

Module 3: Working with C/C++

★ Objective

- ★ Learn basic Eclipse concepts: Perspectives, Views, ...
- ★ Learn how to use Eclipse to manage a remote project
- ★ Learn how to use Eclipse to develop C programs
- ★ Learn how to launch and run a remote C program

★ Contents

- ★ Brief introduction to the C/C++ Development Tools (CDT)
- ★ Create a simple remote application
- ★ Learn to launch a remote C application

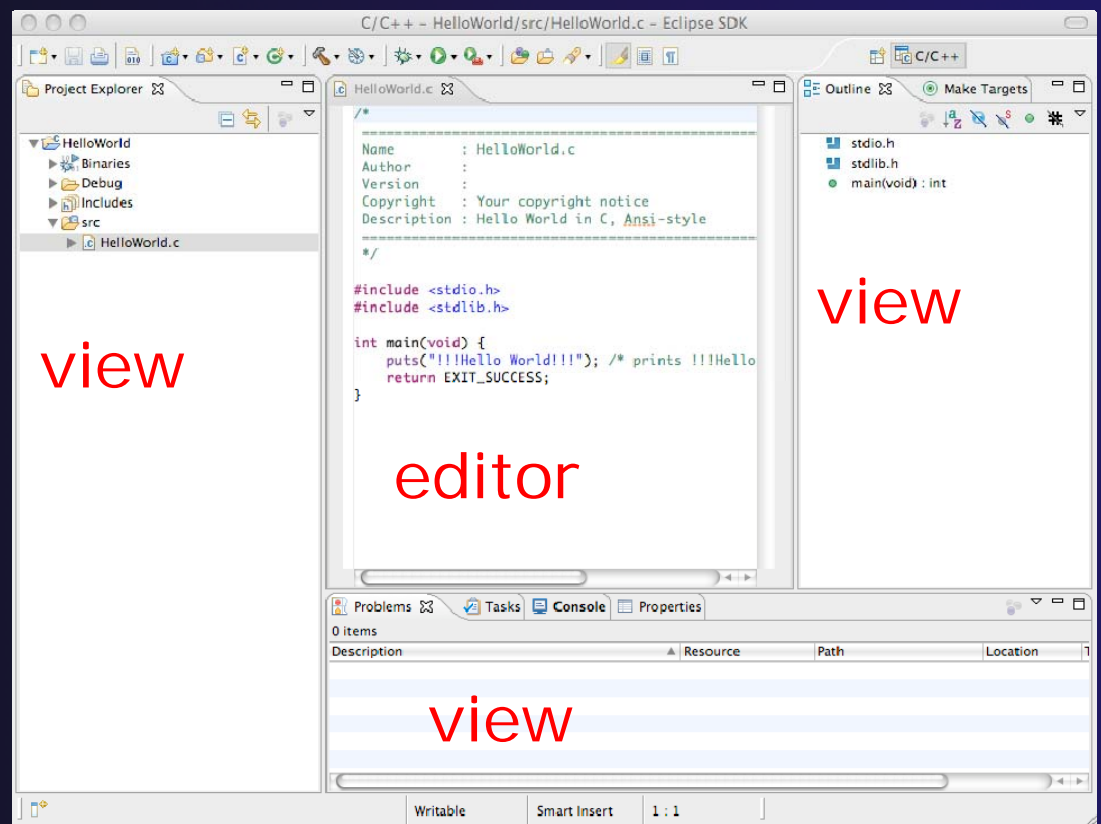
Login Information

- ★ The hands on portion of this module will be done on a remote system at **SDSC**, thank you to SDSC!
 - ★ **trestles.sdsc.edu**
- ★ See the following URL for more information on the system
 - ★ <http://www.sdsc.edu/us/resources/trestles/>
 - ★ Each student will be assigned an ID and password at the start of the tutorial
- ★ Please use only this ID
 - ★ We are also working to make this work with Ranger and Kraken, this work is not complete...

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

Module 3



perspective

3-2

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

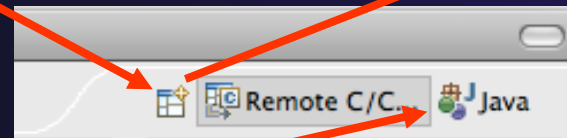
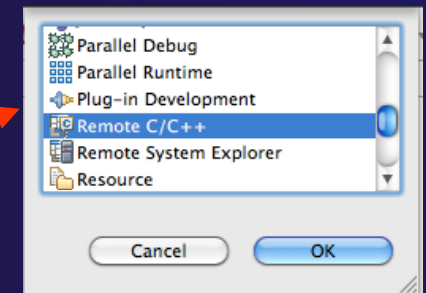
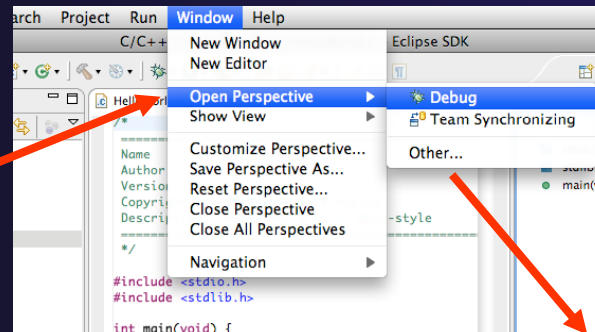
- ★ Choose the **Window>Open Perspective** menu option

- ★ Then choose **Other...**

- ★ Click on the **Open Perspective** button in the upper right corner of screen

- ★ Click on a perspective shortcut button

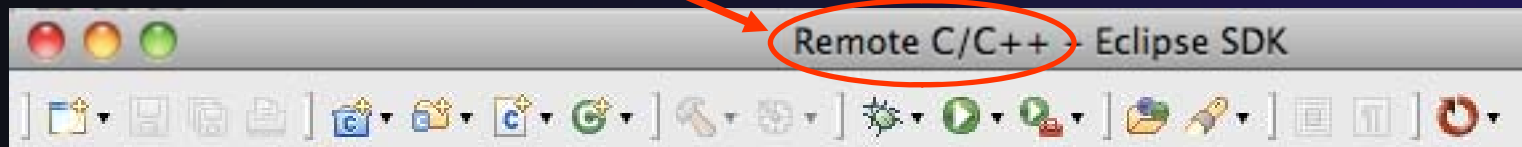
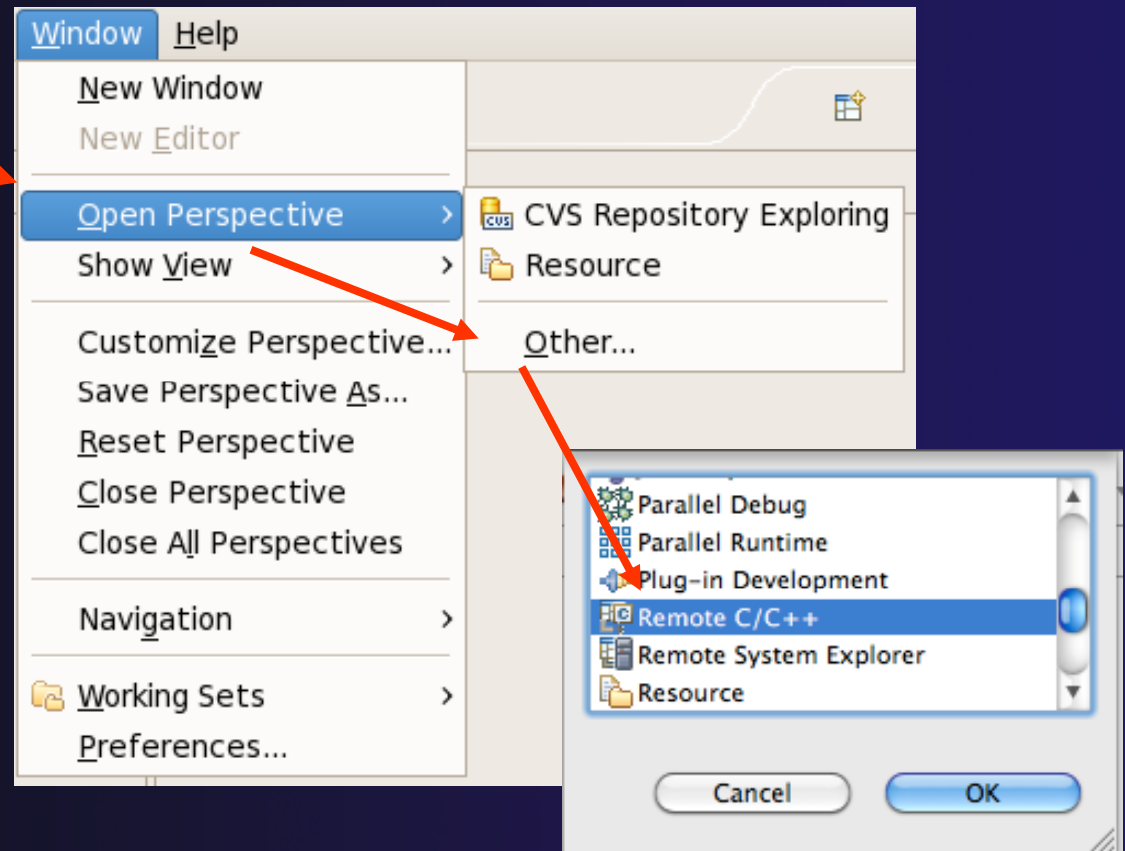
- ★ Switch perspective on next slide...





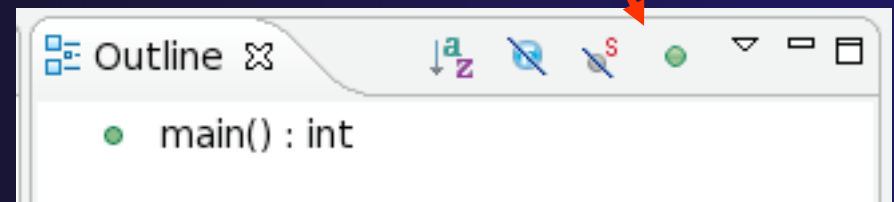
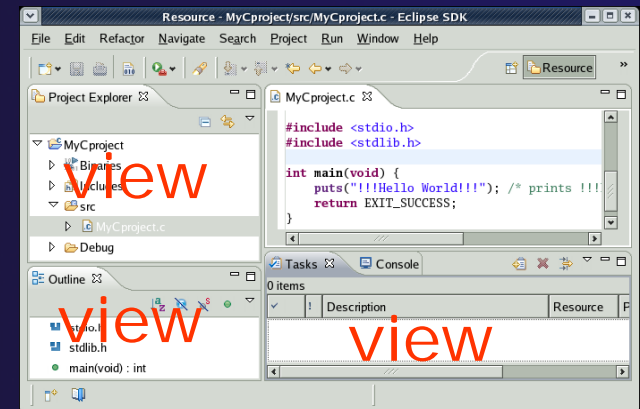
Switch to Remote C/C++ Perspective

- ★ Select **Window>Open Perspective**
- ★ Then choose **Other...**
- ★ Only needed if you're not already in the perspective
- ★ What Perspective am in in?
See title Bar



