Module 3: Working with C/C++

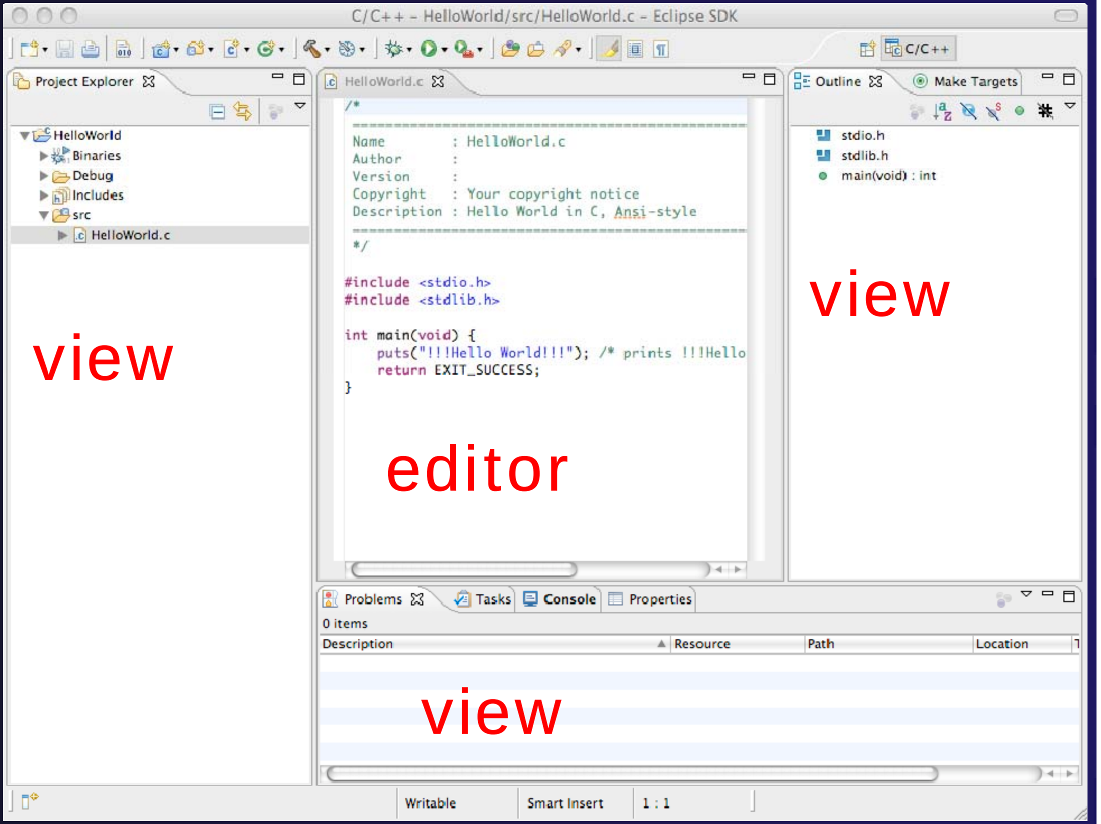
- Objective
  - Learn basic Eclipse concepts: Perspectives, Views, ...
  - Learn how to use Eclipse to manage a remote project
  - Learn how to use Eclipse to develop C programs
  - Learn how to launch and run a remote C program
- Contents
  - Brief introduction to the C/C++ Development Tools (CDT)
  - 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



```
/*
 ==========================================================
 Name        : HelloWorld.c
 Author      :
 Version     :
 Copyright   : Your copyright notice
 Description : Hello World in C, Ansi-style
 ==========================================================
 */

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

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

perspective

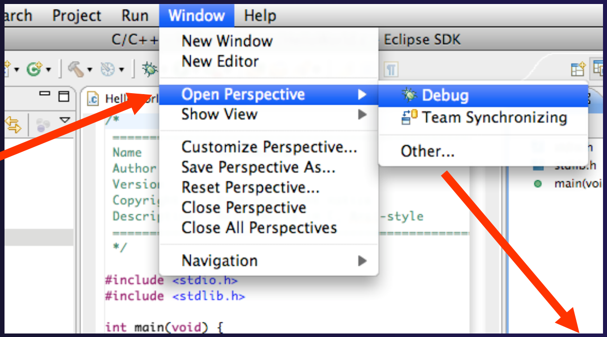
Perspectives

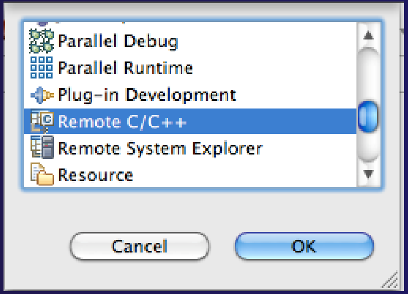
- Perspectives define the layout of views and editors in the workbench
- They are *task oriented*, i.e. they contain specific views for doing certain tasks:
  - There is a **Resource Perspective** for manipulating resources
  - **C/C++ Perspective** for manipulating compiled code
  - **Debug Perspective**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

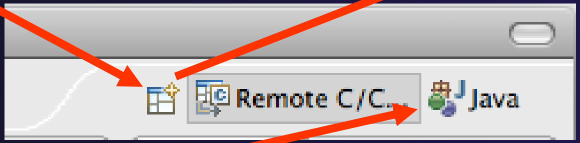
- If you are on the Welcome screen now, select "Go to Workbench" now

Switching Perspectives

- Three ways of changing perspectives
  - 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…

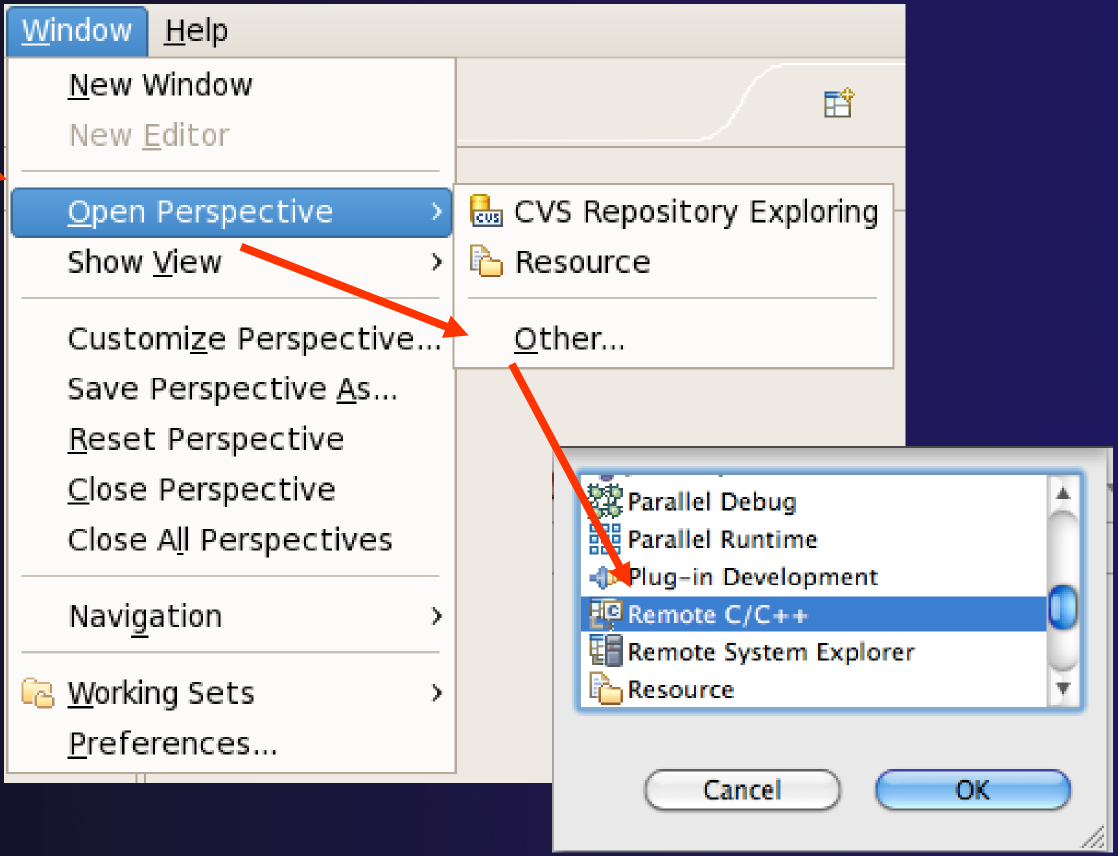




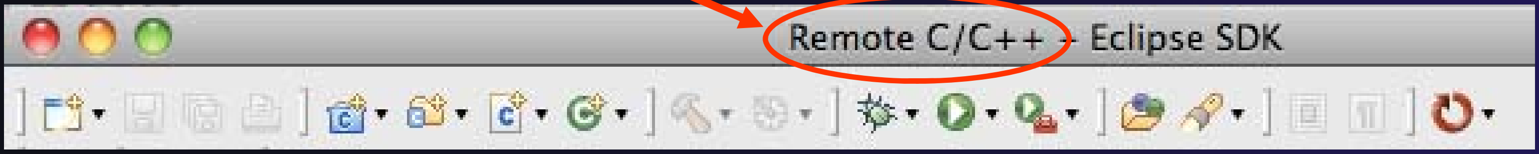


Switch to Remote C/C++ Perspective

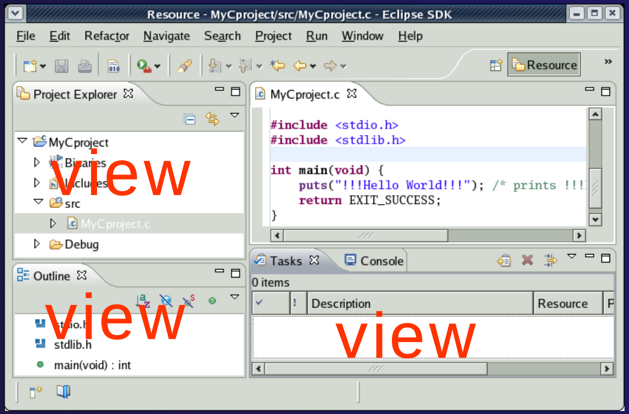
- Select **Window>Open Perspective**
- Then choose **Other…**
- Only needed if you're not already in the perspective



- What Perspective am in in?
  See title Bar

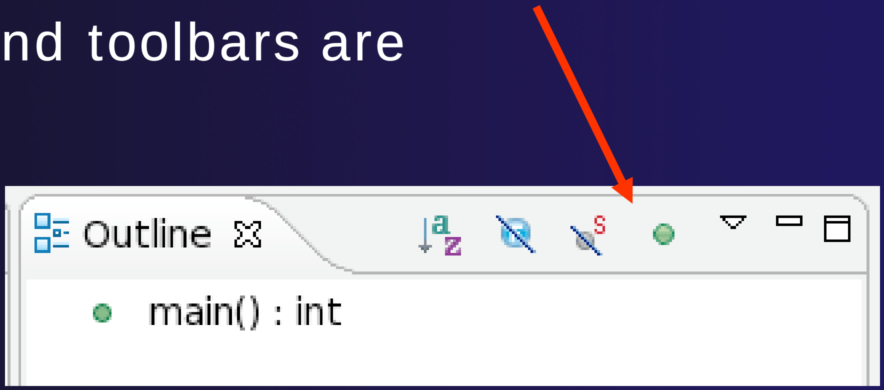


Views



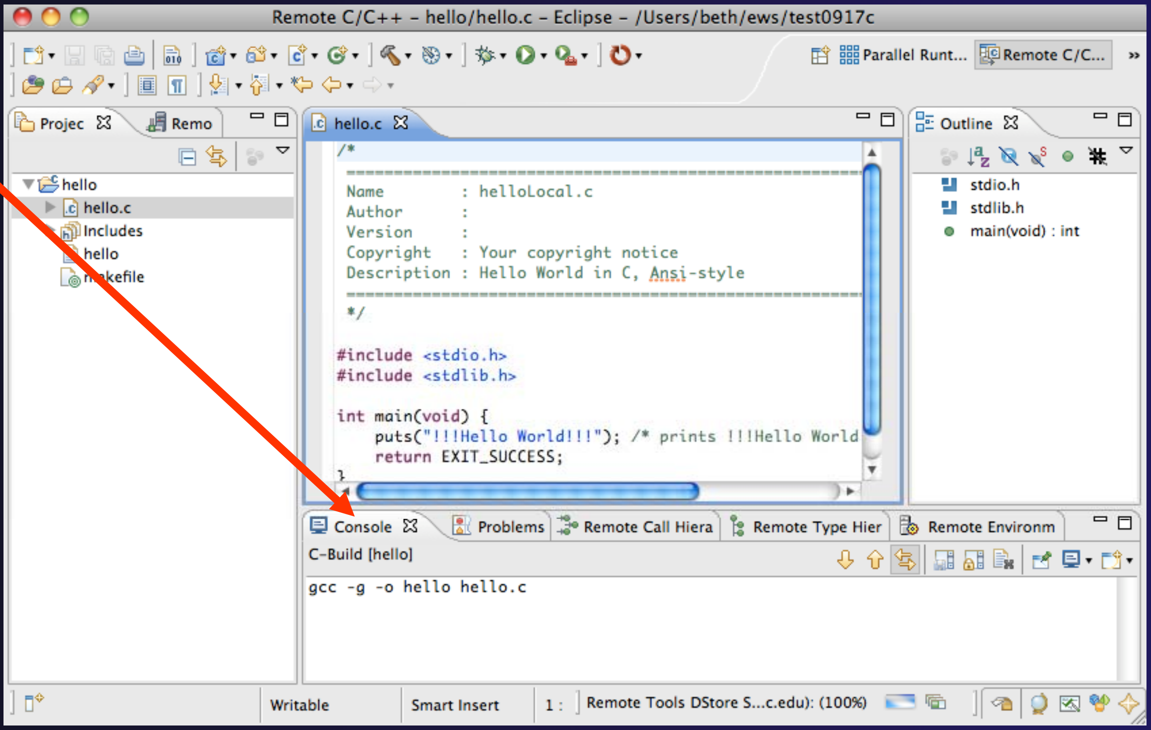
```
#include <stdio.h>
#include <stdlib.h>

