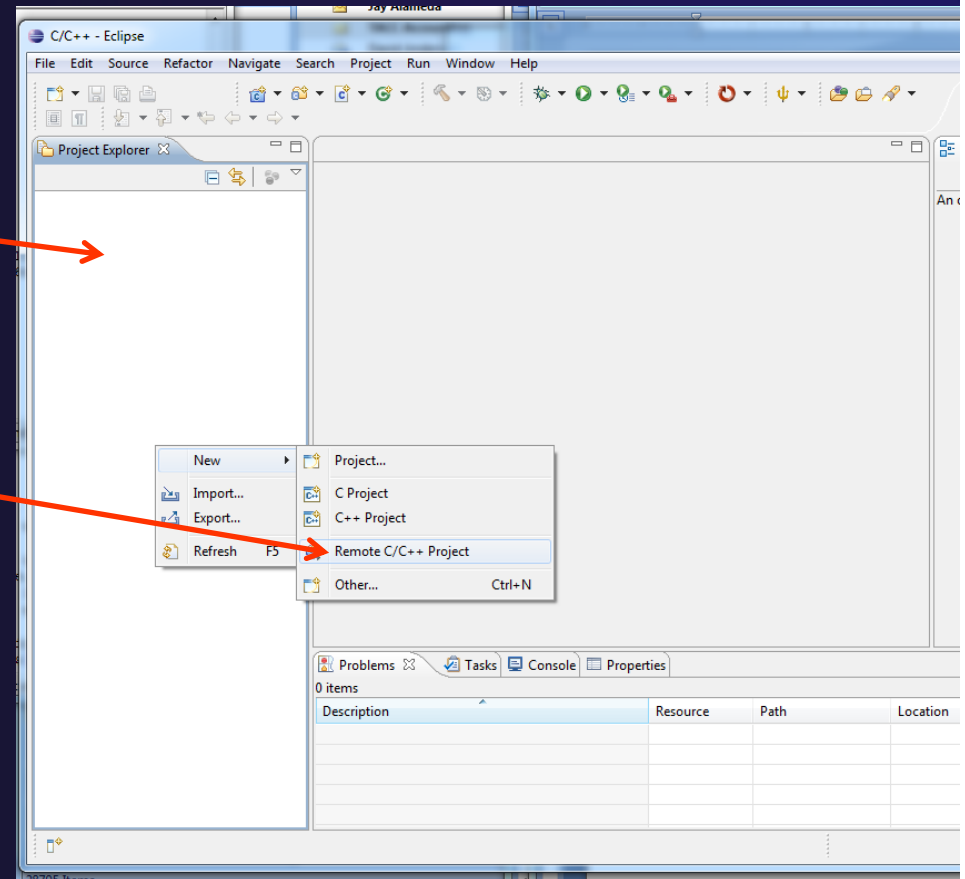


# Preparation for Remote C/C++ Project

- ★ General notes:
  - ★ This has been tested and found to work on the following machines: Queenbee
- ★ The example shown is worked on Queenbee
- ★ There are “out of band” setup steps that you need to do prior to starting the tutorial
  - ★ Copy sample MPI code to your directory
  - ★ Set up environment on remote machine
- ★ Please refer to setup instructions for your machine

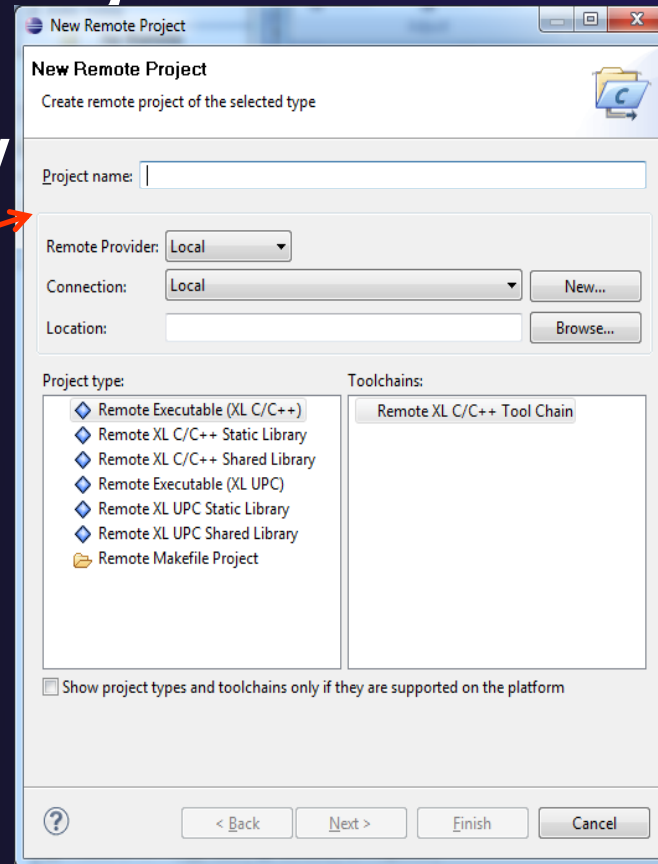
# Remote C/C++ Project

- ★ While in the C/C++ perspective:
  - ★ Right click in the Project Explorer
  - ★ Select **New > Remote C/C++ Project**



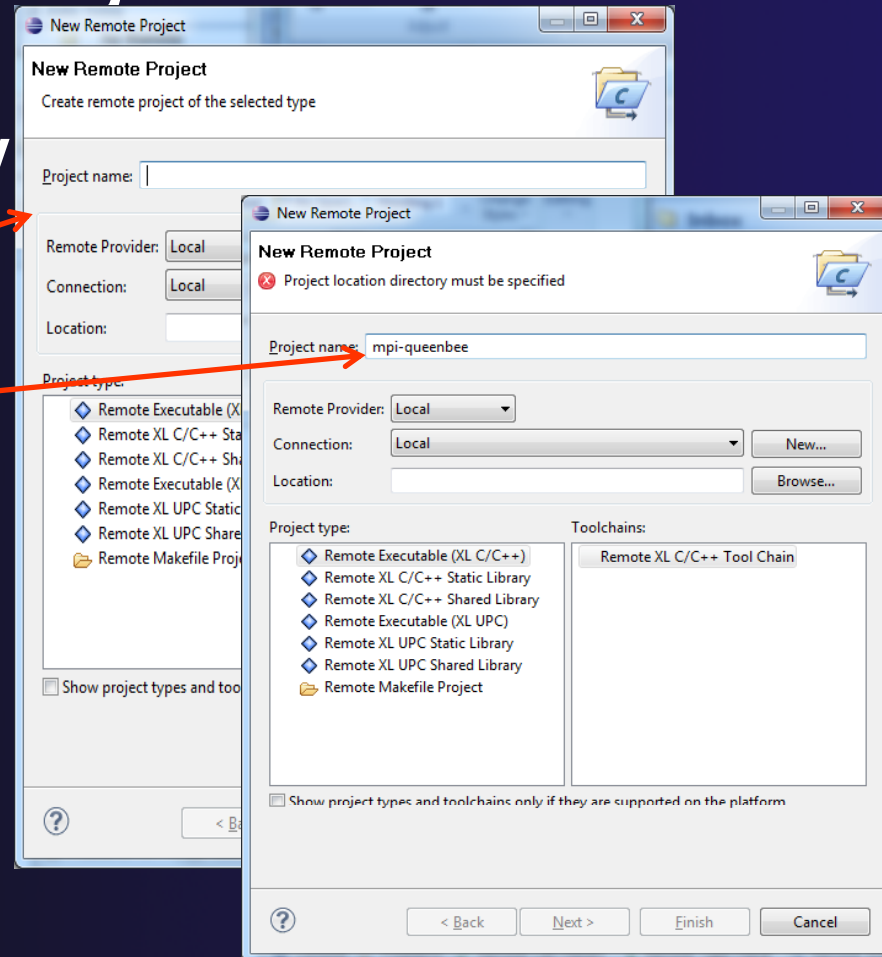
# Remote Project Wizard

- ★ This starts the **New Remote Project Wizard**



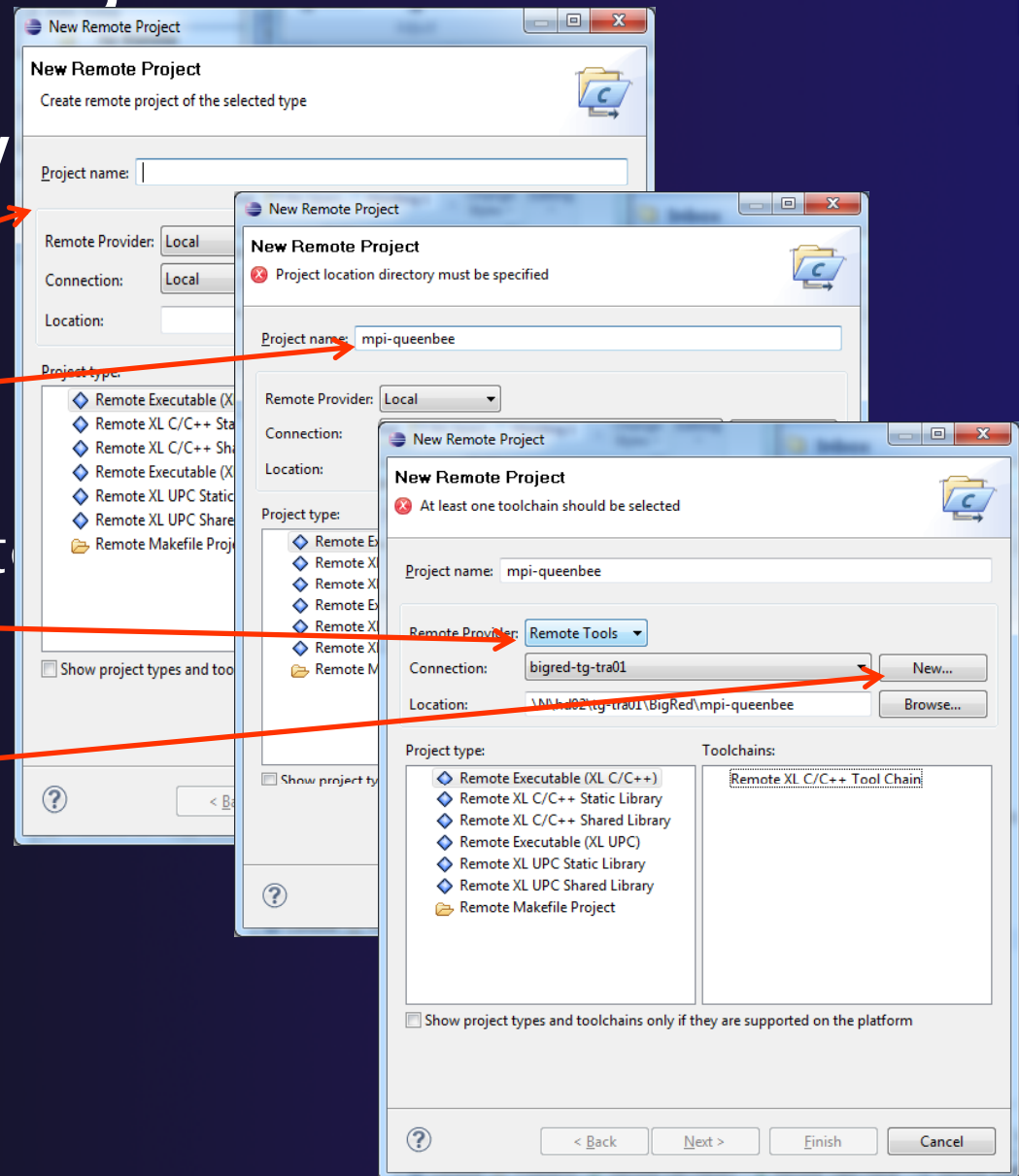
# Remote Project Wizard

- ★ This starts the **New Remote Project Wizard**
- ★ Name the project



# Remote Project Wizard

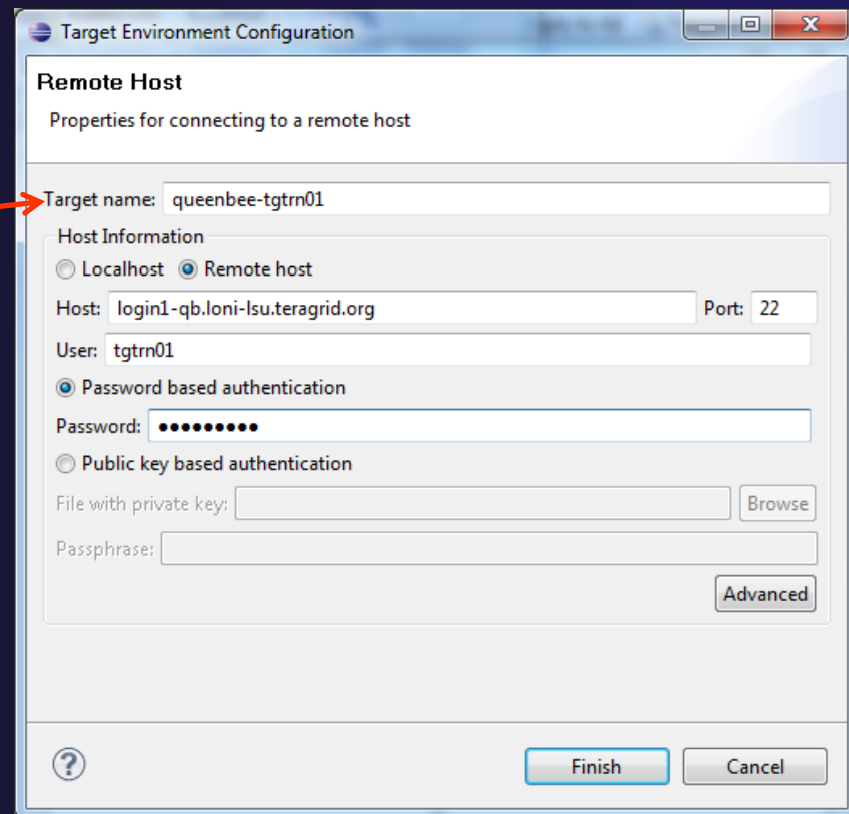
- ★ This starts the **New Remote Project Wizard**
- ★ Name the project
- ★ Select **Remote Tools** as the Remote Provider
- ★ Select **New...** Connection



# New Remote Connection



- ★ Remote Host Configuration
  - ★ Name the remote target



Target Environment Configuration

**Remote Host**  
Properties for connecting to a remote host

Target name: queenbee-tgtrn01

Host Information

Localhost  Remote host

Host: login1-qb.loni-lsu.teragrid.org Port: 22

User: tgtrn01

Password based authentication

Password: ●●●●●●●●

Public key based authentication

File with private key:  Browse

Passphrase:

Advanced

Finish Cancel

# New Remote Connection



- ★ Remote Host Configuration
  - ★ Name the remote target
  - ★ Select Remote Host, and fill in the hostname
    - ★ Use the host you were assigned

Target Environment Configuration

**Remote Host**  
Properties for connecting to a remote host

Target name: queenbee-tgtrn01

Host Information

Localhost  Remote host

Host: login1-qb.loni-lsu.teragrid.org Port: 22

User: tgtrn01

Password based authentication

Password: ●●●●●●●●

Public key based authentication

File with private key:  Browse

Passphrase:

Advanced

Finish Cancel

# New Remote Connection



- ★ Remote Host Configuration
  - ★ Name the remote target
  - ★ Select Remote Host, and fill in the hostname
    - ★ Use the host you were assigned
  - ★ And add in your username and password (this is currently not editable once saved)

Target Environment Configuration

**Remote Host**  
Properties for connecting to a remote host

Target name: queenbee-tgtrn01

Host Information

Localhost  Remote host

Host: login1-qb.loni-lsu.teragrid.org Port: 22

User: tgtrn01

Password based authentication

Password: .....

Public key based authentication

File with private key:  Browse

Passphrase:

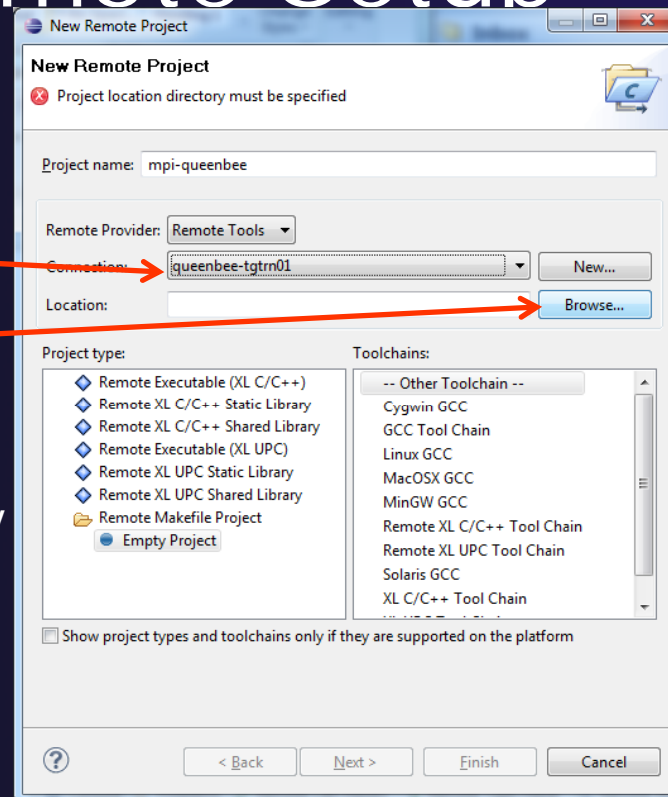
Advanced

Finish Cancel



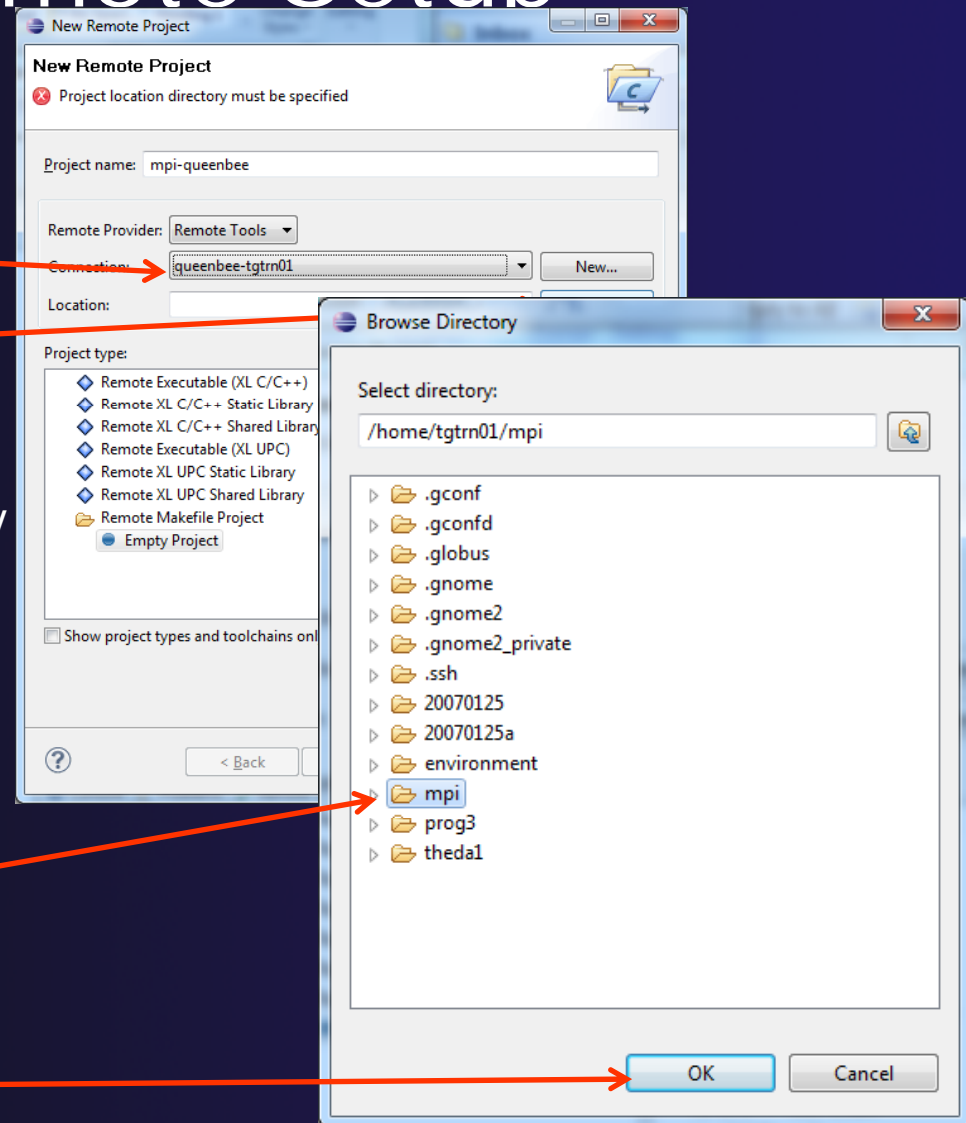
# Finish Remote Setup

- ★ Choose new remote connection
- ★ Then select **Browse** to choose one's working directory
- ★ N.B. this is conceptually similar to "importing" existing code into a project, note that you can also create an empty directory and start coding away!



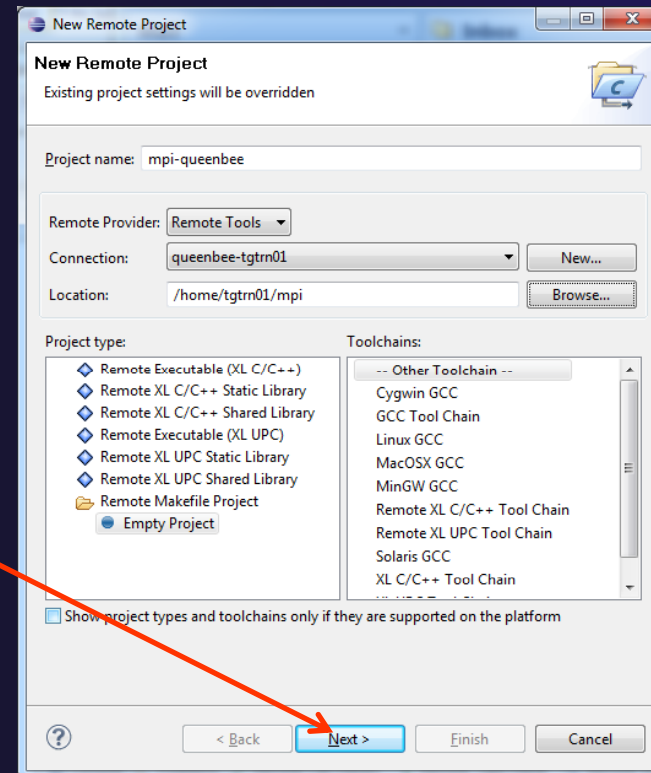
# Finish Remote Setup

- ★ Choose new remote connection
- ★ Then select **Browse** to choose one's working directory
  - ★ N.B. this is conceptually similar to "importing" existing code into a project, note that you can also create an empty directory and start coding away!
- ★ Choose the *mpi* sample code directory, and then select OK



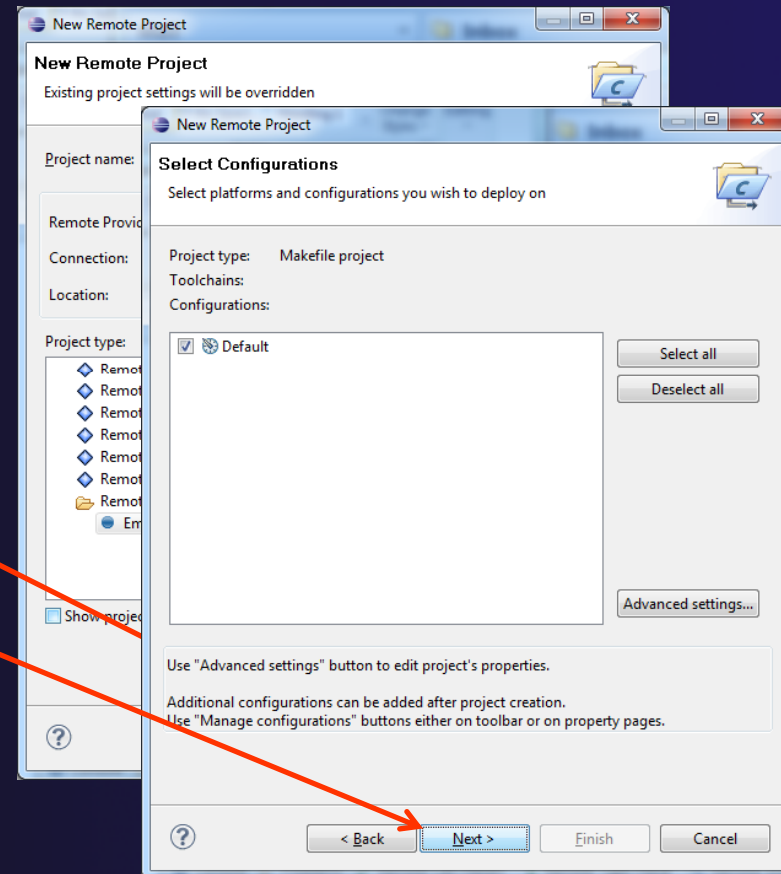
# Finish Remote Setup

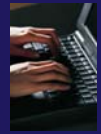
- ★ We'll step through the rest of the wizard to finish the remote project
- ★ Select **Next**



# Finish Remote Setup

- ★ We'll step through the rest of the wizard to finish the remote project
  - ★ Select **Next**
  - ★ And **Next**





# Finish Remote Setup

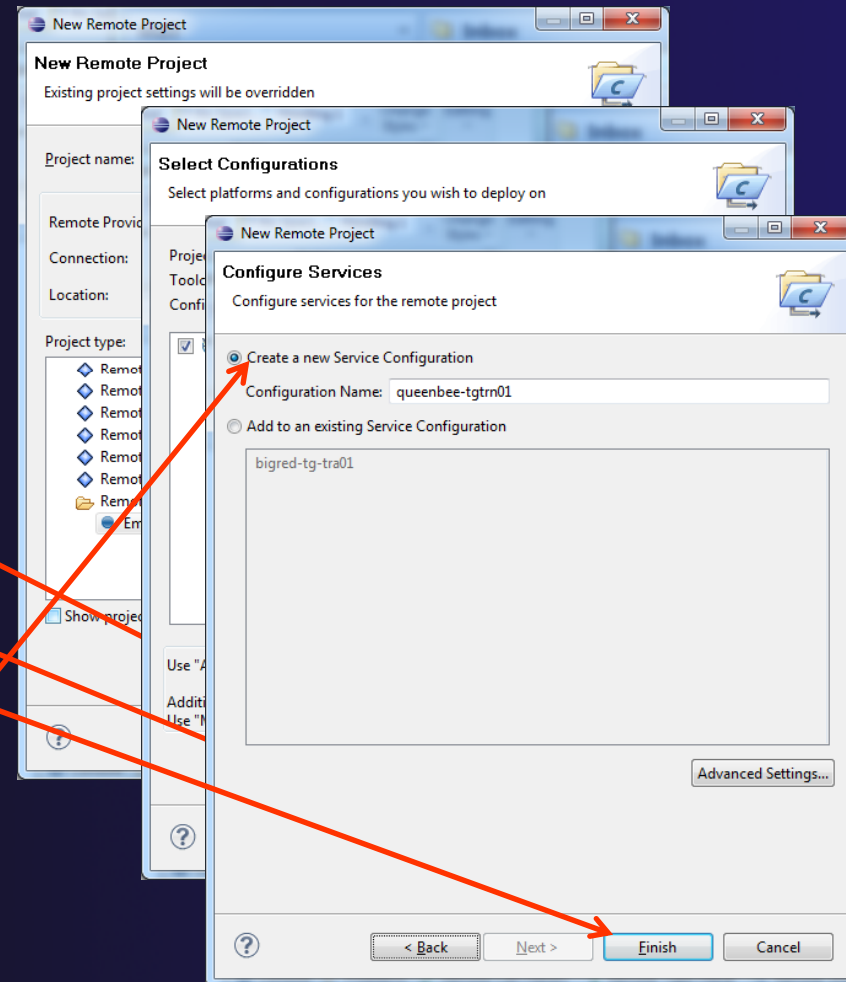
★ We'll step through the rest of the wizard to finish the remote project

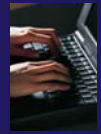
★ Select **Next**

★ And **Next**

★ And **Finish**

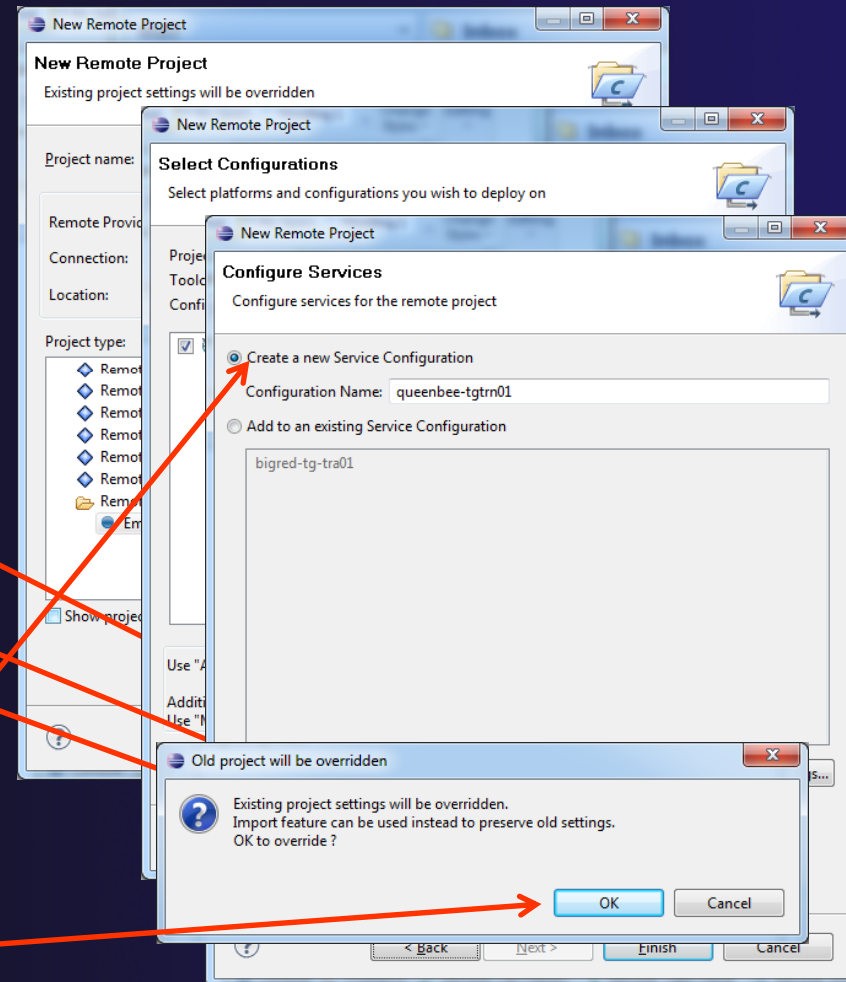
★ N.B. We'll be creating a new service configuration





# Finish Remote Setup

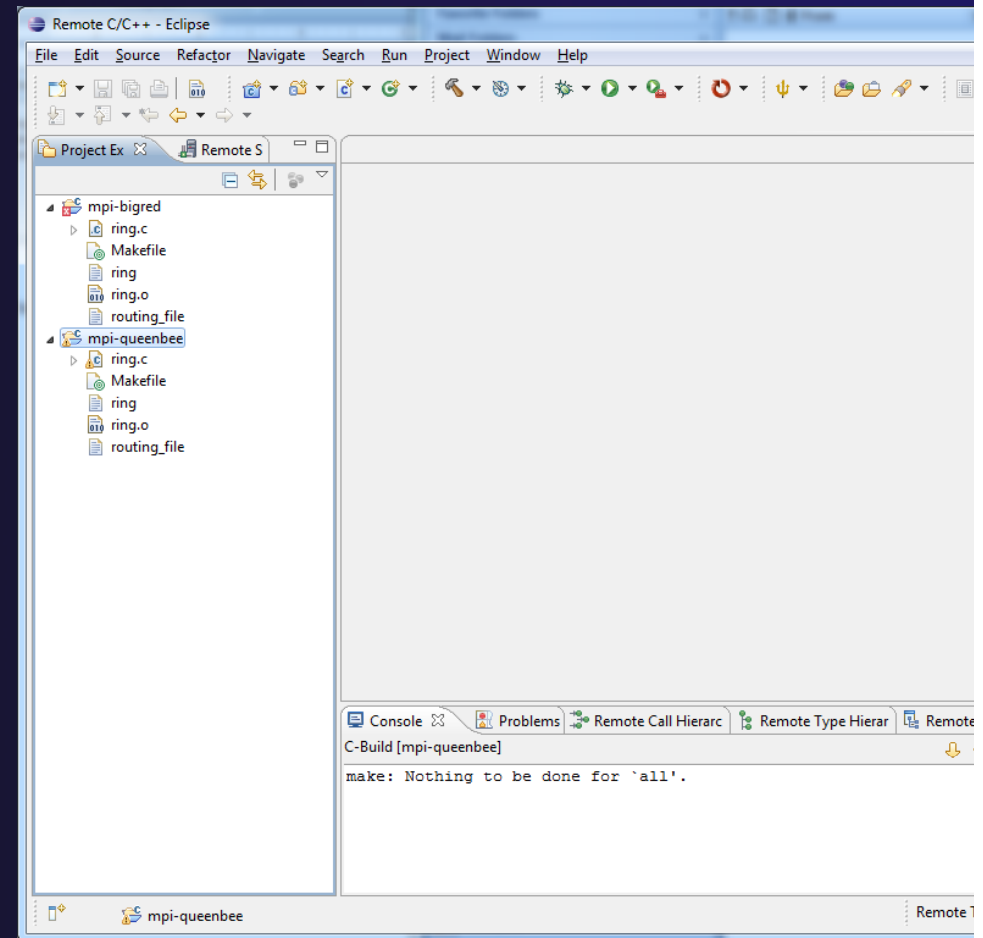
- ★ We'll step through the rest of the wizard to finish the remote project
  - ★ Select **Next**
  - ★ And **Next**
  - ★ And **Finish**
    - ★ N.B. We'll be creating a new service configuration
  - ★ Select **OK** to override existing project settings



# Remote C/C++ Project

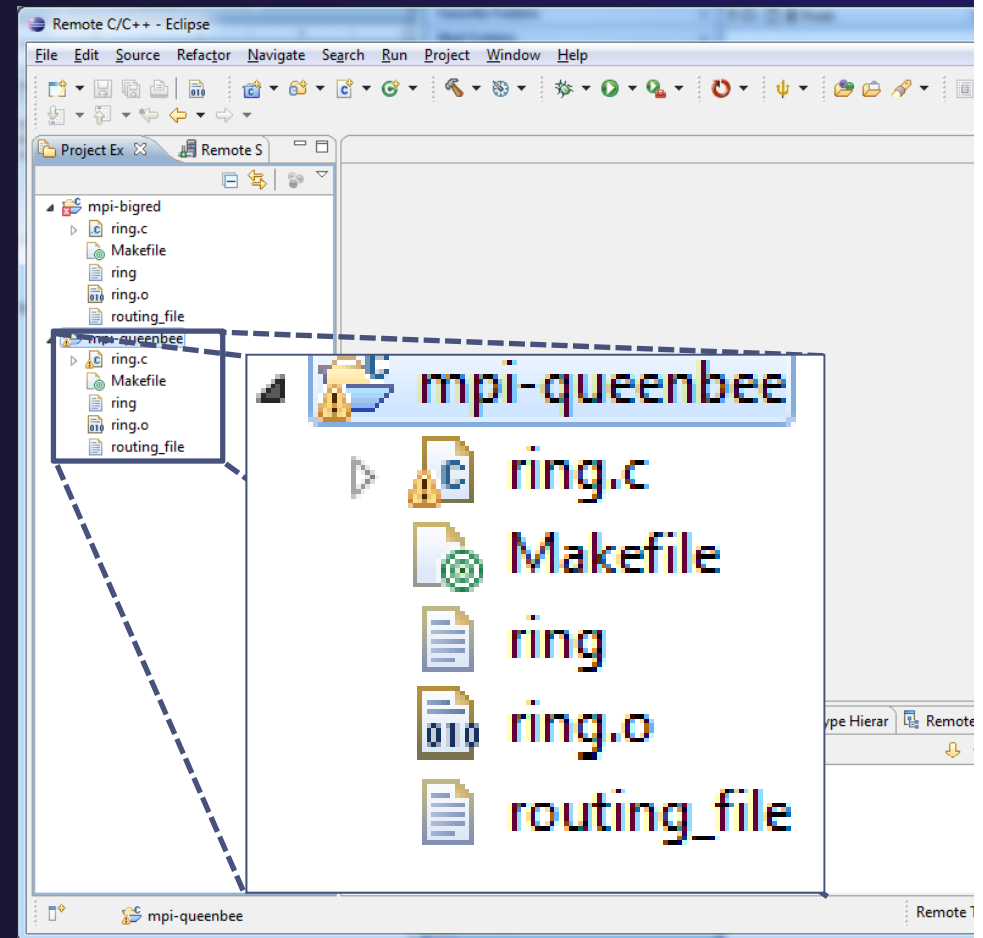


- ★ If prompted to switch to the Remote C/C++ perspective, select **OK**.