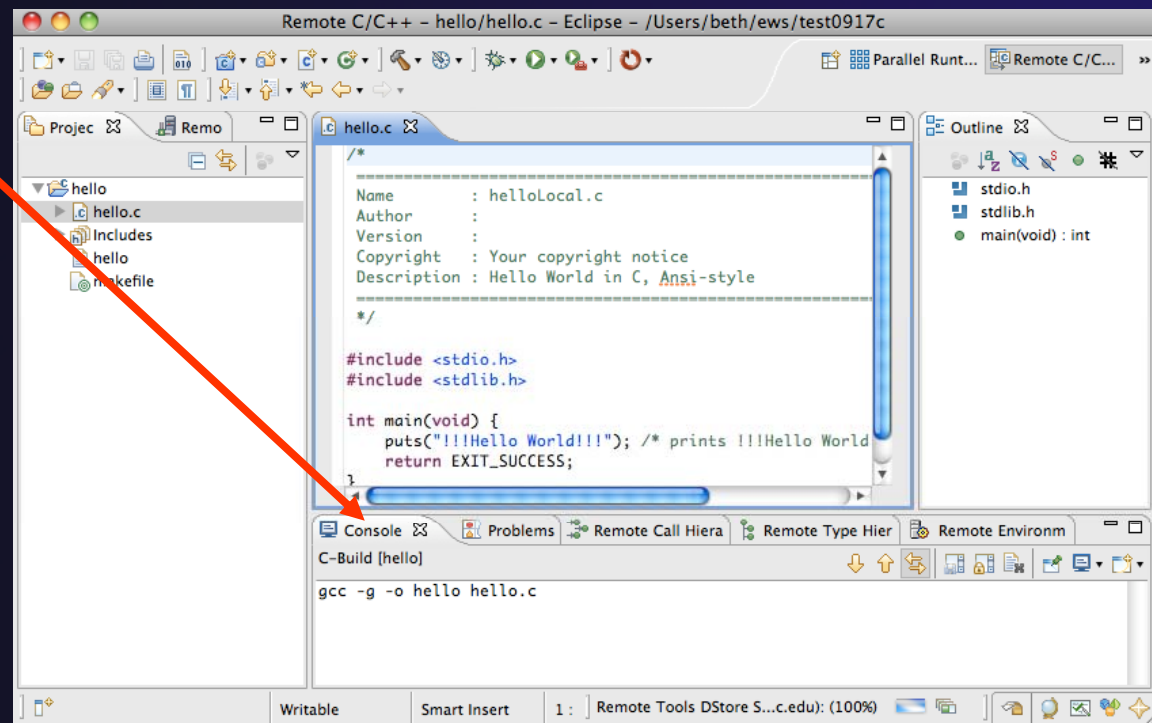
Views

- ★ 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



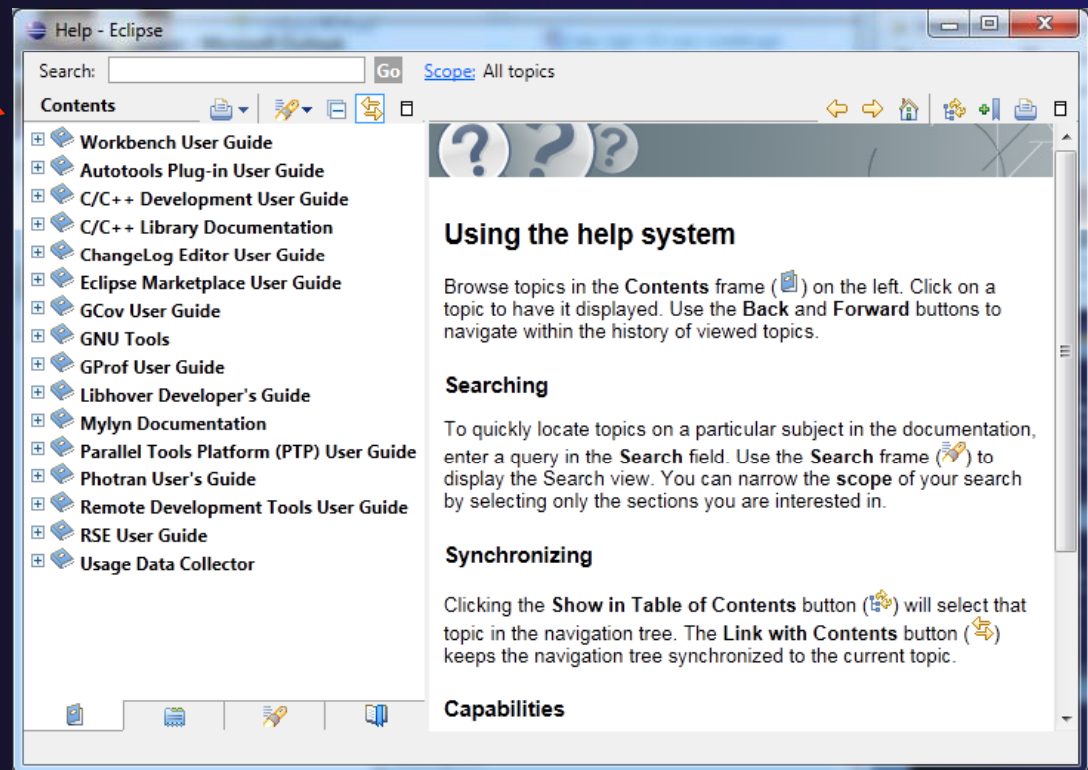
Stacked Views

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

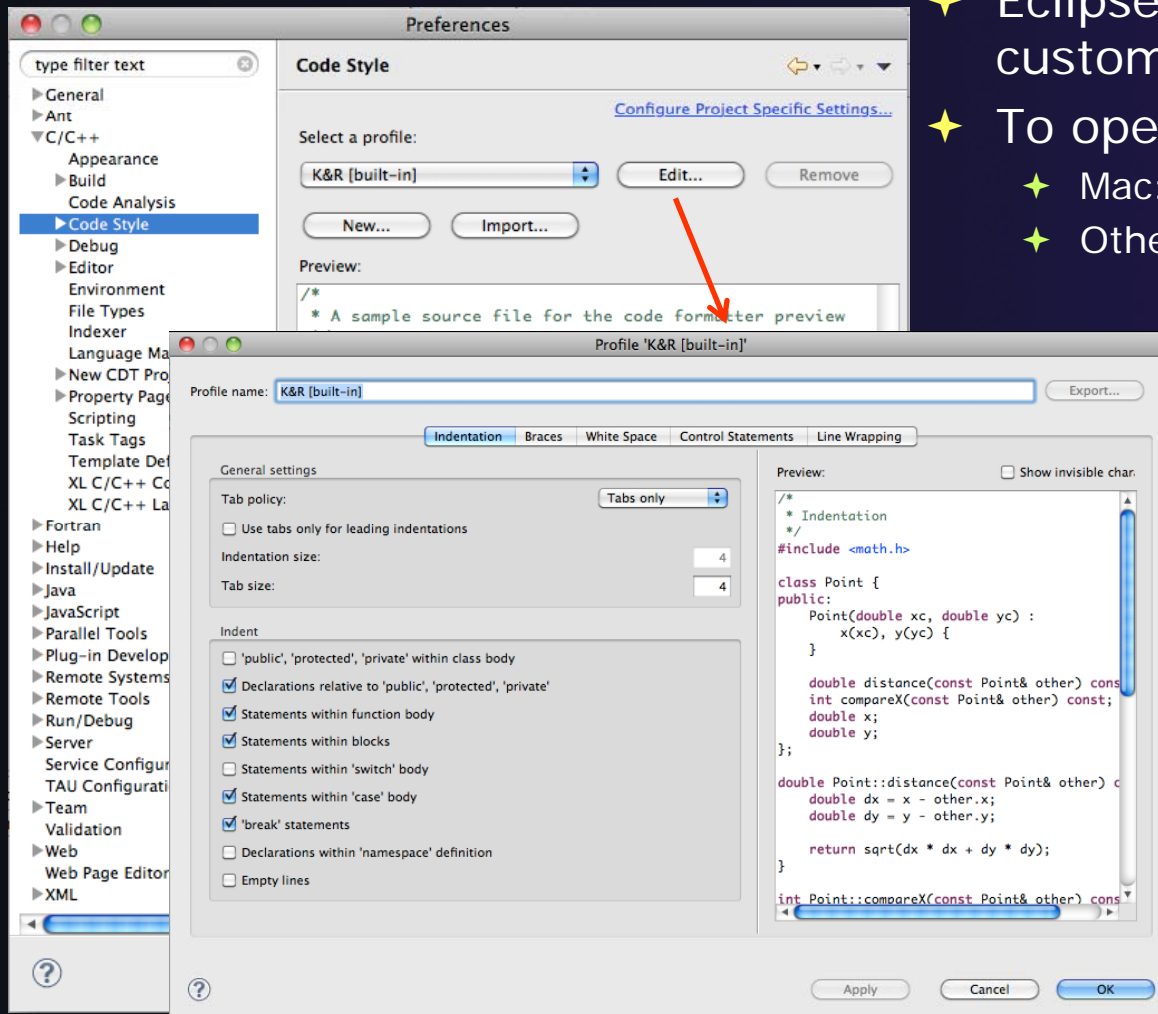


Help

- ★ 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.)



Preferences



- ✦ Eclipse Preferences allow customization of almost everything
- ✦ To open use
 - ✦ Mac: **Eclipse>Preferences...**
 - ✦ Others: **Windows>Preferences...**
- ✦ The C/C++ preferences allow many options to be altered
- ✦ In this example the Code Style preferences are shown
 - ✦ These allow code to be automatically formatted in different ways

Types of C/C++ Projects

- ★ C/C++ Projects can be
 - ★ **Local** – source is located on local machine, builds happen locally
 - ★ **Remote** – source is either located on remote machine, or synchronized with remote machine, builds take place on remote machine
 - ★ **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 the local machine or on a remote system
 - ★ MPI needs to be installed
 - ★ An application built locally probably can't be run on a remote machine unless their architectures are the same
- ★ We will show you how to create, build and run the program on a remote machine
 - ★ We will create a remote Makefile project

Remote Projects

★ “Traditional” Remote Projects

- ★ Source is located on remote machine
- ★ Eclipse is installed on the local machine and can be used for:
 - ★ Editing
 - ★ Building
 - ★ Running
 - ★ Debugging
- ★ Source indexing is performed on remote machine
 - ★ Enables call hierarchy, type hierarchy, include browser, search, outline view, and more...
- ★ Builds are performed on remote machine
 - ★ Supports both managed and makefile projects
- ★ Application is run and debugged remotely using the PTP resource managers

★ Synchronized Projects

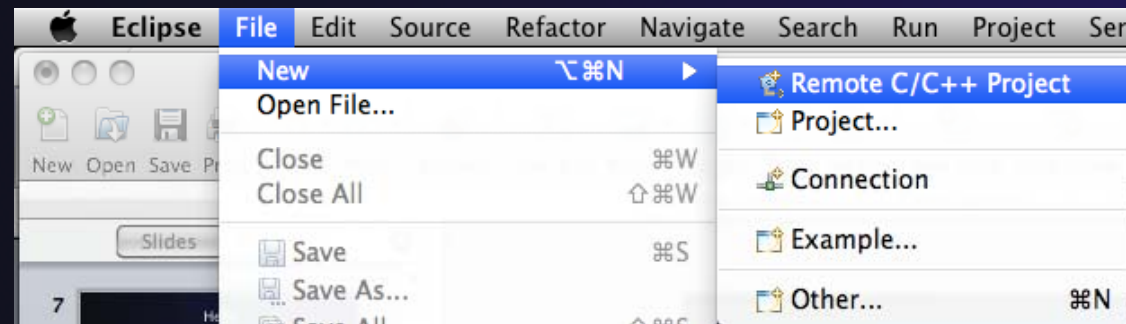
- ★ Source is located on *both* the local system and on a remote target system. The two copies are kept in sync by Eclipse.
- ★ Eclipse is installed on the local machine and can be used for:
 - ★ Editing
 - ★ Building
 - ★ Running
 - ★ Debugging
 - ★ *Development can continue “off-line”*
- ★ Source indexing is performed on *local* machine
 - ★ Enables call hierarchy, type hierarchy, include browser, search, outline view, and more...
- ★ Builds are performed on *one or more* remote machines
 - ★ Supports both managed and makefile projects
- ★ Application is run and debugged remotely using the PTP resource managers

Traditional Remote Projects



Creating a Remote C/C++ Project

- ★ Use **File>New>Remote C/C++ Project** to open the new project wizard
- ★ The wizard will take you through the steps for creating the project

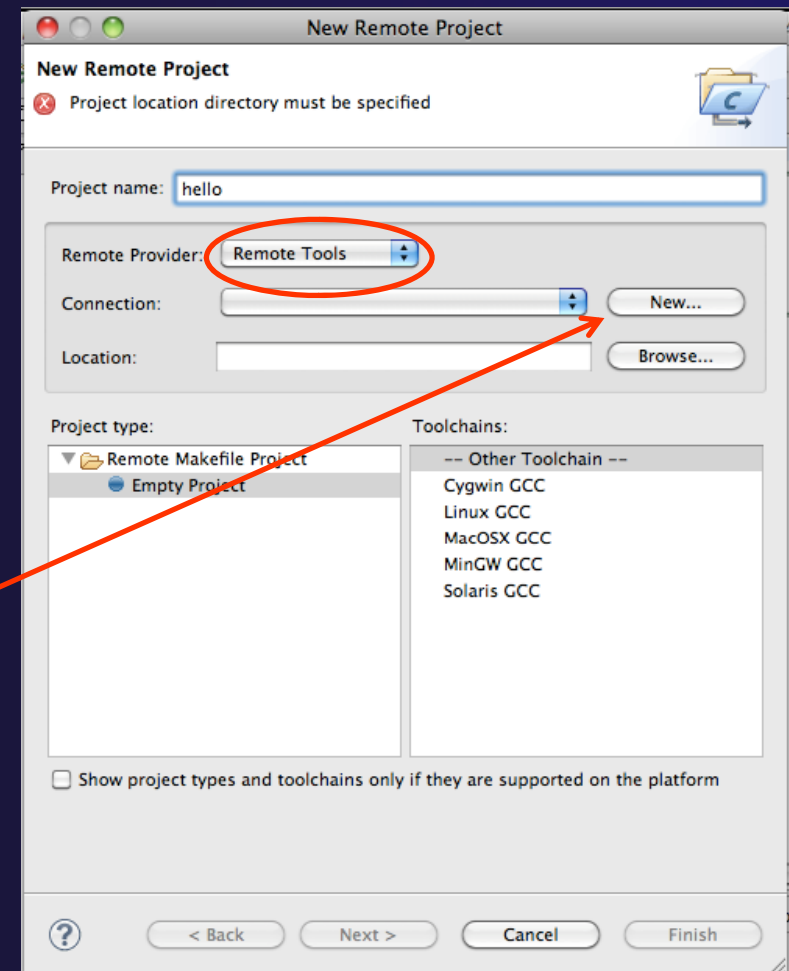


Don't see the "Remote C/C++ Project" choice?
Make sure you are in the Remote C/C++ Perspective



New Remote Project Wizard

- ✦ Enter project name, e.g. "hello"
- ✦ Select a **Remote Provider**
 - ✦ Remote providers supply different ways of accessing remote (or local) systems
 - ✦ Choose **Remote Tools**
- ✦ A **Connection** specifies how to connect to the remote host
 - ✦ Click on the **New...** button to create a new connection





Remote Host Configuration

- ✦ Enter a connection name (can be anything) for the **Target name**
 - ✦ Use "abe.ncsa.uiuc.edu"
- ✦ The host is remote, so the **Remote host** option should be checked
- ✦ Enter the host name or IP address of the remote host for the **Host**
 - ✦ Use "abe.ncsa.uiuc.edu"
- ✦ Enter the user name and password supplied at the beginning of the tutorial for the **User** and **Password**
- ✦ Note: if your remote machine uses OTP for authentication, *leave the password field blank*
- ✦ Click **Finish**

Target Environment Configuration

Generic Remote Host
Properties for connecting to a generic host

Target name:

Host Information

☐ Localhost ☒ Remote host

Host:

User:

☒ Password based authentication

Password:

☐ Public key based authentication

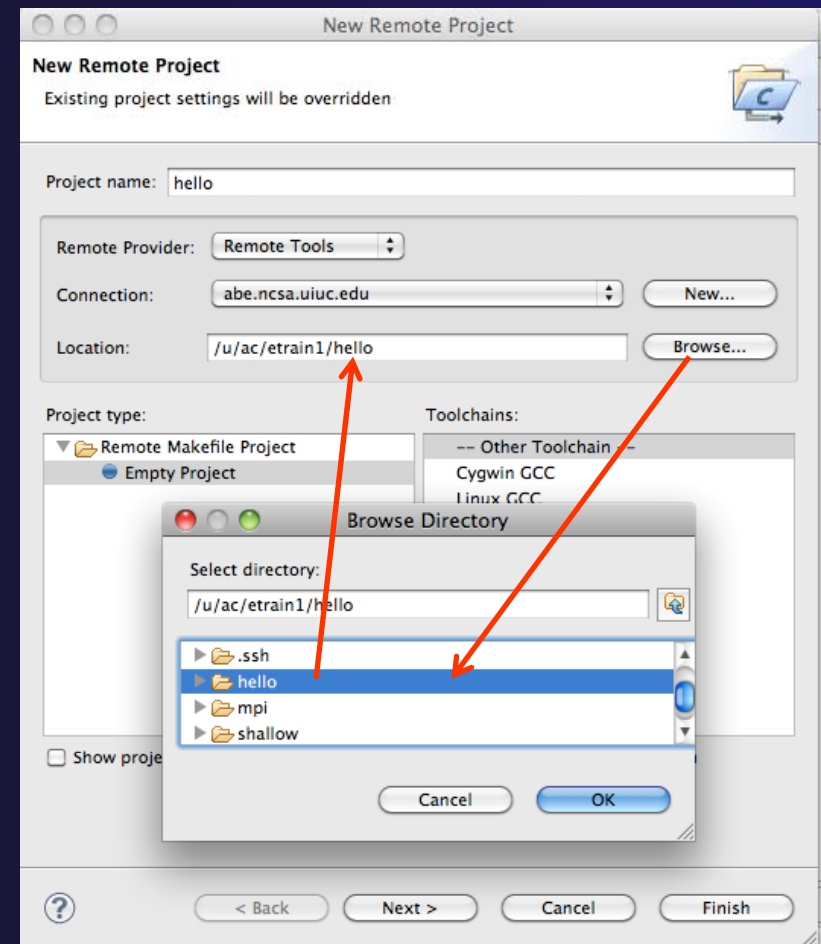
File with private key:

Passphrase:

Project Location



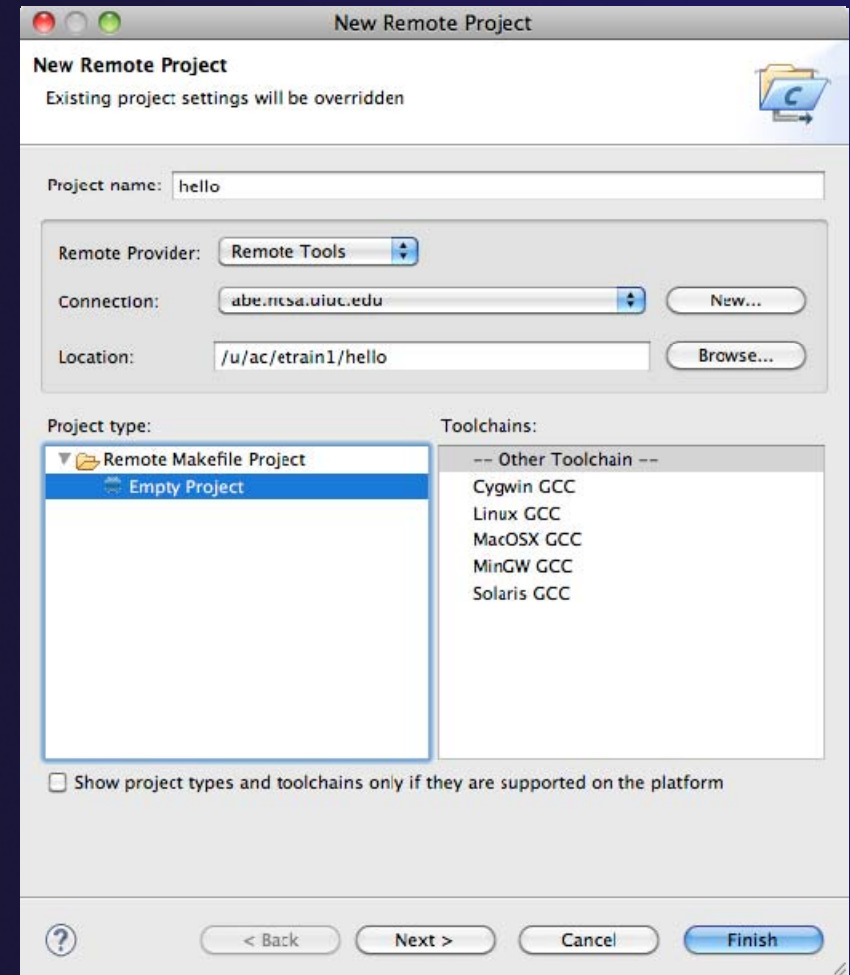
- ★ The **Location** is the directory on the remote host containing the source and executable files
- ★ Click on the browse button to browse for folders on the remote machine
 - ★ You should see the folders in your home directory
 - ★ Choose the "hello" directory
- ★ Click **OK**





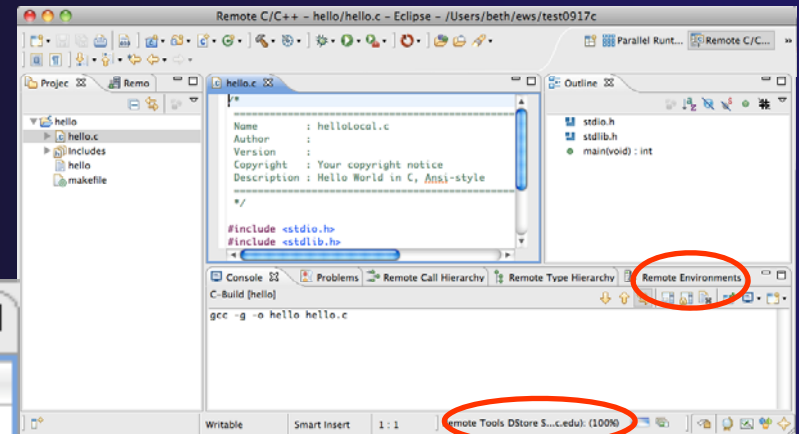
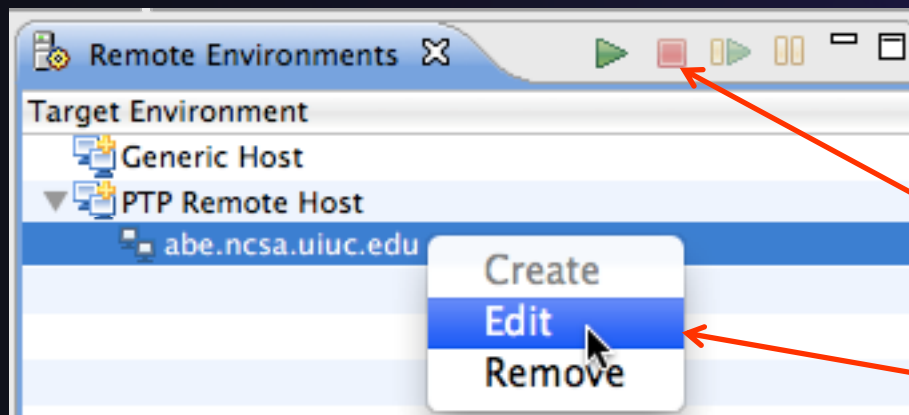
Project Type

- ★ The **Project type** determines information about the project
 - ★ If the project is managed or unmanaged (described later)
 - ★ The tool chain (compiler, linker, etc.) to use when building
 - ★ If the project creates an executable, static, or shared library
 - ★ Options available depend on whether the project is local or remote
- ★ Under **Remote Makefile Project**, select **Empty Project**
- ★ For **Toolchains**, select **Other Toolchain**
- ★ Click on **Finish** to complete the wizard



Changing Remote Connection Information

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



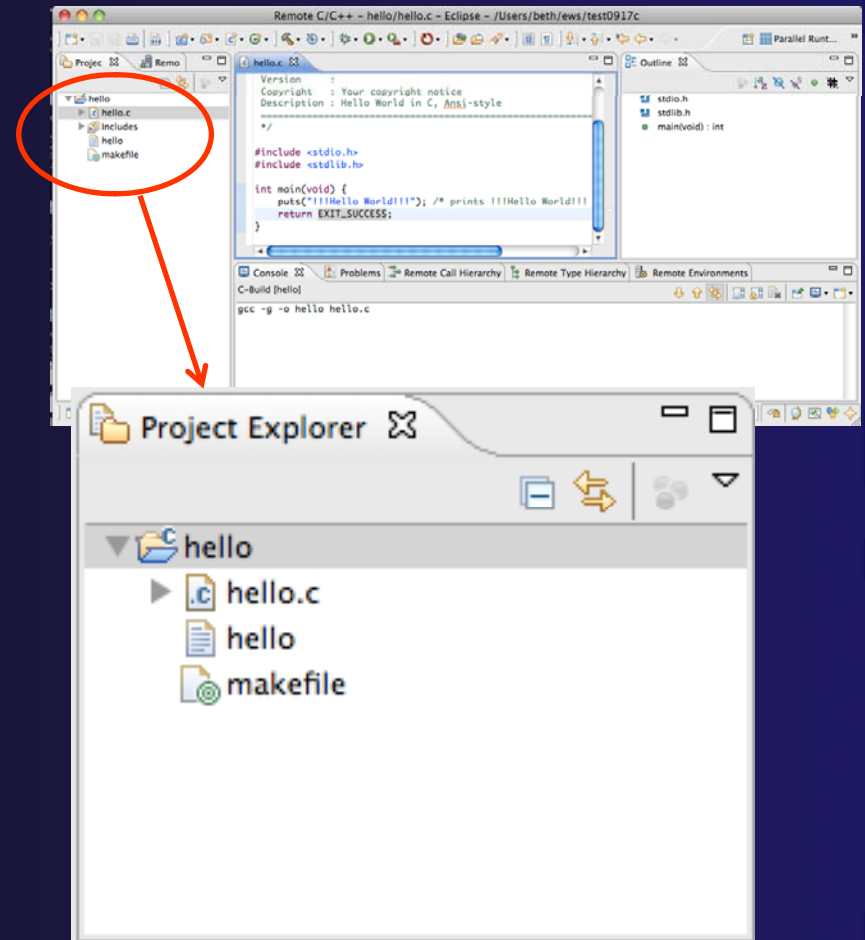
- ★ Stop the remote connection first
- ★ Right-click and select **Edit**

- ★ Note: running server is shown in lower right
 - ★ Opening any remote file restarts it

Remote Tools DStore S...c.edu): (100%)

Project Explorer View

- ✦ Shows the user's projects
- ✦ Each project contains
 - ✦ Source files
 - ✦ Executable files
 - ✦ Folders
 - ✦ Metadata (not visible)
- ✦ Can have any number of projects
- ✦ We only have a single project so far



Synchronized Remote Projects

Goals

- ✦ Check out code from CVS
- ✦ Set up synchronized project with code from CVS
- ✦ Manage builds as with traditional remote projects

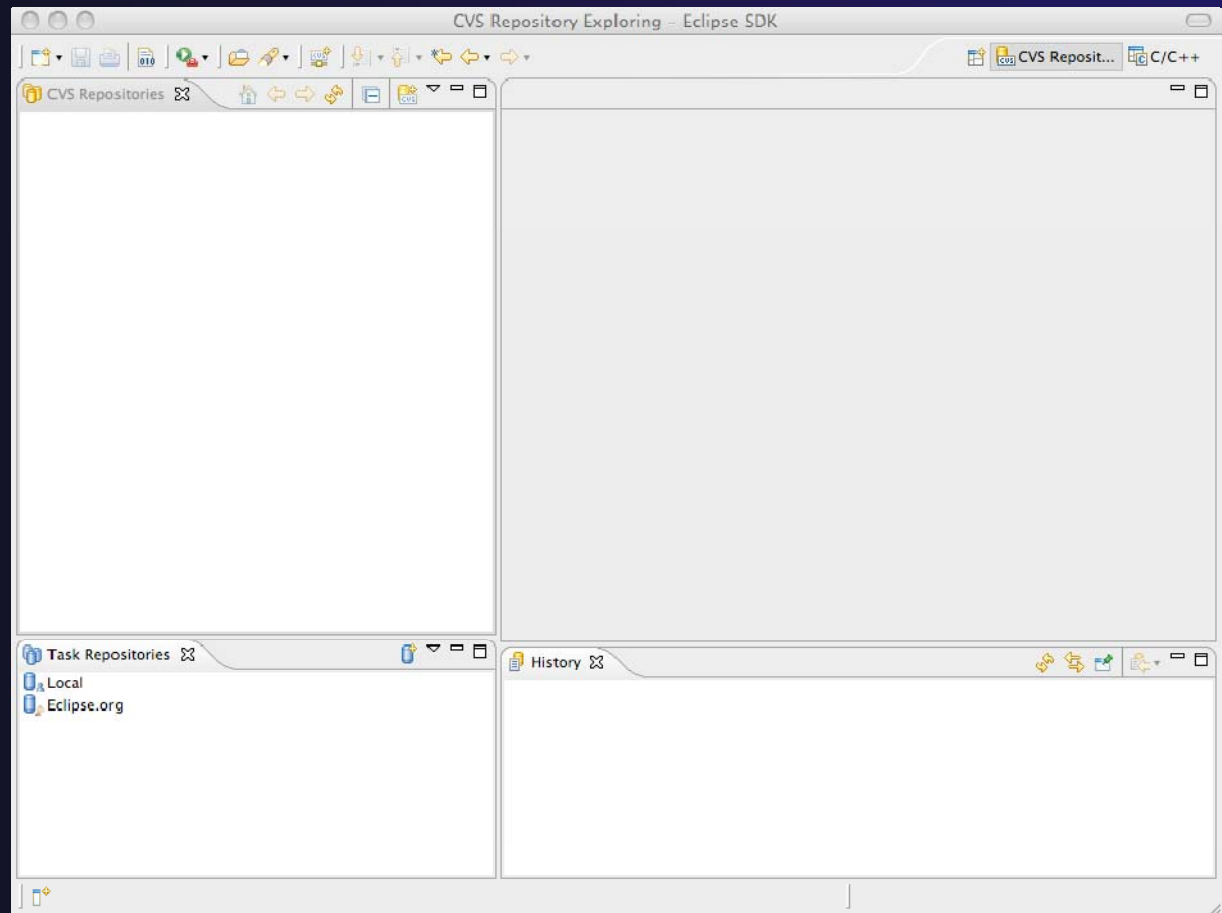
Creating the Project

- ✦ Configuring version control
- ✦ Checking out the source code
- ✦ Team support