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

- 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

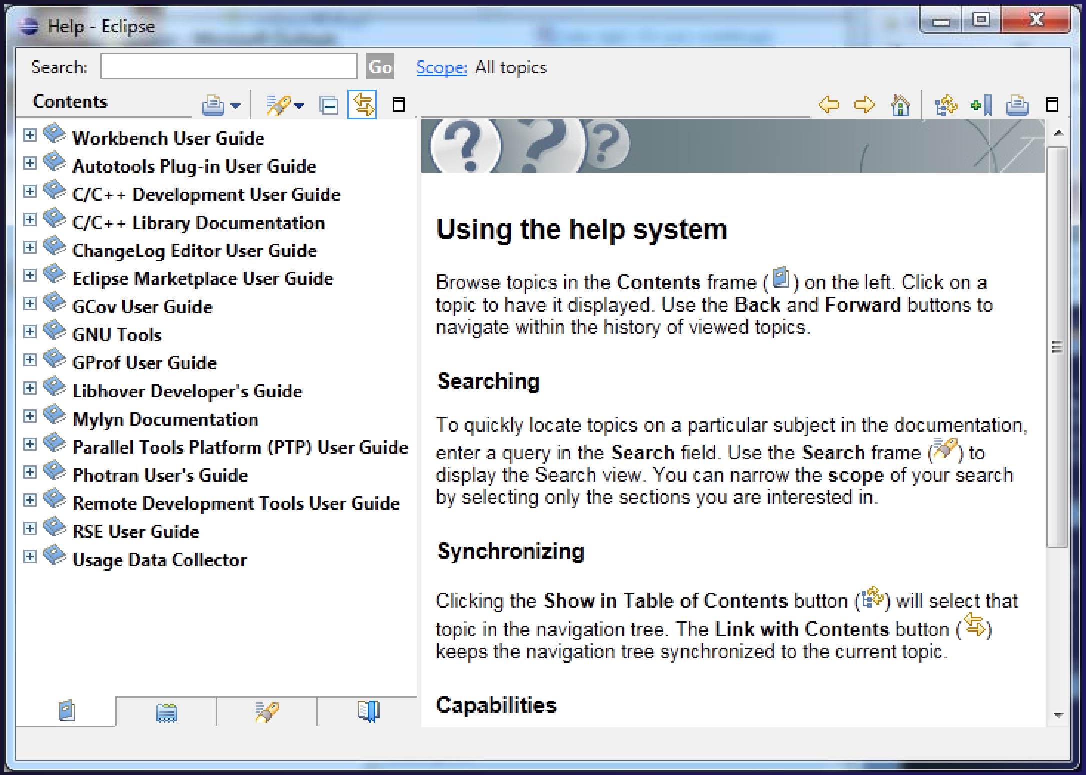


```
/*
 ==========================================================================
 Name        : helloLocal.c
 Author      :
 Version     :
 Copyright   : Your copyright notice
 Description : Hello World in C, Ansi-style
 ==========================================================================
 */

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

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