# Remote C/C++ Project

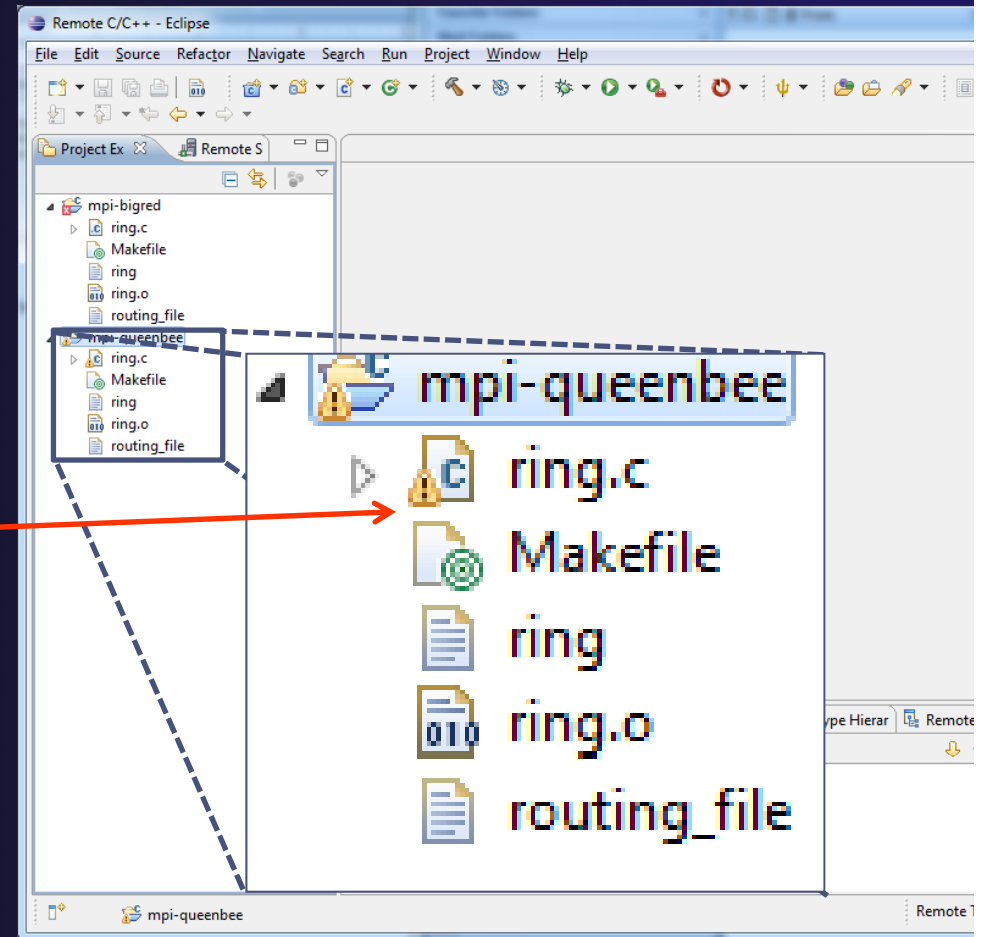
- ★ If prompted to switch to the Remote C/C++ perspective, select **OK**.





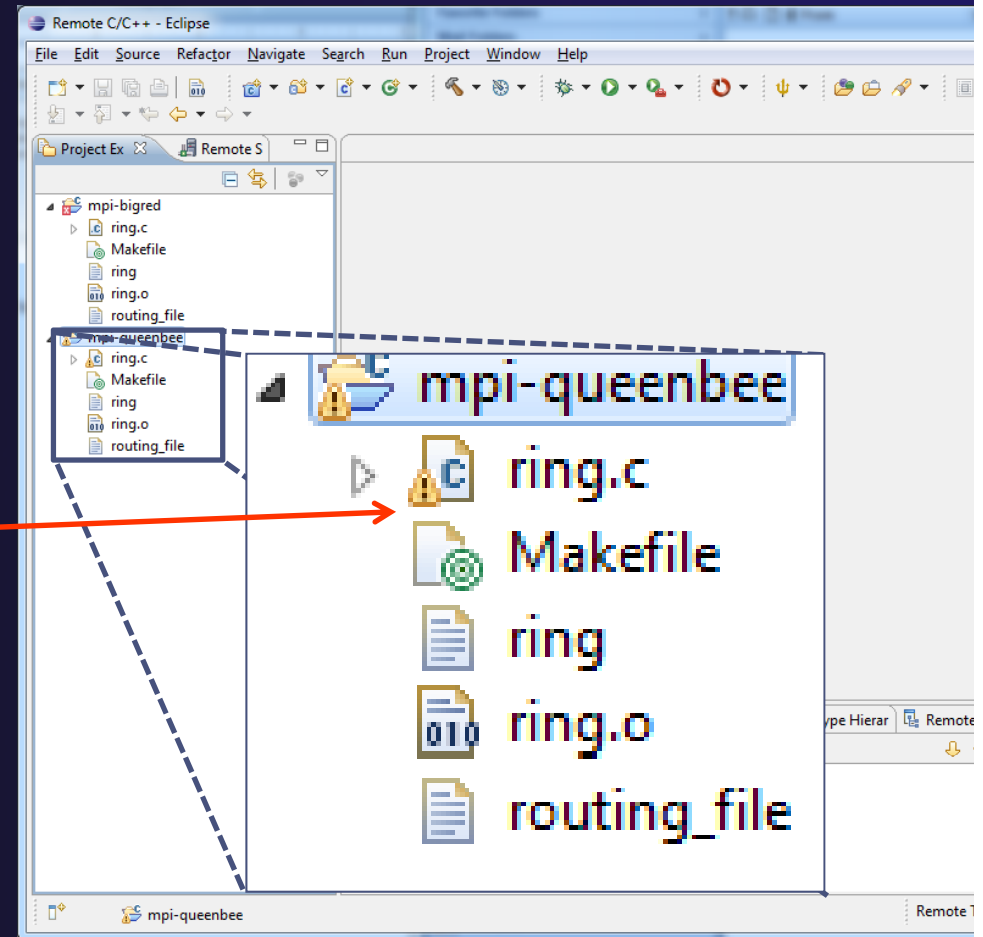
# Remote C/C++ Project

- ★ If prompted to switch to the Remote C/C++ perspective, select **OK**.
- ★ Note that there is an issue with the ring.c source



# Remote C/C++ Project

- ★ If prompted to switch to the Remote C/C++ perspective, select **OK**.
- ★ Note that there is an issue with the ring.c source
- ★ Double click on **ring.c** to load it into the editor, to see if we can determine the issue



# Ring.c



- ★ Note the header files marked in red

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mpi.h>

/* command line configurables */
int Ntrips; /* -t <ntrips> */
int Verbose; /* -v */

int parse_command_line_args(int argc, char **argv, int my_id)
{
    int i;
    int error;

    /* default values */
    Ntrips = 1;
    Verbose = 0;

    for (i = 1, error = 0; !error && i < argc; i++)
    {
        if (!strcmp(argv[i], "-t"))
        {
            if (i + 1 < argc && (Ntrips = atoi(argv[i+1])) > 0)
            {
                i++;
            }
        }
    }
}
```

# Ring.c



- ★ Note the header files marked in red
- ★ Also, note that the remote indexer has produced a source outline

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mpi.h>

/* command line configurables */
int Ntrips; /* -t <ntrips> */
int Verbose; /* -v */

int parse_command_line_args(int argc, char **argv, int my_id)
{
    int i;
    int error;

    /* default values */
    Ntrips = 1;
    Verbose = 0;

    for (i = 1, error = 0; !error && i < argc; i++)
    {
        if (!strcmp(argv[i], "-t"))
        {
            if (i + 1 < argc && (Ntrips = atoi(argv[i+1])) > 0)
                i++;
        }
    }
}
```

Console: C-Build [mpi-queenbee] make: Nothing to be done for 'all'.

## ring.c



- ★ Note the header files marked in red
- ★ Also, note that the remote indexer has produced a source outline
- ★ We need to amend the include paths for this project...

```

Remote C/C++ - mpi-queenbee/ring.c - Eclipse
File Edit Source Refactor Navigate Search Run Project Window Help
Project Explorer Remote S
  mpi-bigred
  ring.c
  Makefile
  ring.o
  routing_file
  mpi-queenbee
  ring.c
  Makefile
  ring.o
  routing_file

ring.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mpi.h>

/* command line configurables */
int Ntrips; /* -t <ntrips> */
int Verbose; /* -v */

int parse_command_line_args(int argc, char **argv, int my_id)
{
    int i;
    int error;

    /* default values */
    Ntrips = 1;
    Verbose = 0;

    for (i = 1, error = 0; !error && i < argc; i++)
    {
        if (!strcmp(argv[i], "-t"))
        {
            if (i + 1 < argc && (Ntrips = atoi(argv[i+1])) > 0)
                i++;
        }
    }
}

Outline
  stdio.h
  stdlib.h
  string.h
  mpi.h
  Ntrips: int
  Verbose: int
  parse_command_line_ar
  main(int, char**): int

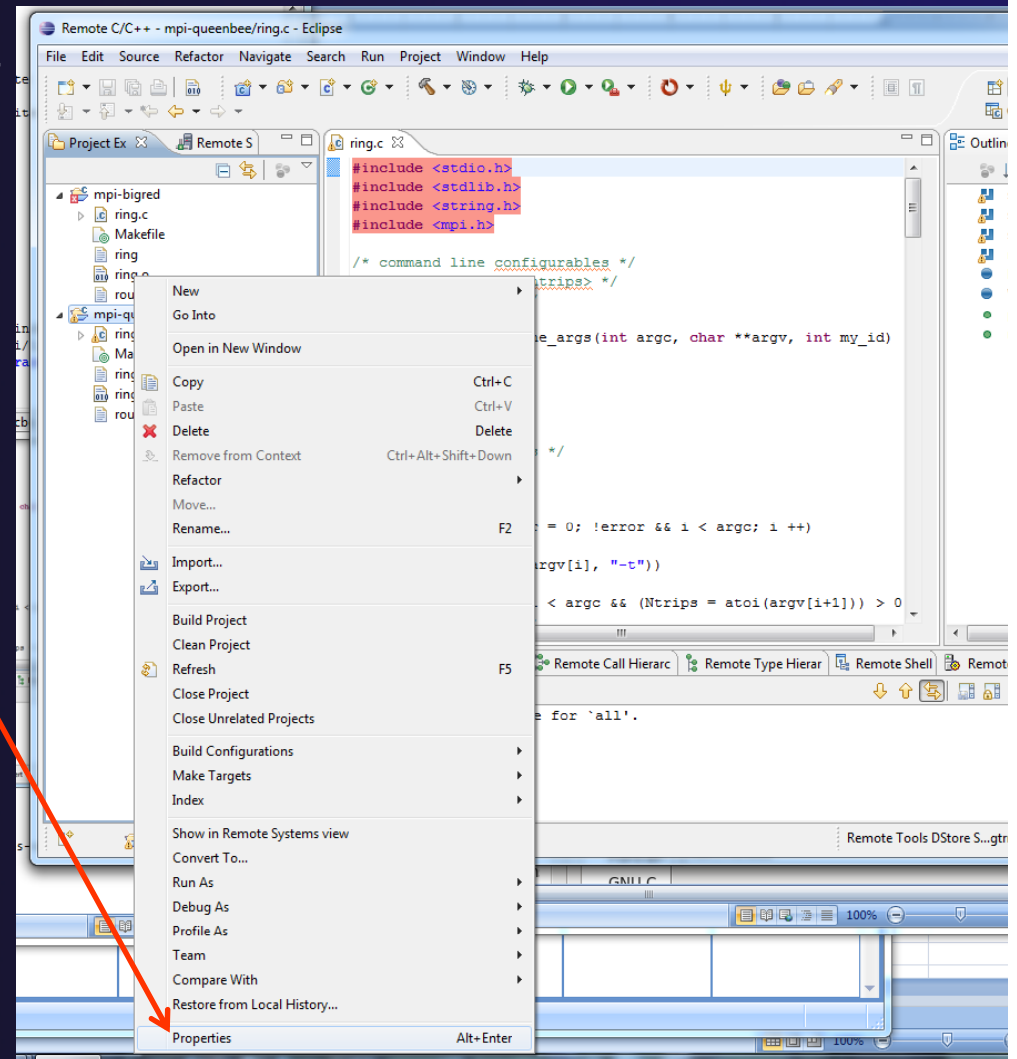
Console
C-Build [mpi-queenbee]
make: Nothing to be done for `all'.

Writable Smart Insert 1:1 Remote Tools DStore S...gtrn01): (100%)

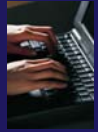
```

# Include paths

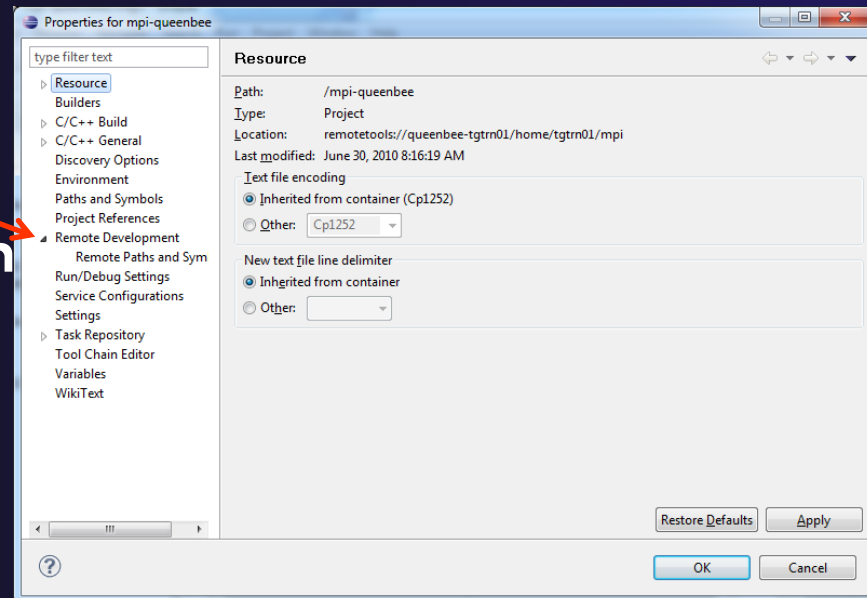
- ★ Right click on project name, then select **Properties**



# Project properties



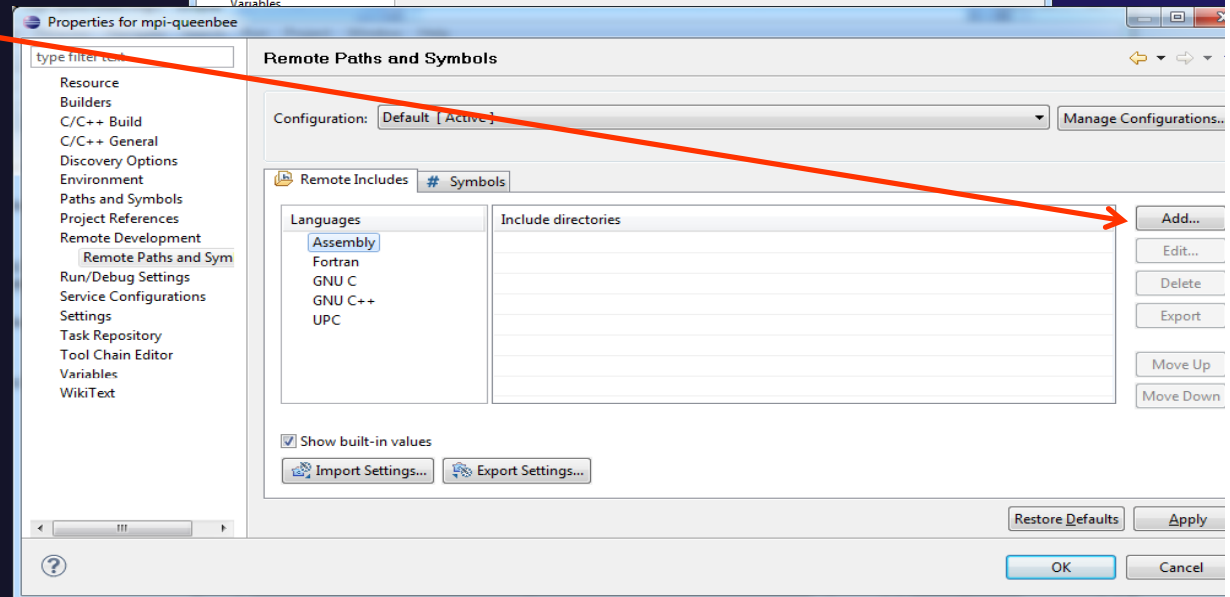
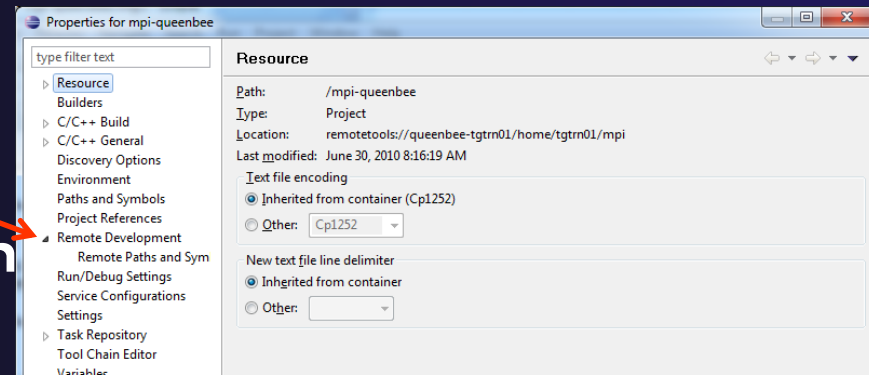
- ★ Open **Remote Development**,
- ★ Select **Remote Paths and Symbols**



# Project properties



- ✦ Open **Remote Development**,
- ✦ Select **Remote Path and Symbols**
- ✦ Click **Add**

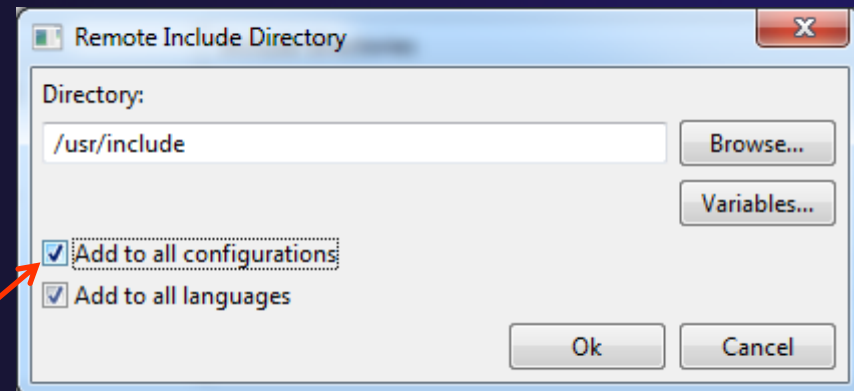






# Remote Include Directory

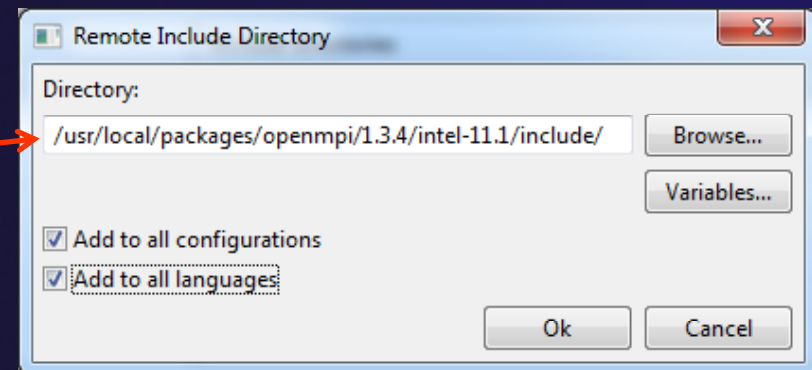
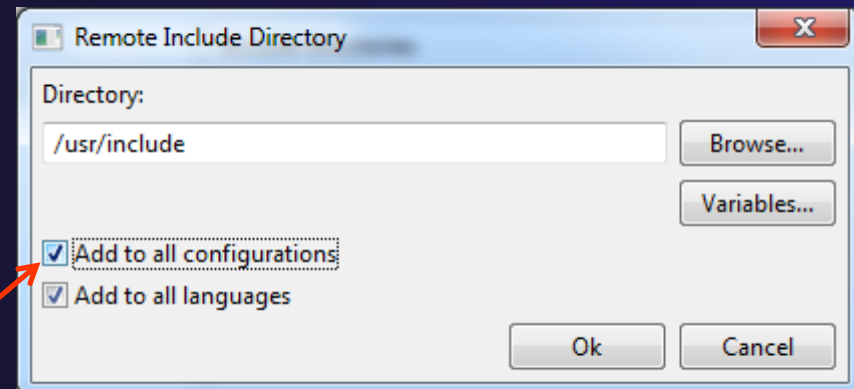
- ★ Add /usr/include for queenbee
  - ★ Note to check “add to all configurations” and “add to all languages”
  - ★ Hit OK
- ★ Include paths for all other machines are in the machine instructions





# Remote Include Directory

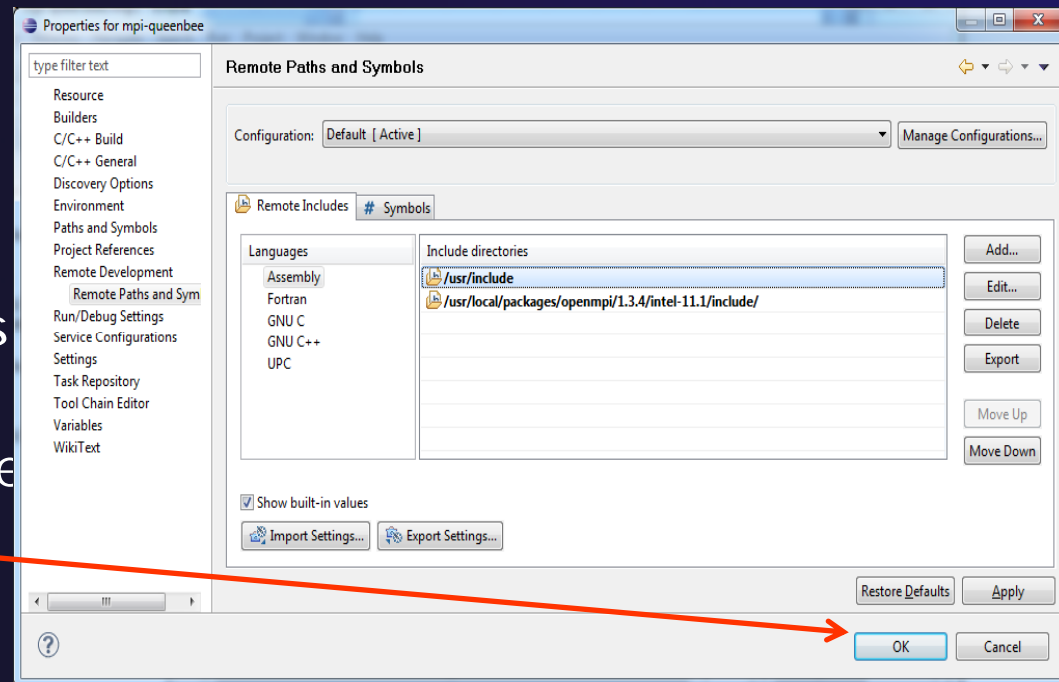
- ★ Add /usr/include for queenbee
  - ★ Note to check “add to all configurations” and “add to all languages”
  - ★ Hit OK
- ★ Repeat for MPI includes...



# Remote include paths



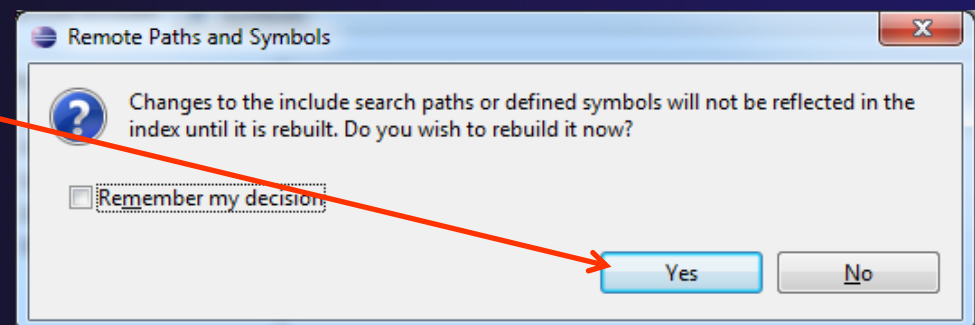
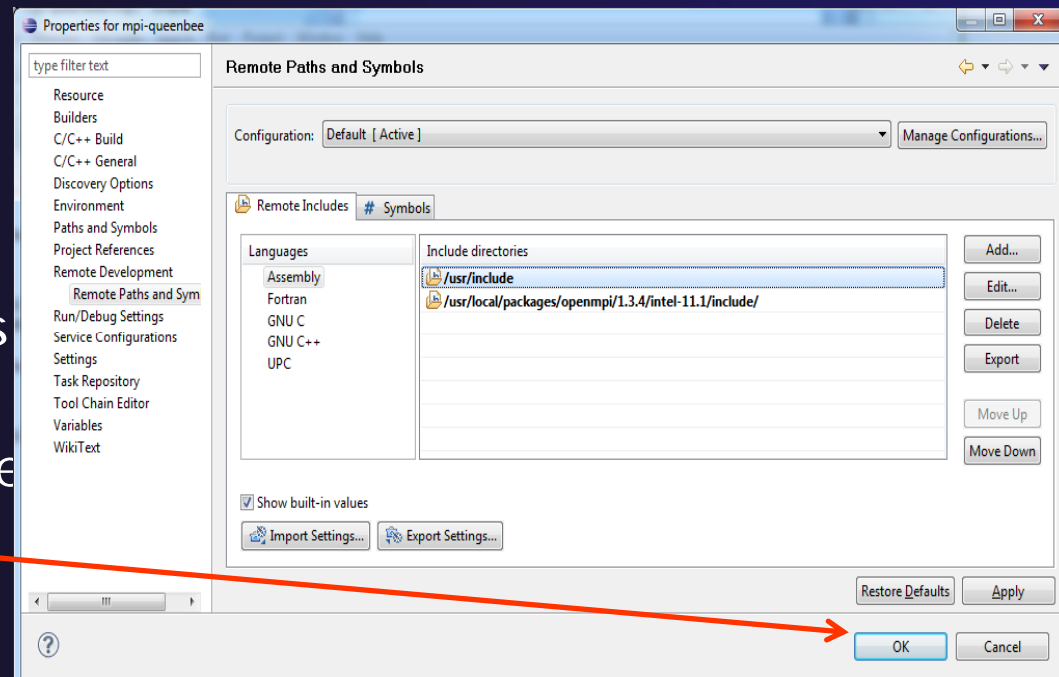
- ★ And you have the paths you need for your platform
  - ★ Note that the details differ by platform
  - ★ Press "OK" to finalize the changes



# Remote include paths

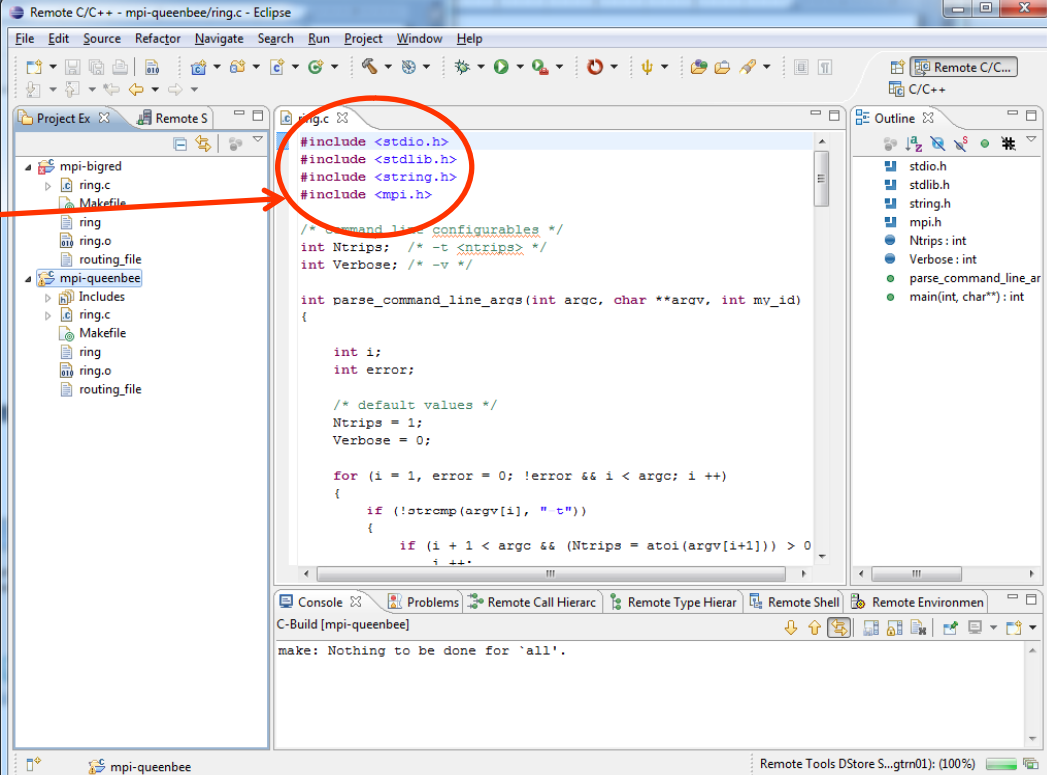


- ★ And you have the paths you need for your platform
  - ★ Note that the details differ by platform
  - ★ Press "OK" to finalize the changes
- ★ Allow Eclipse to rebuild the index by pressing "Yes"