Connecting to a Repository



- ★ Select **Window** ► **Open Perspective** ► **Other...**
- ★ Select **CVS Repository Exploring** then **OK**



Specify Repository Location



- ★ Right-click in the **CVS Repositories** view, then select **New ► Repository Location...**
- ★ Set **Host** to the hostname of remote machine
- ★ Set **Repository path** to the CVS repository path
- ★ Fill in **Username** and **Password**
- ★ Set **Connection type** to **extssh** to use an ssh connection

Add CVS Repository

Add a new CVS Repository to the CVS Repositories view

Location

Host: cvs.ncsa.uiuc.edu

Repository path: /CVS/ptp-samples

Authentication

User: anonymous

Password:

Connection

Connection type: pserver

☒ Use default port

☐ Use port:

☒ Validate connection on finish

☐ Save password (could trigger secure storage login)

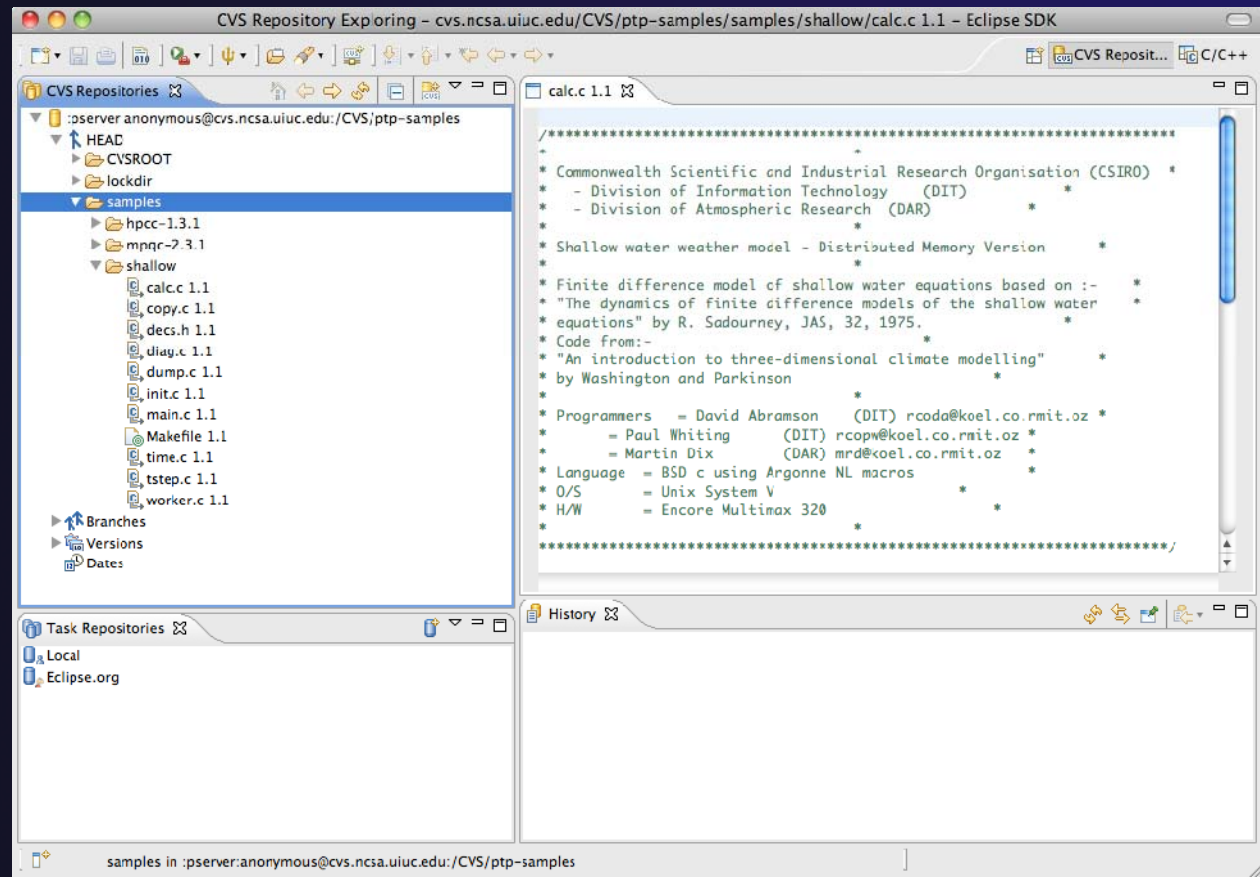
To manage your password, please see ['Secure Storage' Configure connection preferences...](#)

Cancel Finish

CVS Repository Exploring

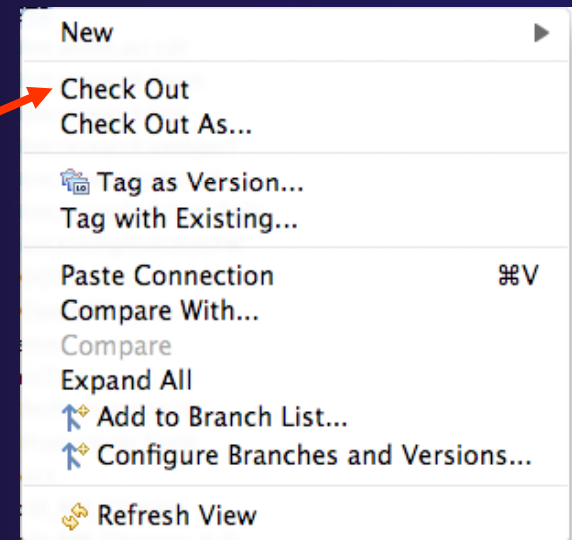


- ✦ Open the repository in the **CVS Repository** view
- ✦ Open **HEAD** to view files and folders in the CVS head
- ✦ Open **Branches** or **Versions** to view CVS branches or versions respectively
- ✦ Right-click on the repository and select **Refresh Branches...** to see all branches and versions



Checking out code in Eclipse

- ★ If the project exists in the repository as an **Eclipse Project**, then one can simply “Check Out” the code
- ★ In CVS Repositories view, right-click on project and select **Project ► Check Out**
- ★ Our example doesn't have Eclipse Project information – this code was checked in with command line tools.
- ★ Our next slide shows how to add Eclipse Project information automatically as you check out the code.

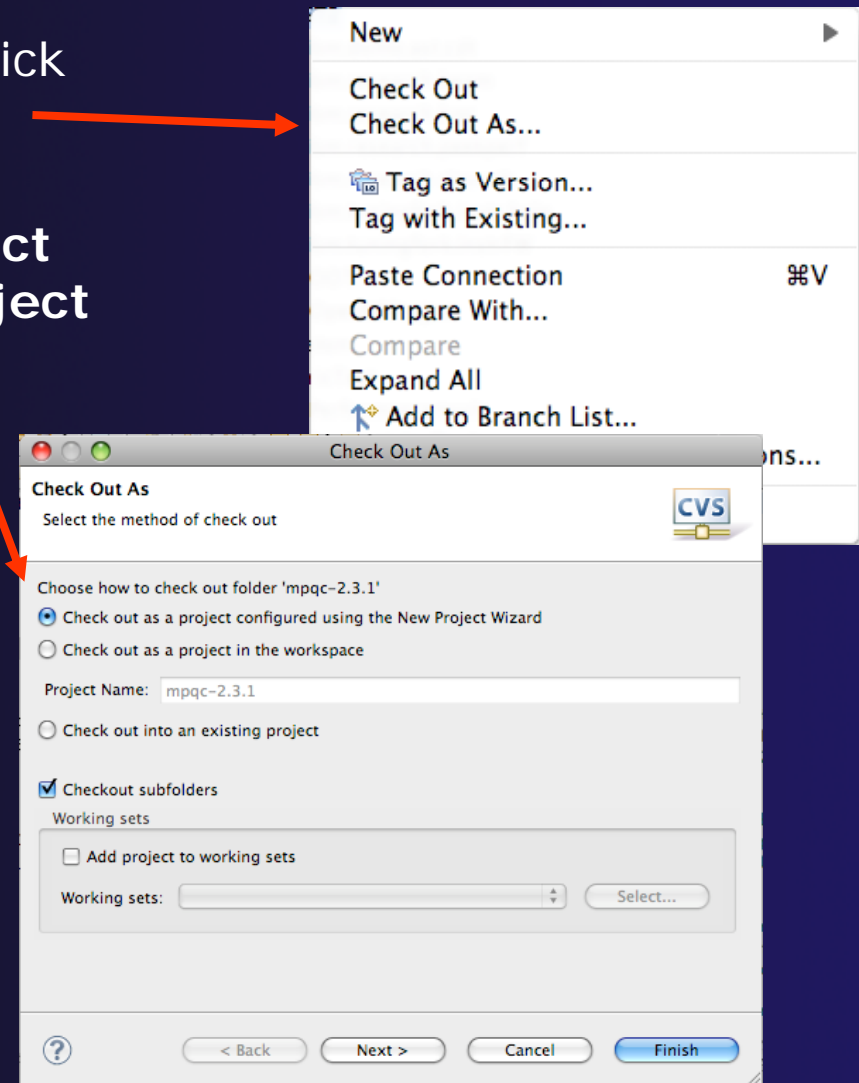


Check out as an Eclipse Project



- ★ In CVS Repositories view, right-click on project and select **Project ► Check Out As...**
- ★ Make sure **Check out as a project configured using the New Project Wizard** is selected
- ★ Leave **Checkout subfolders** checked
- ★ Select **Finish**

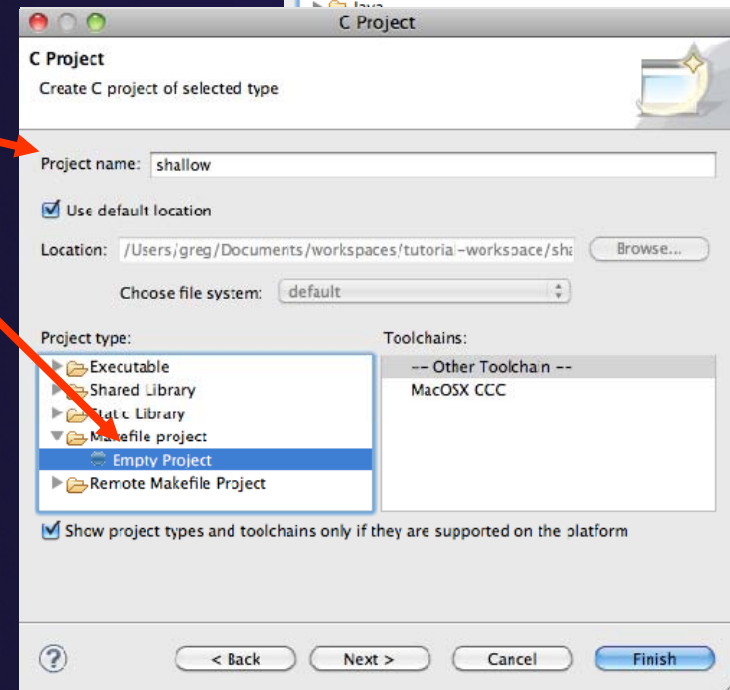
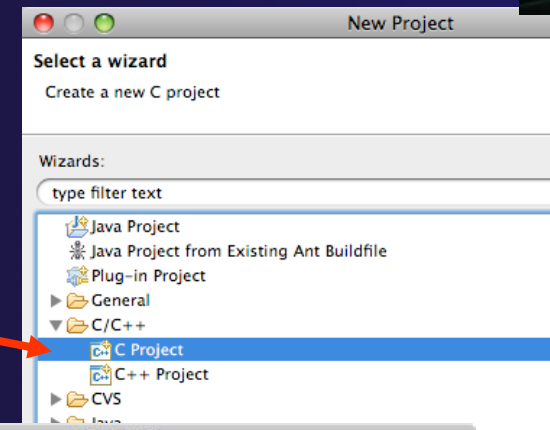
The wizard that runs next will add Eclipse information to the project.