```
C-Build [hello]
gcc -g -o hello hello.c
```

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.)


Help - Eclipse
Search:
Go
Scope: All topics
Contents
Workbench User Guide
Autotools Plug-in User Guide
C/C++ Development User Guide
C/C++ Library Documentation
ChangeLog Editor User Guide
Eclipse Marketplace User Guide
GCov User Guide
GNU Tools
GProf User Guide
Libhover Developer's Guide
Mylyn Documentation
Parallel Tools Platform (PTP) User Guide
Photran User's Guide
Remote Development Tools User Guide
RSE User Guide
Usage Data Collector

Using the help system

Browse topics in the **Contents** frame ( ) on the left. Click on a topic to have it displayed. Use the **Back** and **Forward** buttons to navigate within the history of viewed topics.

Searching

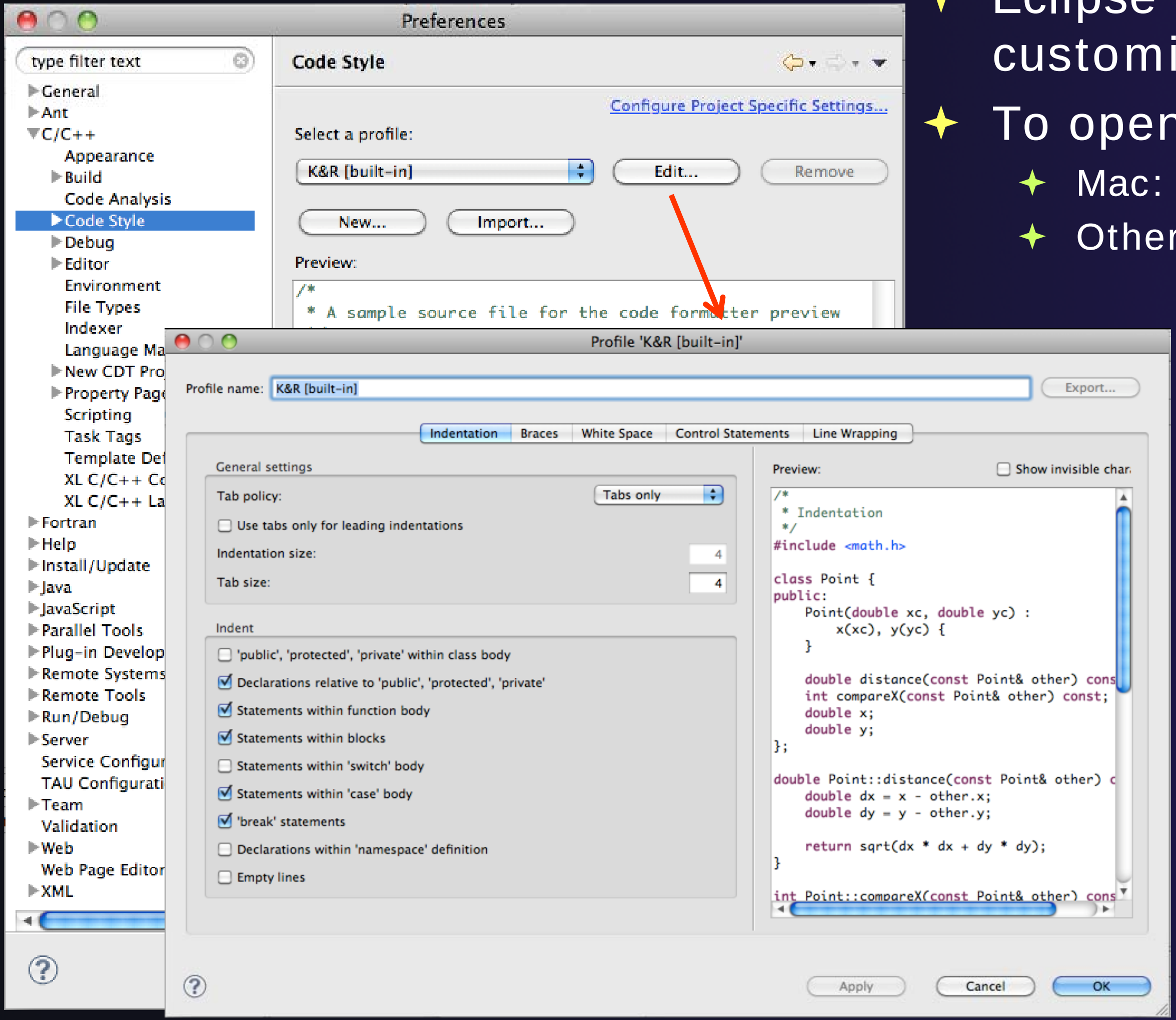
To quickly locate topics on a particular subject in the documentation, enter a query in the **Search** field. Use the **Search** frame ( ) to display the Search view. You can narrow the **scope** of your search by selecting only the sections you are interested in.

Synchronizing

Clicking the **Show in Table of Contents** button ( ) will select that topic in the navigation tree. The **Link with Contents** button ( ) keeps the navigation tree synchronized to the current topic.

Capabilities

Preferences



Select a profile: K&R [built-in]

Preview:

```
/*
 * A sample source file for the code formatter preview
```

Profile name: K&R [built-in]

General settings

Tab policy: Tabs only

- [ ] Use tabs only for leading indentations

Indentation size: 4

Tab size: 4

Indent

- [ ] 'public', 'protected', 'private' within class body
- [x] Declarations relative to 'public', 'protected', 'private'
- [x] Statements within function body
- [x] Statements within blocks
- [ ] Statements within 'switch' body
- [x] Statements within 'case' body
- [x] 'break' statements
- [ ] Declarations within 'namespace' definition
- [ ] Empty lines

Preview: [ ] Show invisible char.

```
/*
 * Indentation
 */
#include <math.h>

class Point {
public:
	Point(double xc, double yc) :
		x(xc), y(yc) {
	}

	double distance(const Point& other) cons
	int compareX(const Point& other) const;
	double x;
	double y;
};

double Point::distance(const Point& other) c
	double dx = x - other.x;
	double dy = y - other.y;