# Include path issues resolved

- ★ Verify that include paths are now resolved



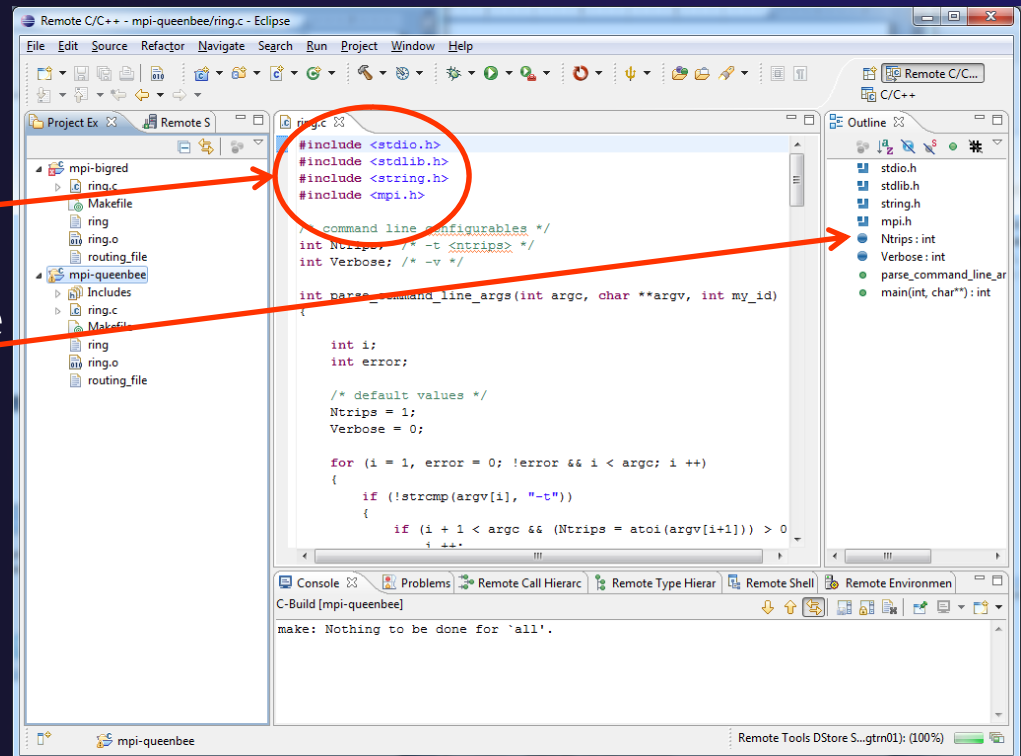
The screenshot shows the Eclipse IDE interface for a remote C/C++ project named 'mpi-queenbee/ring.c'. The Project Explorer on the left shows the project structure, including a sub-project 'mpi-queenbee' with an 'Includes' folder. The main editor displays the source code for 'ring.c', which includes the following headers:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mpi.h>
```

These include statements are circled in red. The console at the bottom shows the output of a 'make' command, indicating that the build process completed successfully with 'Nothing to be done for `all`'.

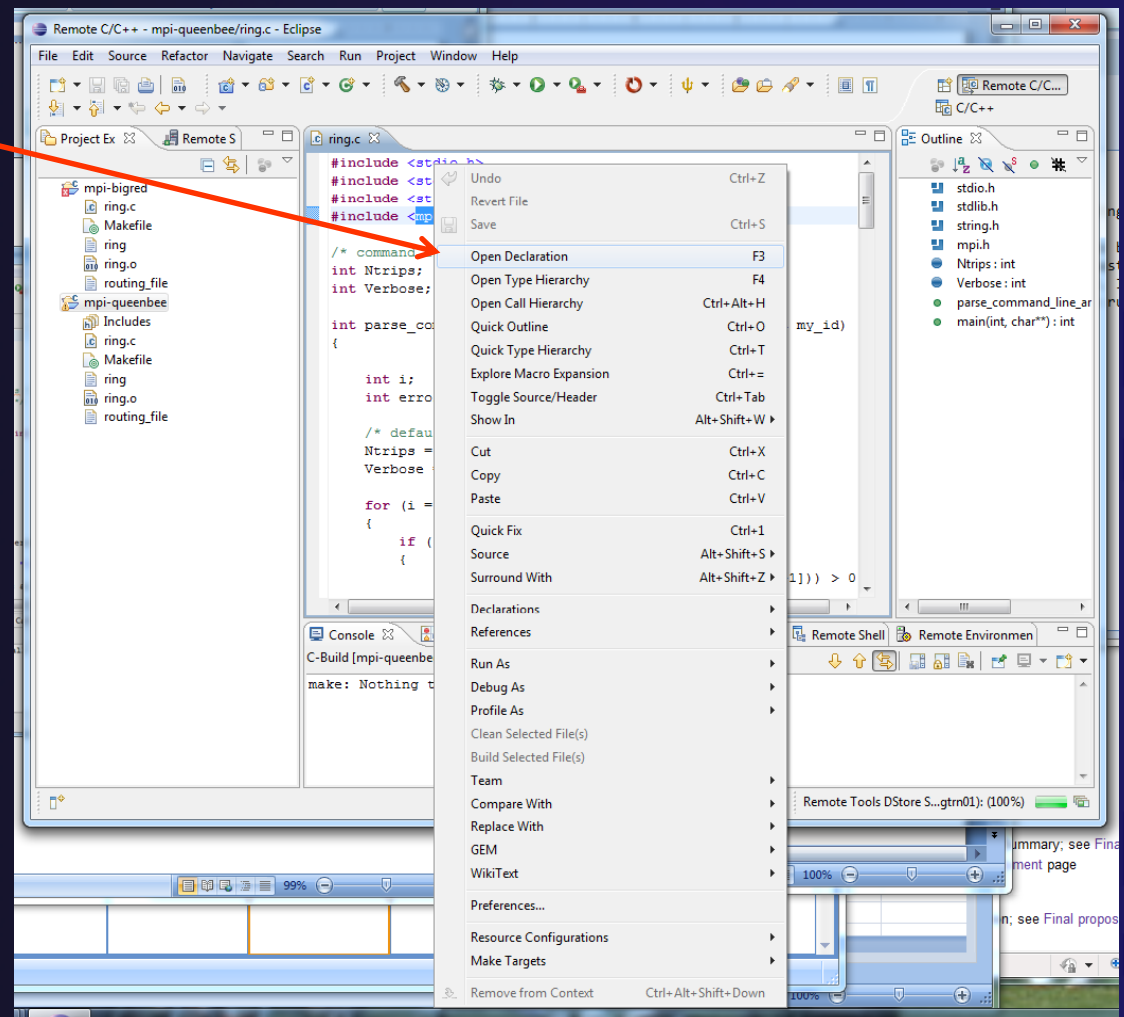
# Include path issues resolved

- ★ Verify that include paths are now resolved
- ★ Also note the outline view



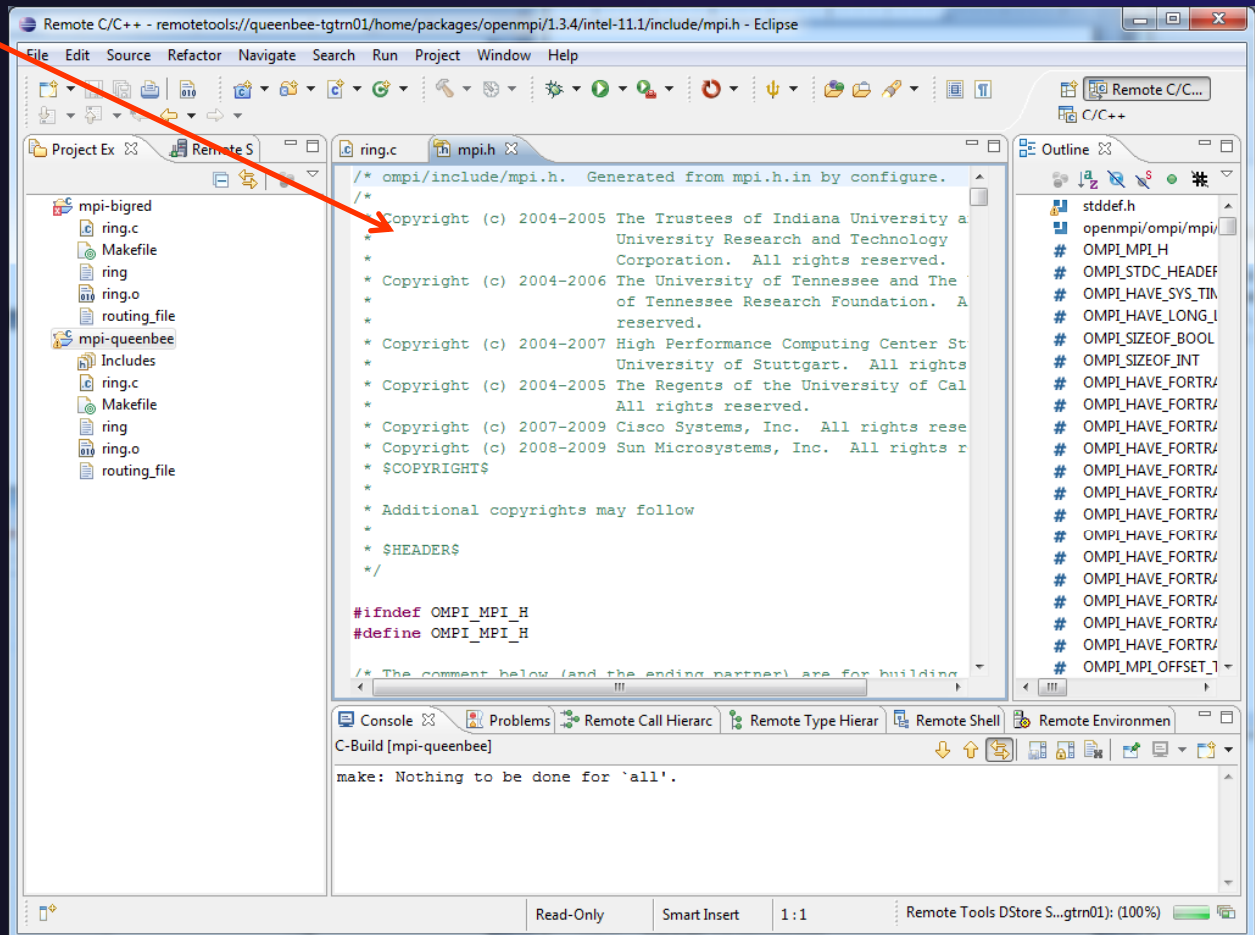
# Follow an include path...

- ★ Right-click on `mpi.h`, and select **Open Declaration**



# Following declarations

- ★ And you can see the contents of mpi.h on your system



```
Remote C/C++ - remotetools://queenbee-tgtrn01/home/packages/openmpi/1.3.4/intel-11.1/include/mpi.h - Eclipse
File Edit Source Refactor Navigate Search Run Project Window Help
Project Ex Remote S ring.c mpi.h Outline
mpi-bigred
  ring.c
  Makefile
  ring
  ring.o
  routing_file
mpi-queenbee
  Includes
  ring.c
  Makefile
  ring
  ring.o
  routing_file

/* ompi/include/mpi.h. Generated from mpi.h.in by configure.
 *
 * Copyright (c) 2004-2005 The Trustees of Indiana University a
 *                           University Research and Technology
 *                           Corporation. All rights reserved.
 * Copyright (c) 2004-2006 The University of Tennessee and The
 *                           of Tennessee Research Foundation. A
 *                           reserved.
 * Copyright (c) 2004-2007 High Performance Computing Center St
 *                           University of Stuttgart. All rights
 * Copyright (c) 2004-2005 The Regents of the University of Cal
 *                           All rights reserved.
 * Copyright (c) 2007-2009 Cisco Systems, Inc. All rights rese
 * Copyright (c) 2008-2009 Sun Microsystems, Inc. All rights r
 * $COPYRIGHT$
 *
 * Additional copyrights may follow
 *
 * $HEADERS
 */

#ifdef OMPI_MPI_H
#define OMPI_MPI_H

/* The comment below (and the ending partner) are for building
```

Console Problems Remote Call Hierarc Remote Type Hierarc Remote Shell Remote Environmen  
C-Build [mpi-queenbee]  
make: Nothing to be done for `all'.

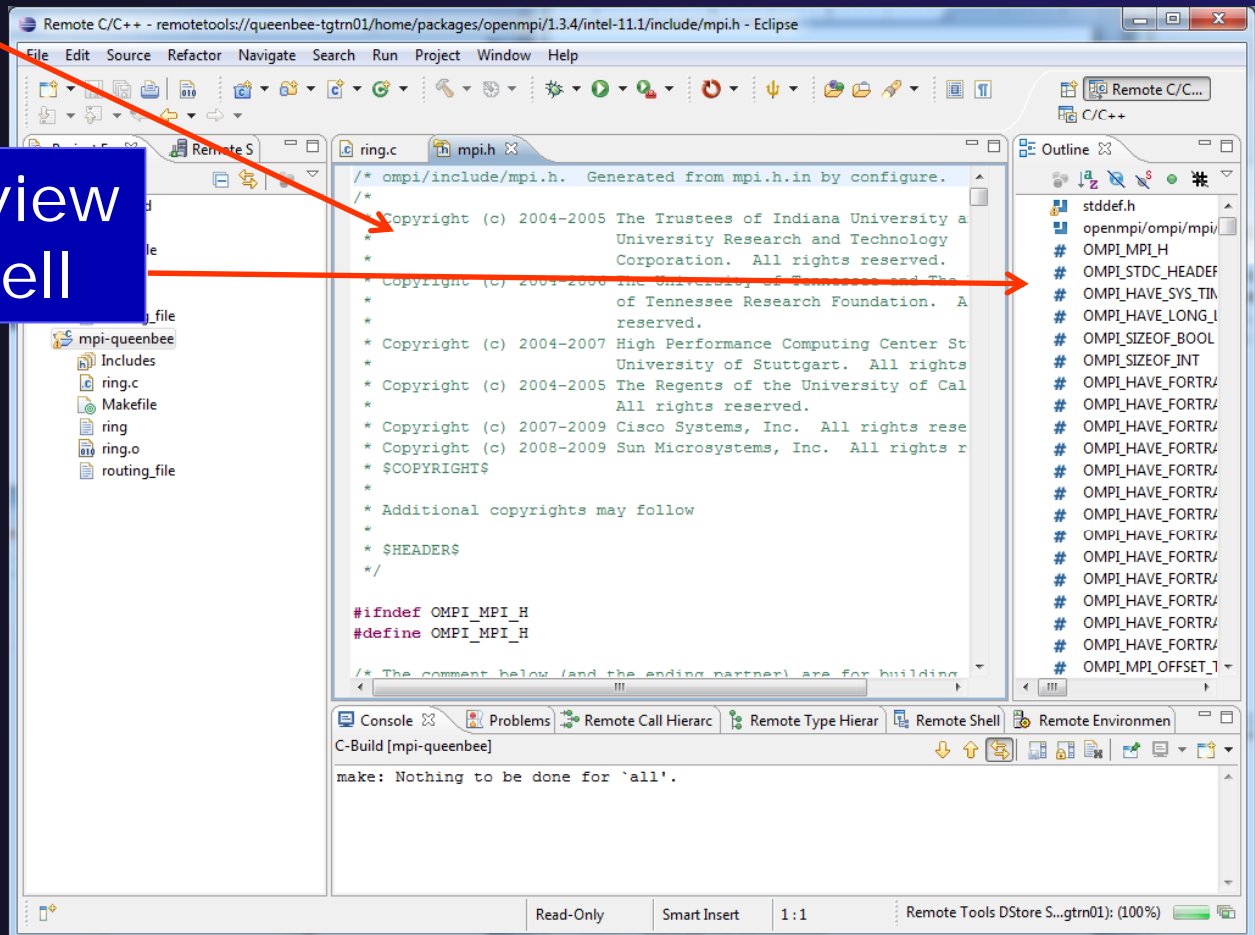
Read-Only Smart Insert 1:1 Remote Tools DStore S...gtrn01: (100%)



# Following declarations

- ★ And you can see the contents of mpi.h on your system

Note outline view changed as well



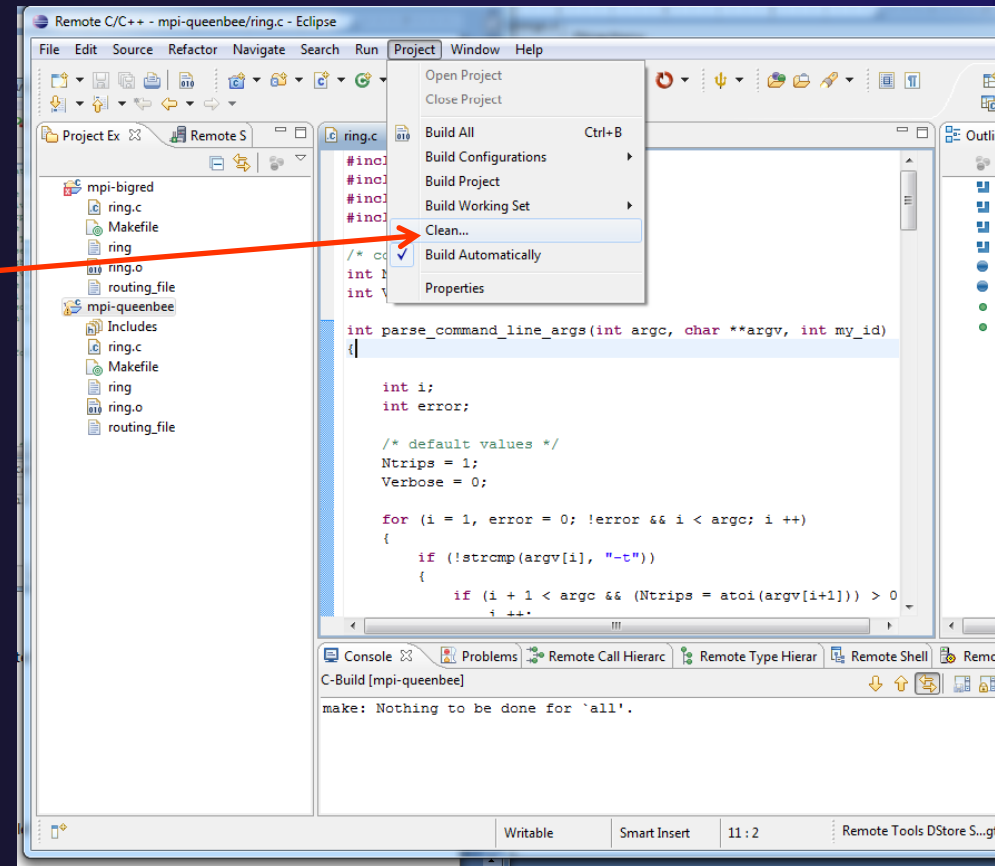
The screenshot shows the Eclipse IDE interface. The main editor displays the contents of the file `mpi.h`, which is a header file for OpenMPI. The file contains copyright information and preprocessor directives. The Outline view on the right shows the structure of the file, with the `OMPI_MPI_H` macro definition highlighted. The Console view at the bottom shows the output of a build command: `make: Nothing to be done for 'all'.`

```
/* ompi/include/mpi.h. Generated from mpi.h.in by configure.
 *
 * Copyright (c) 2004-2005 The Trustees of Indiana University a
 * University Research and Technology Corporation. All rights reserved.
 * Copyright (c) 2004-2006 The University of Tennessee and The
 * of Tennessee Research Foundation. All rights reserved.
 * Copyright (c) 2004-2007 High Performance Computing Center St
 * University of Stuttgart. All rights reserved.
 * Copyright (c) 2004-2005 The Regents of the University of Cal
 * All rights reserved.
 * Copyright (c) 2007-2009 Cisco Systems, Inc. All rights rese
 * Copyright (c) 2008-2009 Sun Microsystems, Inc. All rights r
 * $COPYRIGHT$
 *
 * Additional copyrights may follow
 *
 * $HEADERS
 */
#ifdef OMPI_MPI_H
#define OMPI_MPI_H

/* The comment below (and the ending partner) are for building
```

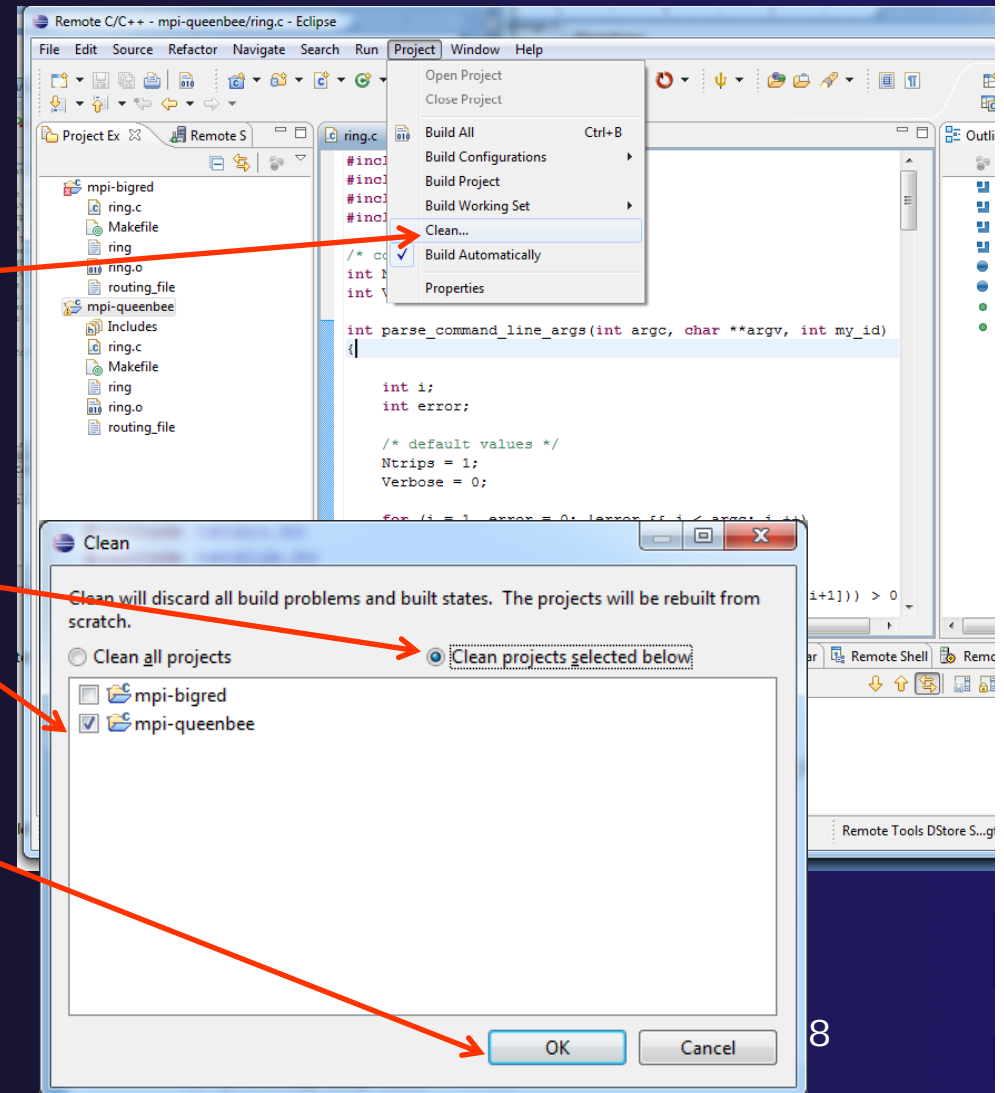
# Building your code

- ★ Let's start our build by doing a clean build
  - ★ Project > Clean...



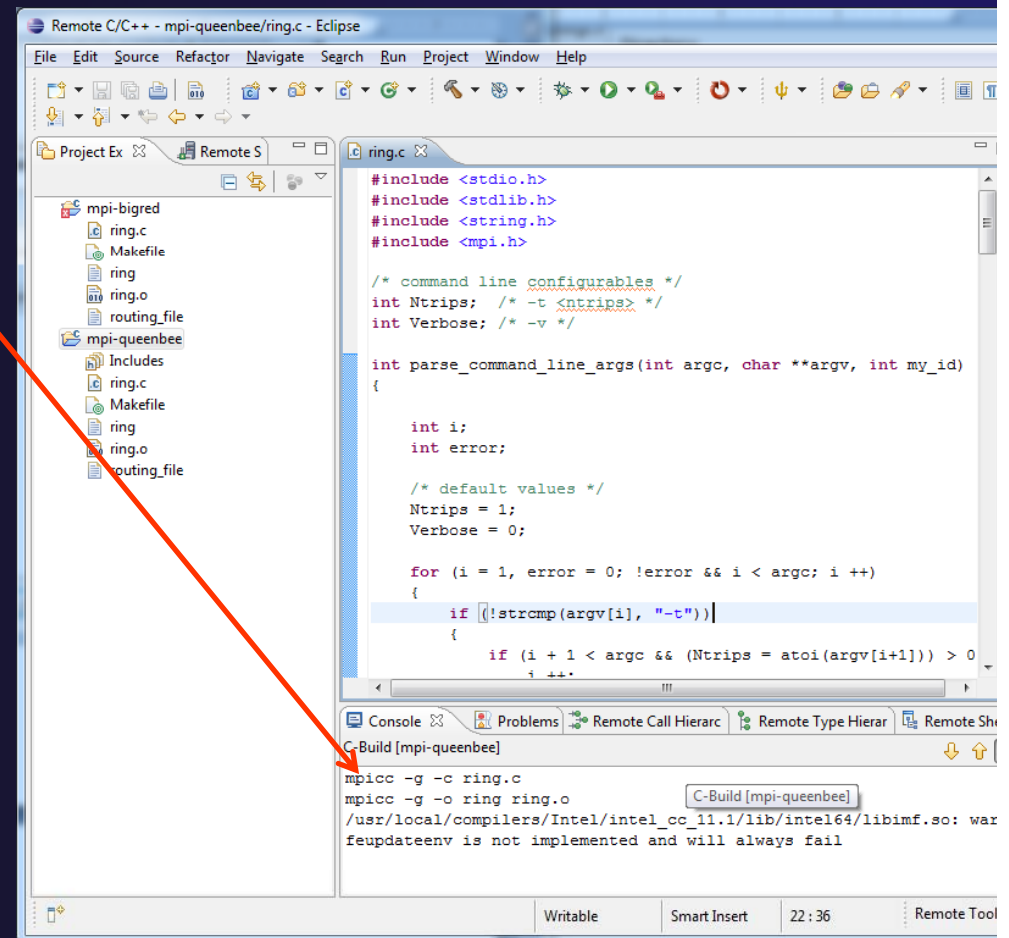
# Building your code

- ★ Let's start our build by doing a clean build
  - ★ **Project > Clean...**
  - ★ Select the project we are working on (and the clean projects selected below button)
  - ★ Press OK



## Build clean

- ★ Note output from compiler/make



The screenshot shows the Eclipse IDE interface. The left sidebar displays a project tree with folders for 'mpi-bigred' and 'mpi-queenbee'. The main editor window shows the source code for 'ring.c', which includes headers for `<stdio.h>`, `<stdlib.h>`, `<string.h>`, and `<mpi.h>`. The code defines command-line options for `Ntrips` and `Verbose`, and contains a `parse_command_line_args` function. The bottom console window shows the output of a clean build:

```
C-Build [mpi-queenbee]
mpicc -g -c ring.c
mpicc -g -o ring ring.o
/usr/local/compilers/Intel/intel_cc_11.1/lib/intel64/libimf.so: warning: feupdateenv is not implemented and will always fail
```

An orange arrow points from the text 'Note output from compiler/make' to the console output.

# Build clean

- ★ Note output from compiler/make
- ★ Ring, ring.o should also appear in your project

```
Remote C/C++ - mpi-queenbee/ring.c - Eclipse
File Edit Source Refactor Navigate Search Run Project Window Help
Project Explorer Remote S
mpi-bigred
  ring.c
  Makefile
  ring
  ring.o
  routing_file
mpi-queenbee
  Includes
  ring.c
  Makefile
  ring
  ring.o
  routing_file
ring.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mpi.h>

/* command line configurables */
int Ntrips; /* -t <ntrips> */
int Verbose; /* -v */

int parse_command_line_args(int argc, char **argv, int my_id)
{
    int i;
    int error;

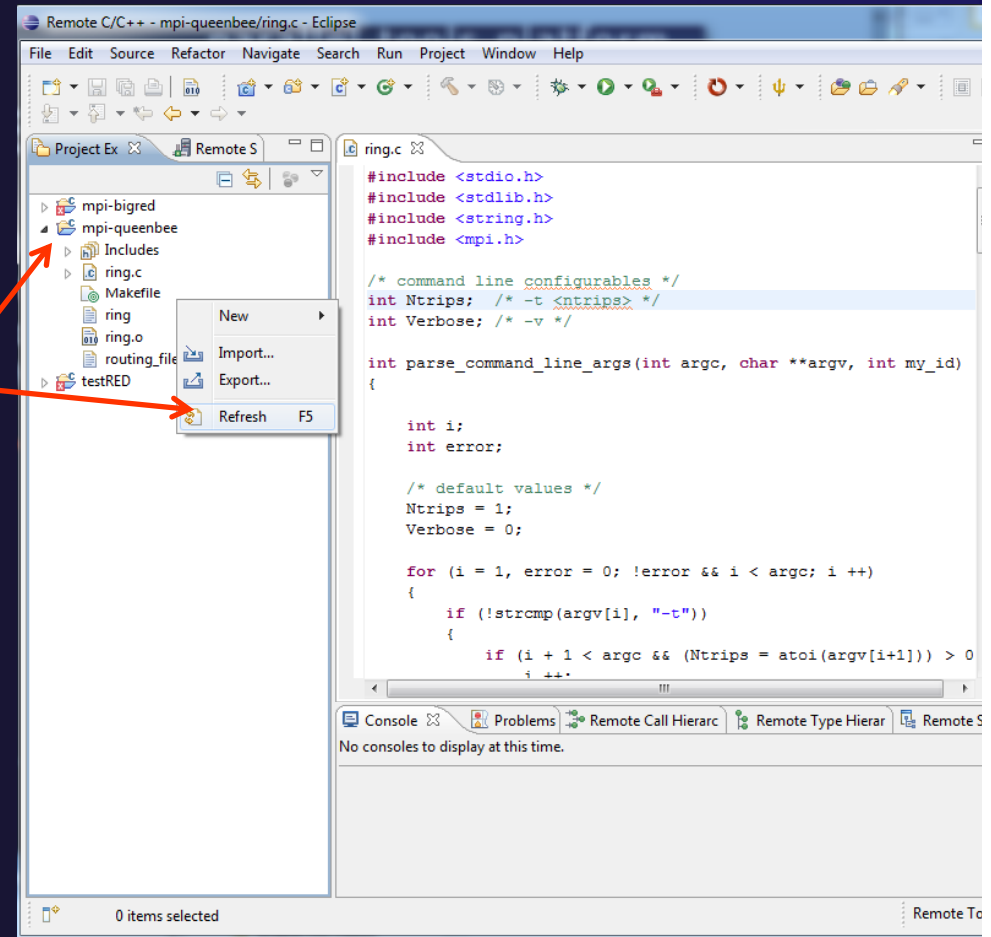
    /* default values */
    Ntrips = 1;
    Verbose = 0;

    for (i = 1, error = 0; !error && i < argc; i++)
    {
        if (!strcmp(argv[i], "-t"))
        {
            if (i + 1 < argc && (Ntrips = atoi(argv[i+1])) > 0
                i++;
        }
    }
}