New Project Wizard: Create a C Project



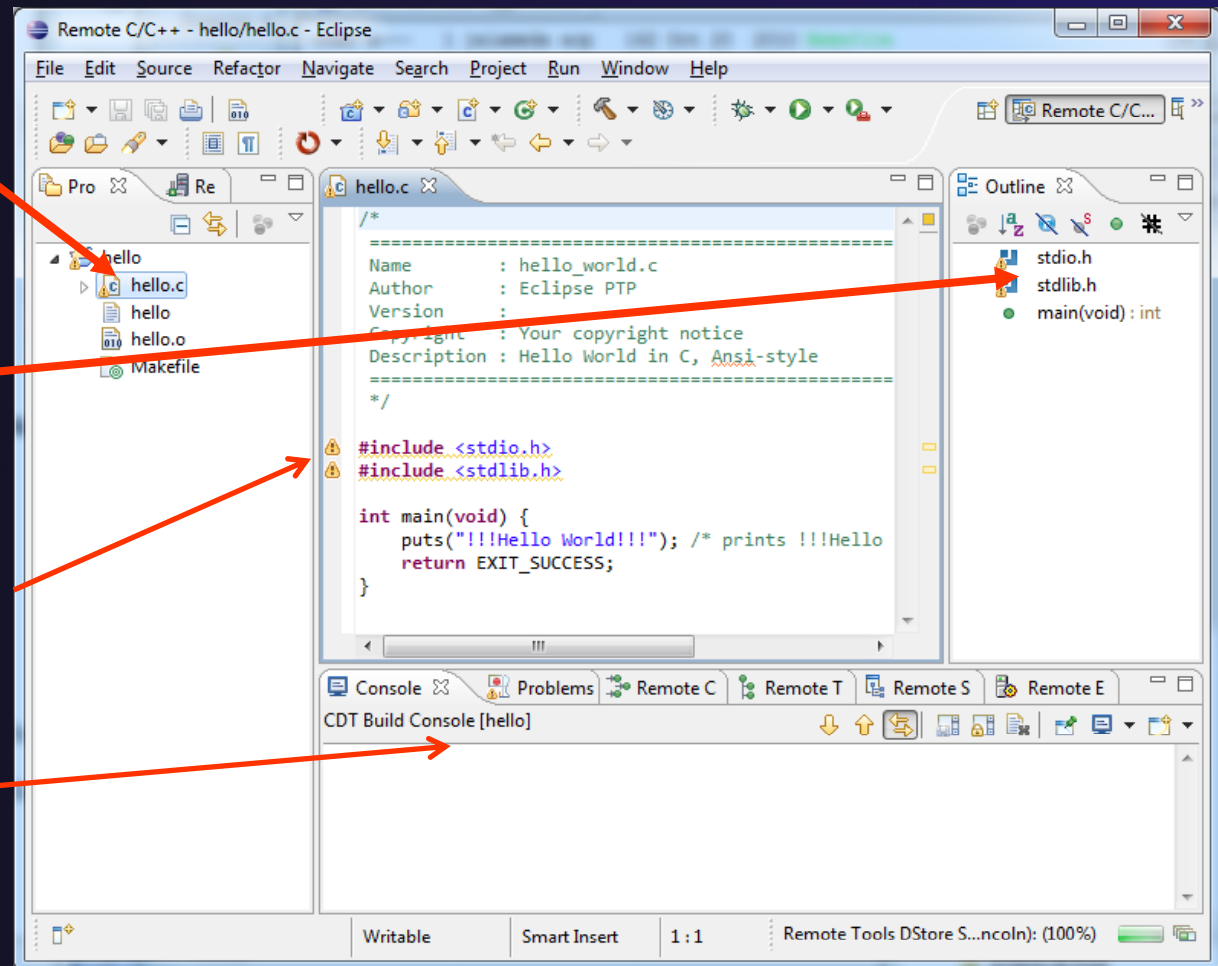
- ★ The **New Project Wizard** is used to create a C project
- ★ Enter **Project name**
- ★ Under **Project Types**, select **Makefile project ▶ Empty Project**
 - ★ Ensures that CDT will use existing makefiles
- ★ Select **Finish**
- ★ When prompted to switch to the **C/C++ Perspective**, select **Yes**



Editor and Outline View

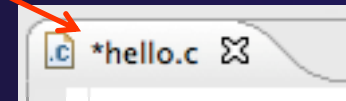
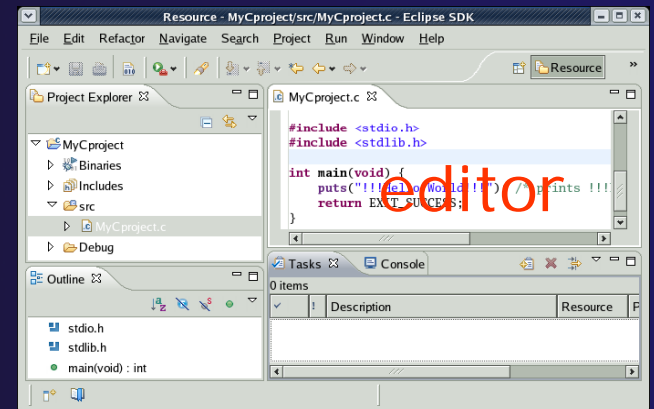


- ★ Double-click on source file to open editor
- ★ Outline view is shown for file in editor
- ★ You should see warnings on the include files: we will fix this later
- ★ Console shows results of build



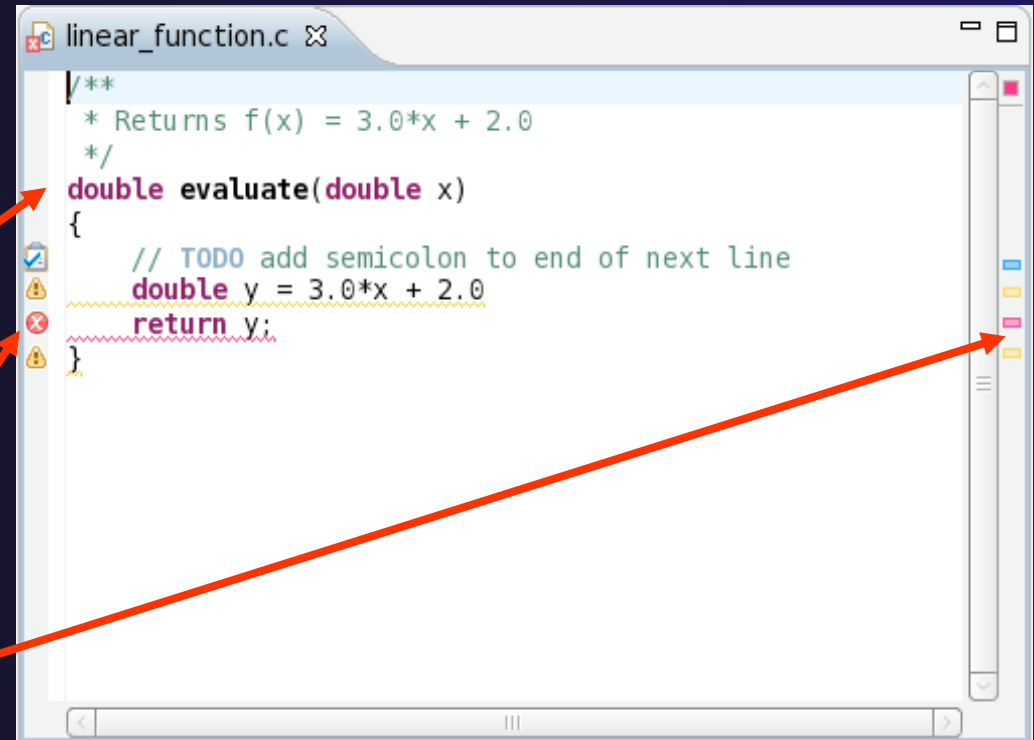
Editors

- ✦ 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
- ✦ Save the changes by using Command/Ctrl-S or **File>Save**

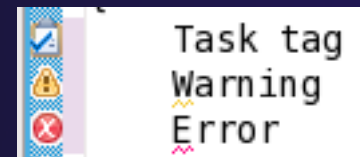


Source Code Editors & Markers

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

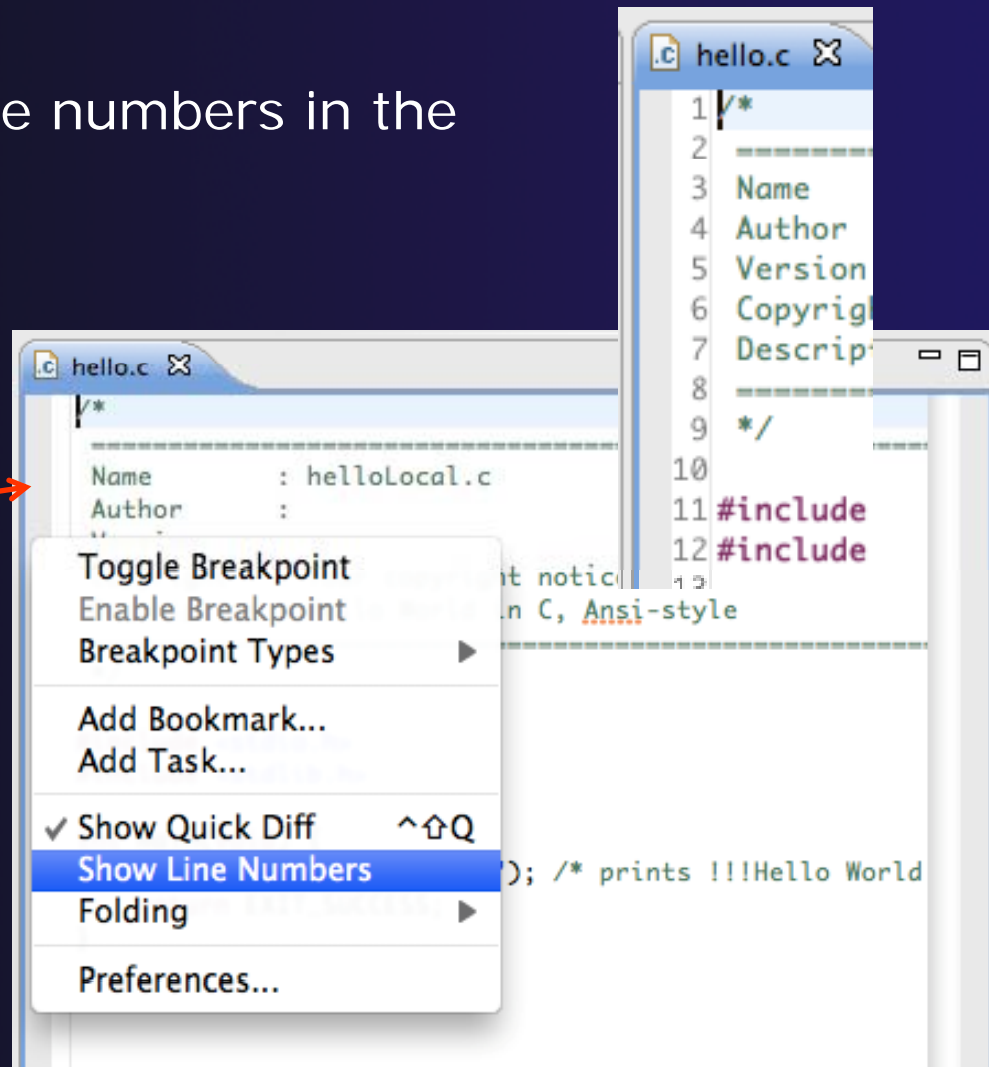


Icons:



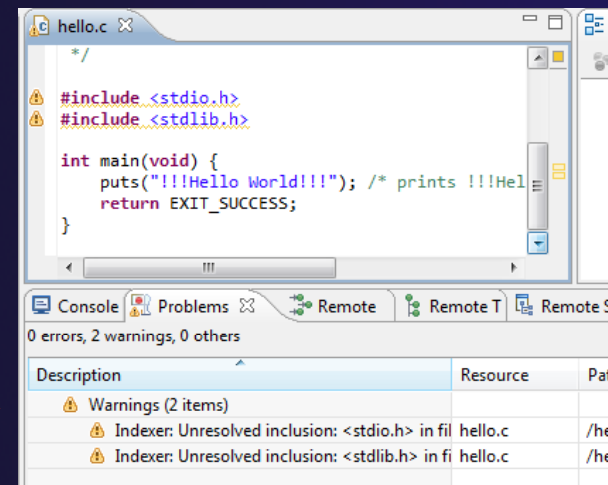
Line Numbers

- ★ 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**



Include File Locations

- ★ Content assist and navigation requires knowledge of include file location on the remote system
- ★ The editor will indicate warnings on lines that have the problem
- ★ **Problems View** will display a warning
- ★ The project properties must be changed to resolve the problem