	return sqrt(dx * dx + dy * dy);
}

int Point::compareX(const Point& other) cons
```

- 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

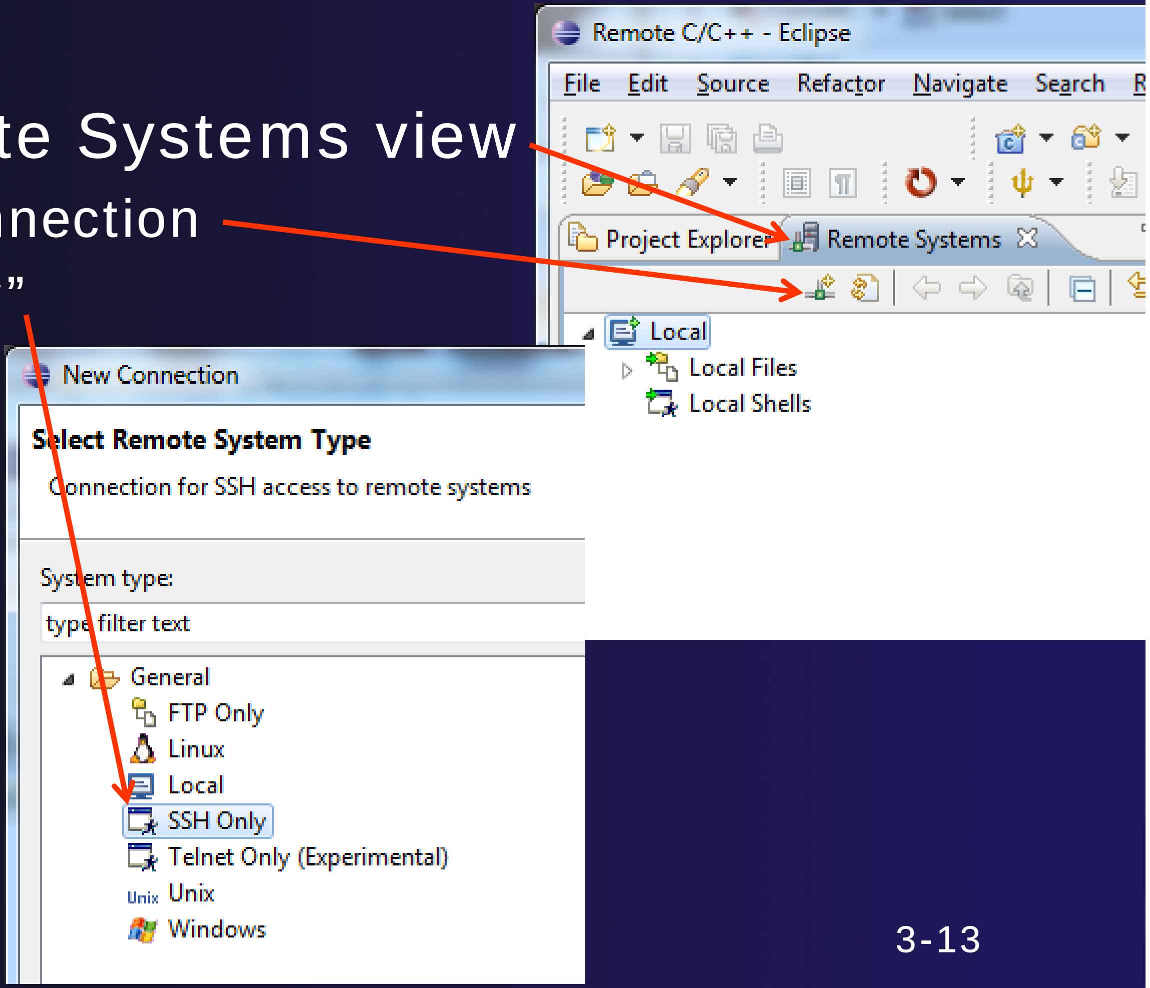
- "Traditional" Remote Projects
- Source is located on remote machine
- Eclipse is installed on the local machine and can be used for:
  - Editing
  - Building
  - Running
  - Debugging
- Source indexing is performed on remote machine
  - Enables call hierarchy, type hierarchy, include browser, search, 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

Preparation steps:

- Make sure you are in the Remote C/C++ perspective
- Select the Remote Systems view
  - Define a new connection
  - Select “SSH Only”
  - Then **Next**



Select Remote System Type

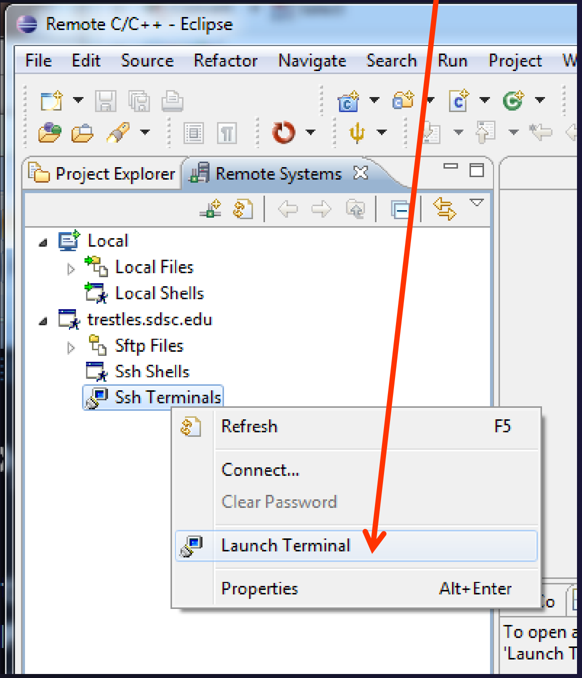
Connection for SSH access to remote systems

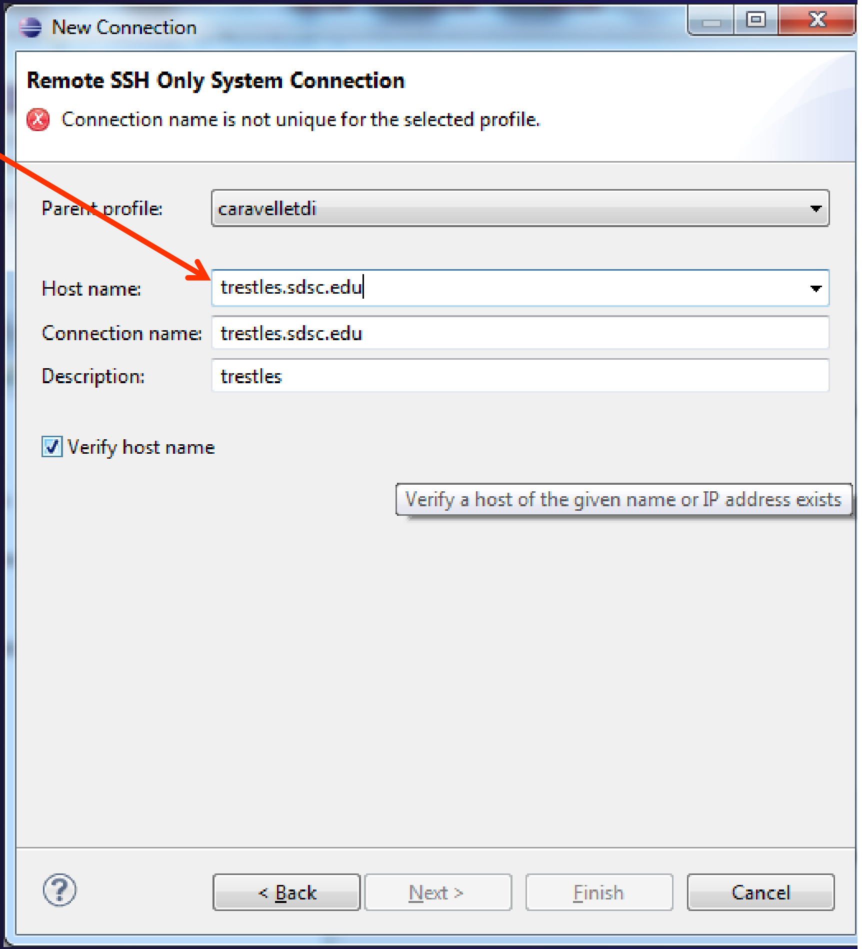
System type:

type filter text

Preparation, continued

- Add trestle's host info
  - Then **Finish**
- Right click on ssh terminal, under trestles




New Connection

Remote SSH Only System Connection

Connection name is not unique for the selected profile.

Parent profile: caravelletdi

Host name: trestles.sdsc.edu

Connection name: trestles.sdsc.edu

Description: trestles

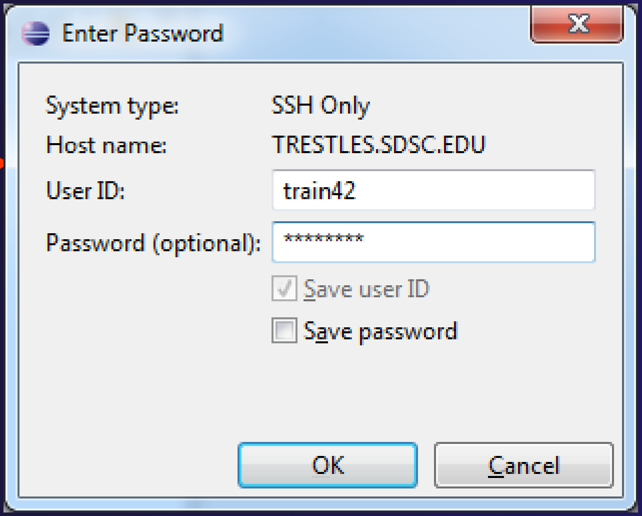
[✓] Verify host name

Verify a host of the given name or IP address exists

< Back | Next > | Finish | Cancel

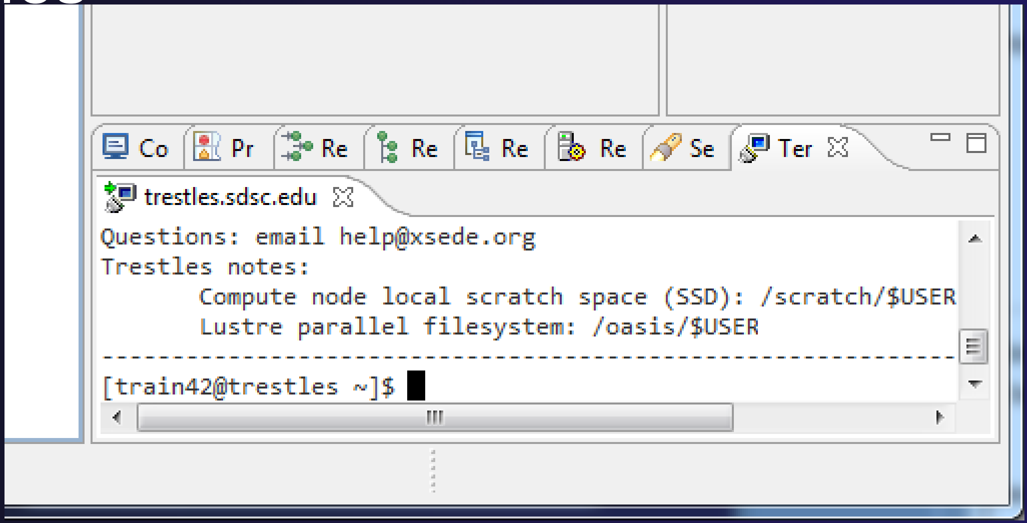
Preparation, continued

- Add your training account login
- Click through any RSA messages
- And now you have a terminal to trestles



System type:	SSH Only
Host name:	TRESTLES.SDSC.EDU
User ID:	train42
Password (optional):	********