Console Problems Remote Call Hierarc Remote Type Hierarc Remote Shu
C-Build [mpi-queenbee]
mpicc -g -c ring.c
mpicc -g -o ring ring.o
/usr/local/compilers/Intel/intel_cc_11.1/lib/intel64/libimf.so: warning: feupdateenv is not implemented and will always fail
Writable Smart Insert 22:36 Remote Tool
```

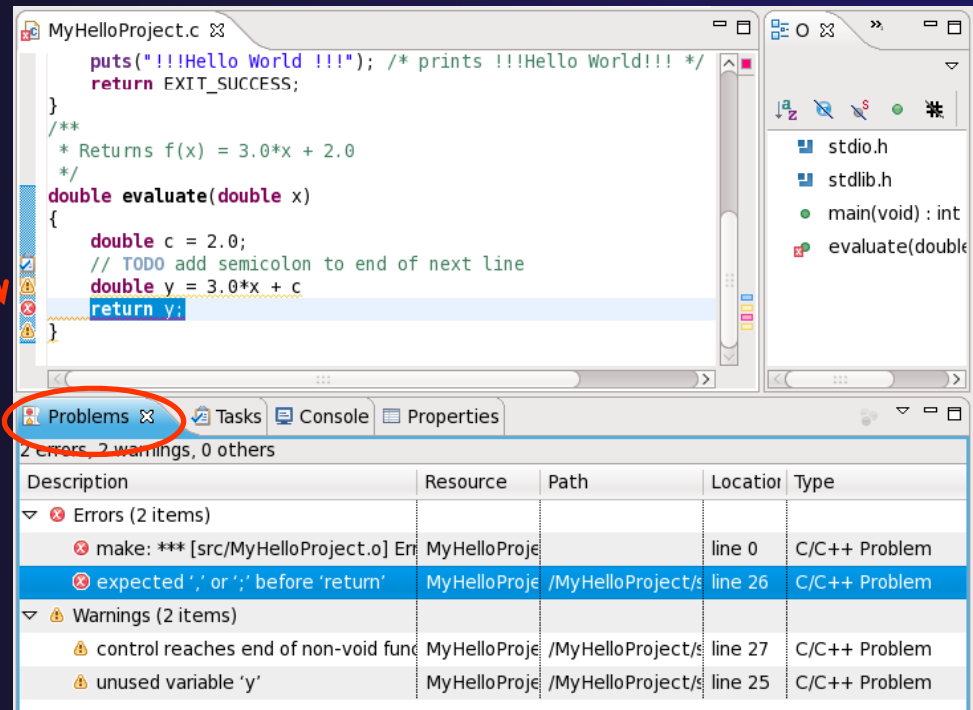
# Build clean

- ★ Note you can refresh your project's file list – right click on Project Explorer, then Refresh
- ★ You can also refresh specific projects, by right-clicking on the project itself



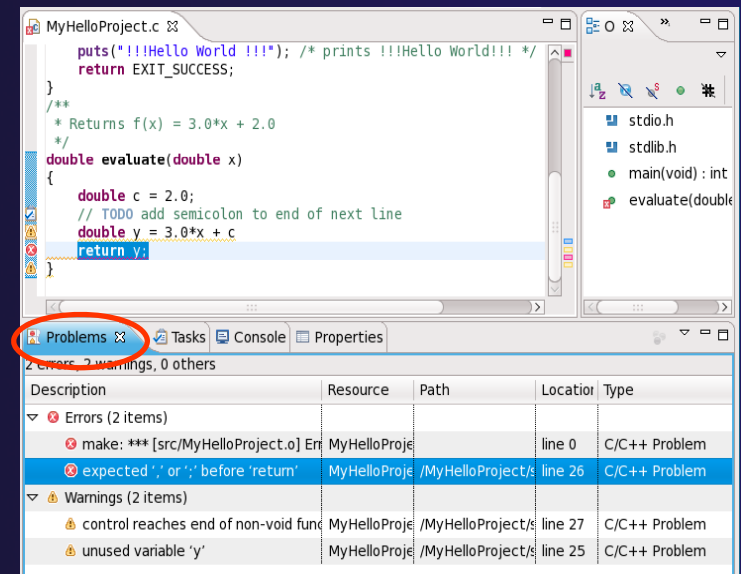
# Build problems?

- ✦ If there are problems, see:
- ✦ Marker on editor line
- ✦ **Problems view**
- ✦ Double-click on line in **Problems** view to go to location of error



# Build problems? Try it

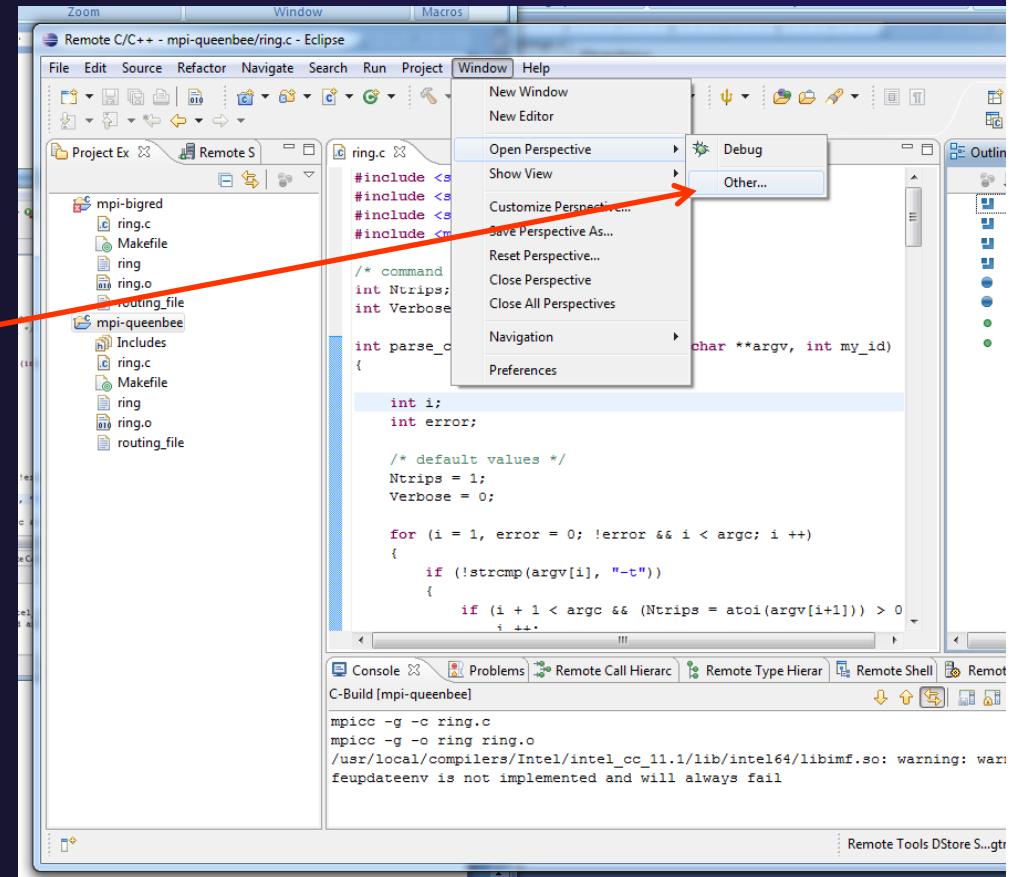
- ✦ Remove a semicolon from a line in your “Hello World” example
- ✦ Save file
- ✦ Rebuild
- ✦ **See the Problems view**
- ✦ Double-click on line in **Problems** view to go to location of error
- ✦ Fix it and rebuild to continue





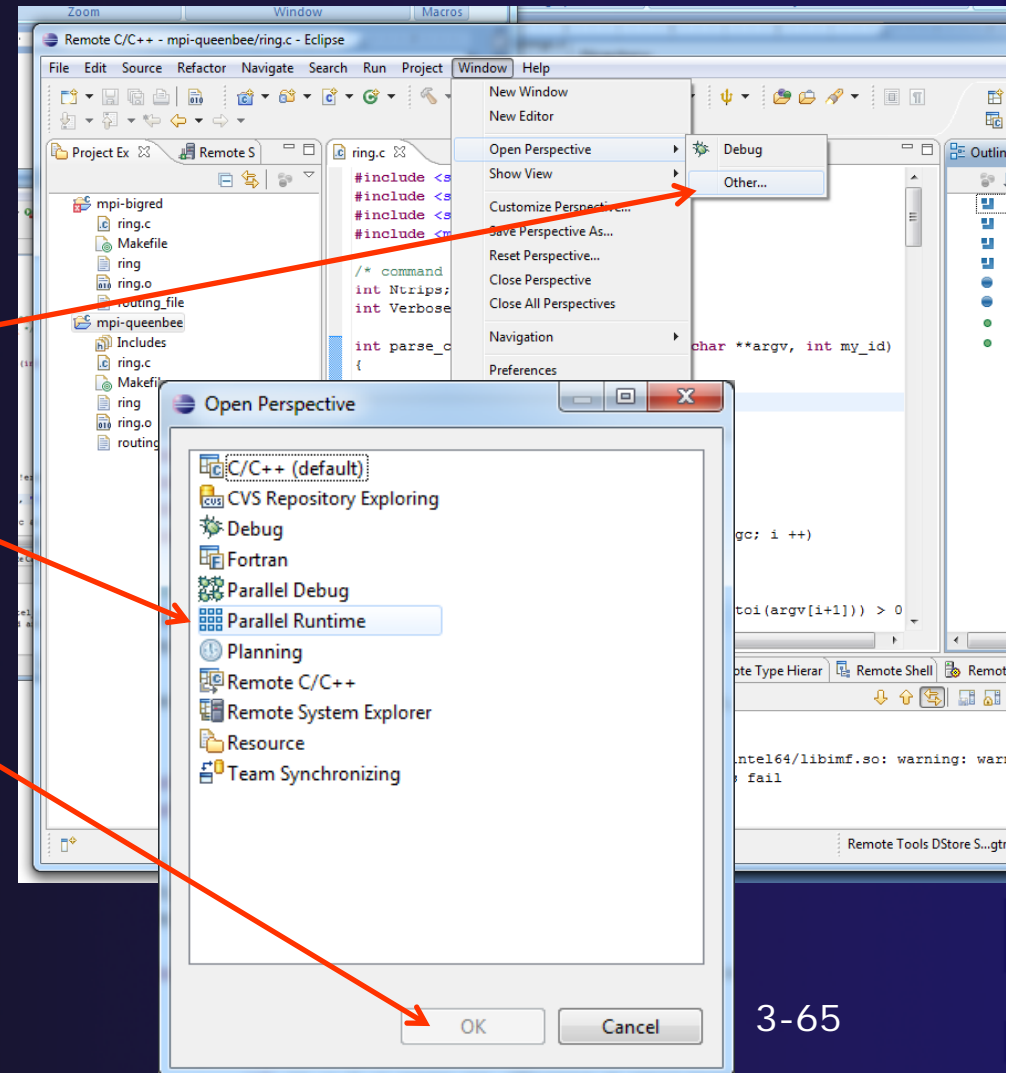
# Running your code

- ✦ Switch to the parallel runtime perspective
- ✦ **Window > Open Perspective > Other...**



# Running your code

- ★ Switch to the parallel runtime perspective
- ★ **Window > Open Perspective > Other...**
- ★ Choose **Parallel Runtime**
- ★ Then press **OK**



# Terminology

- ★ The **PTP Runtime** perspective is provided for monitoring and controlling applications
- ★ Some terminology
  - ★ **Resource manager** - Corresponds to an instance of a resource management system (e.g. a job scheduler). You can have multiple resource managers connected to different machines.
  - ★ **Queue** - A queue of pending jobs
  - ★ **Job** - A single run of a parallel application
  - ★ **Machine** - A parallel computer system
  - ★ **Node** - Some form of computational resource
  - ★ **Process** - An execution unit (may be multiple threads of execution)

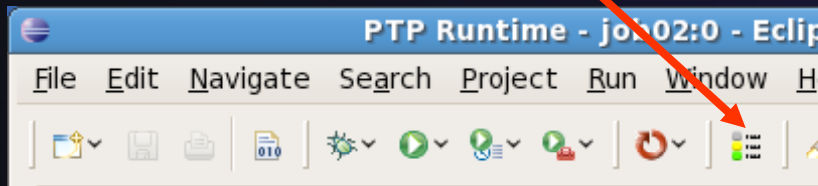
# Resource Managers

- ★ PTP uses the term “resource manager” to refer to any subsystem that controls the resources required for launching a parallel job.
- ★ Examples:
  - ★ Job scheduler (e.g. LoadLeveler)
  - ★ Open MPI Runtime Environment (ORTE)
- ★ Each resource manager controls one target system
- ★ Resource Managers can be local or remote



# About PTP Icons

- ★ Open using legend icon in toolbar



**Legend**

Resource Manager Icons

- STARTING
- STARTED
- STOPPING
- STOPPED
- SUSPENDED
- ERROR

Machine Icons	Node Icons
UP	UP
DOWN	DOWN
ALERT	ERROR
ERROR	UNKNOWN
UNKNOWN	USER EXCLUSIVE
	USER SHARED
	OTHER EXCLUSIVE
	OTHER SHARED
	PROCESS RUNNING
	PROCESS TERMINATED

Job Icons	Process Icons
PENDING	STARTING
STARTED	RUNNING
RUNNING	EXITED NORMALLY
TERMINATED	EXITED WITH SIGNAL
SUSPENDED	SUSPENDED
ERROR	ERROR
UNKNOWN	UNKNOWN

Close



# PTP Runtime Perspective

Resource managers view

Machines view

Node details view

Jobs view

The screenshot shows the Eclipse IDE interface for the PTP Runtime perspective. The top toolbar contains various icons for file operations and development tools. The main workspace is divided into several views:

- Resource Managers:** A view for managing resources, currently empty.
- Machines:** A view for selecting a machine, currently showing "Please select a machine".
- Node Attributes:** A table with columns "Attribute" and "Value".
- Process Info:** A table for process information.
- Jobs List:** A table with columns "State" and "Name".
- Console:** A view showing the output of a C-build process, including commands like "make all-am", "mpicc", "mv", and "mpicc" for various object files.
- Properties:** A view showing a table with columns "Property" and "Value".

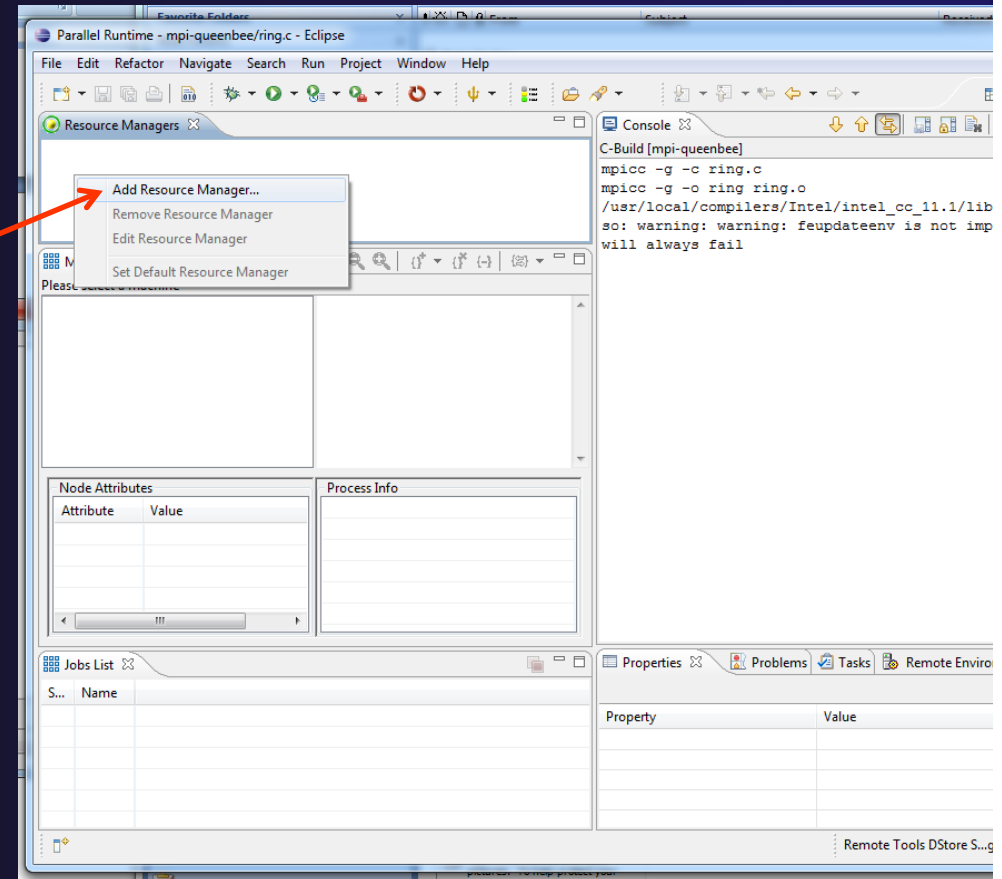
Red arrows point from the text labels on the left to the corresponding views in the screenshot.

Console view

Properties view

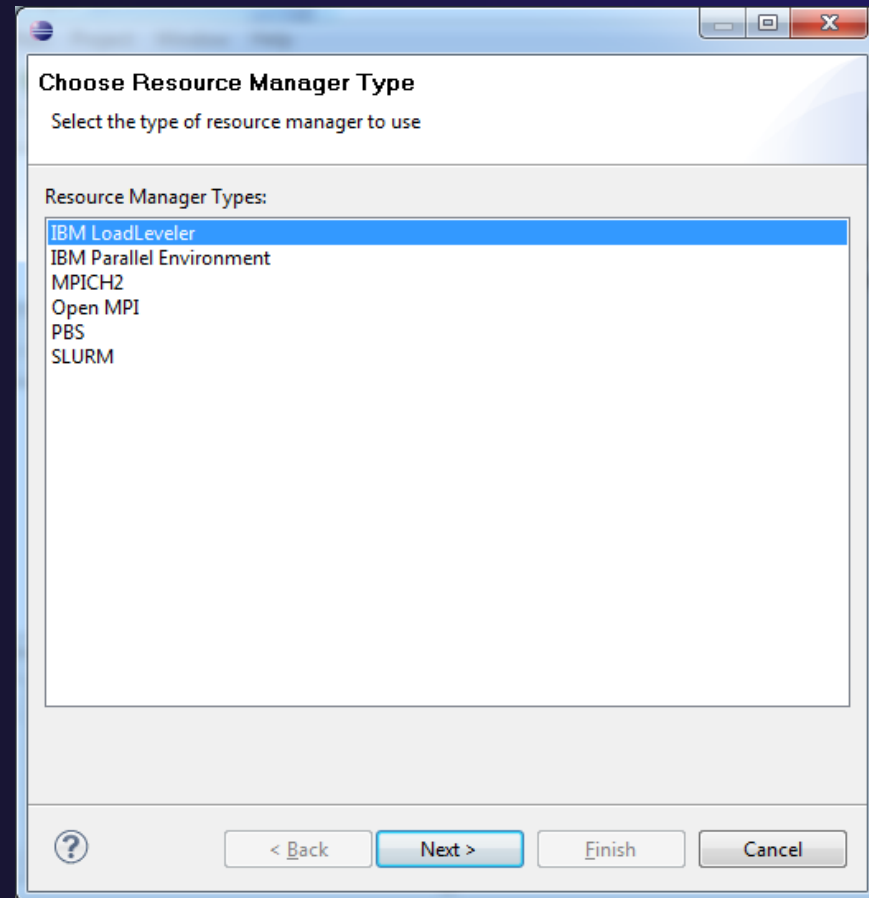
# Add a resource manager

- ★ Right click in resource managers view, select **Add Resource Manager...**



# Add resource manager

- ★ We'll add 2 resource managers for queenbee – one for OpenMPI, another for PBS

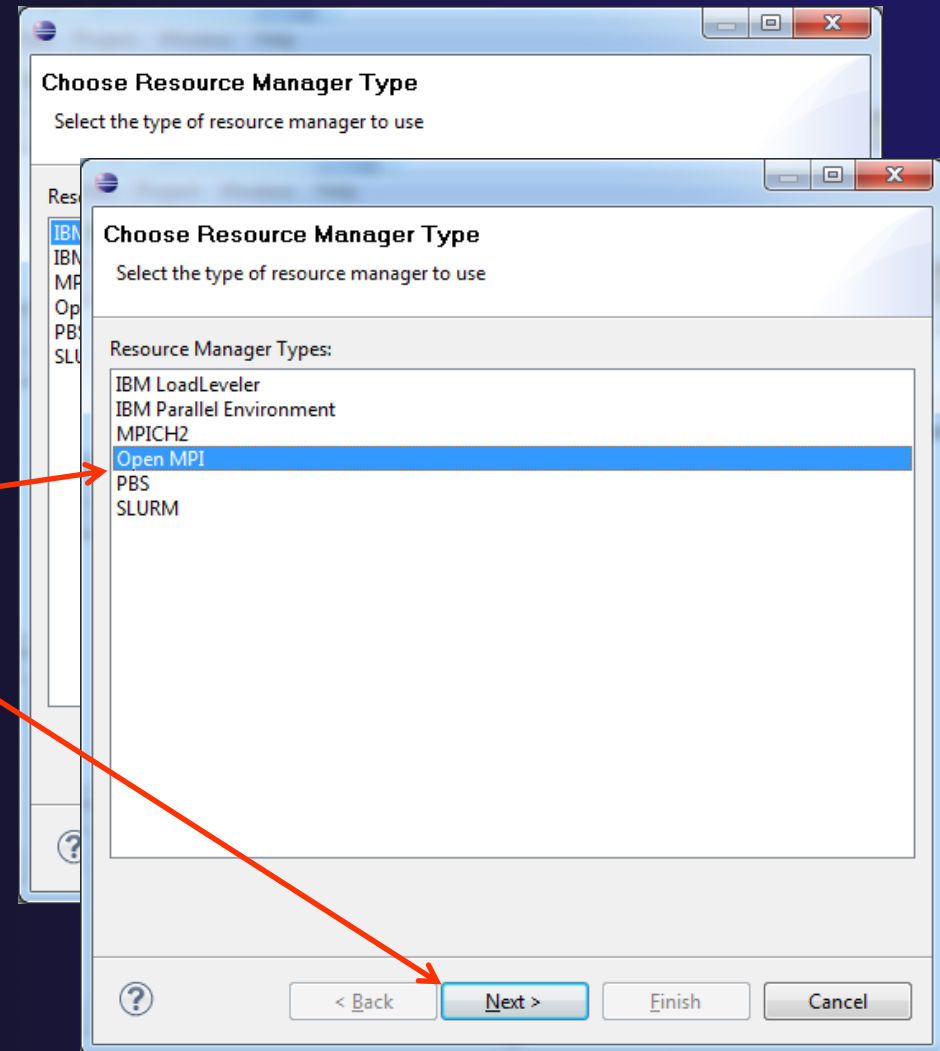




# Add resource manager



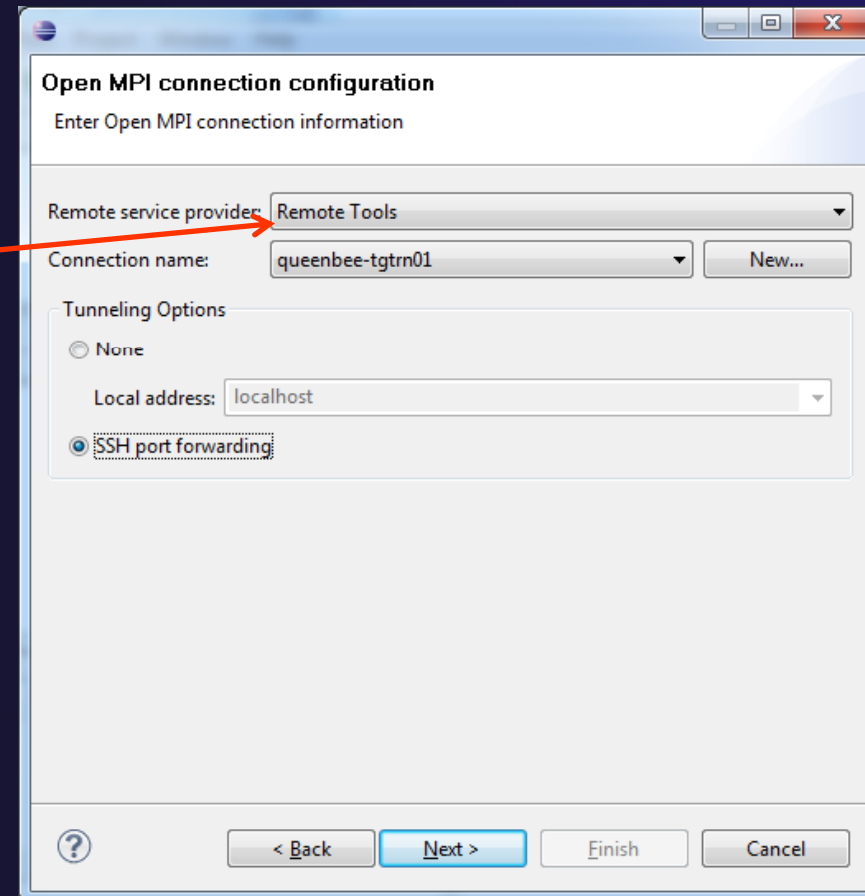
- ★ We'll add 2 resource managers for queenbee – one for OpenMPI, another for PBS
- ★ Choose OpenMPI, then select Next



# OpenMPI Resource Manager



- ★ Select **Remote Tools** as the remote service provider



Open MPI connection configuration

Enter Open MPI connection information

Remote service provider: Remote Tools

Connection name: queenbee-tgtrn01 New...

Tunneling Options

None

Local address: localhost

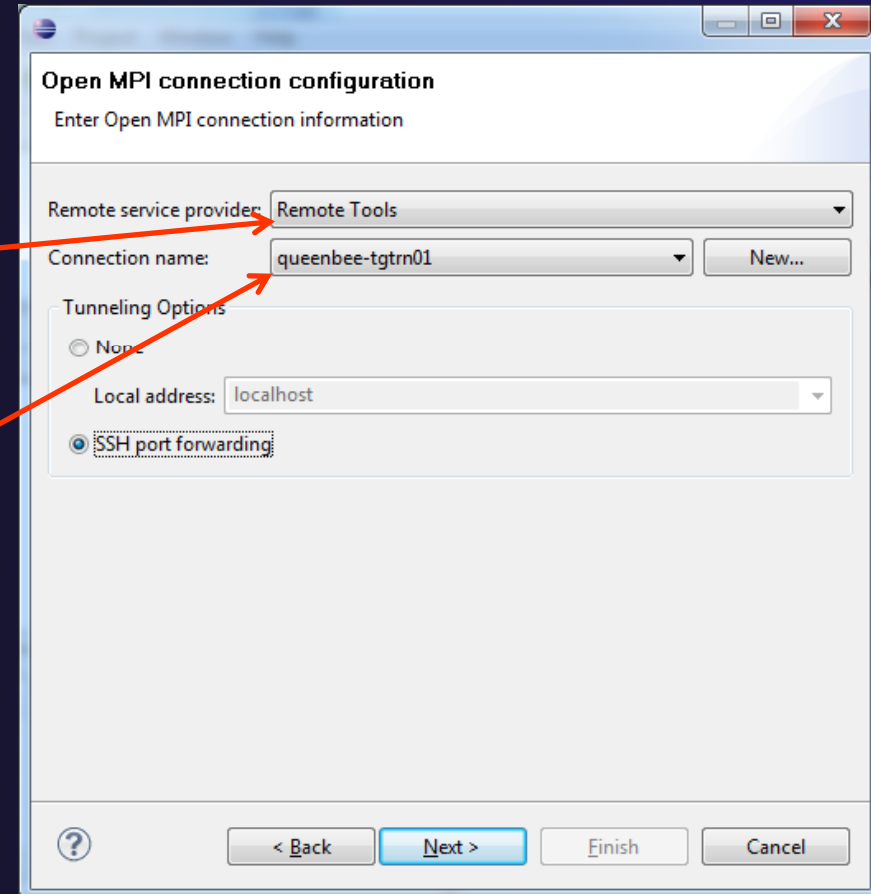
SSH port forwarding

? < Back Next > Finish Cancel

# OpenMPI Resource Manager



- ★ Select **Remote Tools** as the remote service provider
- ★ Then select the appropriate connection name



Open MPI connection configuration

Enter Open MPI connection information

Remote service provider: Remote Tools

Connection name: queenbee-tgtrn01

Tunneling Options

None

Local address: localhost

SSH port forwarding

< Back Next > Finish Cancel

# OpenMPI Resource Manager



- ★ Select **Remote Tools** as the remote service provider
- ★ Then select the appropriate connection name
- ★ **Important:** be sure to click **SSH port forwarding** to enable ssh tunneling of connections back to your laptop

Open MPI connection configuration

Enter Open MPI connection information

Remote service provider: Remote Tools

Connection name: queenbee-tgtrn01 [New...]

Tunneling Options

None

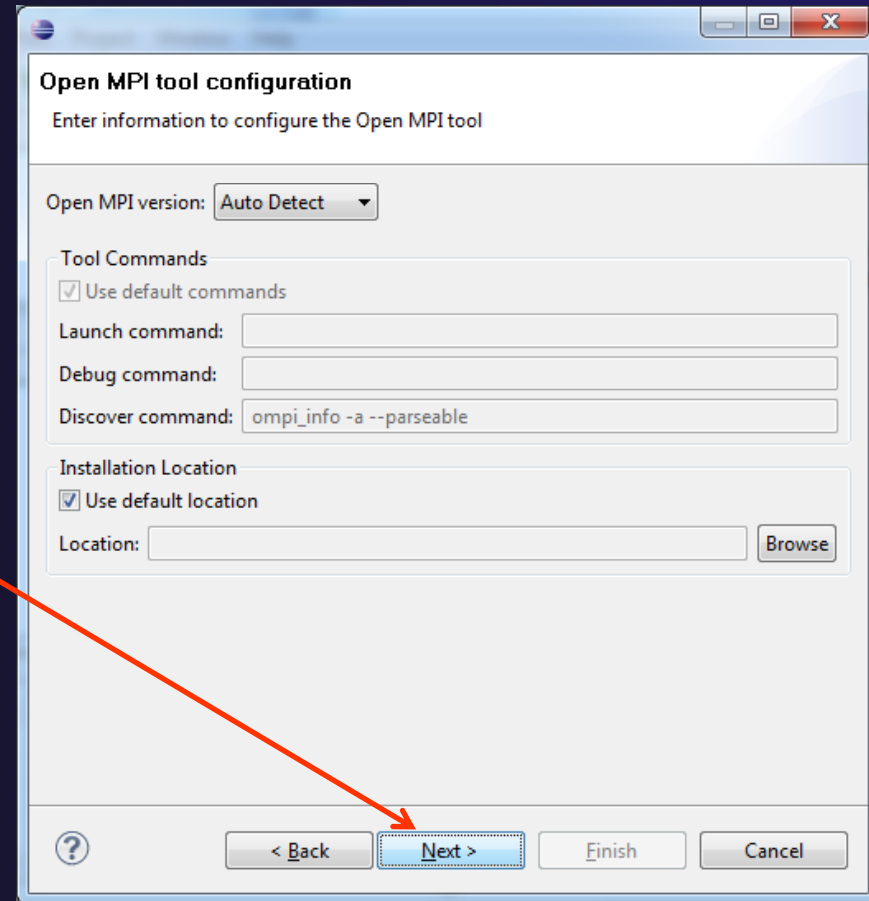
Local address: localhost

SSH port forwarding

< Back Next > Finish Cancel

# OpenMPI wizard

- ★ Leave this page as default values, select **next**



Open MPI tool configuration

Enter information to configure the Open MPI tool

Open MPI version: Auto Detect

Tool Commands

Use default commands

Launch command:

Debug command:

Discover command:

Installation Location

Use default location

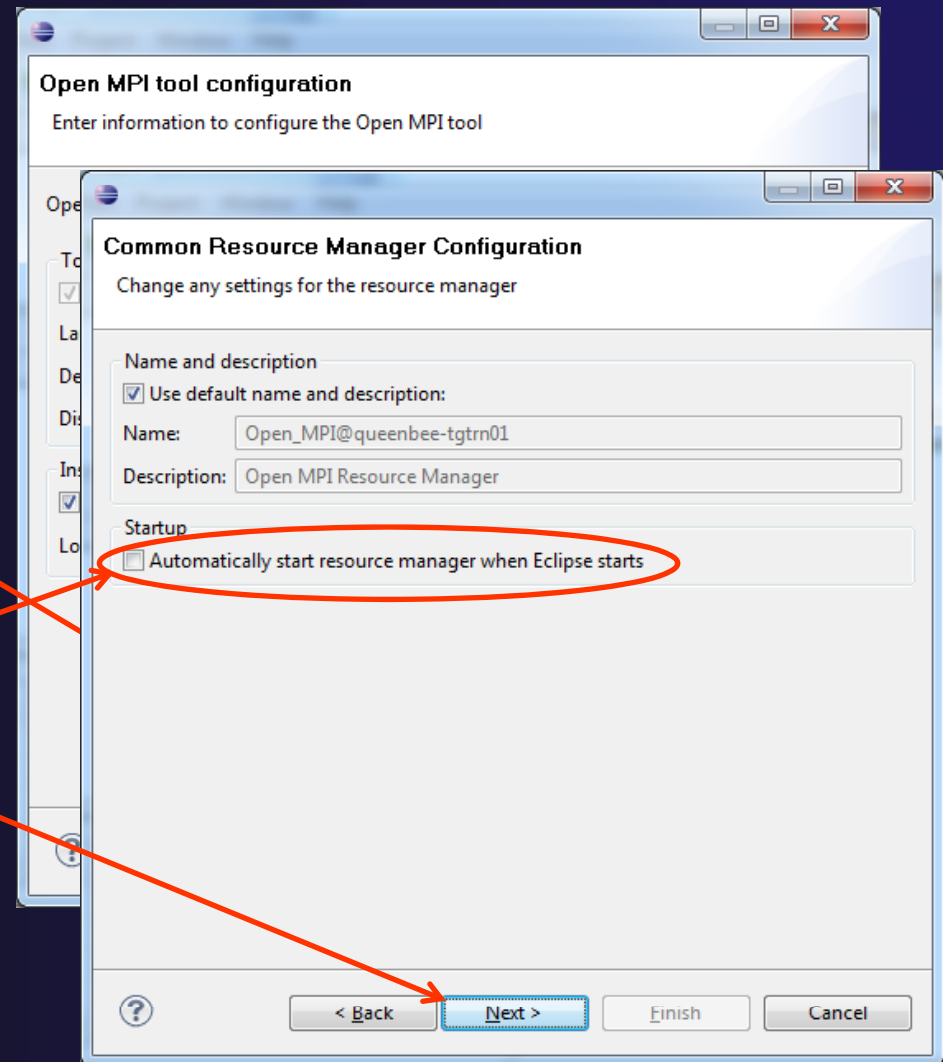
Location:

? < Back **Next >** Finish Cancel

# OpenMPI wizard



- ★ Leave this page as default values, select **next**
- ★ Only start resource manager when you need it (leave Startup unselected)
  - ★ Press **Next**





# OpenMPI wizard

- ★ Leave this page as default values, select **next**
- ★ Only start resource manager when you need it (leave Startup unselected)
  - ★ Press **Next**
- ★ Add to an existing service configuration
  - ★ Select the machine we were working on, then select **Finish**

Open MPI tool configuration  
Enter information to configure the Open MPI tool

Common Resource Manager Configuration  
Change any settings for the resource manager

Select Service Configuration  
Select a service configuration for this resource manager, or create a new one.

Create a new Service Configuration  
Configuration Name:

Add to an existing Service Configuration

Configuration Name	Launch Service
bigred-tg-tra01	Not Configured
queenbee-tgtrn01	Not Configured

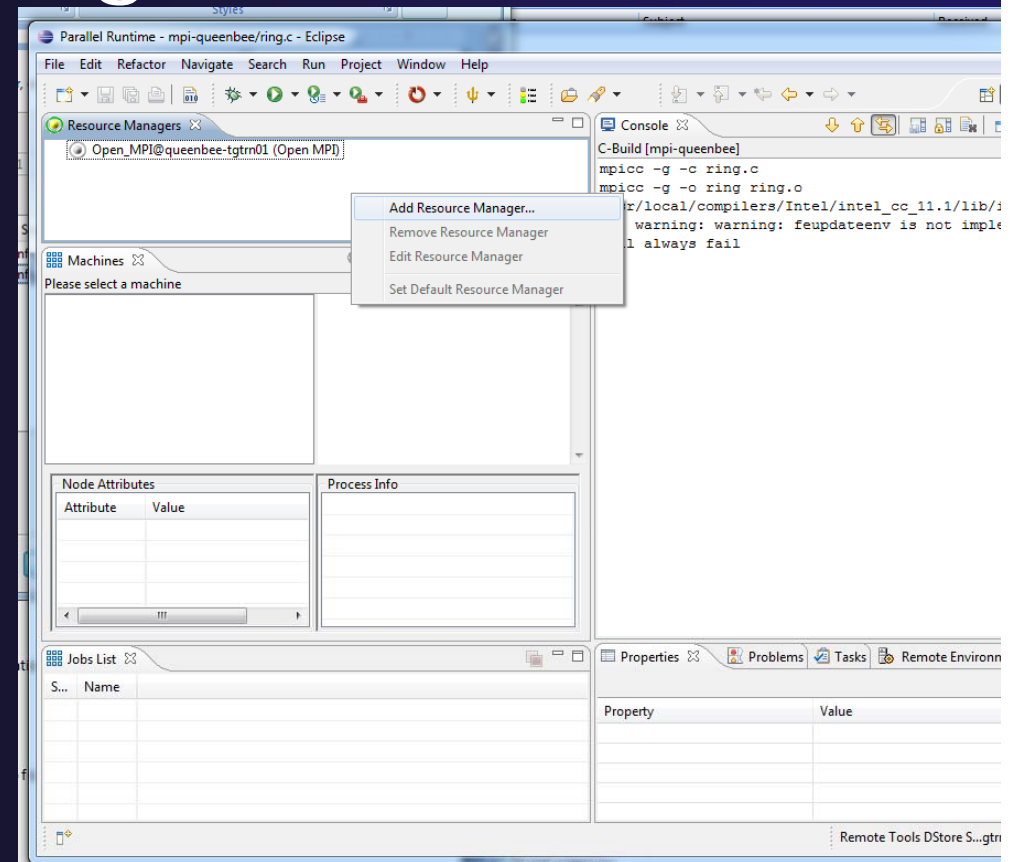
Advanced Settings...

< Back   Next >   **Finish**   Cancel

# Repeat for the PBS Resource Manager



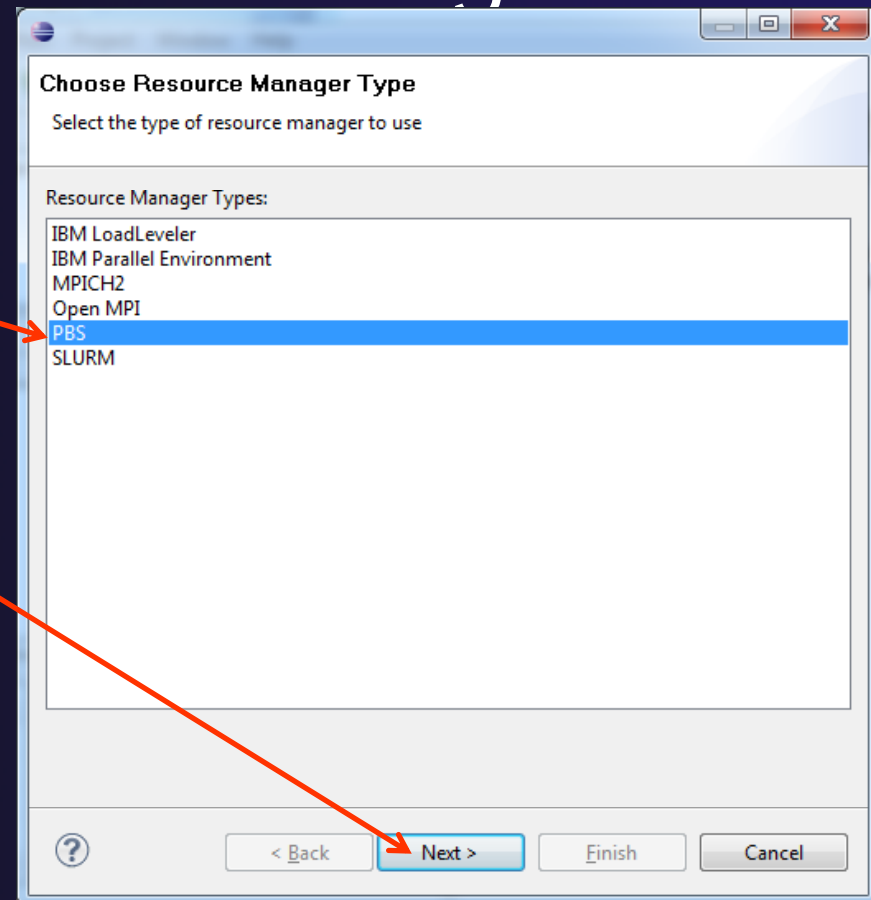
- ★ Right-click in the Resource Manager view, select **Add Resource Manager...** again





# PBS Resource Manager

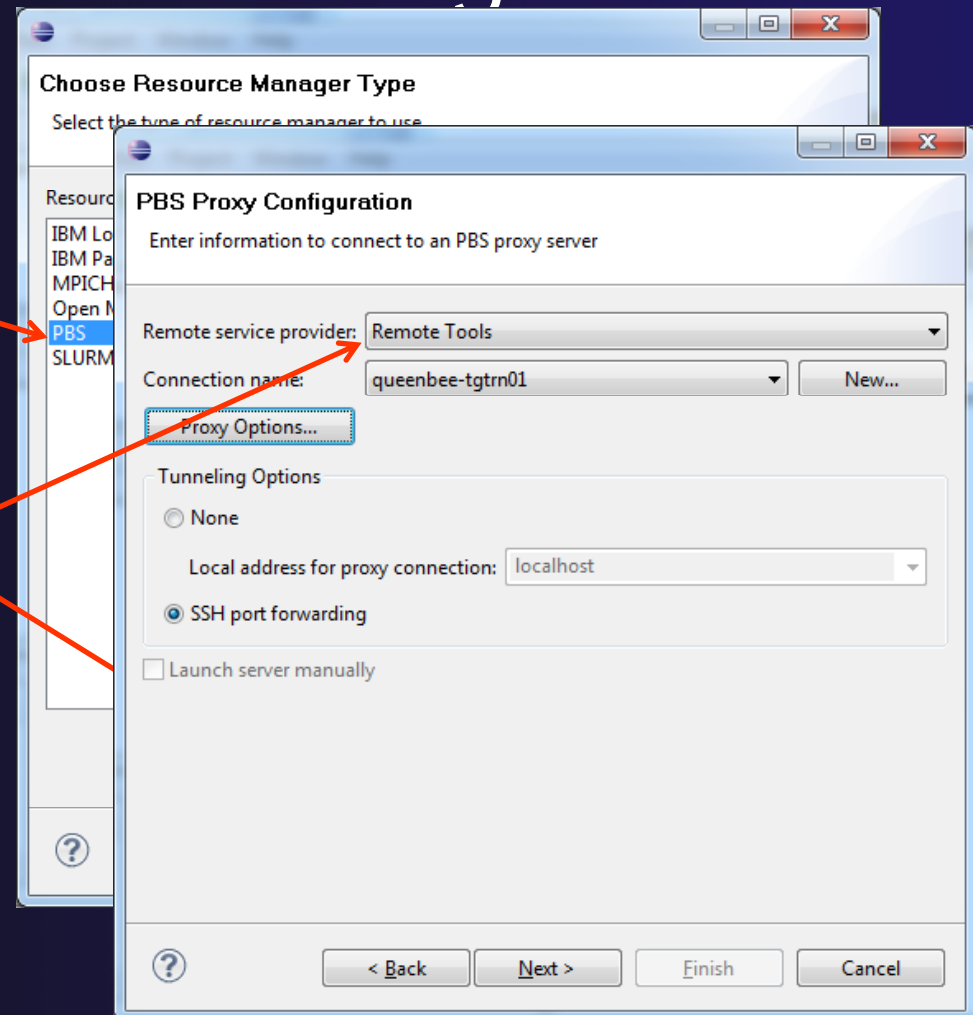
- ★ Select PBS Resource Manager... then **Next >**



# PBS Resource Manager



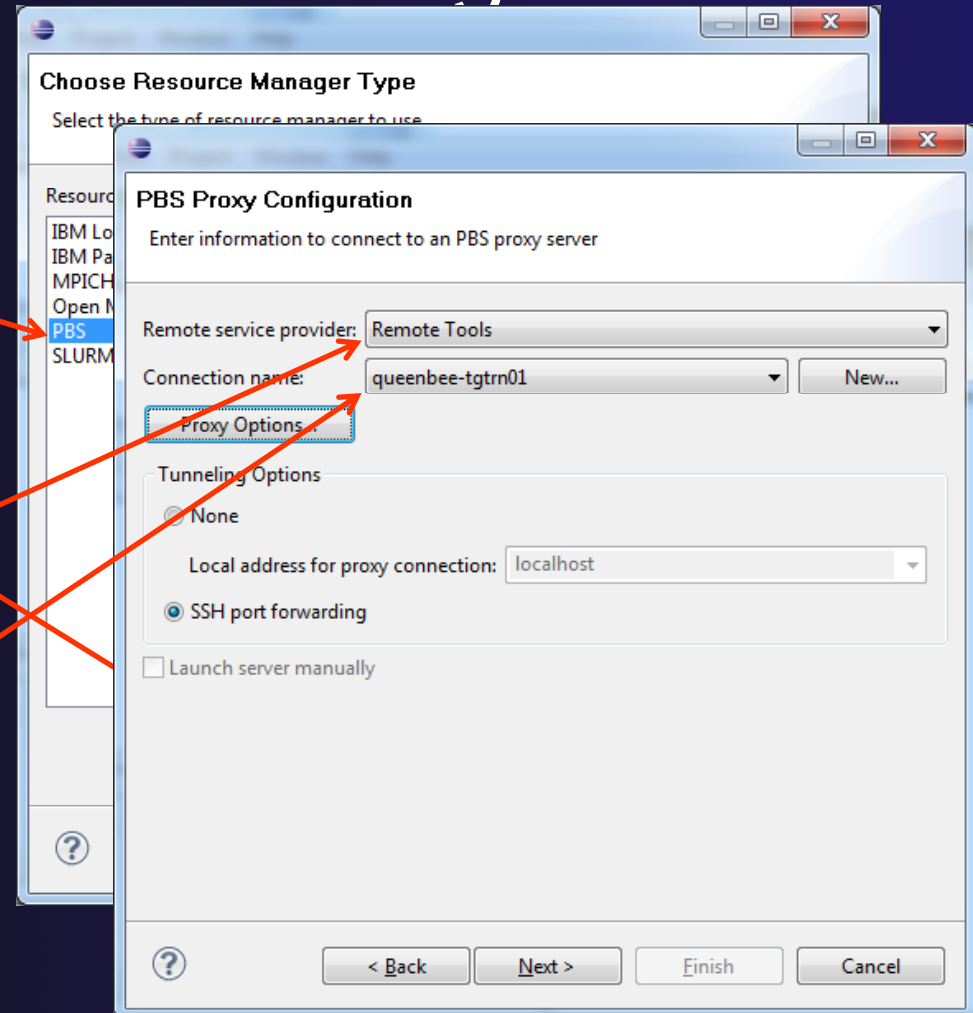
- ★ Select PBS Resource Manager... then **Next >**
- ★ Select **Remote Tools** Remote Service Provider



# PBS Resource Manager



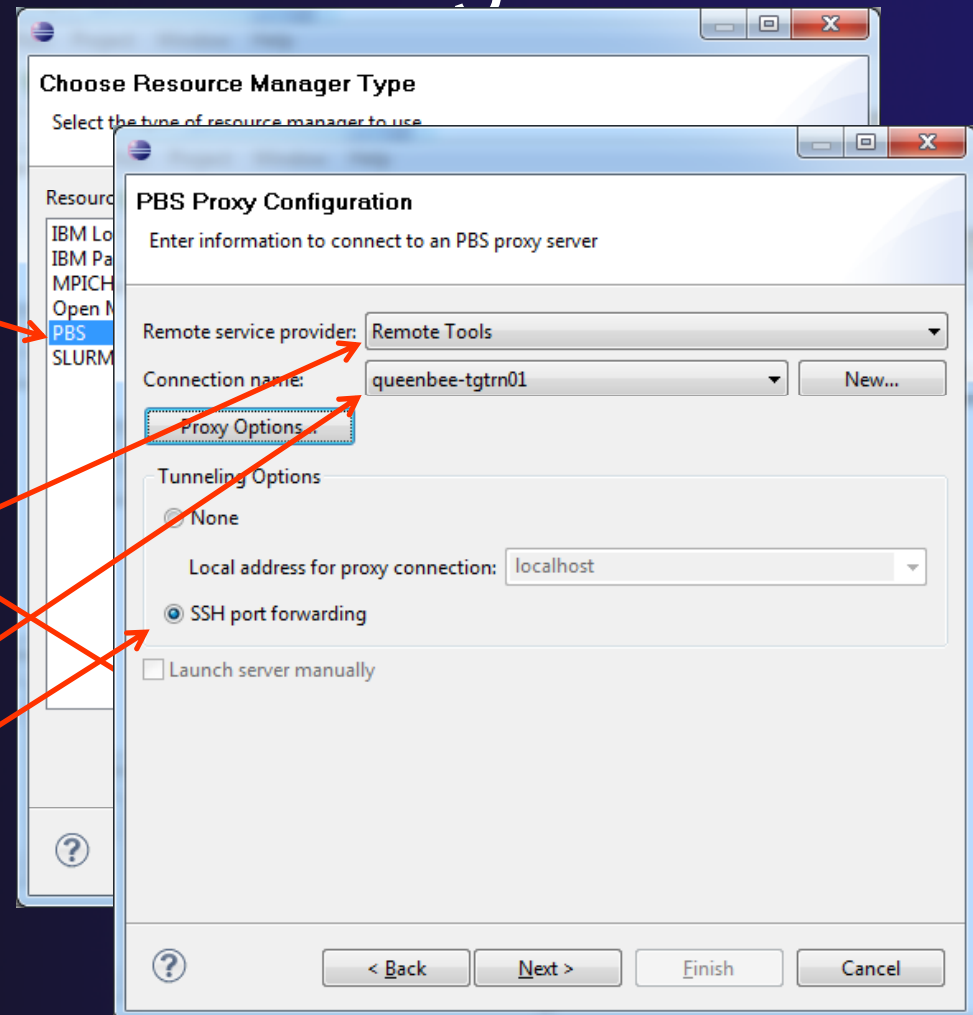
- ★ Select PBS Resource Manager... then **Next >**
- ★ Select **Remote Tools** Remote Service Provider
- ★ And appropriate connection name



# PBS Resource Manager



- ★ Select PBS Resource Manager... then **Next >**
- ★ Select **Remote Tools** Remote Service Provider
- ★ And appropriate connection name
- ★ And enable SSH port forwarding



# PBS Resource Manager



- ★ Leave the PBS Batch Script Configuration as default
  - ★ Provides an opportunity to customize resource manager
- ★ Select **Next**

**PBS Batch Script Configuration**  
Enter information to configure PBS Batch Script Templates

Default Template: default\_template

Edit Template Delete Template

Attribute Placeholders

Name	Default Value	Tool Tip
Account_Name		Format: string
Error_Path		Format: "[hostname]:p
Job_Name		Format: string up to 15
Output_Path		Format: "[hostname]:p
Resource_List.nodes		The value is one or mo
Resource_List.walltime	00:30:00	Format: [[hours:]minu
destination		Format: queue[el@serve

< Back Next > Finish Cancel

# PBS Resource Manager



- ★ Leave the PBS Batch Script Configuration as default
  - ★ Provides an opportunity to customize resource manager
- ★ Select **Next**
- ★ Only start resource manager manually
- ★ And select **Next**

**PBS Batch Script Configuration**  
Enter information to configure PBS Batch Script Templates

**Common Resource Manager Configuration**  
Change any settings for the resource manager

**Name and description**  
 Use default name and description:  
Name: PBS@queenbee-tgtrn01  
Description: PBS Resource Manager

**Startup**  
 Automatically start resource manager when Eclipse starts

? < Back **Next >** Finish Cancel

# PBS Resource Manager



- ★ This time, create a new service configuration
- ★ And select **Finish**

Select Service Configuration

Select a service configuration for this resource manager, or create a new one.

Create a new Service Configuration

Configuration Name:

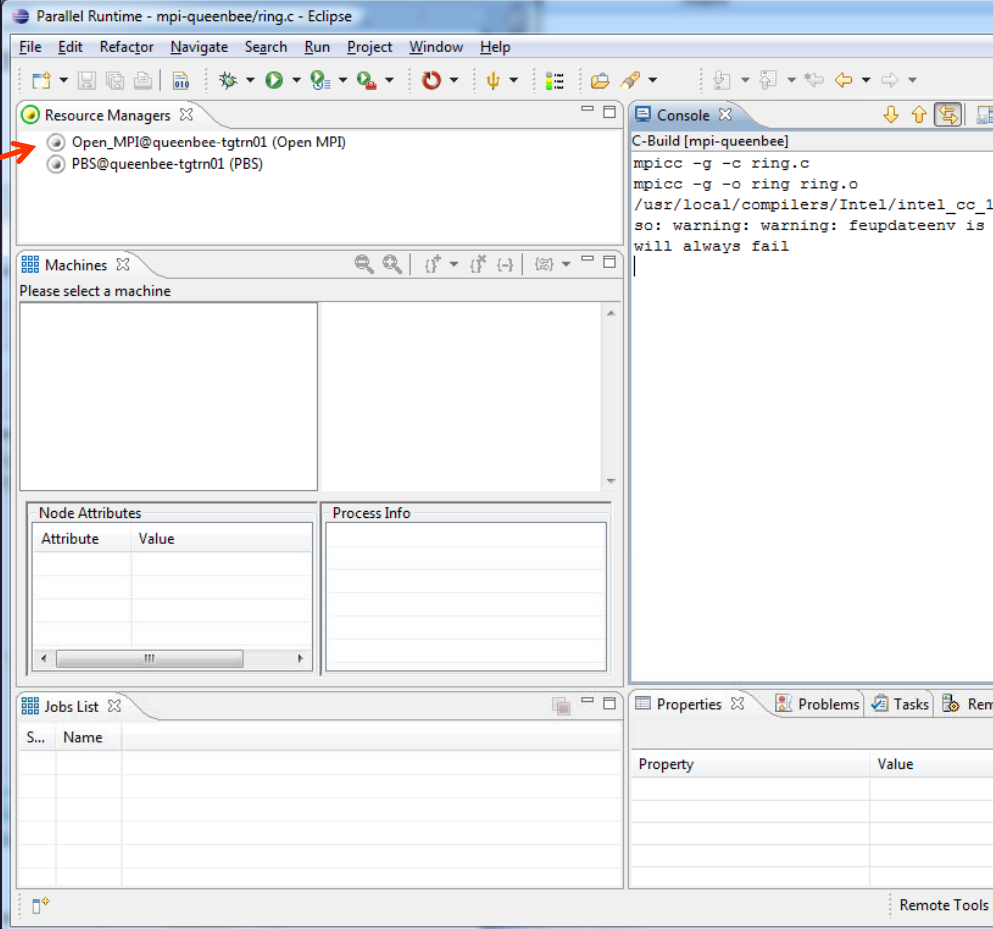
Add to an existing Service Configuration

Configuration Name	Launch Service
bigred-tg-tra01	Not Configured
queenbee-tgtrn01	Open_MPI@queenbee-tgtrn01

Advanced Settings...

# Resource managers

★ Voila! We now have two resource managers



The screenshot shows the Eclipse IDE interface for the Parallel Runtime. The 'Resource Managers' view is active, displaying two configured managers: 'Open\_MPI@queenbee-tgtrn01 (Open MPI)' and 'PBS@queenbee-tgtrn01 (PBS)'. An orange arrow points from the text 'two resource managers' to the first manager. Below this, the 'Machines' view is empty, with a 'Please select a machine' prompt. The 'Node Attributes' and 'Process Info' tables are also empty. The 'Console' view shows the output of a build command: 'mpicc -g -c ring.c', 'mpicc -g -o ring ring.o', and a warning: 'so: warning: warning: feupdateenv is will always fail'. The 'Jobs List' view is empty. The 'Properties' view is also empty.

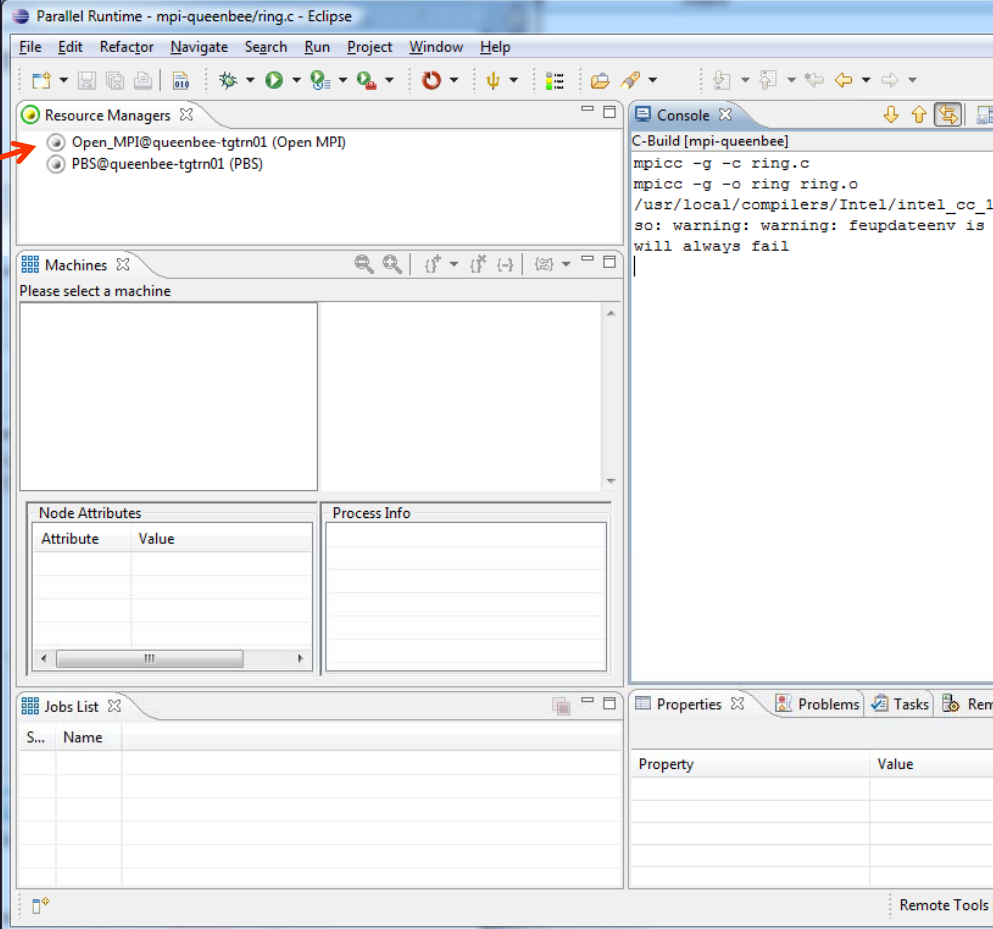
Attribute	Value

Property	Value



# Resource managers

★ Voila! We now have two resource managers

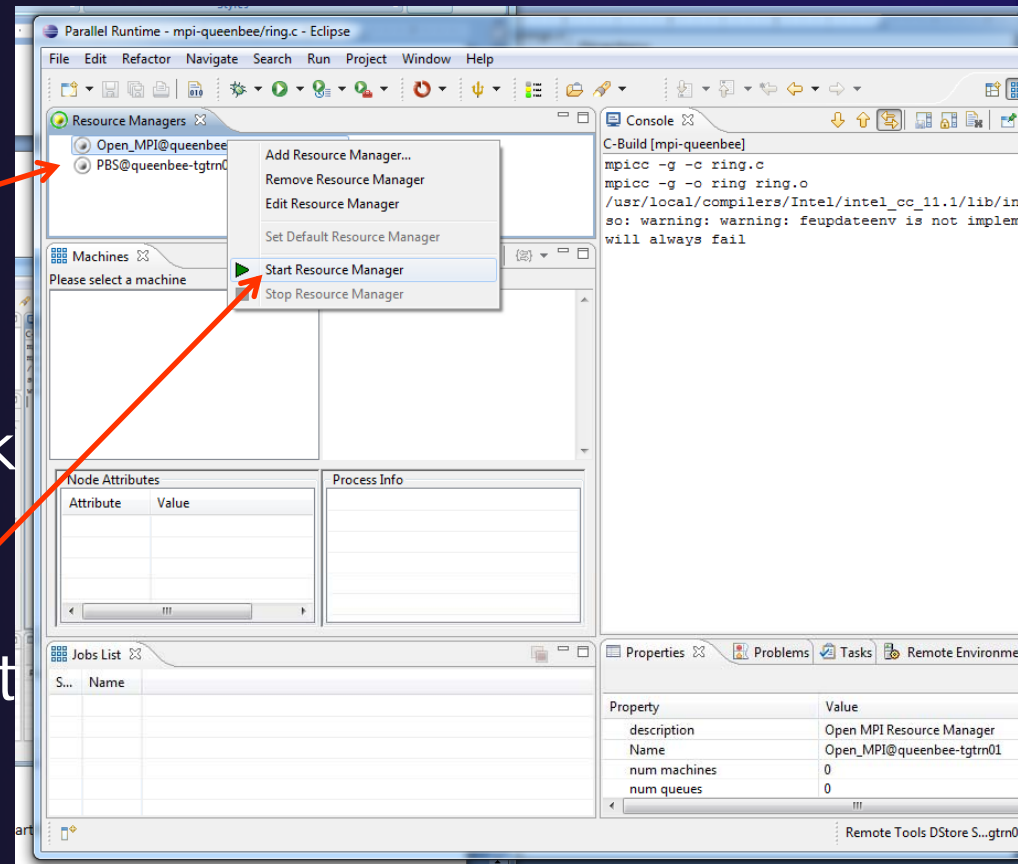


The screenshot shows the Eclipse IDE interface for the Parallel Runtime. The 'Resource Managers' panel is active, displaying two options: 'Open\_MPI@queenbee-tgtrn01 (Open MPI)' and 'PBS@queenbee-tgtrn01 (PBS)'. An orange arrow points from the text 'two resource managers' to the 'Open MPI' option. The 'Machines' panel below it shows a 'Please select a machine' dialog. The 'Console' panel on the right displays the output of a compilation command: 