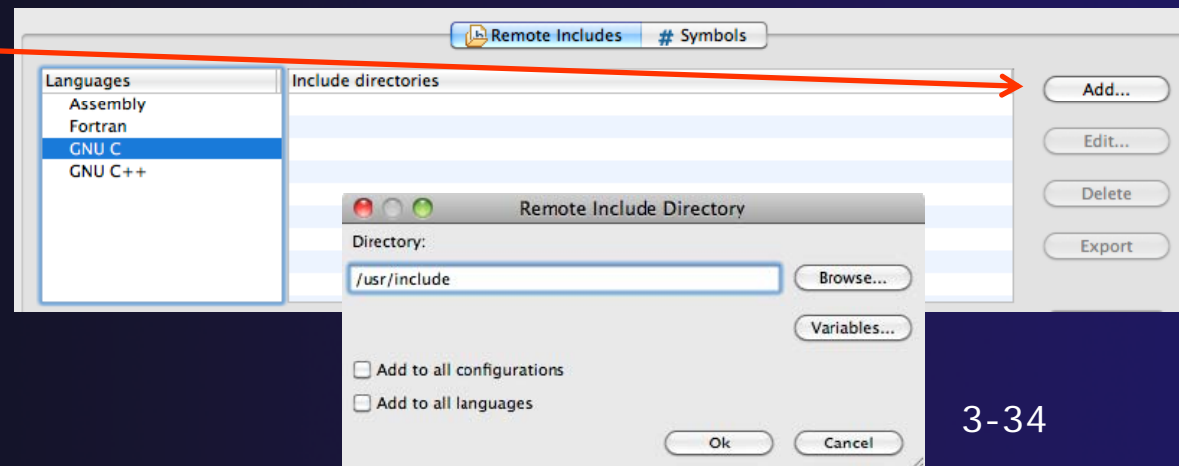
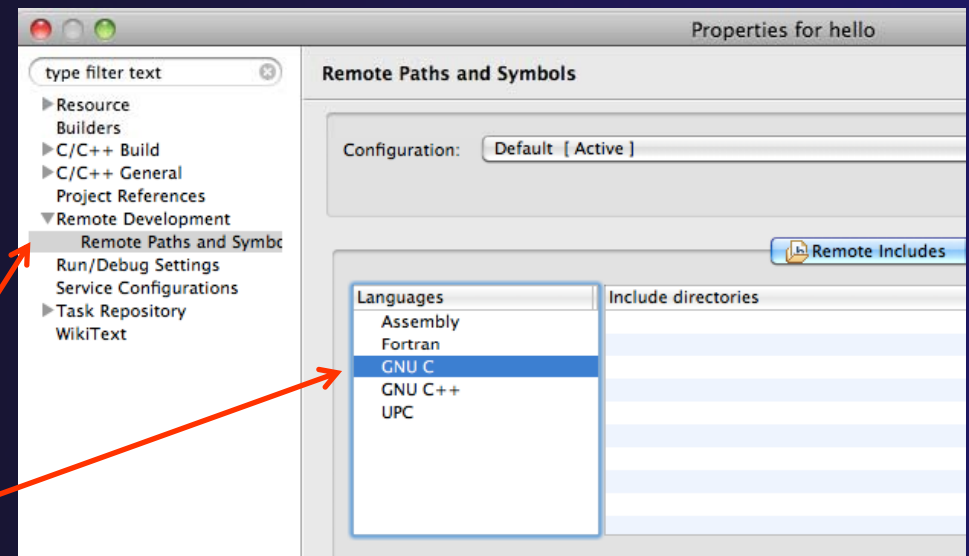


Indexer: Unresolved inclusion: <stdio.h> in file: /u/ac/etrain1/hello/hello.c:11. Please re-configure project's remote include paths or symbols.



Changing the Project Properties

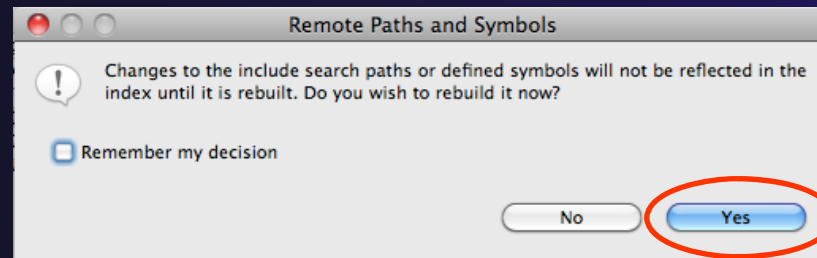
- ✦ Open the project properties by right-clicking on project and select **Properties**
- ✦ Expand **Remote Development**
- ✦ Select **Remote Paths and Symbols**
- ✦ Select **GNU C** to change C paths and symbols
- ✦ Click **Add**
- ✦ Enter `"/usr/include"`
- ✦ Click **OK**





Saving the Project Properties

- ★ Click **OK** to save the Project Properties
- ★ You will be prompted to rebuild the index
 - ★ Select **Yes**



- ★ Red warnings should be gone from editor, since Eclipse knows the location of the include files now

```
hello.c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    puts("!!!Hello World!!!"); /* prints !!!Hello World!!! */
    return EXIT_SUCCESS;
}
```




Navigating to Other Files

- ★ On demand hyperlink
 - ★ Hold down Command/Ctrl key
 - ★ Click on element to navigate to its definition in the header file (Exact key combination depends on your OS)
 - ★ E.g. Command/Ctrl and click on EXIT_SUCCESS
- ★ Open declaration
 - ★ Right-click and select **Open Declaration** will also open the file in which the element is declared
 - ★ E.g. right-click on `stdio.h` and select **Open Declaration**

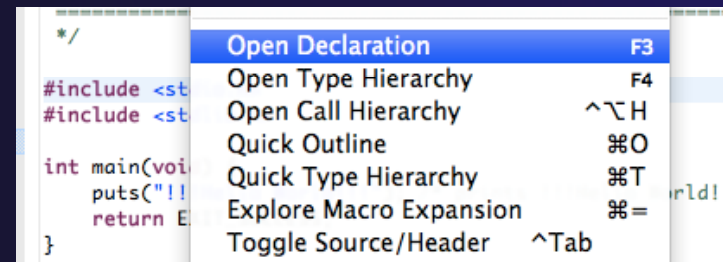
```

hello.c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    puts("!!!Hello World!!!"); /* prints !!!Hello World!!! */
    return EXIT_SUCCESS;
}
  
```

```

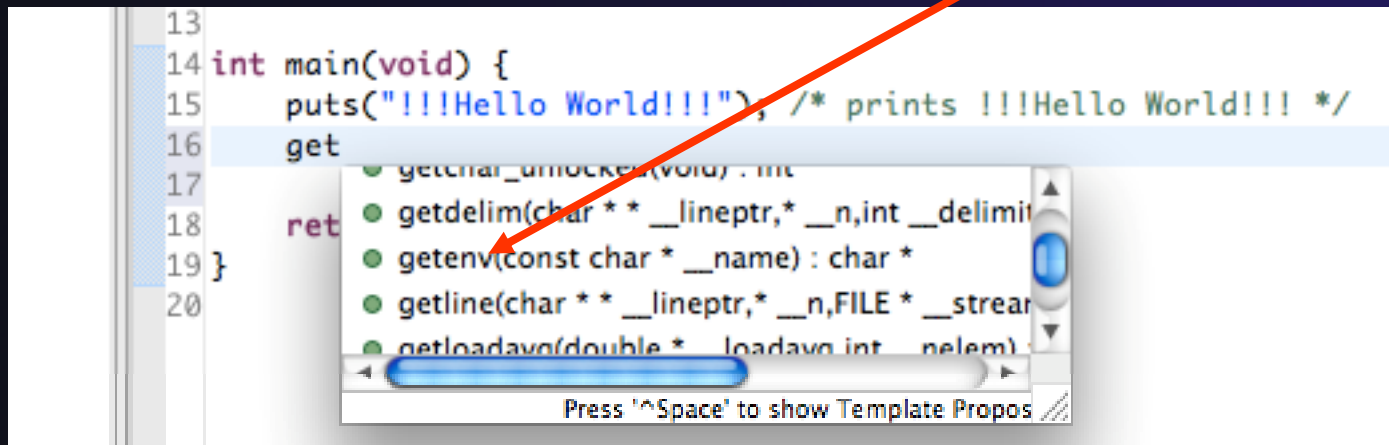
hello.c  stdlib.h
/* We define these the same for all machines.
   Changes from this to the outside world should be done in
#define EXIT_FAILURE 1 /* Failing exit status. */
#define EXIT_SUCCESS 0 /* Successful exit status. */
  
```





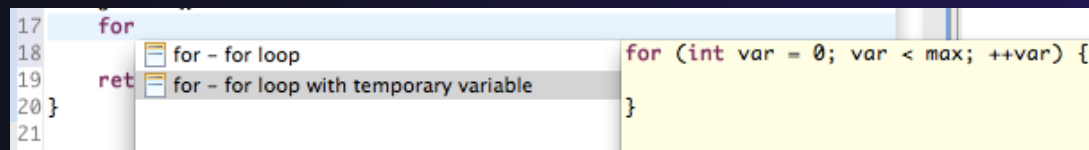
Content Assist & Templates

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




- ✦ Code Templates: type 'for' and Ctrl-space

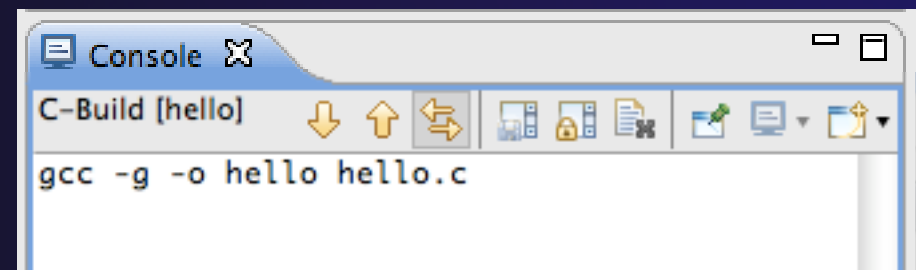
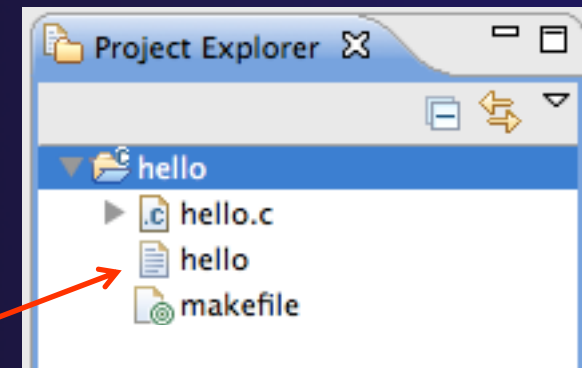
Hit ctrl-space again
for code templates





Building the Project

- ★ The project should build automatically when created
- ★ If there is no makefile, then the build will fail
- ★ To manually build, select the project and press the "build" button 
- ★ Alternatively, select **Project > Build Project**
- ★ The executable should appear in the project
- ★ The **Console** view shows build output



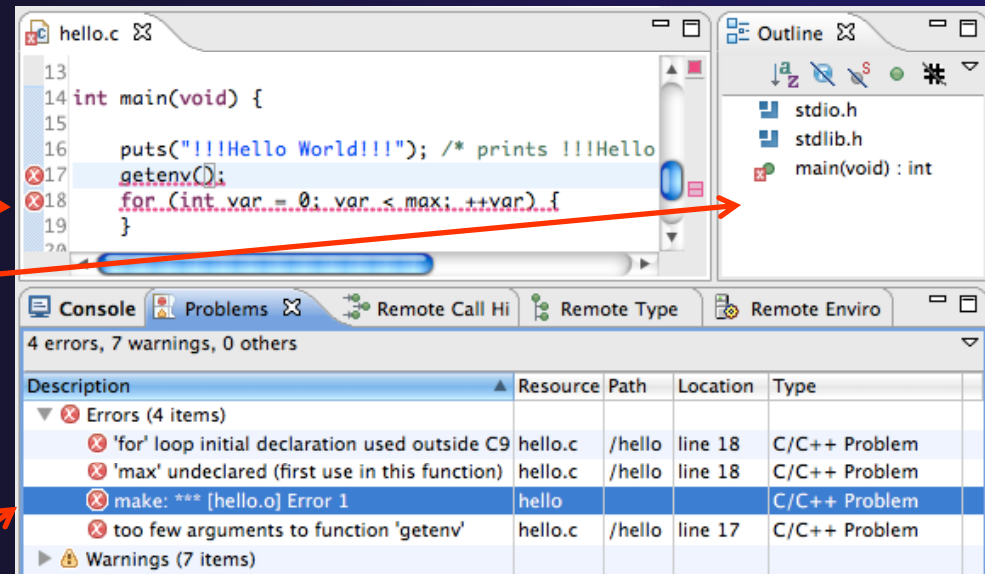
Executable
'hello'

Build Problems

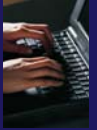


- ★ If there are problems, they will be shown in a variety of ways


- ★ Marker on editor line
- ★ Marker on overview ruler
- ★ Listed in the **Problems view**

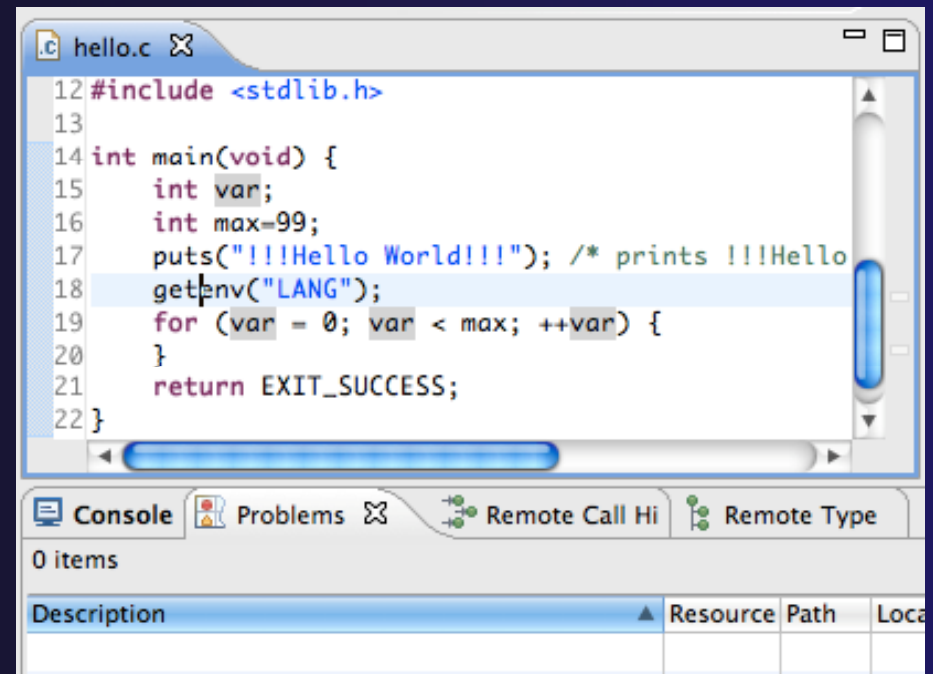


- ★ Double-click on line in **Problems view** to go to location of error



Fix Build Problems

- ★ Fix errors by giving **getenv** an argument and fixing declarations as shown
- ★ Save the file
- ★ Rebuild by pressing build button 
- ★ **Problems view** is now empty