[✓] Save user ID
[ ] Save password



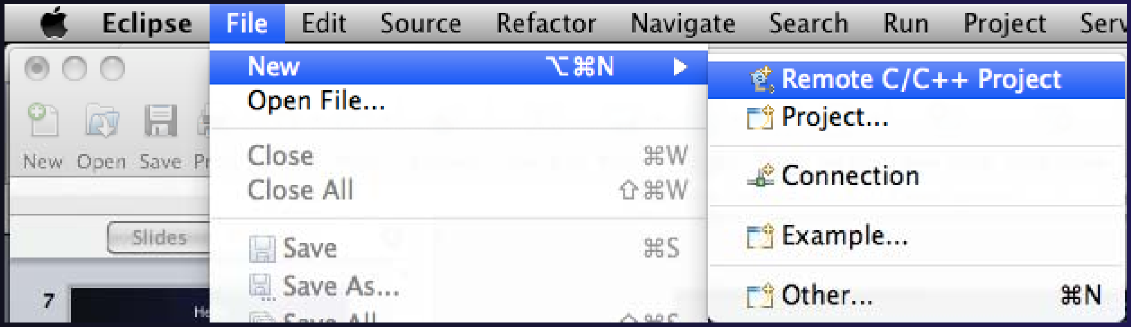
```
Questions: email help@xsede.org
Trestles notes:
        Compute node local scratch space (SSD): /scratch/$USER
        Lustre parallel filesystem: /oasis/$USER
------------------------------------------------------------
[train42@trestles ~]$
```

Why did we do this?

- To show you can gain “traditional” access to a remote host through Eclipse
- And to have you stage some directories:
- Issue the following commands in the terminal
  - cp –r ~ux400689/hello_world .
  - cp –r ~ux400689/shallow .
  - cp –r ~ux400689/mpi .
- This will give us some source code to work with

Creating a Remote C/C++ Project

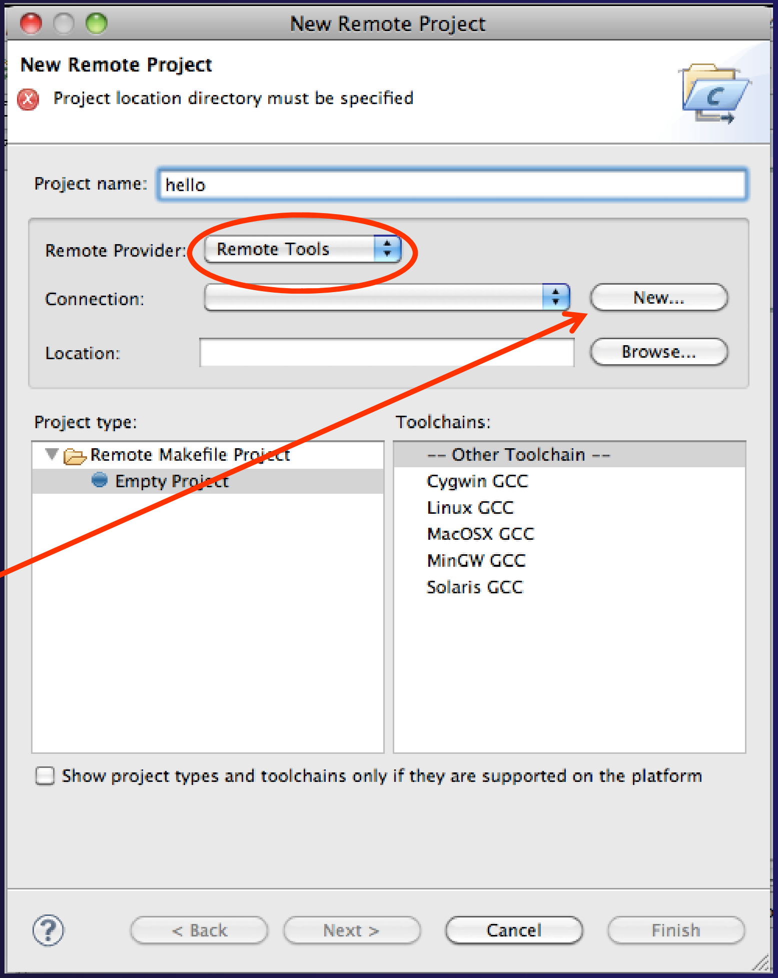
- Use **File>New>Remote C/C++ Project** to open the new project wizard
- The wizard will take you through the steps for creating the project



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



New Remote Project

New Remote Project

Project location directory must be specified

Project name: hello

Remote Provider: Remote Tools

Connection:

Location:

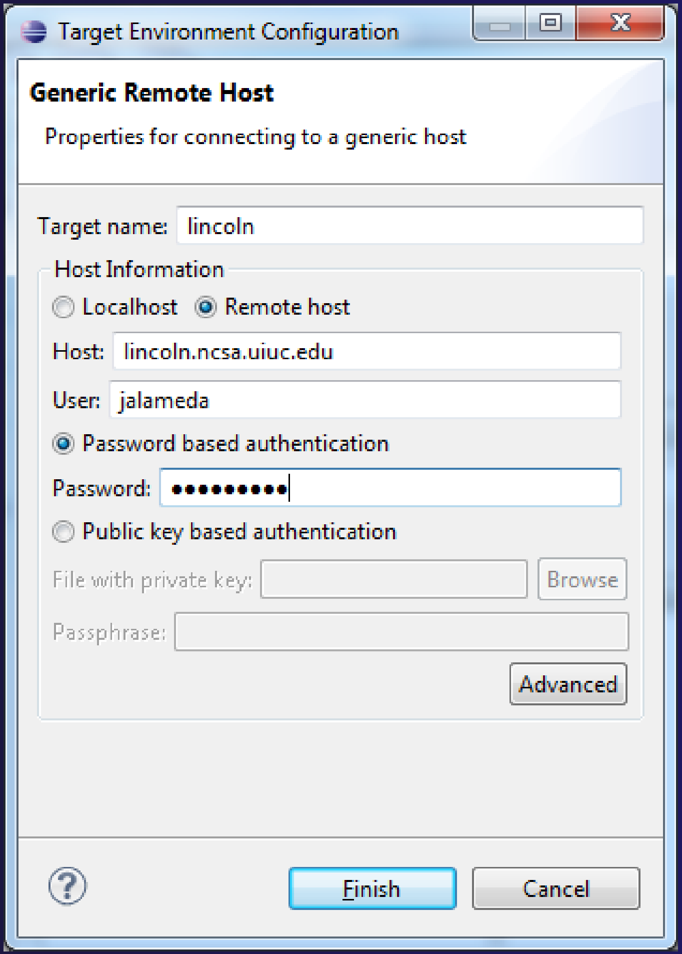
Project type:
- Remote Makefile Project
  - Empty Project

Toolchains:
- -- Other Toolchain --
- Cygwin GCC
- Linux GCC
- MacOSX GCC
- MinGW GCC
- Solaris GCC

☐ Show project types and toolchains only if they are supported on the platform

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**



Generic Remote Host

Properties for connecting to a generic host

Target name: lincoln

Host Information

○ Localhost [✓ Remote host]

Host: lincoln.ncsa.uiuc.edu

User: jalameda

[✓ Password based authentication]