```
C-Build [mpi-queenbee]  
mpicc -g -c ring.c  
mpicc -g -o ring ring.o  
/usr/local/compilers/Intel/intel_cc_1  
so: warning: warning: feupdateenv is  
will always fail
```

# Resource managers

- ✦ Voila! We now have two resource managers
- ✦ To start a resource manager – right click on the resource manager (OpenMPI this time), and select **Start Resource Manager**



# OpenMPI Resource Manager

- ★ Note that Resource Manager is now running (green icon)

The screenshot shows the Eclipse IDE interface with the 'Parallel Runtime - mpi-queenbee/ring.c - Eclipse' window. The 'Resource Managers' view is expanded to show 'Open\_MPI@queenbee-tgtrn01 (Open MPI)' with a green icon, indicating it is running. Below it, the 'Machines' view shows 'Open\_MPI@queenbee-tgtrn01: queenbee-tgtrn01 - Root [1]' with a green icon and the number '0'. The 'Node Attributes' and 'Process Info' tables are empty. The 'Jobs List' view is also empty. The 'Console' view shows the output of the compilation process:

```
C-Build [mpi-queenbee]
mpicc -g -c ring.c
mpicc -g -o ring ring.o
/usr/local/compilers/Intel/intel_cc_
so: warning: feupdateenv is
will always fail
```

The 'Properties' view at the bottom right shows the following table:

Property	Value
description	Open MPI Resou
Name	Open_MPI@que
num machines	0
num queues	0

# OpenMPI Resource Manager

- ★ Note that Resource Manager is now running (green icon)
- ★ Also, queenbee as a system is denoted as up and running

The screenshot displays the Eclipse IDE interface for the OpenMPI Resource Manager. The 'Resource Managers' view shows a tree structure with 'Open\_MPI@queenbee-tgtrn01 (Open MPI)' selected, indicated by a green icon. Below it, the 'Machines' view shows 'Open\_MPI@queenbee-tgtrn01: queenbee-tgtrn01 - Root [1]' with a green icon next to 'queenbee-tgtrn01'. The 'Jobs List' view is empty. The 'Properties' view shows the following details:

Property	Value
description	Open MPI Resou
Name	Open_MPI@que
num machines	0
num queues	0

The console window shows the following output:

```
C-Build [mpi-queenbee]
mpicc -g -c ring.c
mpicc -g -o ring ring.o
/usr/local/compilers/Intel/intel_cc_
so: warning: feupdateenv is
will always fail
```

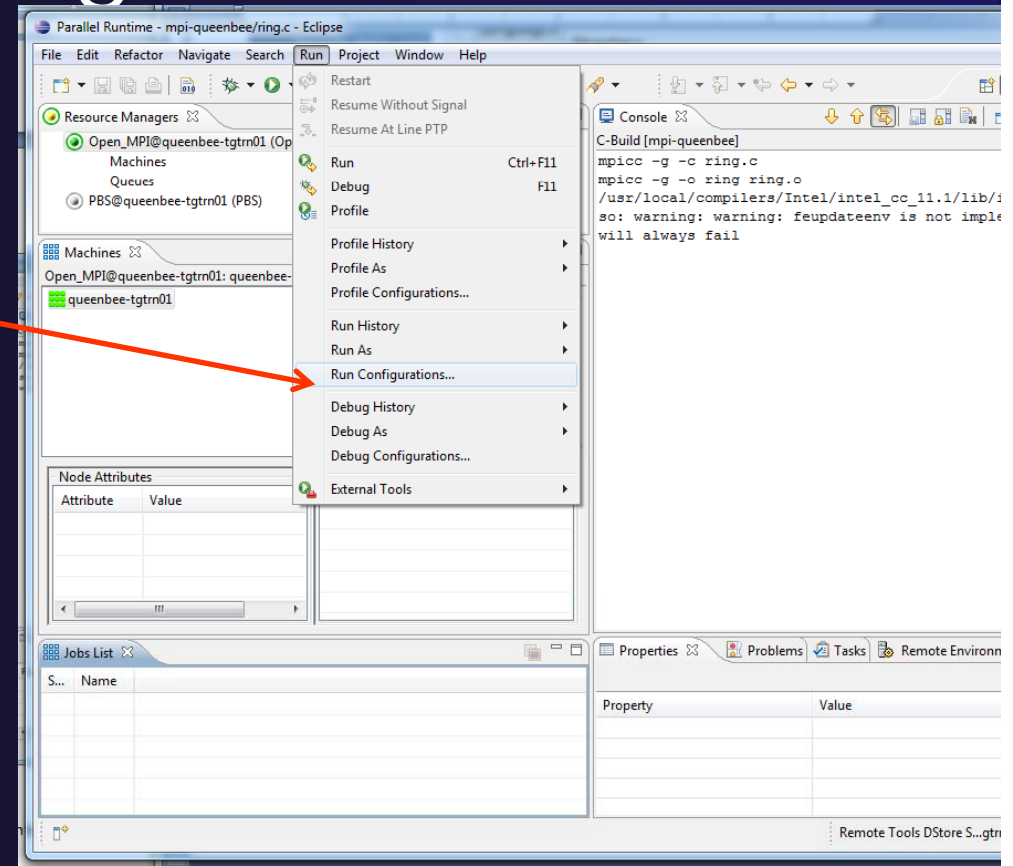
# OpenMPI Resource Manager

- ★ Note that Resource Manager is now running (green icon)
- ★ Also, queenbee as a system is denoted as up and running
- ★ And we have one head node available for interactive use

The screenshot displays the Eclipse IDE interface for the OpenMPI Resource Manager. The 'Resource Managers' view shows a tree structure with 'Open\_MPI@queenbee-tgtrn01 (Open MPI)' selected, which has a green status icon. Below it, the 'Machines' view shows 'Open\_MPI@queenbee-tgtrn01: queenbee-tgtrn01 - Root [1]' with a green status icon. The 'Jobs List' view is empty. The 'Console' view shows the output of a compilation command: 'mpicc -g -c ring.c', 'mpicc -g -o ring ring.o', and a warning: 'warning: feupdateenv is will always fail'. The 'Properties' view shows the configuration for the resource manager: description: Open MPI Resou, Name: Open\_MPI@que, num machines: 0, num queues: 0.

# Run Configuration

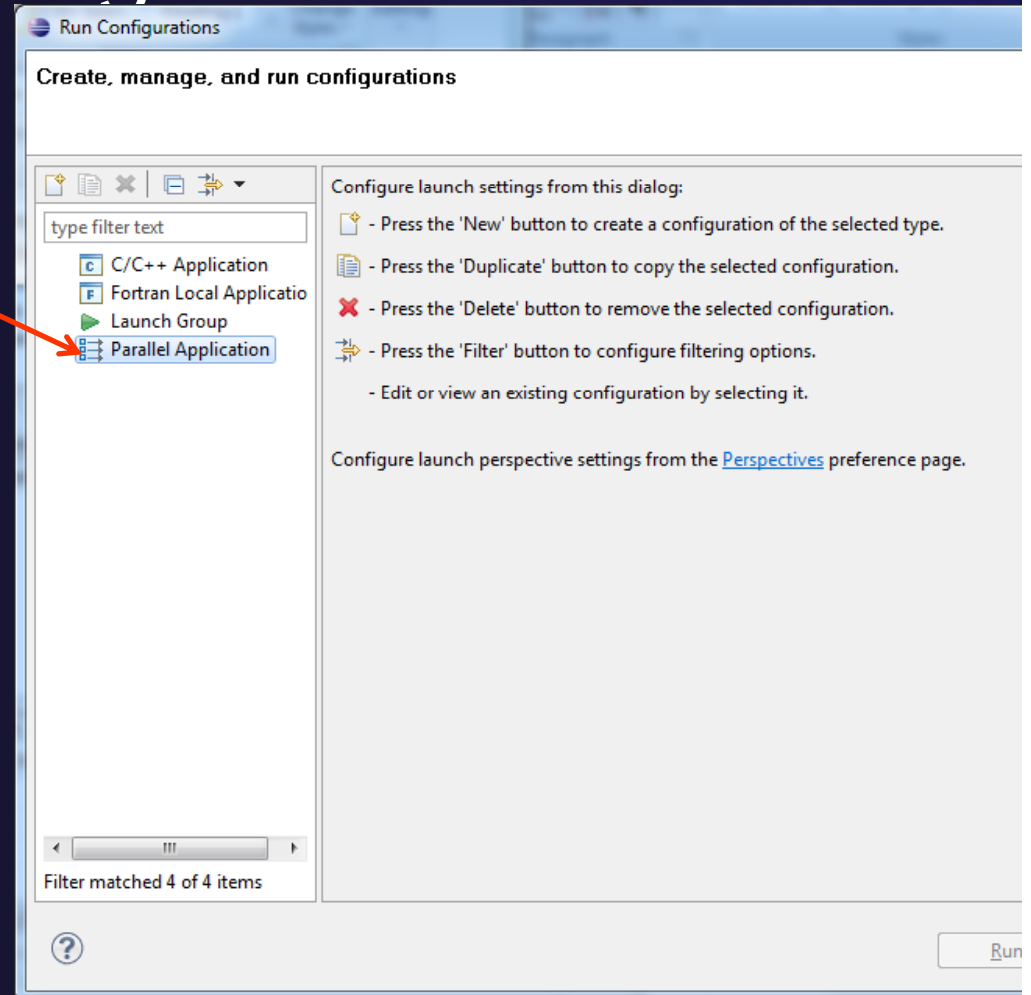
- ★ To run our code, select **Run > Run Configurations**



# Run Configuration



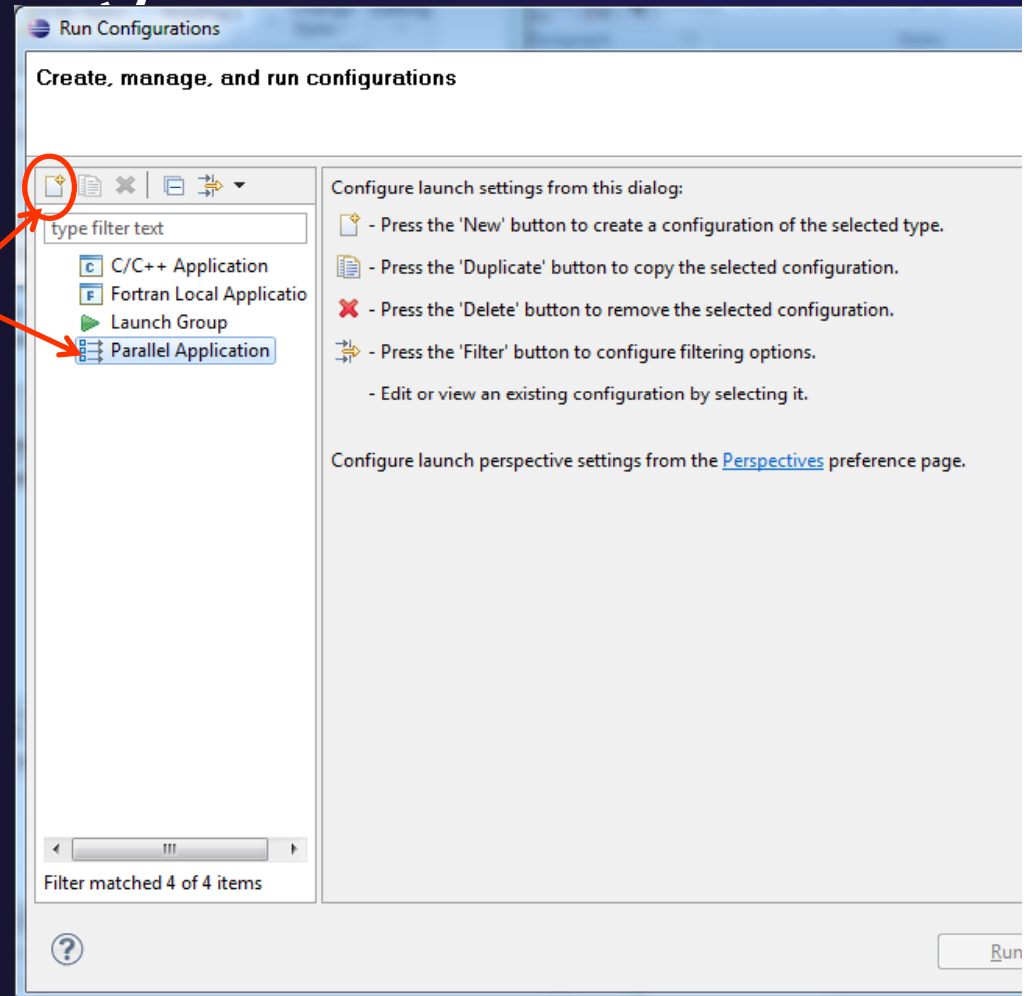
- ★ Sneaky detail –
  - ★ Select **Parallel Application**, then



# Run Configuration



- ★ Sneaky detail –
  - ★ Select **Parallel Application**, then
  - ★ Select the **New** button

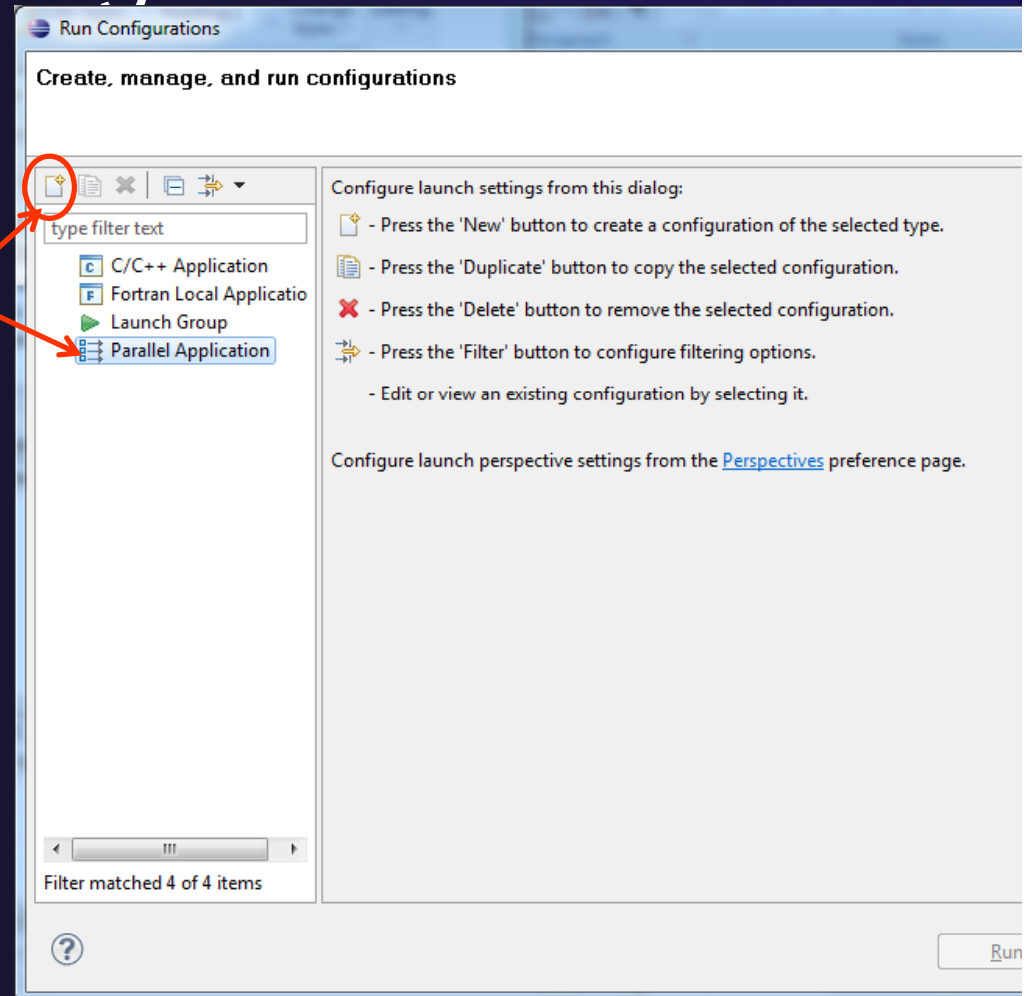




# Run Configuration



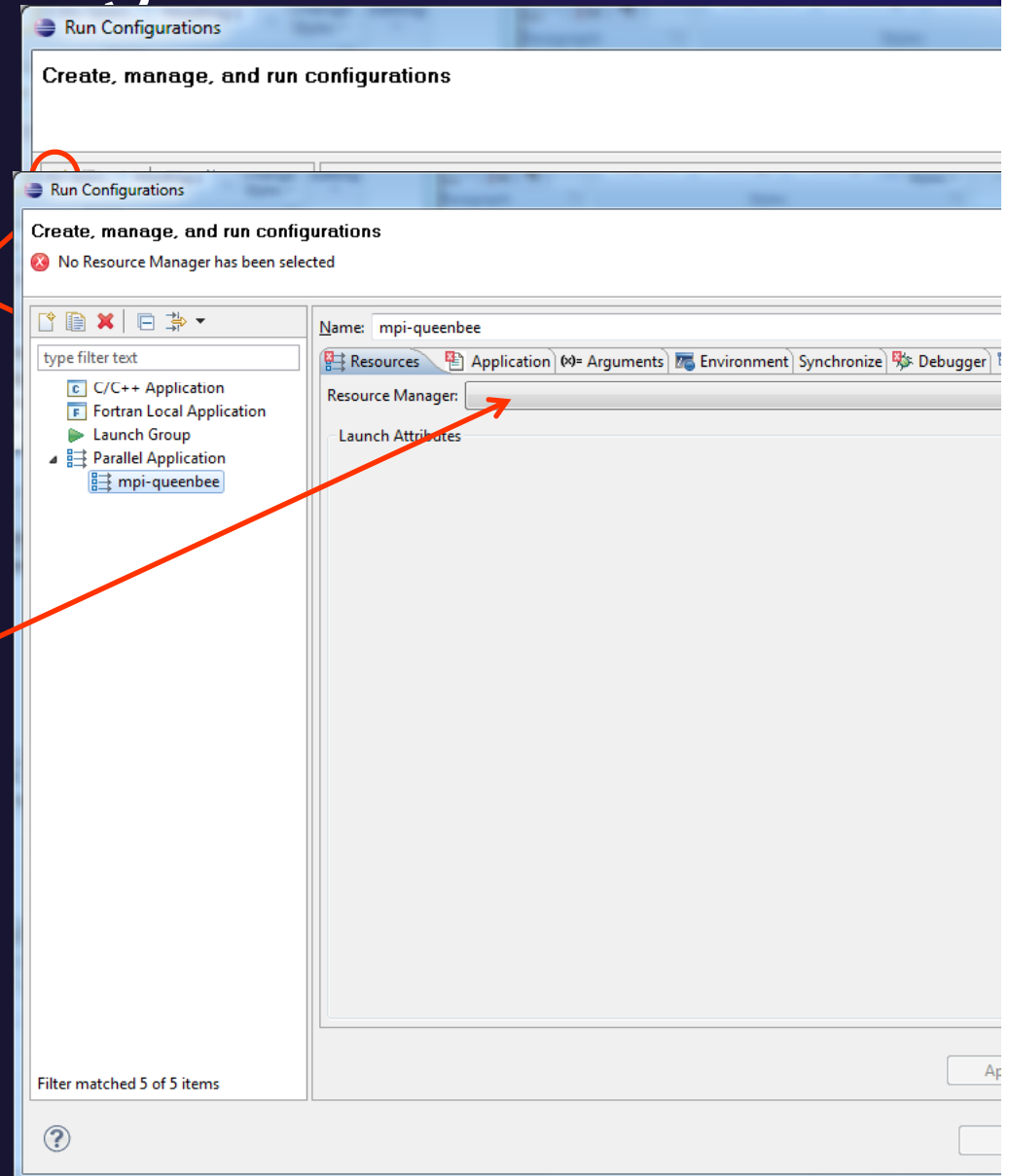
- ★ Sneaky detail –
  - ★ Select **Parallel Application**, then
  - ★ Select the **New** button



# Run Configuration

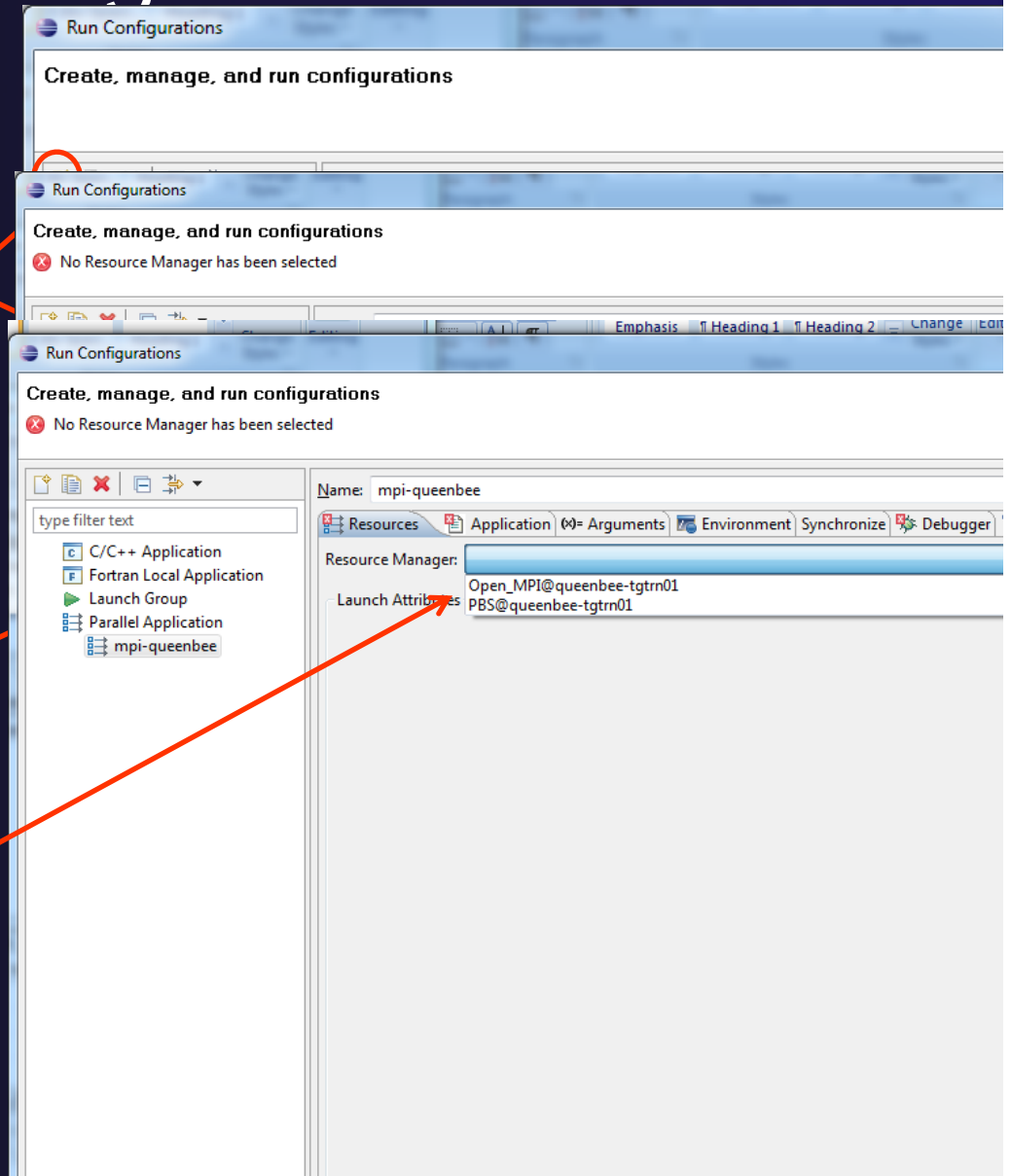


- ★ Sneaky detail –
  - ★ Select **Parallel Application**, then
  - ★ Select the **New** button
- ★ Now need to setup the Resource Manager (pull down resource manager menu)



# Run Configuration

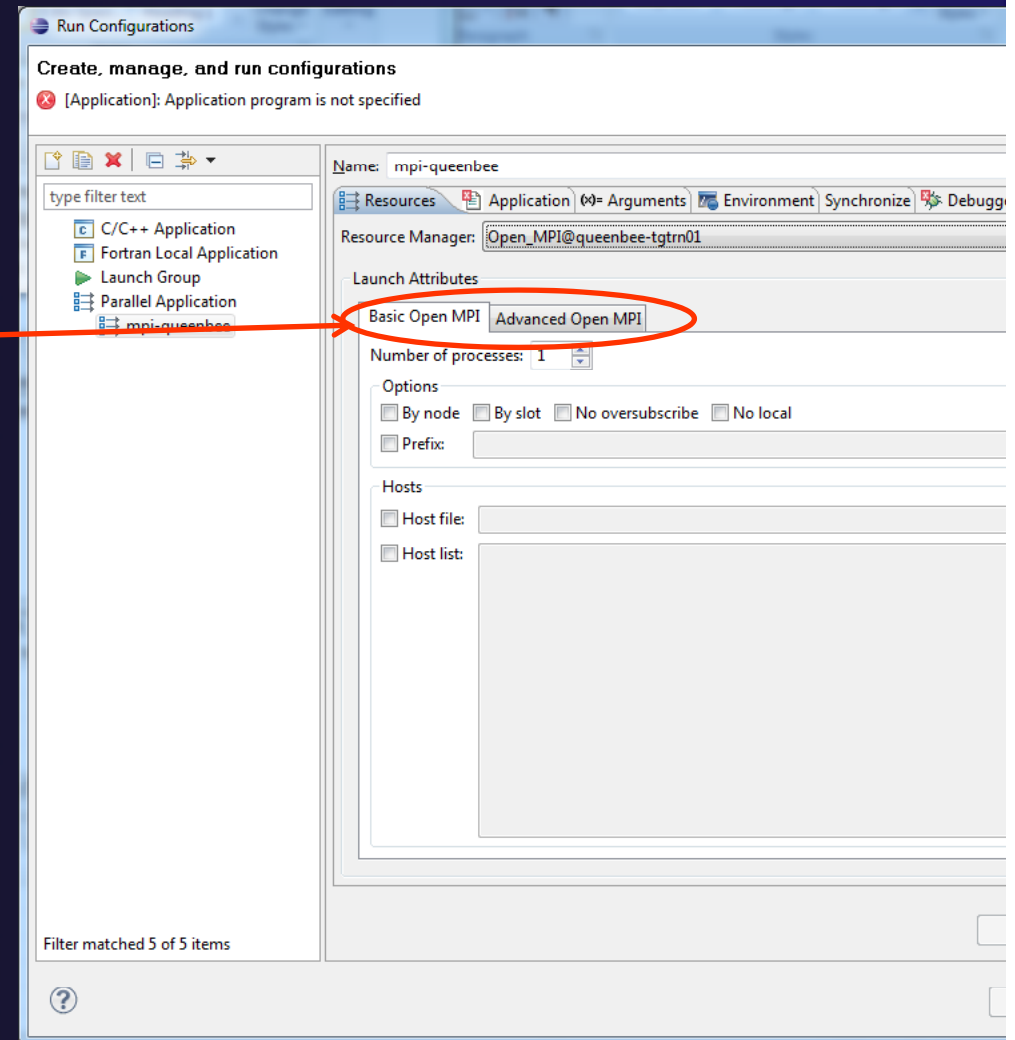
- ★ Sneaky detail –
  - ★ Select **Parallel Application**, then
  - ★ Select the **New** button
- ★ Now need to setup the Resource Manager (pull down resource manager menu)
  - ★ Choose OpenMPI



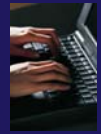
# OpenMPI Configuration



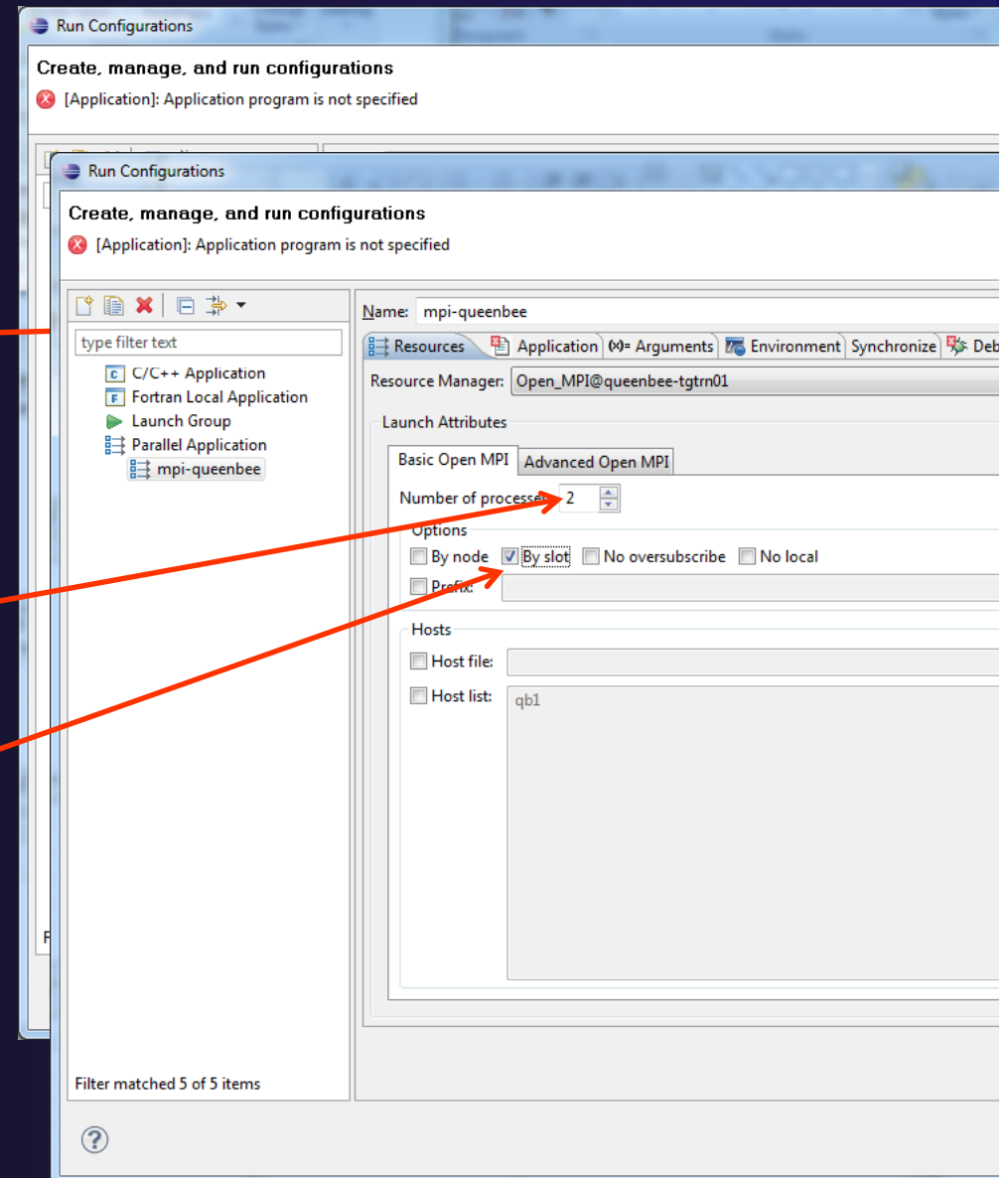
- ★ Note that the OpenMPI has both basic and advanced attributes...
- ★ We'll stick with basic for today



# OpenMPI Configuration



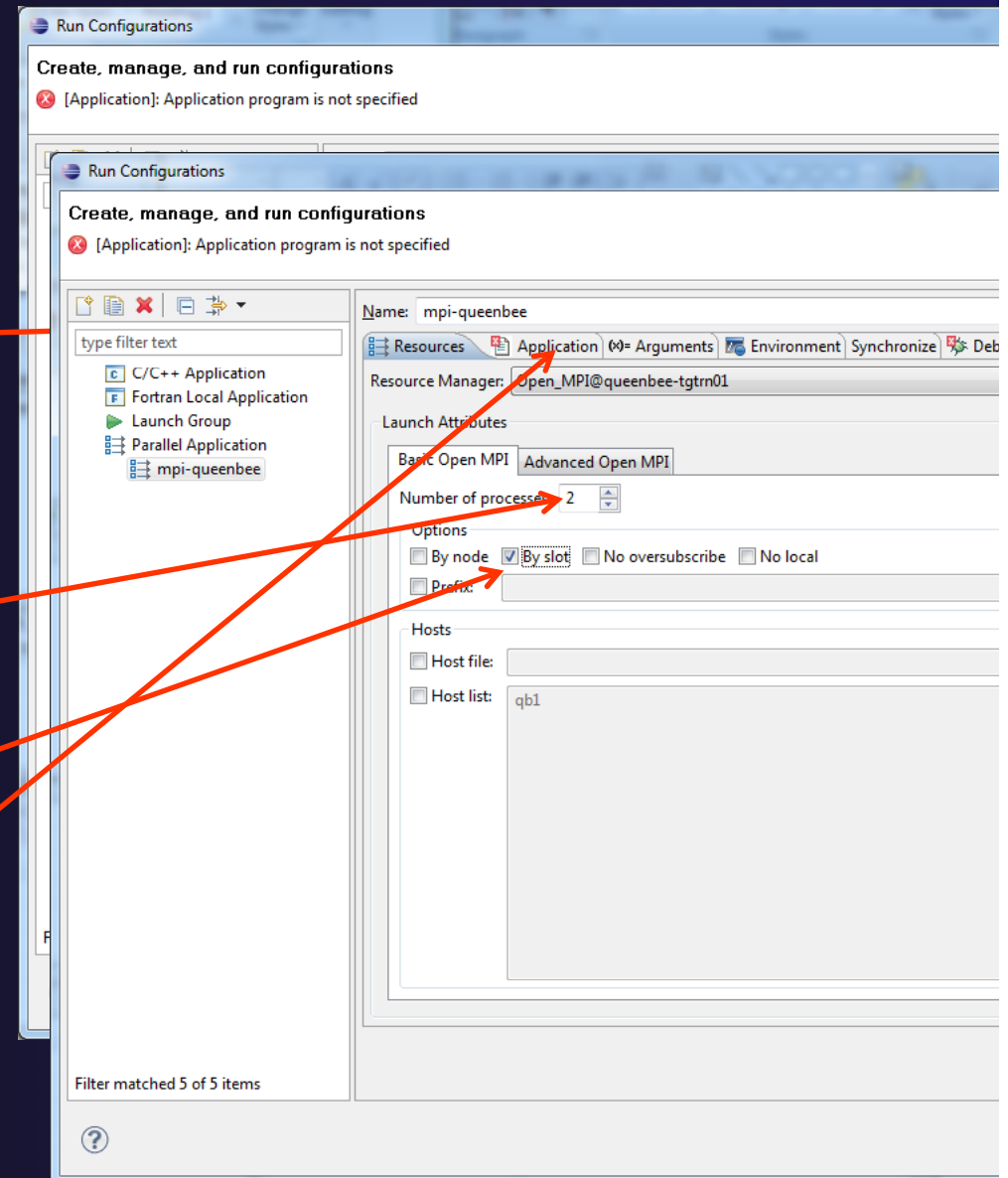
- ★ Note that the OpenMPI has both basic and advanced attributes...
  - ★ We'll stick with basic for today
- ★ Configure 2 processes
  - ★ And allocate "by slot"



# OpenMPI Configuration



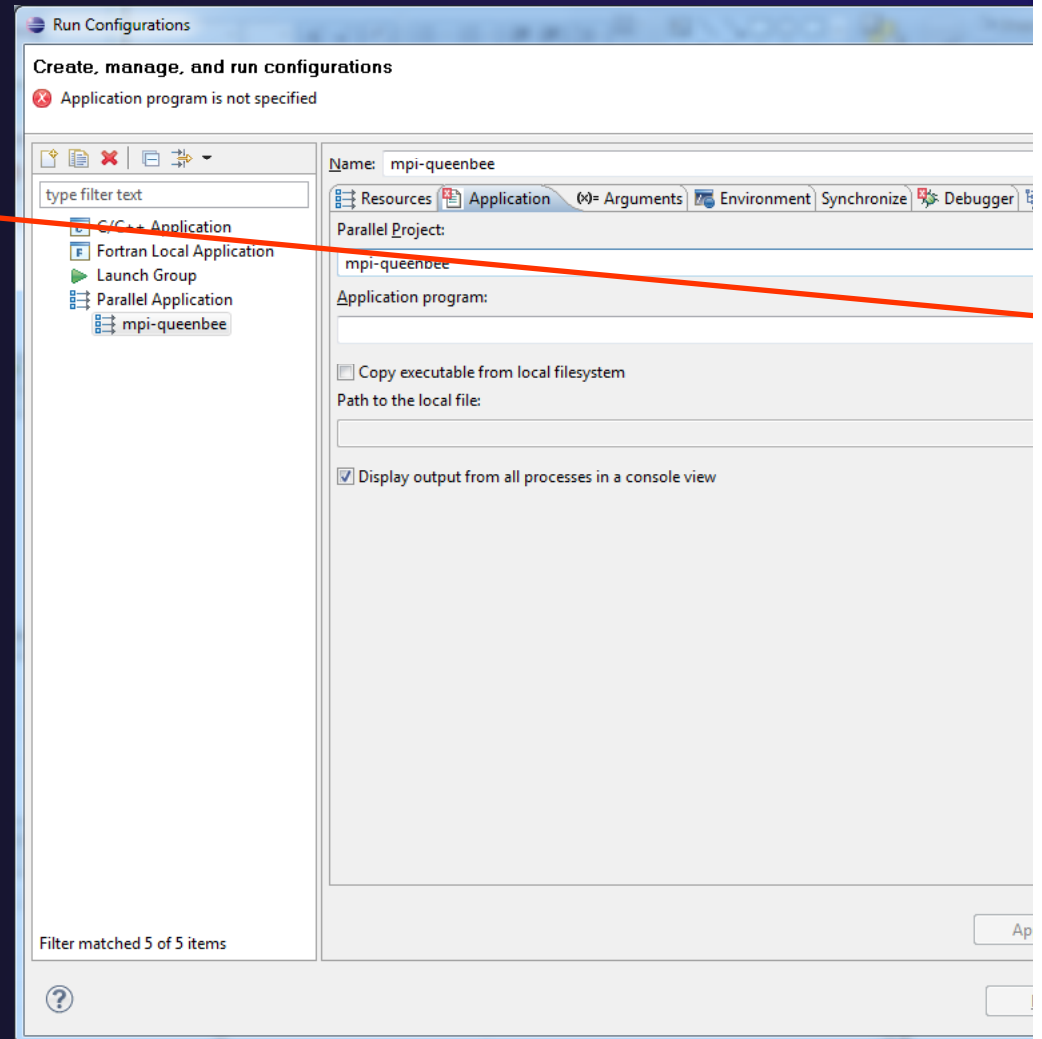
- ★ Note that the OpenMPI has both basic and advanced attributes...
  - ★ We'll stick with basic for today
- ★ Configure 2 processes
  - ★ And allocate "by slot"
- ★ Select the **Application** tab



# OpenMPI Configuration



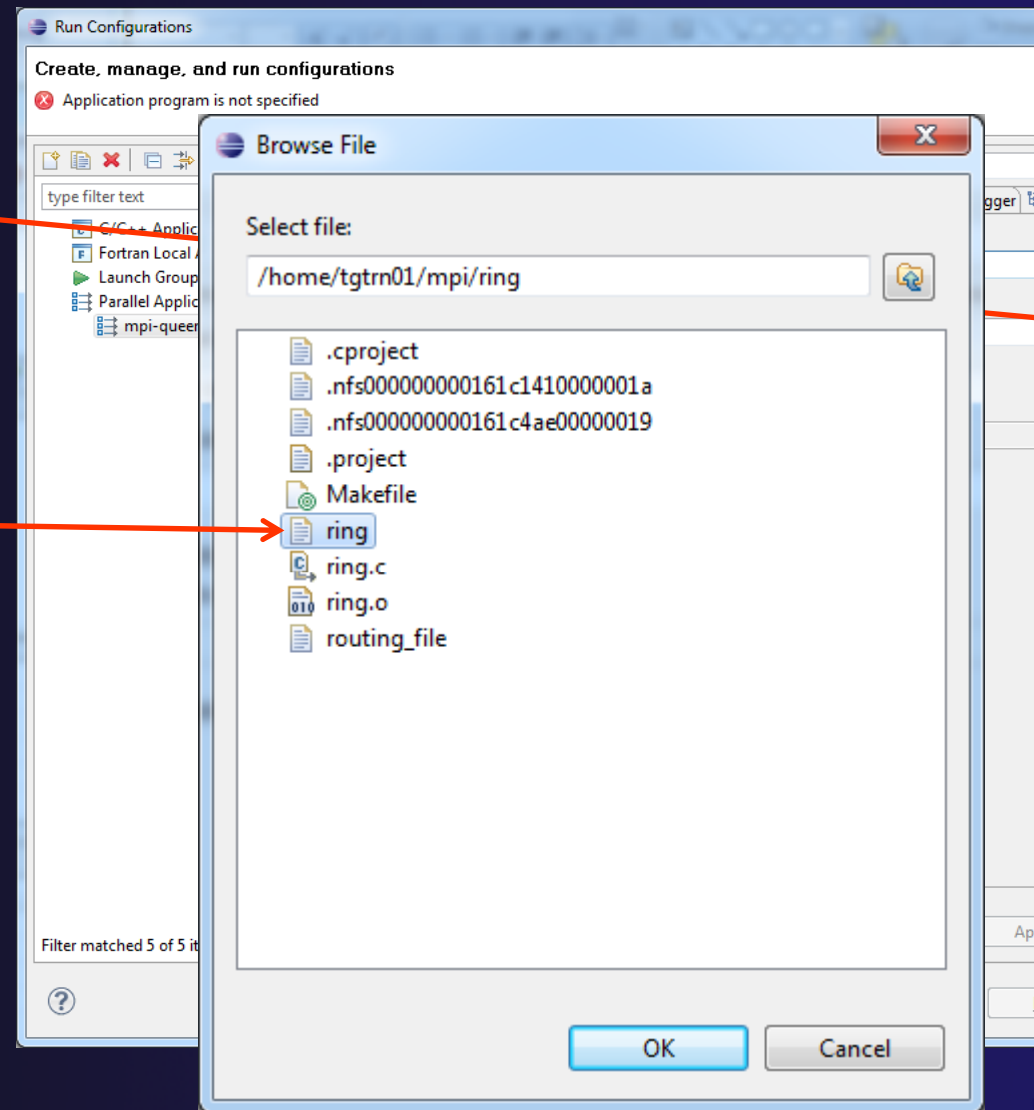
★ Browse for the application code



# OpenMPI Configuration



- ★ **Browse** for the application code
- ★ Select the ring executable we built previously, then press **OK**

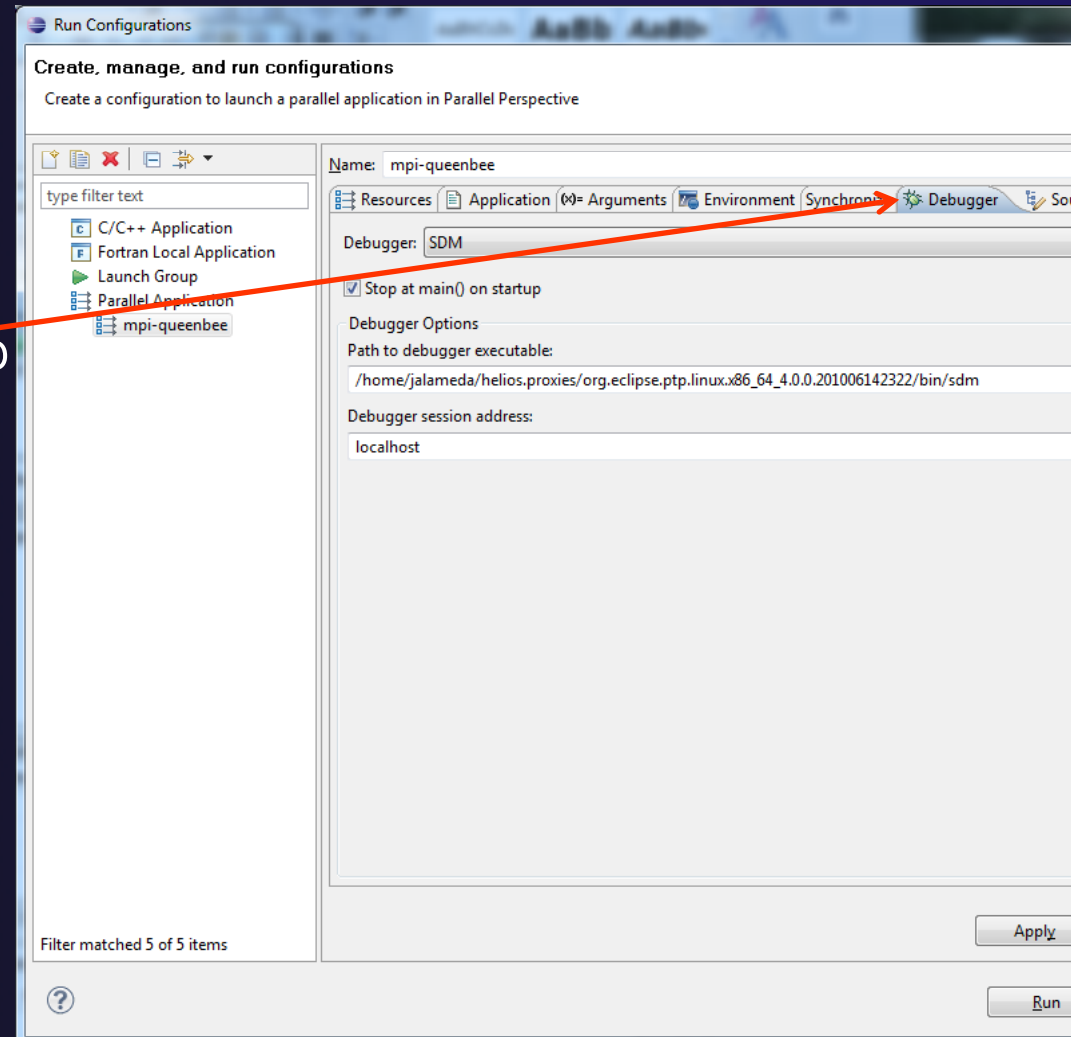




# OpenMPI Configuration



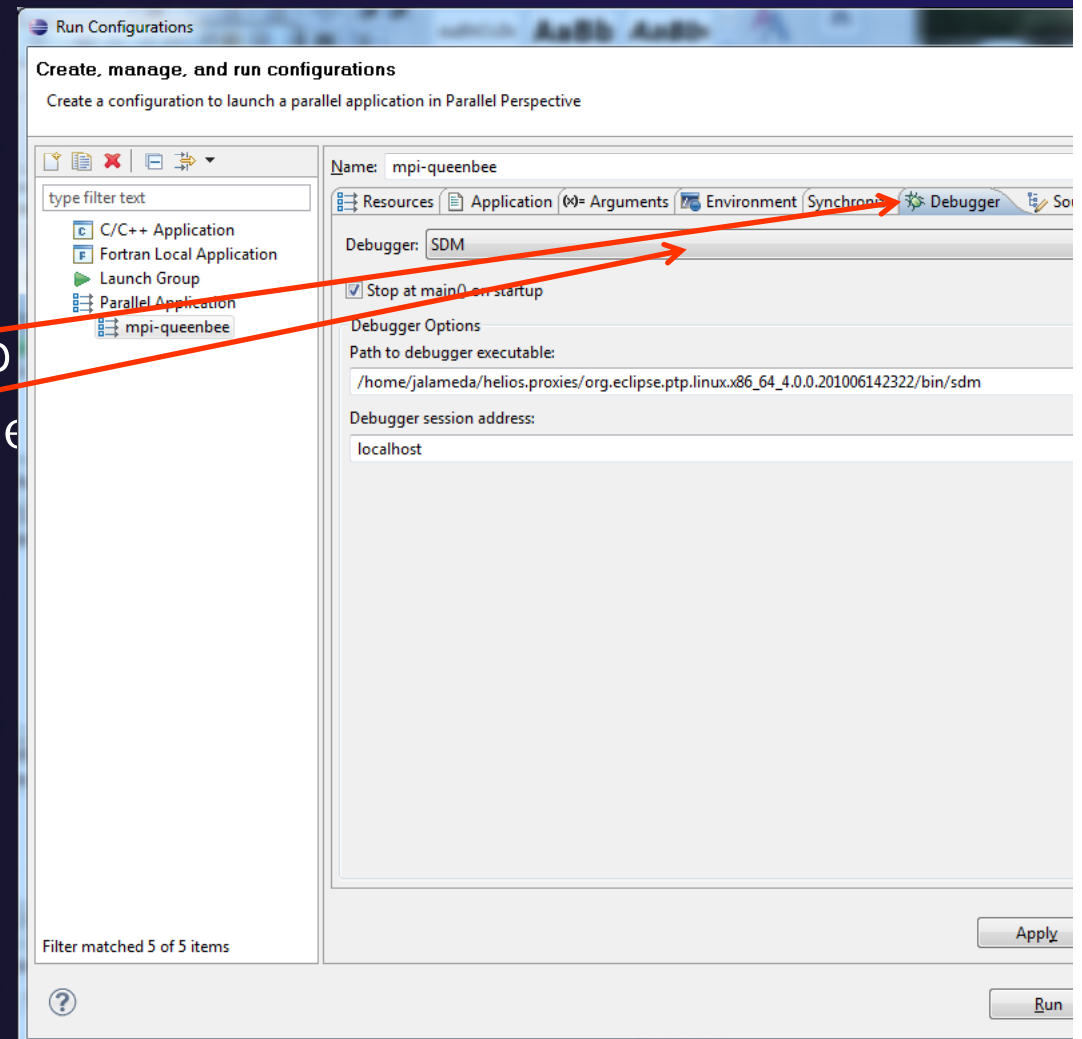
- ★ We'll need to configure the debugger as well
- ★ Select debugger tab



# OpenMPI Configuration



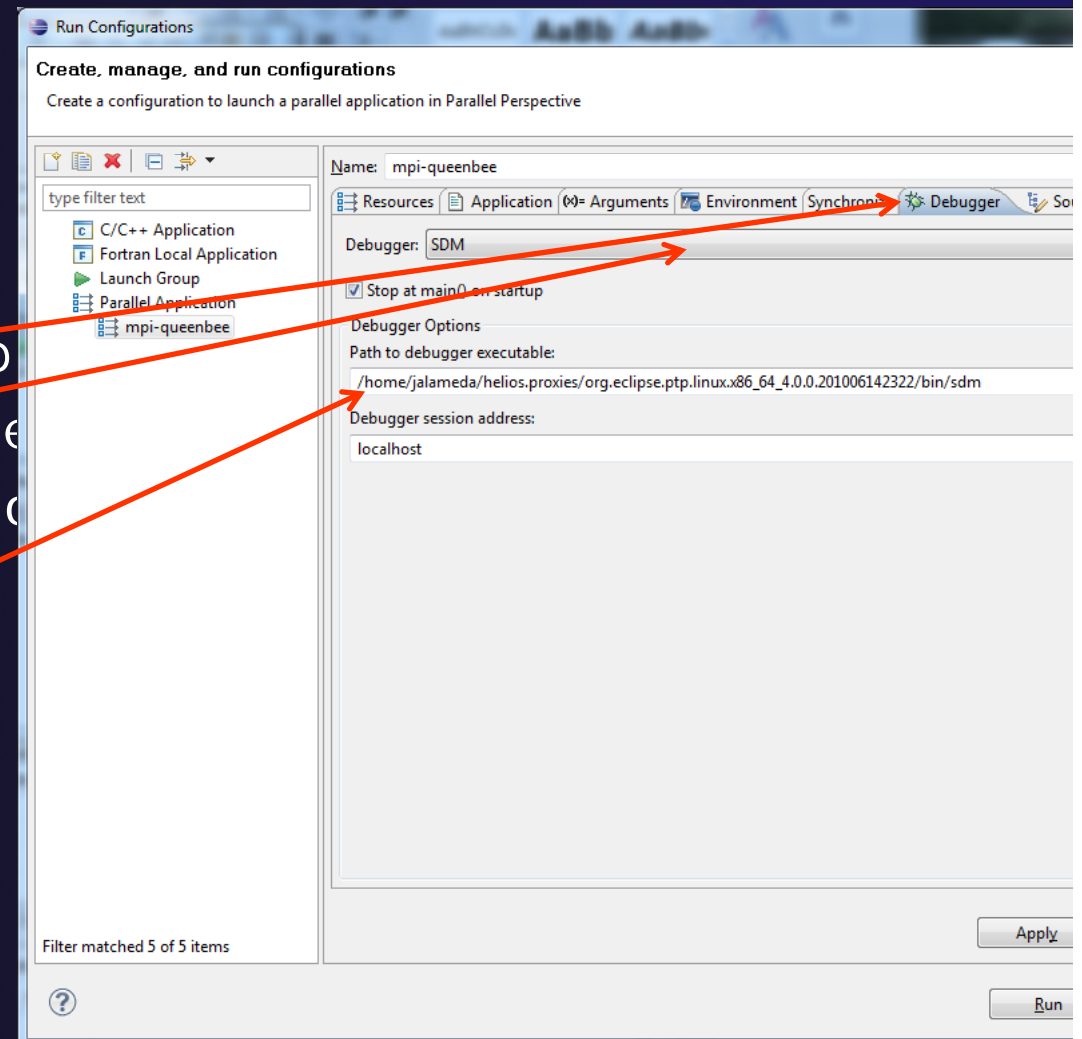
- ★ We'll need to configure the debugger as well
  - ★ Select debugger tab
  - ★ Select **SDM** debugger



# OpenMPI Configuration



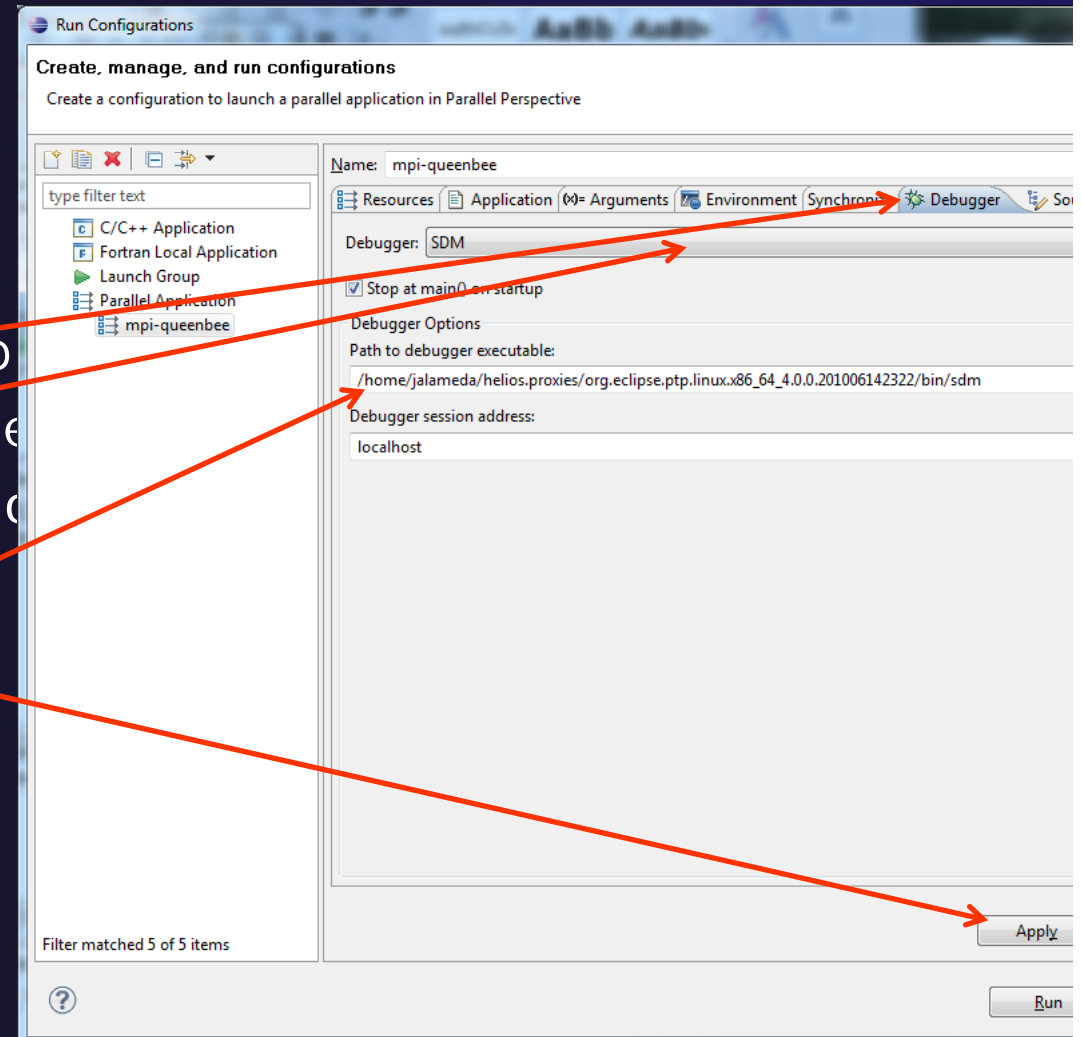
- ★ We'll need to configure the debugger as well
  - ★ Select debugger tab
  - ★ Select **SDM** debugger
  - ★ Browse to the location of the executables



# OpenMPI Configuration



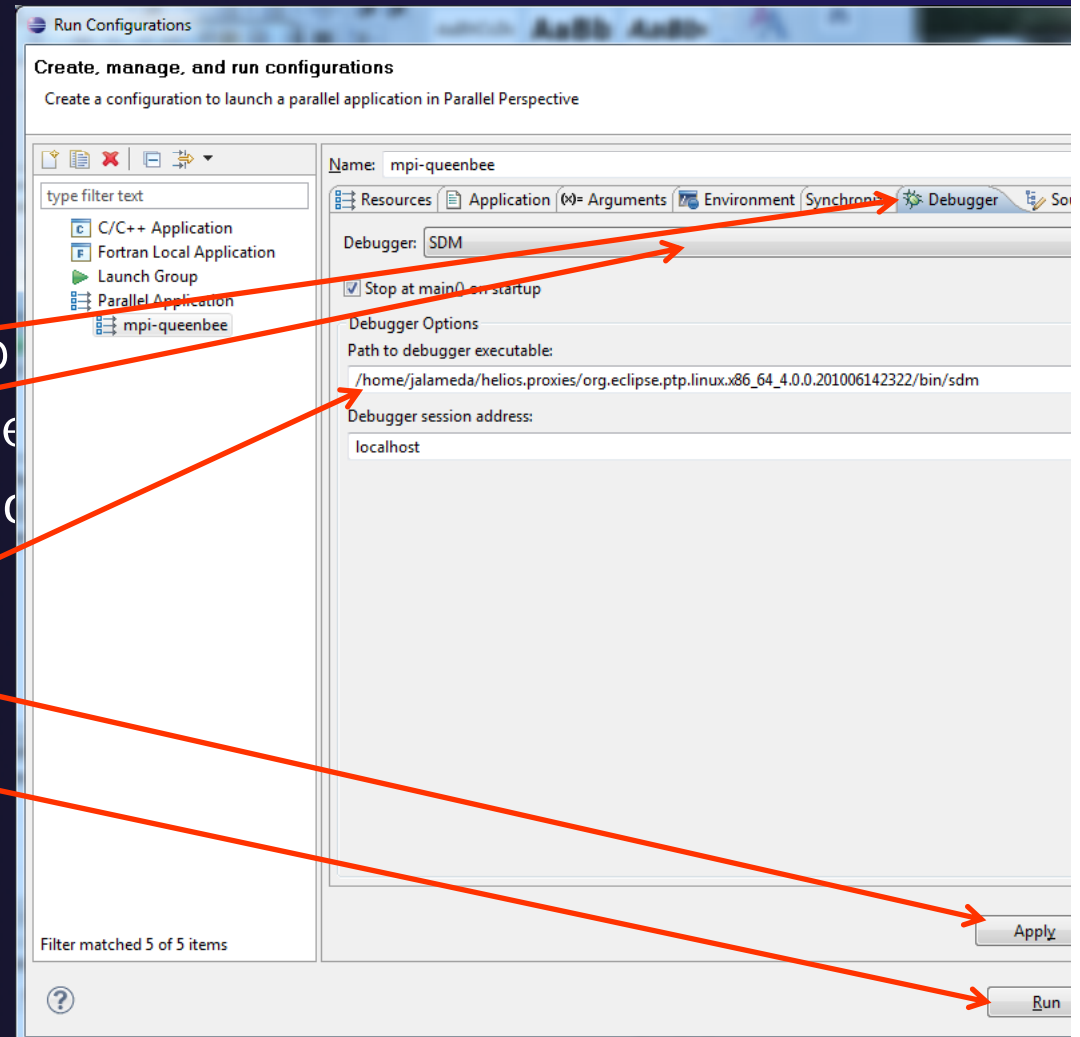
- ★ We'll need to configure the debugger as well
  - ★ Select debugger tab
  - ★ Select **SDM** debugger
  - ★ Browse to the location of the executables
- ★ Press **Apply**



# OpenMPI Configuration



- ★ We'll need to configure the debugger as well
  - ★ Select debugger tab
  - ★ Select **SDM** debugger
  - ★ Browse to the location of the executables
- ★ Press **Apply**, then press **Run**



# Interactive code run



- ★ Note successful completion (refer to icons previously)

The screenshot displays the Eclipse IDE with the Parallel Runtime plugin. The main window shows the 'Parallel Runtime - mpi-queenbee/ring.c - Eclipse' interface. The 'Resource Managers' panel shows two managers: 'Open\_MPI@queenbee-tgtrn01 (Open MPI)' and 'PBS@queenbee-tgtrn01 (PBS)'. The 'Machines' panel shows a tree view with 'Open\_MPI@queenbee-tgtrn01: queenbee-tgtrn01 - Root [1]' and 'queenbee-tgtrn01'. A green checkmark icon is visible next to 'queenbee-tgtrn01', indicating successful completion. The 'Jobs List' panel shows a table with one job:

Name	Status	Executable Name	Executable Path	User	Argume
job0	NORMAL	ring	/home/tgtrn01/mpi		[]

The 'Properties' panel shows the following details for the selected job:

Property	Value
description	Open MPI Resource Man
Name	Open_MPI@queenbee-t
num machines	0
num queues	0

## Interactive code run



- ★ Note successful completion (refer to icons previously)
- ★ Code output to console
- ★ Click on node to get process information

The screenshot shows the Eclipse IDE with the Parallel Runtime plugin. The 'Resource Managers' view shows 'Open\_MPI@queenbee-tgtrn01 (Open MPI)' and 'PBS@queenbee-tgtrn01 (PBS)'. The 'Machines' view shows 'Open\_MPI@queenbee-tgtrn01: queenbee-tgtrn01 - Root [1]' and 'queenbee-tgtrn01'. The 'Jobs List' view shows a table with one job:

Name	Status	Executable Name	Executable Path	User	Argume
job0	NORMAL	ring	/home/tgtrn01/mpi		[]

The 'Console' view shows the output: 'Open\_MPI@queenbee-tgtrn01:default:job0 Master: end of trip 1 of 1: after receivi'. The 'Properties' view shows the following details:

Property	Value
description	Open MPI Resource Man
Name	Open_MPI@queenbee-t
num machines	0
num queues	0

# Process information

- ★ Note information on the 2 MPI ranks

The screenshot displays the Eclipse IDE's Parallel Runtime interface. The 'Machines' view shows a node 'queenbee-tgtrn01' with 0 MPI ranks. The 'Process Info' view shows two MPI ranks: 'job0:job0.0' and 'job0:job0.1'. The 'Jobs List' view shows a job 'job0' with status 'NORMAL' and executable 'ring'. The 'Properties' view shows properties for the job: Name 'qb1', Node Number '0', and Open MPI number of nodes '1'.

Attribute	Value
Name	qb1
Node Num...	0
Open MPI ...	1

Process Info
job0:job0.0
job0:job0.1

S...	Name	Status	Executable Name	Executable Path	User	Argu...
●	job0	NORMAL	ring	/home/tgtrn01/mpi		[]

Property	Value
Name	qb1