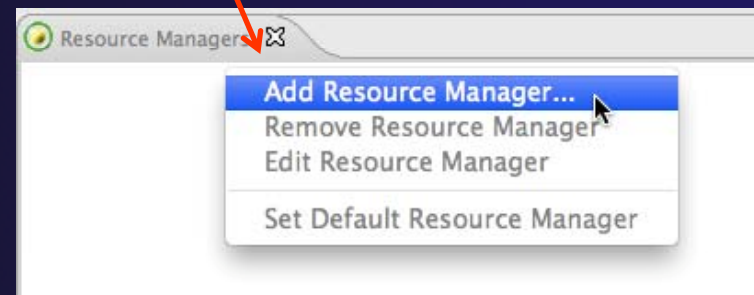
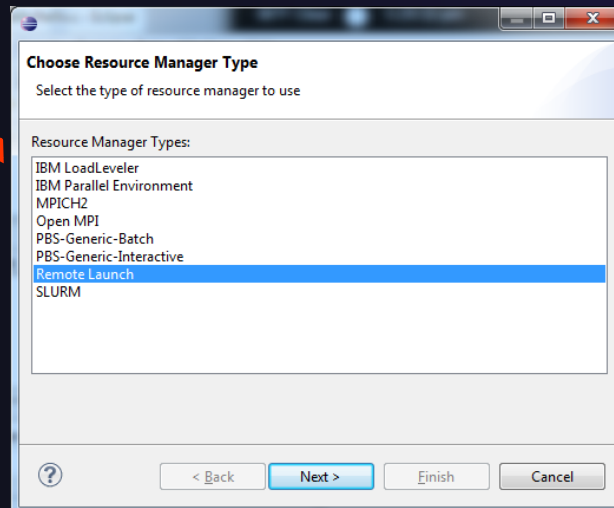
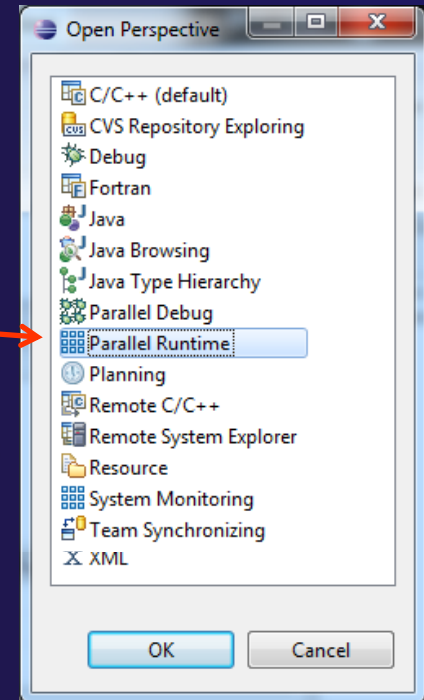
```
12 #include <stdlib.h>
13
14 int main(void) {
15     int var;
16     int max=99;
17     puts("!!!Hello World!!!"); /* prints !!!Hello
18     getenv("LANG");
19     for (var = 0; var < max; ++var) {
20     }
21     return EXIT_SUCCESS;
22 }
```

The screenshot shows a code editor window titled 'hello.c' with the above code. Below the editor is a toolbar with 'Console', 'Problems', 'Remote Call Hi', and 'Remote Type' buttons. The 'Problems' view is active and shows '0 items'. Below this is a table with columns 'Description', 'Resource', 'Path', and 'Local'.

Description	Resource	Path	Local

Create a Resource Manager

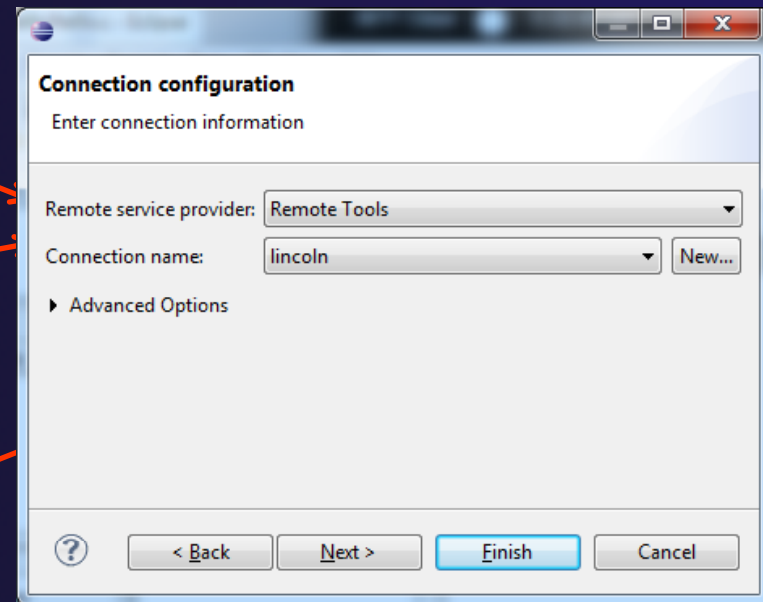
- ★ A *Resource Manager* specifies how/where programs will be launched
- ★ Switch to the **Parallel Runtime** perspective
 - ★ **Window > Open Perspective...**
- ★ In the **Resource Managers** view, right-click and select **Add Resource Manager...**
- ★ Select **Remote Launch** and **Next >**



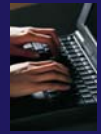
Configure the Resource Manager



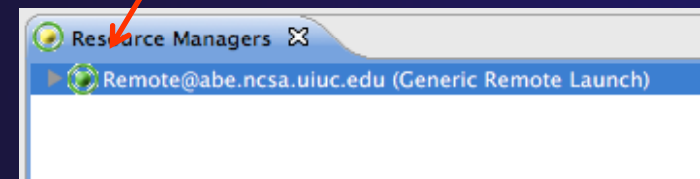
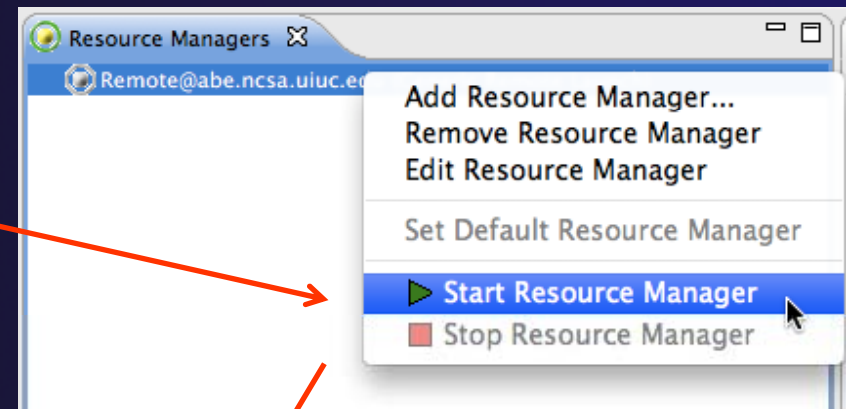
- ★ Choose **Remote Tools** for **Remote service provider**
- ★ Choose "abe.ncsa.uiuc.edu" for **Connection name**
 - ★ This was the connection used when the project was created
- ★ Click **Finish**



Start the Resource Manager



- ★ Right-click on the new resource manager and select **Start Resource Manager** from the menu
- ★ If the resource manager starts successfully, the icon should turn green
- ★ An icon color of red indicates a problem occurred



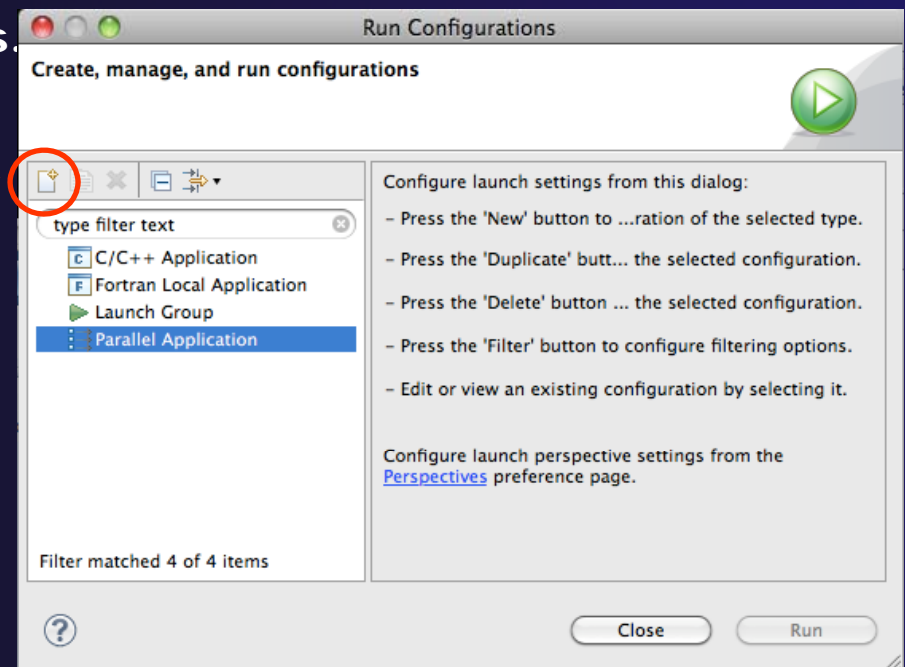
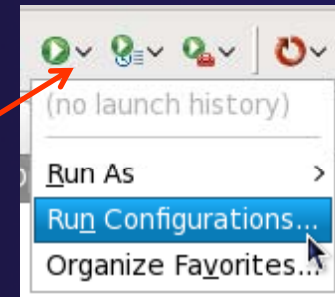
```
greg@localhost:~  
File Edit View Terminal Help  
[greg@localhost ~]$ eclipse/eclipse  
Kerberos username [greg]:  
Kerberos password for greg:  
█
```

NOTE: On some Linux systems, starting a resource manager may appear to hang. Open the window you launched Eclipse from and check if there is a prompt for a kerberos username. Hit "enter" twice if you see the prompt.

Create a Run Configuration

To run the application, create a Run Configuration

- ✦ Open the run configurations dialog
 - ✦ Click on the arrow next to the run button
 - ✦ Or use **Run>Run Configurations...**
- ✦ Select **Parallel Application**
- ✦ Select the **New** button

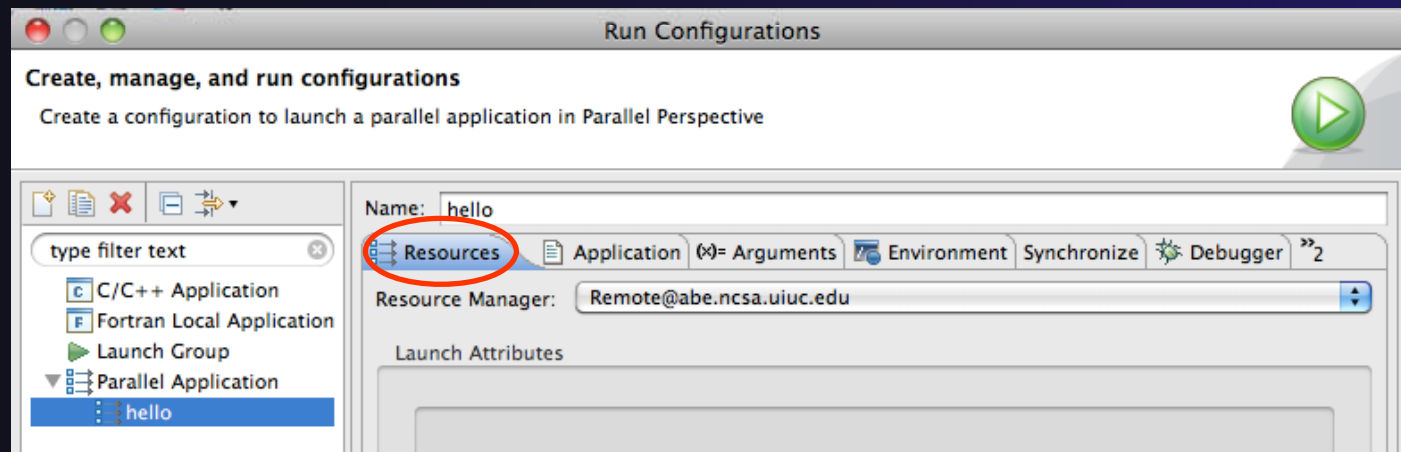


Depending on which flavor of Eclipse you installed, you might have more choices of application types



Complete the Resources Tab

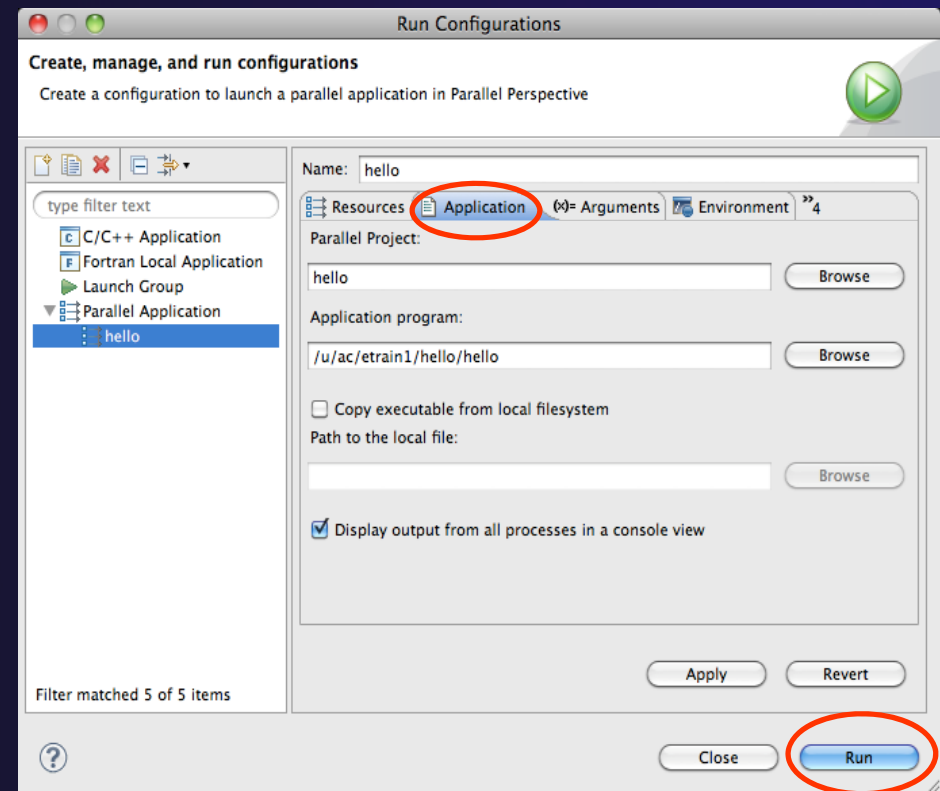
- ★ Select your Resource Manager
 - ★ Should be selected automatically if it has been started
- ★ The Remote Launch doesn't require additional attributes
 - ★ Other resource managers may have additional attributes, such as a queue name, etc.