Password: ••••••••••

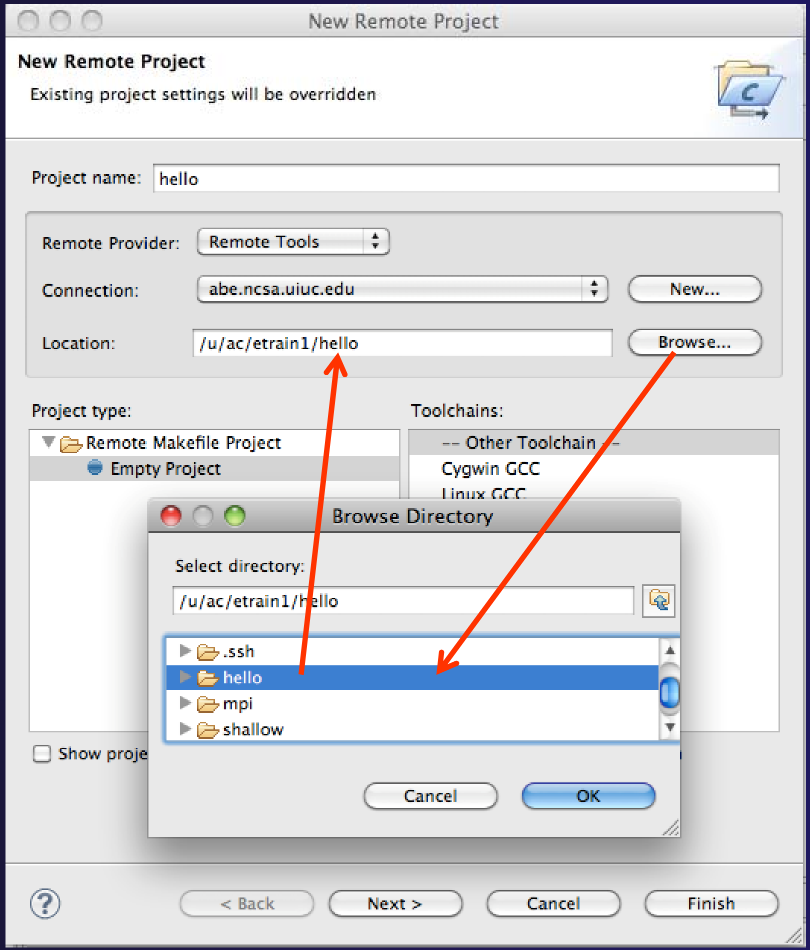
○ Public key based authentication

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**



New Remote Project

Existing project settings will be overridden

Project name: hello

Remote Provider: Remote Tools

Connection: abe.ncsa.uiuc.edu

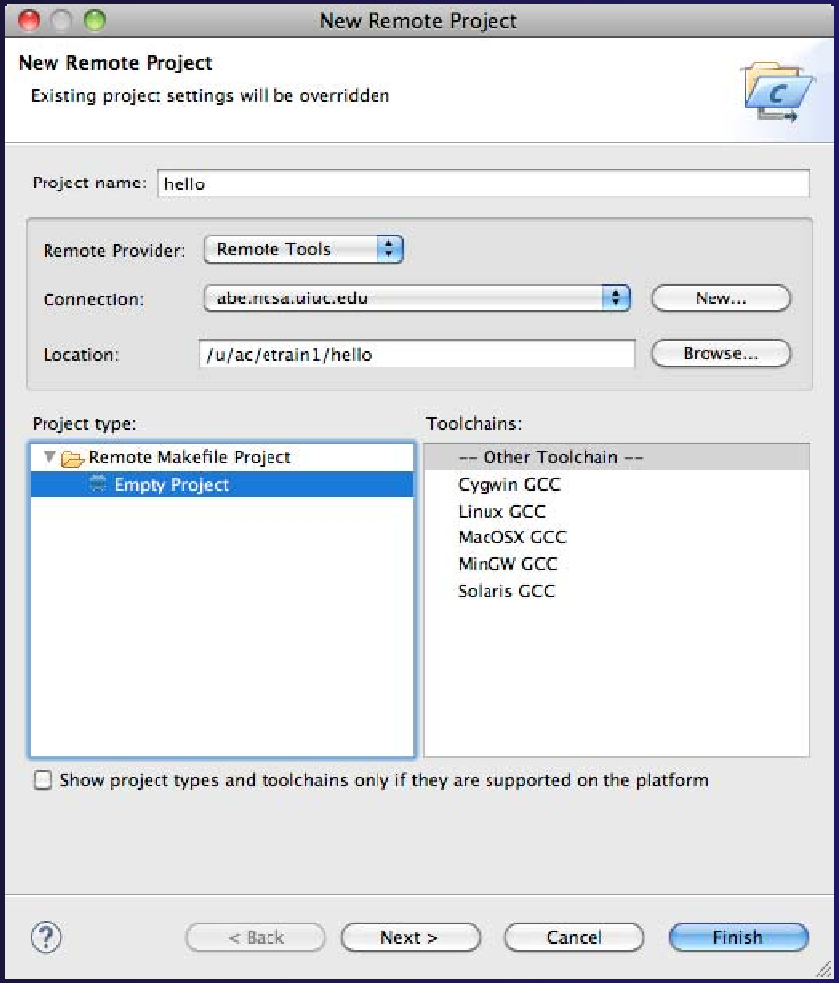
Location: /u/ac/etrain1/hello

Browse Directory

Select directory: /u/ac/etrain1/hello

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



New Remote Project

New Remote Project

Existing project: settings will be overridden

Project name: hello

Remote Provider: Remote Tools

Connection: abe.ncsa.uiuc.edu

Location: /u/ac/etrain1/hello

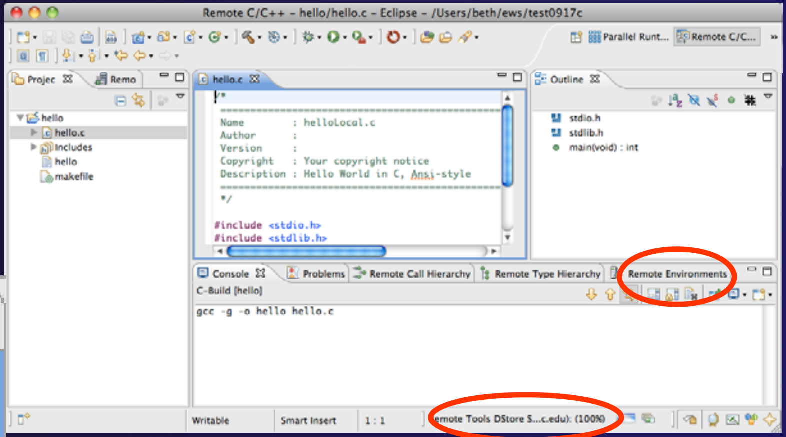
Project type:
- Remote Makefile Project
  - Empty Project

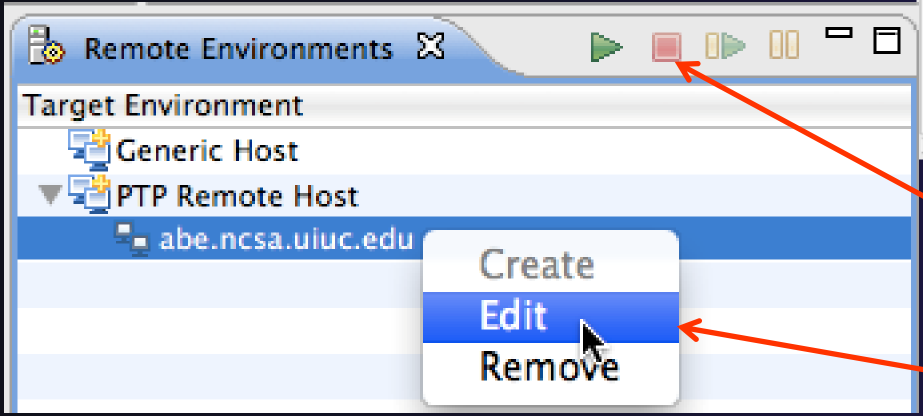
Toolchains:
- -- Other Toolchain --
- Cygwin GCC
- Linux GCC
- MacOSX GCC
- MinGW GCC
- Solaris GCC

☐ Show project types and toolchains only if they are supported on the platform

Changing Remote Connection Information

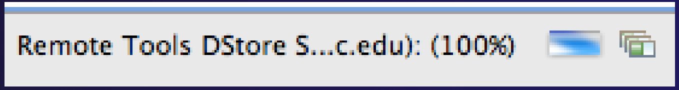
- If you need to change remote connection information (such as username or password), use the **Remote Environments** view


Remote C/C++ - hello/hello.c - Eclipse - /Users/beth/ews/test0917c
Remote Environments
emote Tools DStore S...c.edu): (100%)



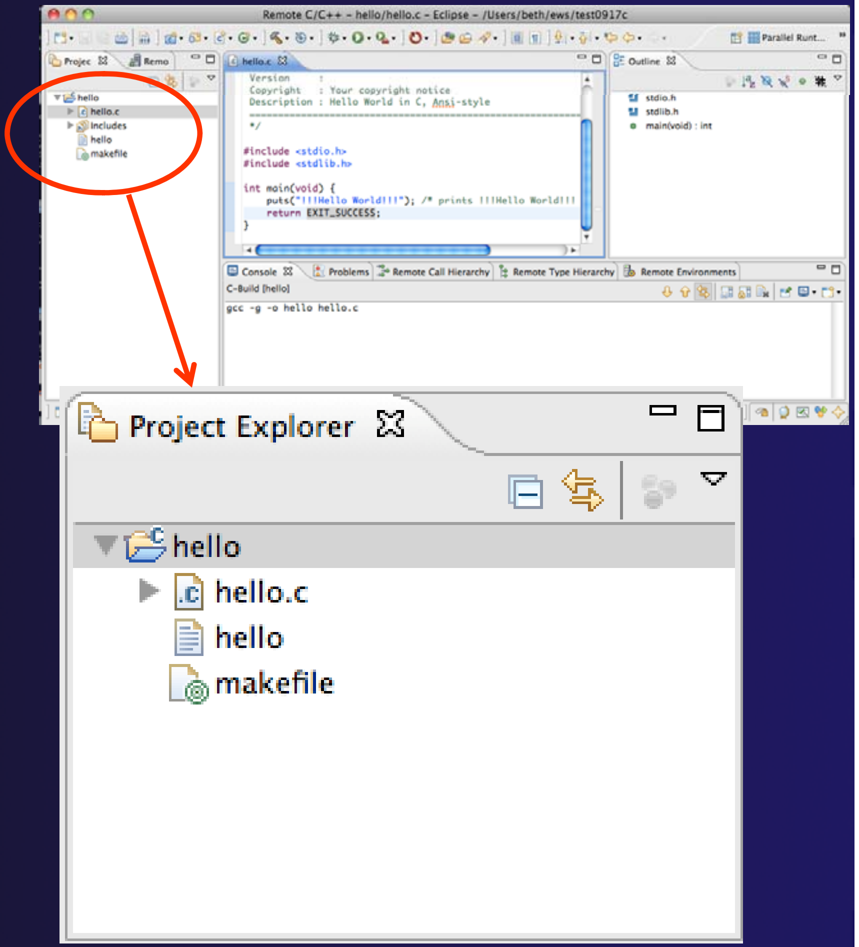
- Stop the remote connection first
- Right-click and select **Edit**

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



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

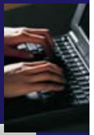

Remote C/C++ - hello/hello.c - Eclipse - /Users/beth/ews/test0917c