Node Number	0
Open MPI number of nodes	1



# Process information

- ★ Note information on the 2 MPI ranks

The screenshot displays the Eclipse IDE interface for the Parallel Runtime environment. The 'Resource Managers' view shows 'Open\_MPI@queenbee-tgtrn01 (Open MPI)' selected. The 'Machines' view shows the machine 'queenbee-tgtrn01' with a process ID of 0. The 'Node Attributes' table is as follows:

Attribute	Value
Name	qb1
Node Num...	0
Open MPI ...	1

The 'Process Info' table shows two MPI ranks:

Process ID
job0:job0.0
job0:job0.1

The 'Jobs List' view shows a table with the following data:

S...	Name	Status	Executable Name	Executable Path	User	Argu...
●	job0	NORMAL	ring	/home/tgtrn01/mpi		[]

The 'Console' view shows the output: 'Open\_MPI@queenbee-tgtrn01:default:job0 Master: end of trip 1 of 1: after rec...'. The 'Properties' view shows the following data:

Property	Value
Name	qb1
Node Number	0
Open MPI number of nodes	1

An orange arrow points from the text 'Note information on the 2 MPI ranks' to the 'Process Info' table.

# Process information

- ★ Note information on the 2 MPI ranks

The screenshot displays the Eclipse IDE's Parallel Runtime interface. The 'Machines' view shows a node 'queenbee-tgtrn01' with 0 MPI ranks. The 'Process Info' view shows two MPI ranks: 'job0:job0.0' and 'job0:job0.1'. The 'Jobs List' view shows a job 'job0' with status 'NORMAL' and executable 'ring'. The 'Properties' view shows properties for the job: Name 'qb1', Node Number '0', and Open MPI number of nodes '1'.

Attribute	Value
Name	qb1
Node Num...	0
Open MPI ...	1

S...	Name	Status	Executable Name	Executable Path	User	Argu...
●	job0	NORMAL	ring	/home/tgtrn01/mpi		[]

Property	Value
Name	qb1
Node Number	0
Open MPI number of nodes	1

# Process information

- ★ Note information on the 2 MPI ranks
  - ★ Rank 1
  - ★ Rank 1 output

The screenshot displays the Eclipse IDE interface for the Parallel Runtime environment. The main window shows the 'job0:job0.1' job details, including the process name, PID (N/A), and status (COMPLETED). The 'Program output' section is visible but empty. The 'Resource Managers' panel shows 'Open\_MPI@queenbee-tgtrn01 (Open MPI)' and 'PBS@queenbee-tgtrn01 (PBS)'. The 'Machines' panel shows 'Open\_MPI@queenbee-tgtrn01: queenbee-tgtrn01 - Root [1]' and 'queenbee-tgtrn01'. The 'Node Attributes' panel shows a table with columns 'Attribute' and 'Value':

Attribute	Value
Name	qb1
Node Num...	0
Open MPI ...	1

The 'Process Info' panel shows a table with columns 'Name' and 'Value':

Name	Value
job0:job0.0	
job0:job0.1	

The 'Jobs List' panel shows a table with columns 'S...', 'Name', 'Status', 'Executable Name', 'Executable Path', 'User', and 'Argume':

S...	Name	Status	Executable Name	Executable Path	User	Argume
●	job0	NORMAL	ring	/home/tgtrn01/mpi		{}

The 'Properties' panel shows a table with columns 'Property' and 'Value':

Property	Value
Name	qb1
Node Number	0
Open MPI number of nodes	1

At the bottom right, the status bar shows 'COMPLETED' and 'PID: N/A'.

# Process information

- ★ Note information on the 2 MPI ranks
  - ★ Rank 1
  - ★ Rank 1 output
  - ★ Rank 0
  - ★ Rank 0 output

The screenshot shows the Eclipse IDE interface for the Parallel Runtime platform. The main window displays the following views:

- Resource Managers:** Shows 'Open\_MPI@queenbee-tgtrn01 (Open MPI)' and 'PBS@queenbee-tgtrn01 (PBS)'.
- Machines:** Shows 'Open\_MPI@queenbee-tgtrn01: queenbee-tgtrn01 - Root [1]' and a table with one entry: 'queenbee-tgtrn01' with a value of '0'.
- Node Attributes:** A table with the following data:
 

Attribute	Value
Name	qb1
Node Num...	0
Open MPI ...	1
- Process Info:** A table with the following data:
 

Process
job0:job0.0
job0:job0.1
- Jobs List:** A table with the following data:
 

S...	Name	Status	Executable Name	Executable Path	User	Argume
●	job0	NORMAL	ring	/home/tgtrn01/mpi		[]
- Process details (right panel):** Shows 'job0:job0.1' with 'PID: N/A' and 'Status: COMPLETED'. The 'Program output' section contains the text: 'Master: end of trip 1 of 1: after receiving passed\_num=2 (should be =trip\*numprocs=2) from source=1'.
- Properties (bottom right):** A table with the following data:
 

Property	Value
Name	qb1
Node Number	0
Open MPI number of nodes	1

Red arrows in the image point from the text 'Rank 0' and 'Rank 0 output' in the slide to the 'Process Info' and 'Program output' sections of the screenshot, respectively.

# Module 5: Parallel Debugging

## ★ Objective

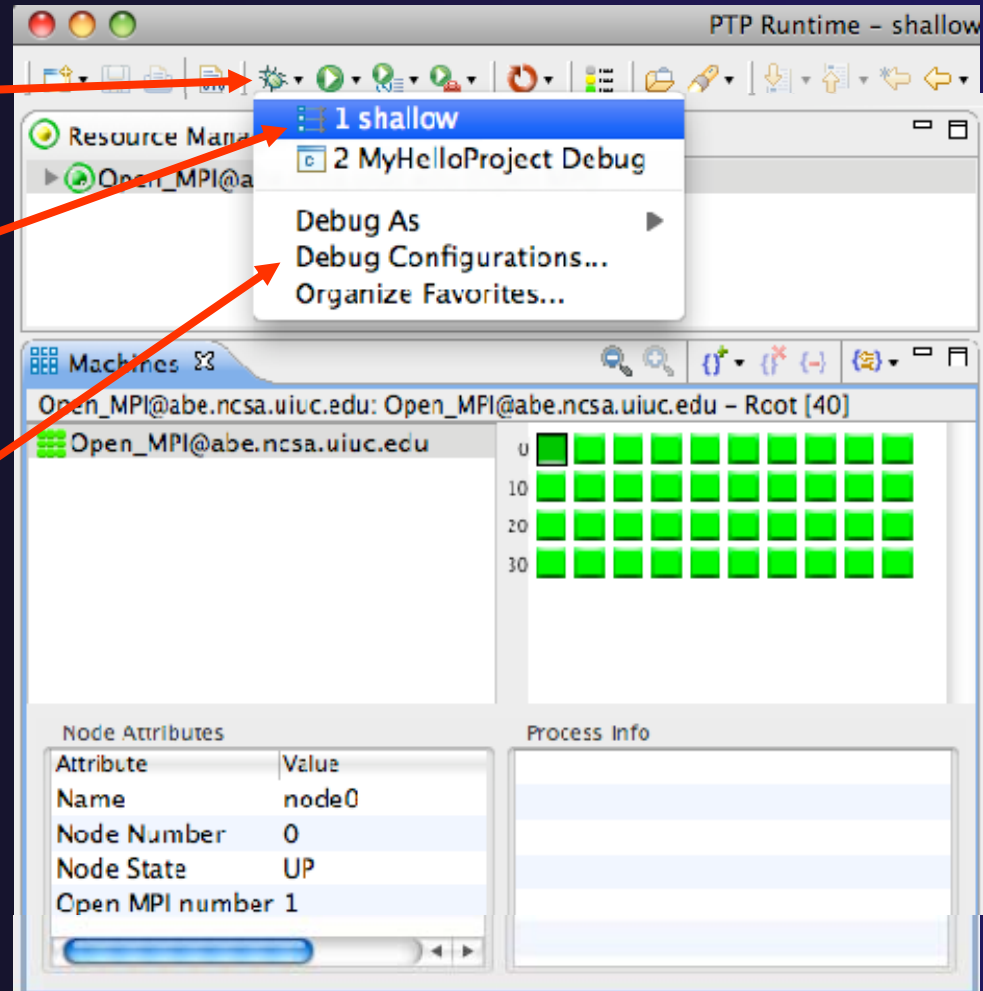
- ★ Learn the basics of debugging parallel programs with PTP

## ★ Contents

- ★ Launching a parallel debug session
- ★ The PTP Debug Perspective
- ★ Controlling individual processes
- ★ Parallel Breakpoints
- ★ Terminating processes

# Launching A Debug Session

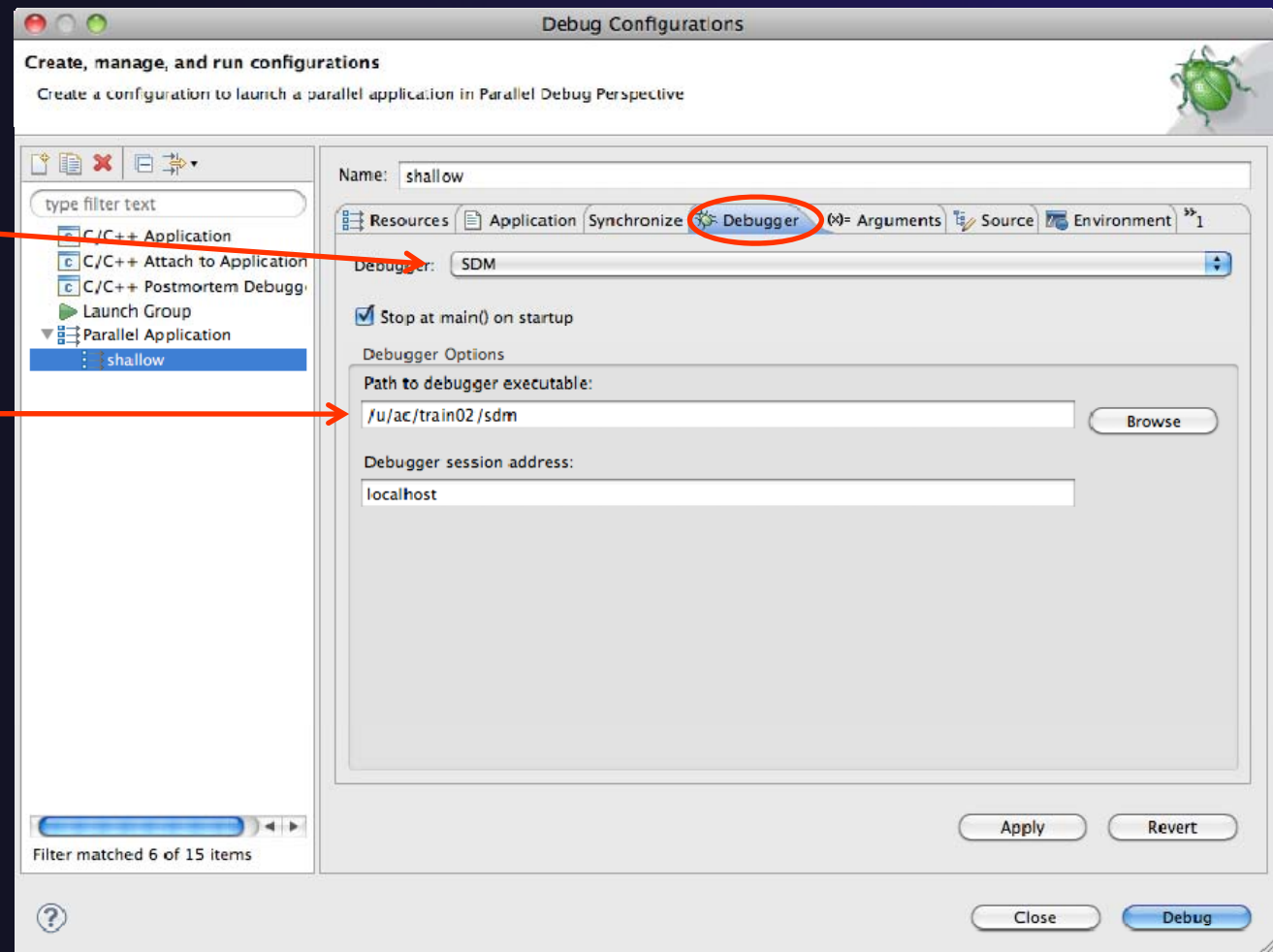
- ★ Use the drop-down next to the debug button (bug icon) instead of run button
- ★ Select the project to launch
- ★ The debug launch will use the same number of processes that the normal launch used
- ★ First, select **Debug Configurations...** to verify the debugger settings





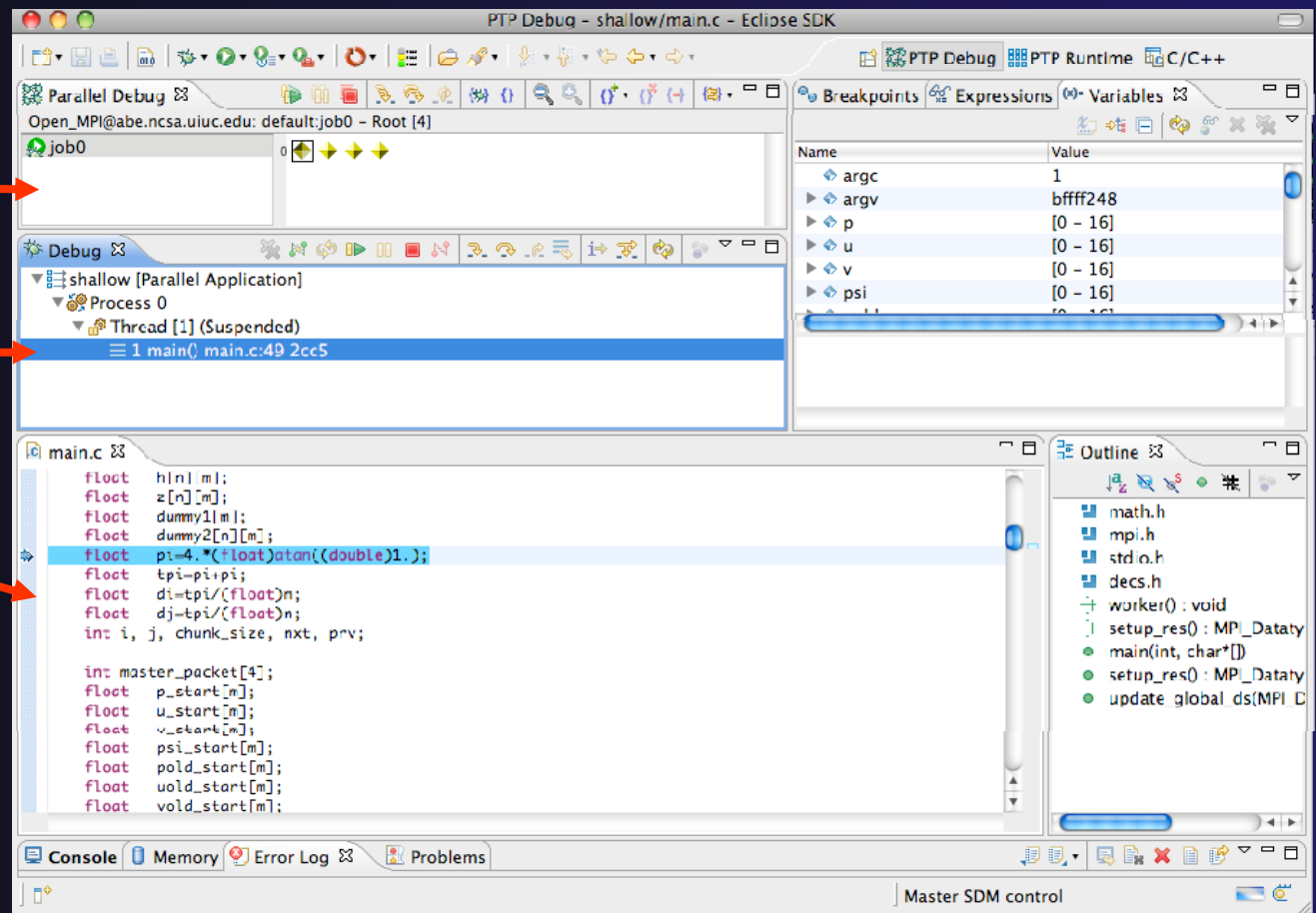
# Verify the Debugger Tab

- ★ Select **Debugger** tab
- ★ Make sure **SDM** is selected in the **Debugger** dropdown
- ★ Use the **Browse** button to select the debugger executable if required
  - ★ If launching remotely, the debugger executable must also be located remotely
- ★ Debugger session address should not need to be changed
- ★ Click on **Debug** to launch the program



# The PTP Debug Perspective (1)

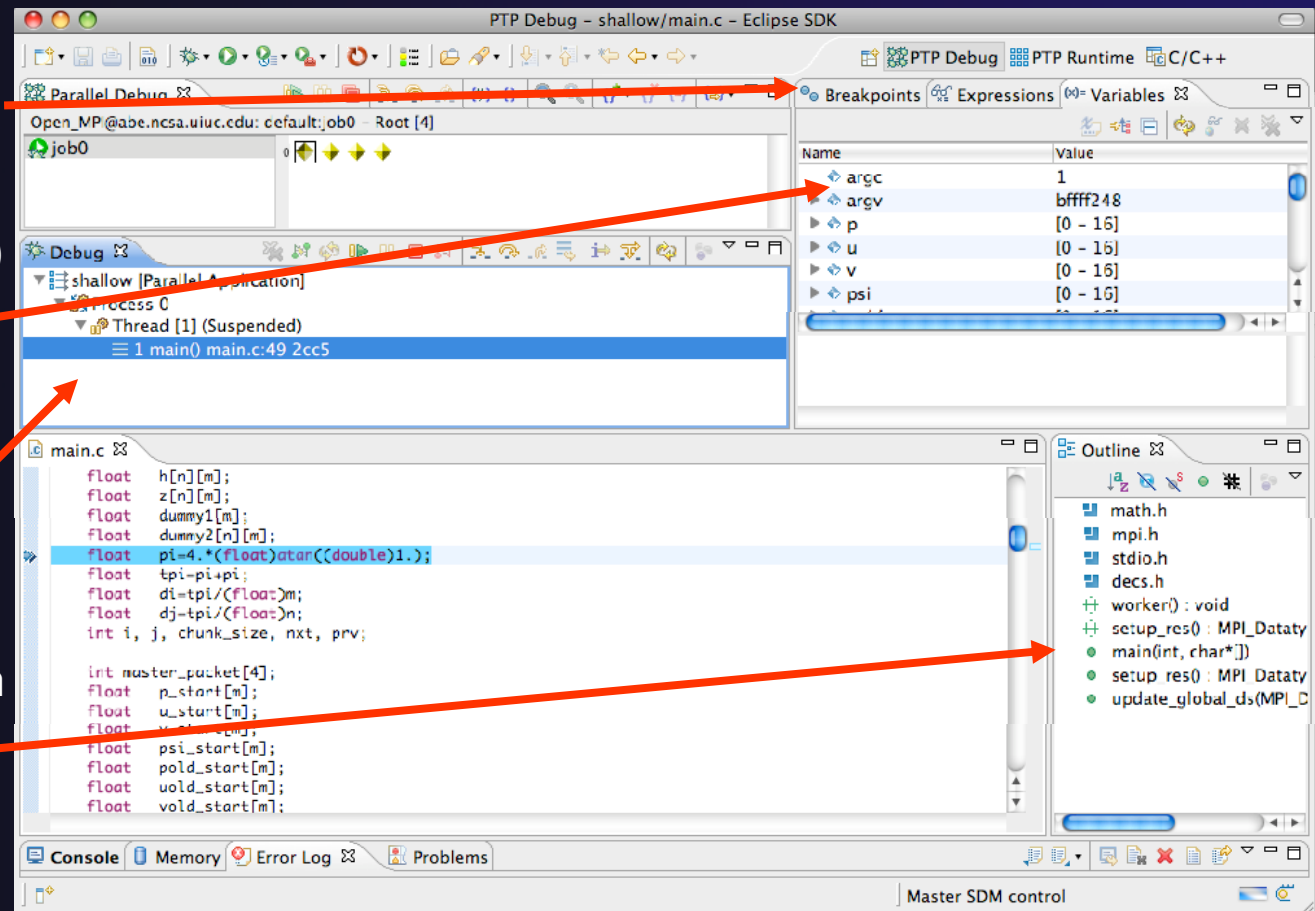
- ★ **Parallel Debug view** shows job and processes being debugged
- ★ **Debug view** shows threads and call stack for individual processes
- ★ **Source view** shows a **current line marker** for all processes





# The PTP Debug Perspective (2)

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





# Stepping All Processes

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

The screenshot displays the Eclipse IDE interface during a parallel debug session. The top window, titled "PTP Debug - shallow/main.c - Eclipse", shows the "Parallel Debug View" with a tree structure under "Open\_MPI@abe.ncsa.uiuc.edu: default:job0 - Root [4]". A process named "job0" is visible with several yellow step-through icons. A red arrow points to the "Step Over" button in the toolbar. Below this, the "Debug View" shows a tree structure for "shallow [Parallel Application]" with "Process 0" and "Thread [1] (Suspended)" expanded. The current line of code is "1 main0 main.c:49 2cc5".

The bottom window, titled "main.c", shows the source code with the following content:

```

float h[n][m];
float z[n][m];
float dummy1[m];
float dummy2[n][m];
float pi=4.*(float)atan((double)1.);
float tpi=pi+pi;
float di=tpi/(float)m;
float dj=tpi/(float)n;
int i, j, chunk_size, nxt, prv;

int master_packet[4];
float p_start[m];
float u_start[m];
float v_start[m];
float psi_start[m];
float pold_start[m];
float vold_start[m];
float vold_start[m];

```

A red arrow points to the line "float tpi=pi+pi;" in the source code, which is highlighted in blue, indicating it is the current line of execution.

# Stepping An Individual Process



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

PTP Debug - shallow/main.c - Eclipse SDK

Parallel Debug

Open\_MPI@abe.ncsa.uiuc.edu: default:job0 - Root [4]

job0

Debug

shallow [Parallel Application...]

Process 0 (Suspended)

Thread [1] (Suspended)

1 main() main.c:51 2cde

main.c

```
float h[n][m];
float z[n][m];
float dummy1[m];
float dummy2[n][m];
float pi=4.*(float)atan((double)1.);
float tpi=pi+pi;
float di=tpi/(float)m;
float dj=tpi/(float)n;
int i, j, chunk_size, nxt, prv;

int master_packet[4];
float p_start[m];
float u_start[m];
float v_start[m];
float psi_start[m];
float pold_start[m];
float uold_start[m];
float vold_start[m];
```

## Current Line Marker

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

# Colors And Markers

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

```

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

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

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

The screenshot shows a code editor window titled 'main.c'. The code is as follows:
   
int proc\_cnt;
   
int tid;
   
MPI\_Datatype \* res\_type;
   
MPI\_Init(&argc, &argv);
   
MPI\_Comm\_size(MPI\_COMM\_WORLD, &proc\_cnt);
   
MPI\_Comm\_rank(MPI\_COMM\_WORLD, &tid);
   
if ( proc\_cnt < 2 )
   
{
   
 fprintf(stderr, "must have at least 2 processes, not %d\n", proc\_cnt);
   
 MPI\_Finalize();
   
 return 1;
   
}

Markers are visible in the left margin: a blue arrow points to the line 'MPI\_Comm\_size...', an orange arrow points to the line 'MPI\_Comm\_rank...', and a green arrow points to the line 'if ( proc\_cnt < 2 )'.



Multiple processes marker



Registered process marker



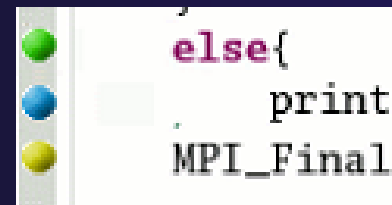
Un-registered process marker



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


# Breakpoints

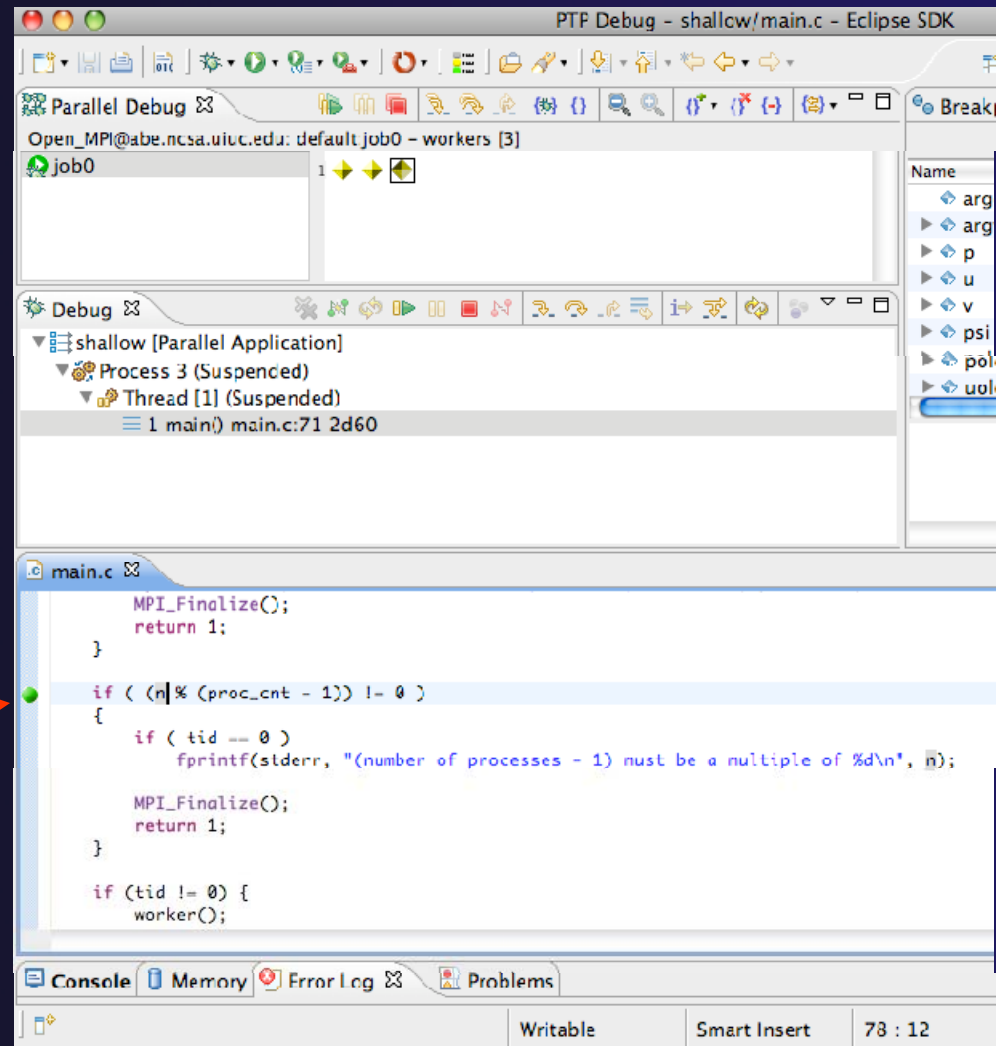
- ★ Apply only to processes in the particular set that is active in the **Parallel Debug view** when the breakpoint is created
- ★ Breakpoints are colored depending on the active process set and the set the breakpoint applies to:
  - ★ **Green** indicates the breakpoint set is the same as the active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





# Creating A Breakpoint

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





# Global Breakpoints

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

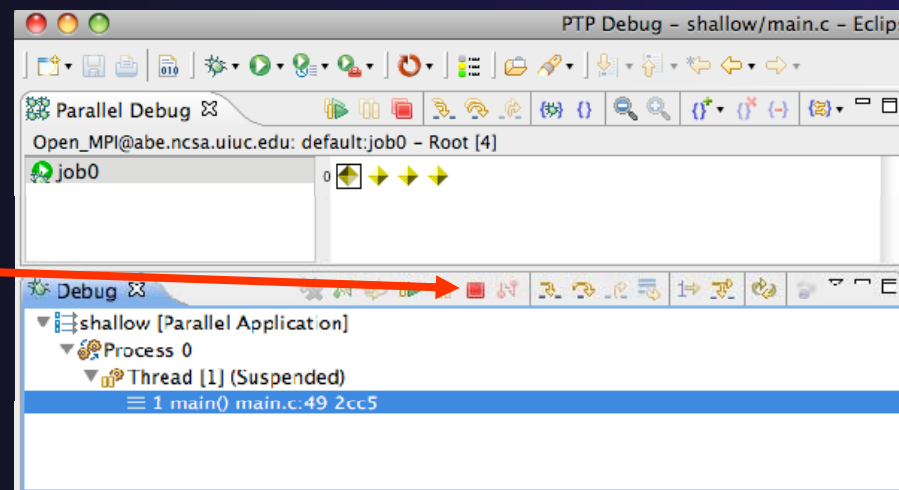
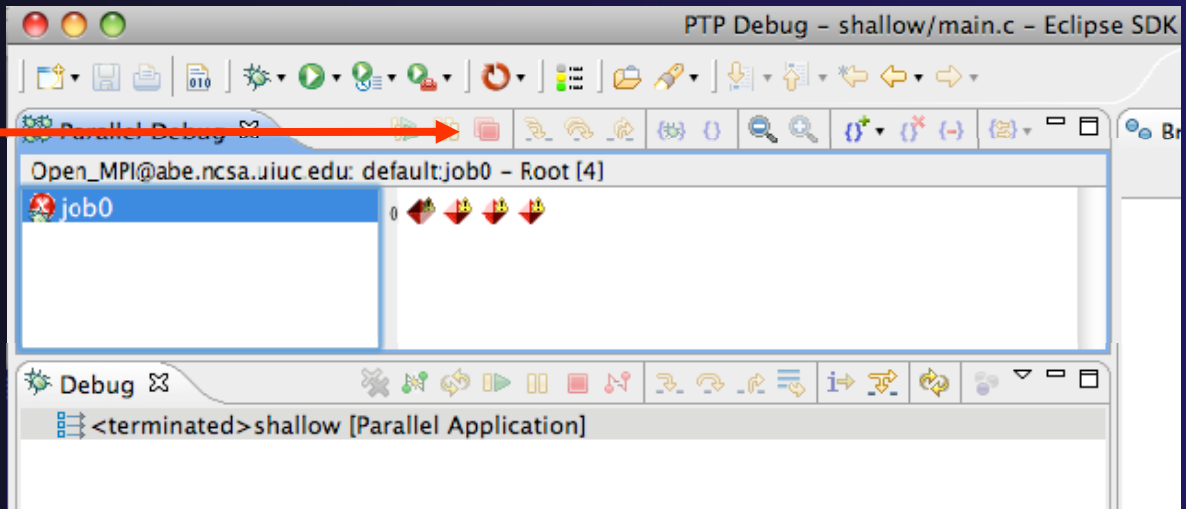
```
if (my_rank != 0) {  
    /* create message */  
    sprintf(message, "Greetin
```





# Terminating A Debug Session

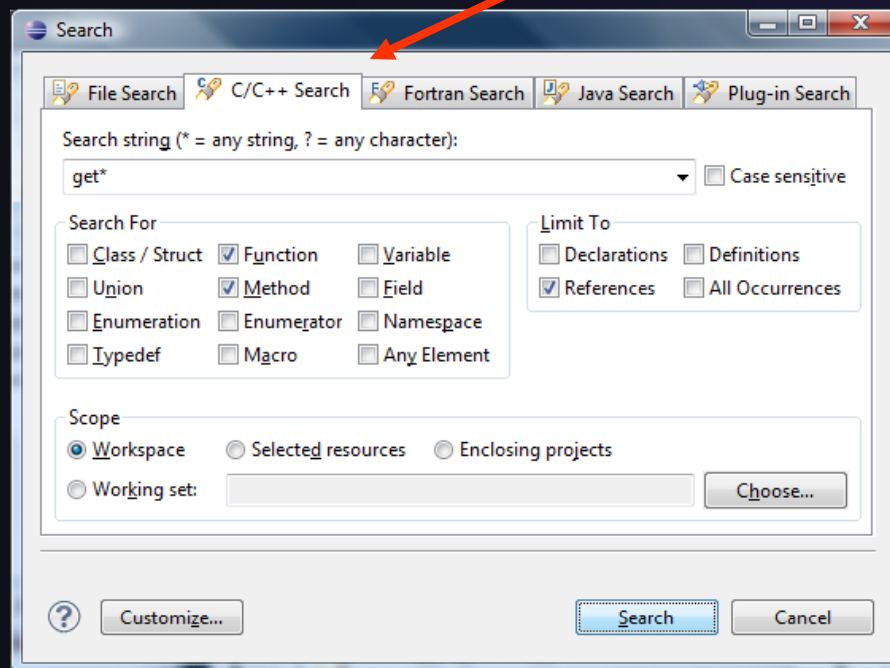
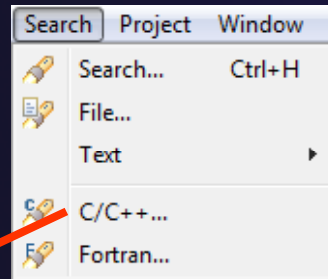
- ★ Click on the **Terminate** icon in the **Parallel Debug view** to terminate all processes in the active set
- ★ Make sure the **Root** set is active if you want to terminate all processes
- ★ You can also use the terminate icon in the **Debug view** to terminate the currently selected process



## Other CDT features

- ✦ Searching
- ✦ Open Declaration / hyperlinking between files in the editor
- ✦ Rename in file (in-place in editor)
- ✦ Refactoring
  - ✦ Rename refactoring / Preview panes
  - ✦ Extract Constant refactoring
  - ✦ Other refactorings in CDT

# Language-Based Searching

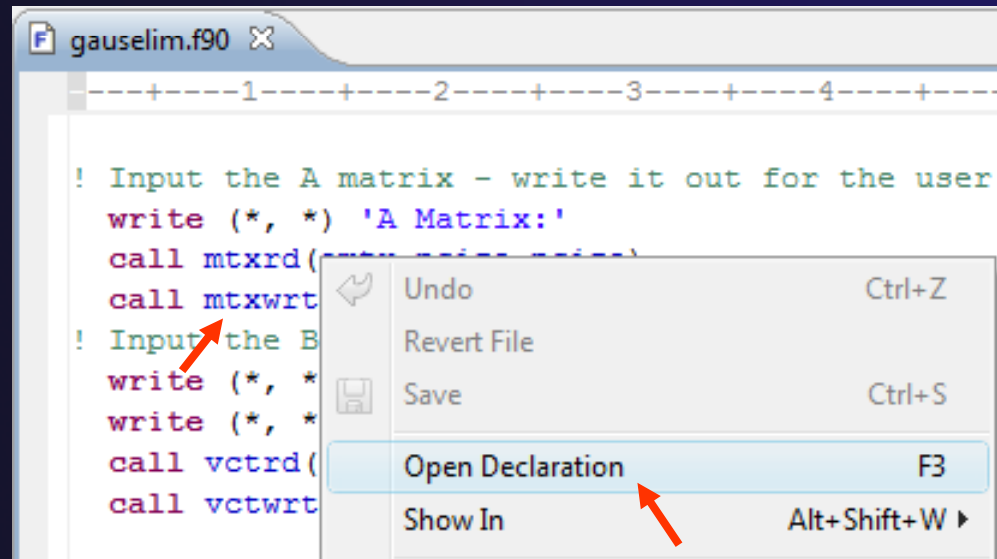


- ★ “Knows” what things can be declared in each language (functions, variables, classes, modules, etc.)
- ★ E.g., search for every call to a function whose name starts with “get”
- ★ Search can be project- or workspace-wide

# Open Declaration

## Need this for C

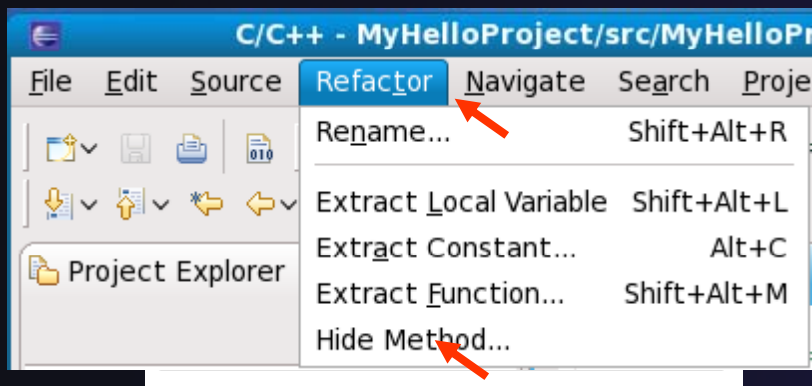
- ✦ Jumps to the declaration of a variable, function, etc., even if it's in a different file
- ✦ Right-click on an identifier
- ✦ Click **Open Declaration**



*Tip: Open Declaration works in C/C++, and it works in Fortran, but it cannot jump "across languages"*

# Rename Refactoring

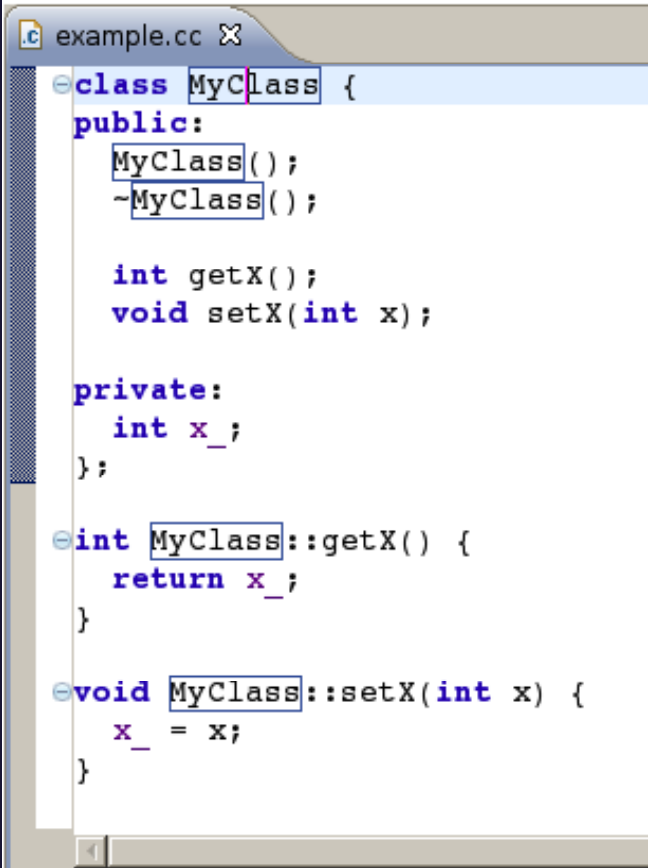
- ✦ Changes the name of a variable, function, etc., *including every use*  
(change is semantic, not textual, and can be workspace-wide)
- ✦ Only proceeds if the new name will be legal  
(aware of scoping rules, namespaces, etc.)



- ✦ Select **C/C++ Perspective**
- ✦ Open a source file
- ✦ Click in editor view on declaration of a variable
- ✦ Select menu item **Refactor ▶ Rename**
  - ✦ Or use context menu
- ✦ Enter new name

# CDT Rename in File

- ★ 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.



```
example.cc X
class MyClass {
public:
    MyClass();
    ~MyClass();

    int getX();
    void setX(int x);

private:
    int x_;
};

int MyClass::getX() {
    return x_;
}

void MyClass::setX(int x) {
    x_ = x;
}
```

# Module 8: Other Tools and Wrap-up

## ✦ Objective

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

## ✦ Contents

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



NCSA

## 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
  - ★ Scalable debugger
  - ★ Workflow tools  
(<https://wiki.ncsa.uiuc.edu/display/MRD PUB/MRD+Public+Space+Home+Page>)
- ★ Part of the enhanced computational environment described at:  
<http://www.ncsa.illinois.edu/BlueWaters/ece.html>





# Online Information

- ✦ Information about PTP
  - ✦ Main web site for downloads, documentation, etc.
    - ✦ <http://eclipse.org/ptp>
  - ✦ Developers' 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 –
    - ★ <http://dev.eclipse.org/mailman/listinfo/photran-dev>

# Getting Involved

- ★ See <http://eclipse.org/ptp>
- ★ Read the developer documentation on the wiki
- ★ Join the mailing lists
- ★ Attend the monthly developer meetings
  - ★ Teleconference each second Tuesday, 1:00 pm ET
  
- ★ PTP will only succeed with your participation!