Complete the Application Tab



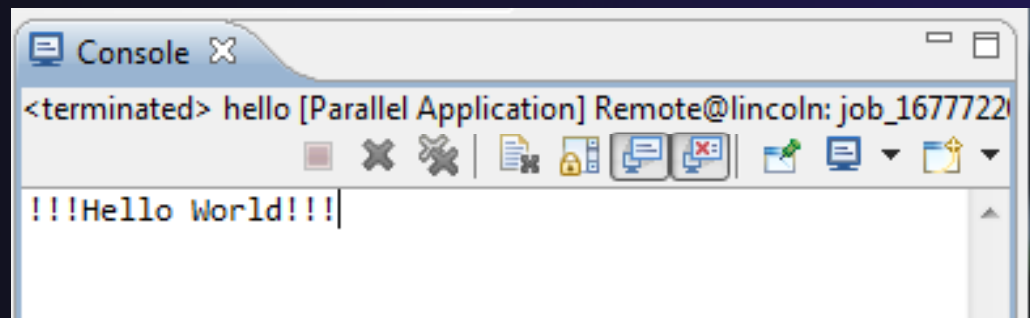
- ★ Make sure "hello" is selected for the **Parallel Project**
- ★ Browse to find the executable file for the **Application program**
- ★ Launch the application by clicking the **Run** button





Viewing Program Output

- ★ When the program runs, the **Console** view should automatically become active
- ★ Any output will be displayed in this view
 - ★ Stdout is shown in black
 - ★ Stderr is shown in red

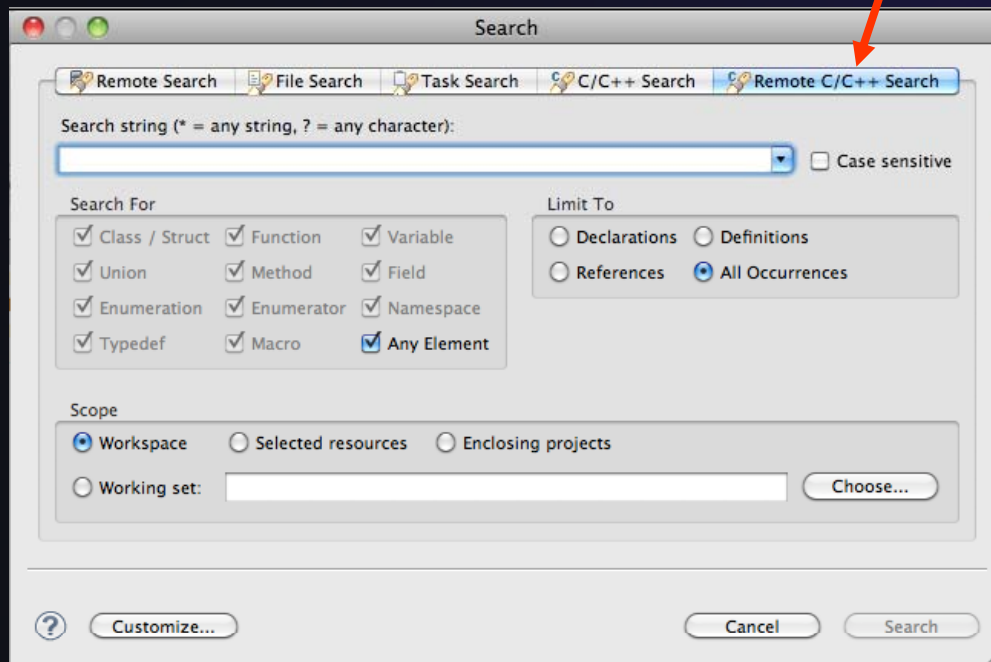
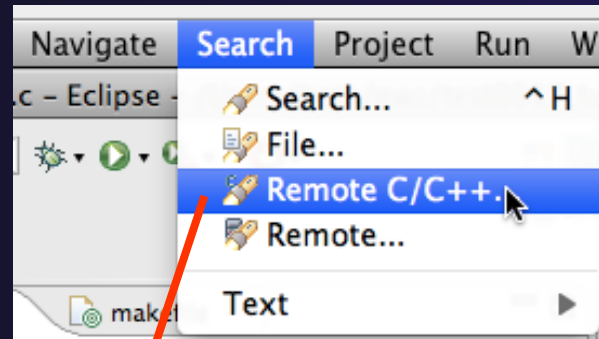


Other CDT features

- ✦ Searching
- ✦ Mark Occurrences
- ✦ Open Declaration / hyperlinking between files in the editor

First, return to the "Remote C/C++
Perspective"

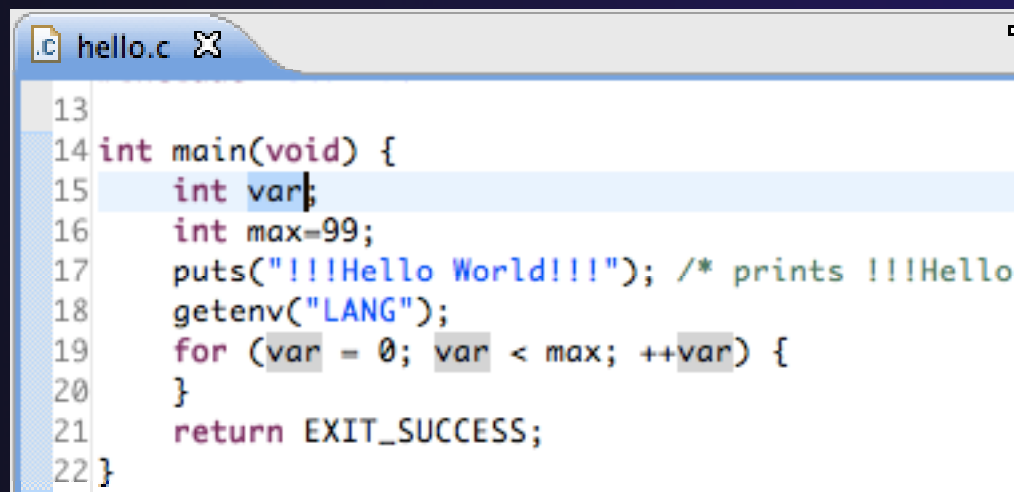
Language-Based Searching



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

Mark Occurrences

- ✦ 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



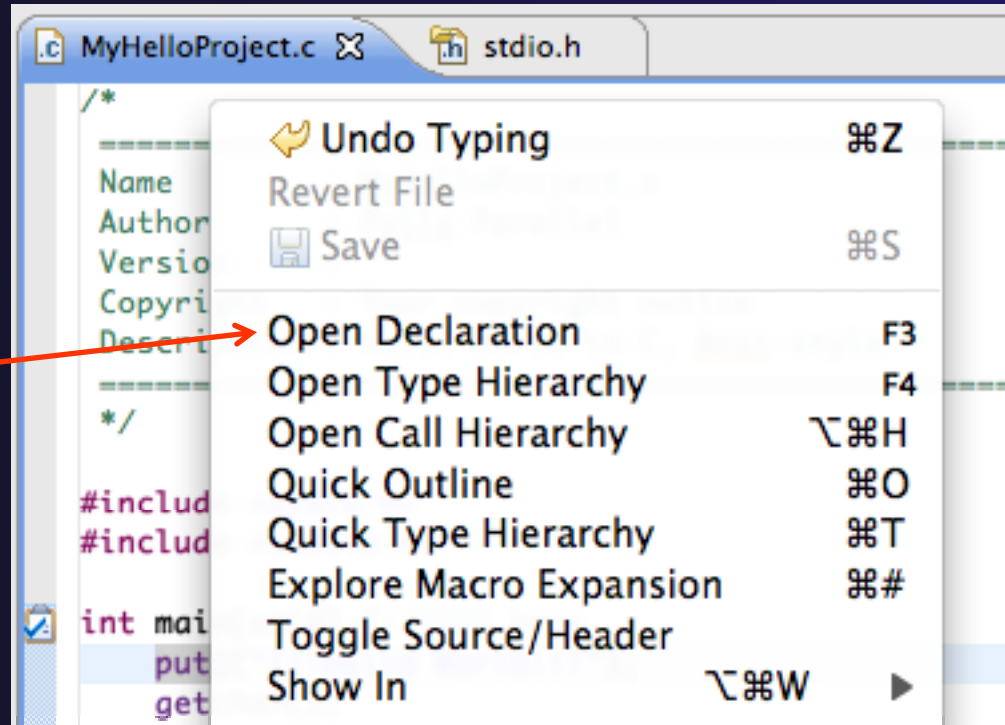
The screenshot shows a CDT editor window titled 'hello.c'. The code is as follows:

```
13
14 int main(void) {
15     int var;
16     int max=99;
17     puts("!!!Hello World!!!"); /* prints !!!Hello
18     getenv("LANG");
19     for (var = 0; var < max; ++var) {
20     }
21     return EXIT_SUCCESS;
22 }
```

In this image, the occurrences of the variable 'var' are highlighted with a light blue background. The highlights are on line 15 ('int var;'), line 19 ('for (var = 0; var < max; ++var)'), and line 20 (the closing brace of the for loop).

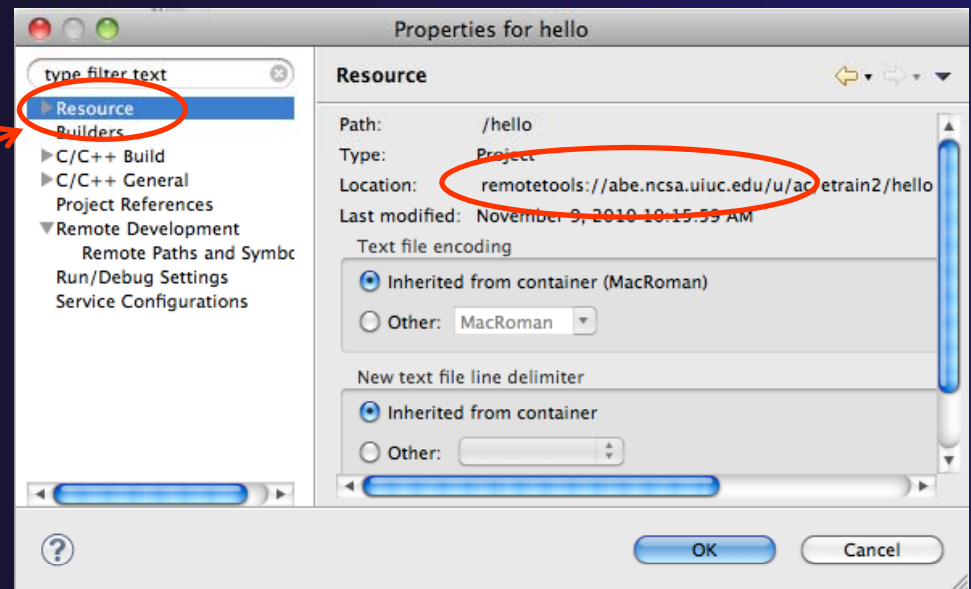
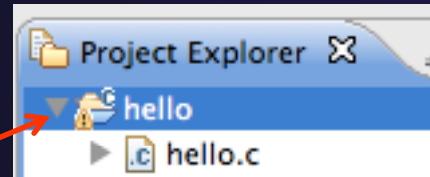
Open Declaration

- ★ 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**
- ★ Can also Ctrl-click (Mac: Cmd-click) on an identifier to “hyperlink” to its declaration



Remote Projects - Location

- ★ How to tell where a project resides?
- ★ Right-click Project
- ★ Select **Properties...**
- ★ In Properties dialog, select **Resource**



Remote Projects - Reopening

- ★ When re-opening Eclipse workbench, remote projects will be closed
- ★ To re-open a closed project, Right-click on closed project and select **Open Project**
- ★ Open project shows folder icon, and can be expanded to show contents of project