```
 Version     :
 Copyright   : Your copyright notice
 Description : Hello World in C, Ansi-style
 ============================================================
 */

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

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

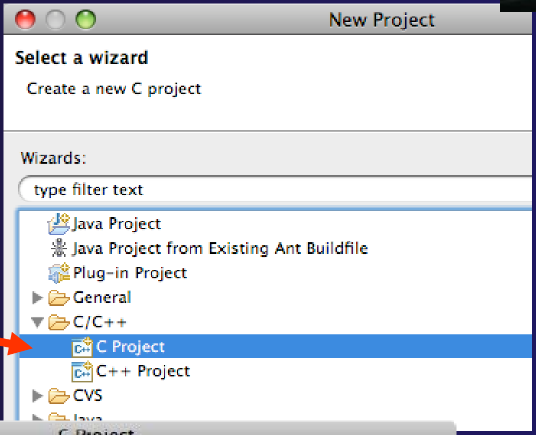
C-Build [hello]

```
gcc -g -o hello hello.c
```

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**



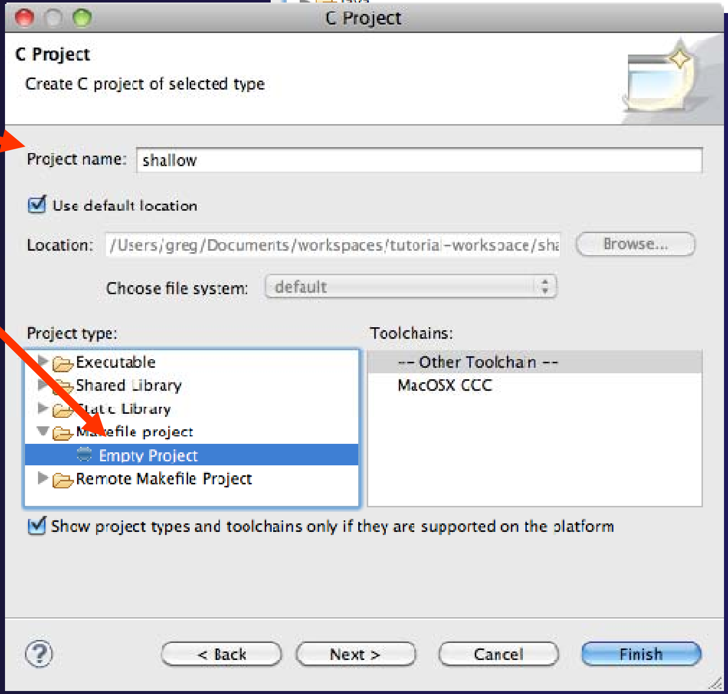
New Project

Select a wizard

Create a new C project

Wizards:

type filter text



C Project

C Project

Create C project of selected type

Project name: shallow

Use default location ✓

Location: /Users/greg/Documents/workspaces/tutorial-workspace/sha

Choose file system: default

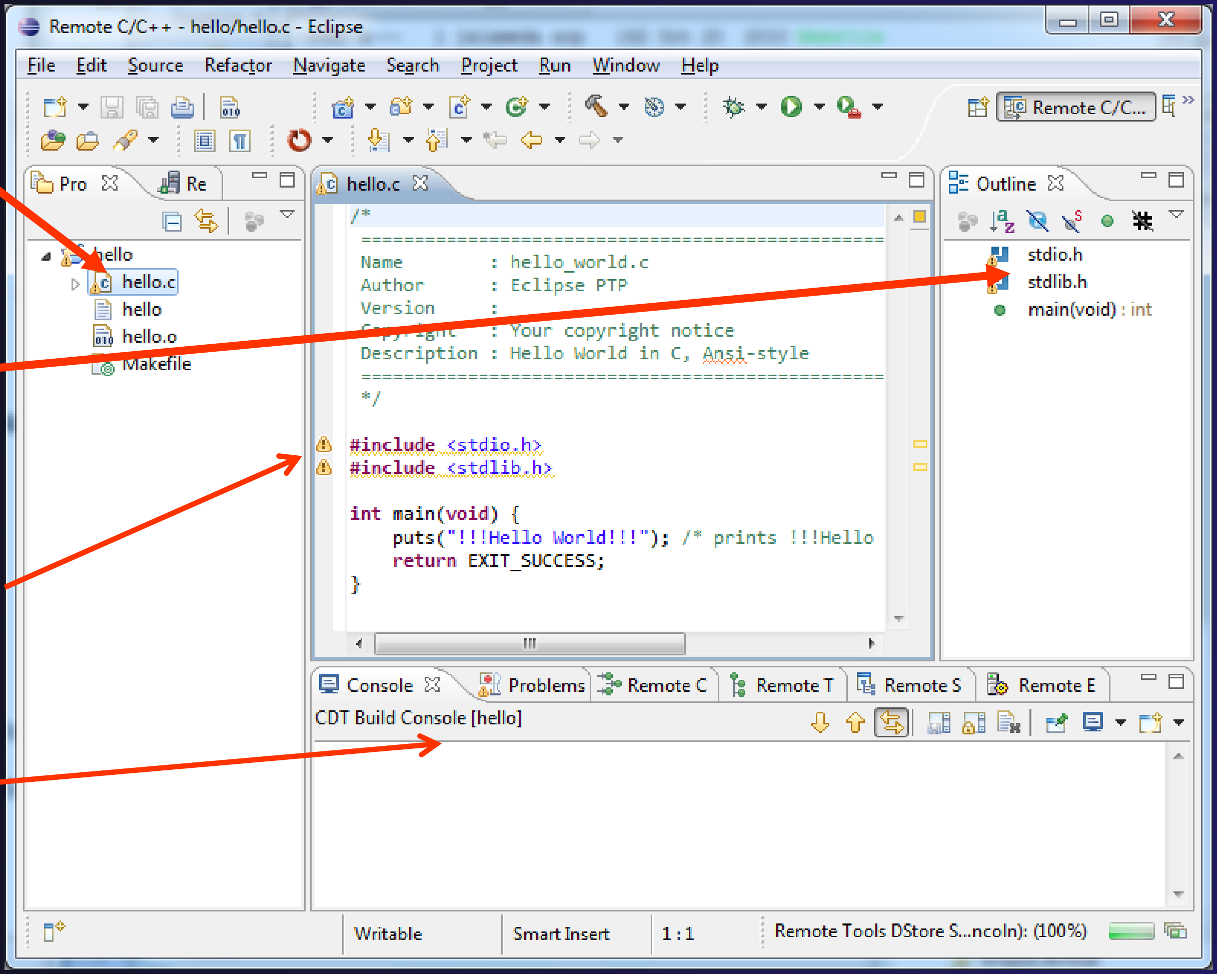
Project type:

Toolchains:

✓ Show project types and toolchains only if they are supported on the platform

Editor and Outline View

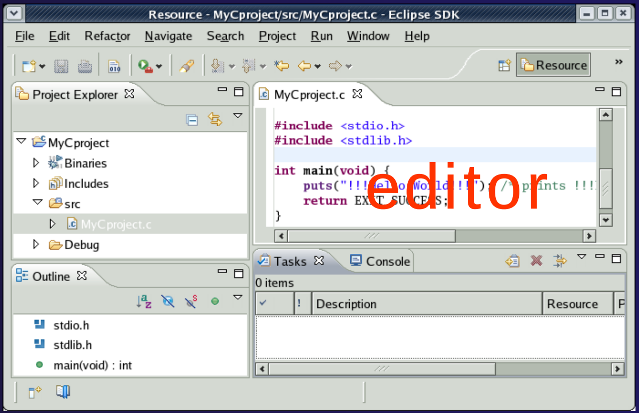
- Double-click on source file to open editor
- Outline view is shown for file in editor
- You should see warnings on the include files: we will fix this later
- Console shows results of build



```
/*
 ============================================================
 Name        : hello_world.c
 Author      : Eclipse PTP
 Version     :
 Copyright   : Your copyright notice
 Description : Hello World in C, Ansi-style
 ============================================================
 */

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

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

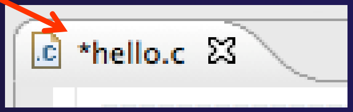
```
CDT Build Console [hello]
```

Editors



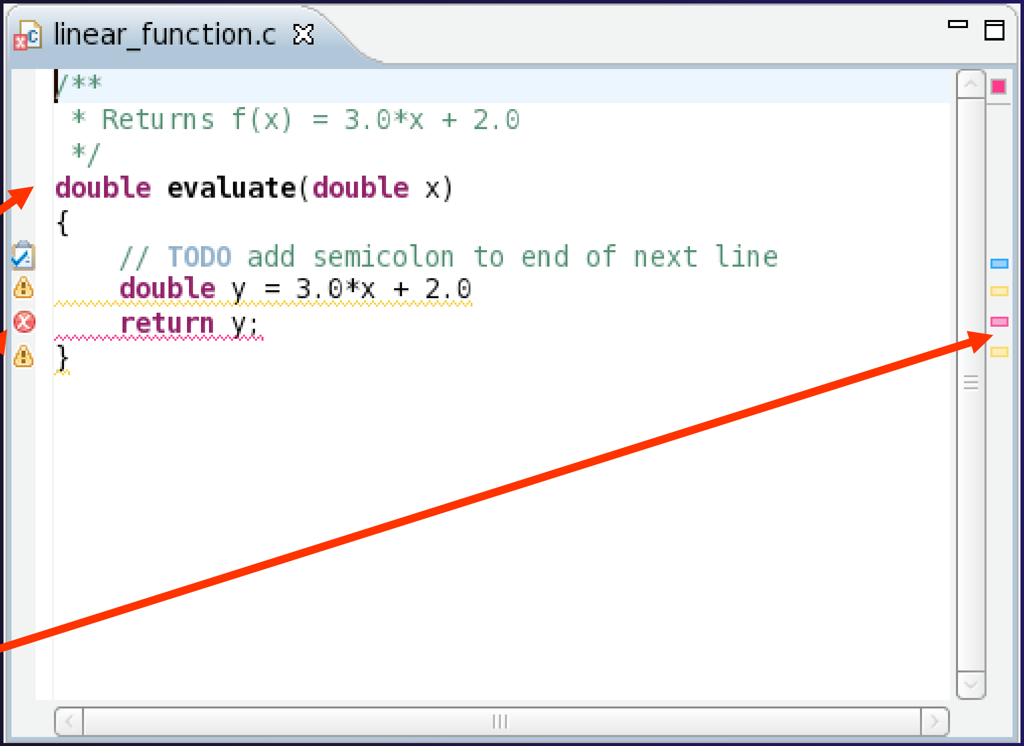
```
#include <stdio.h>
#include <stdlib.h>

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

- 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**



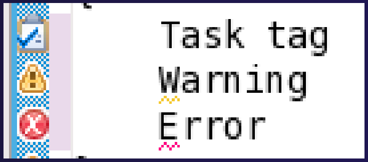
Source Code Editors & Markers

- 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



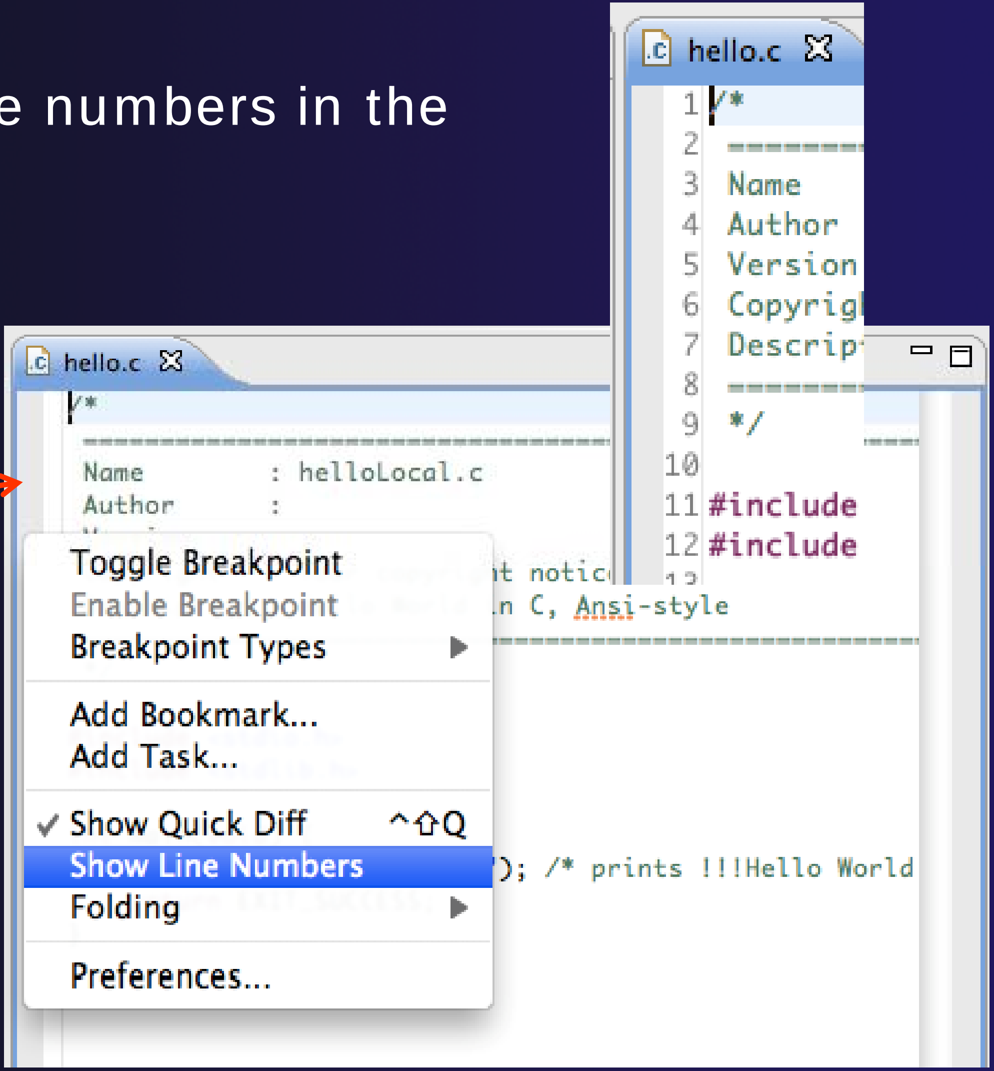
```
/**
 * Returns f(x) = 3.0*x + 2.0
 */
double evaluate(double x)
{
    // TODO add semicolon to end of next line
    double y = 3.0*x + 2.0
    return y;
}
```

Icons:



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**

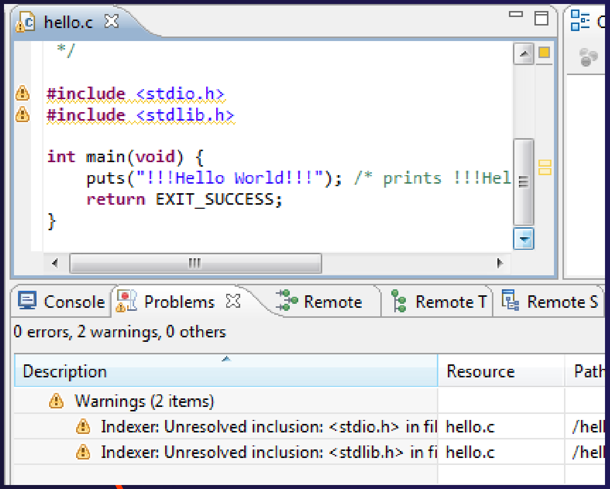


```
/*
================================
Name         : helloLocal.c
Author       :
```

```
1 /*
2  ========
3  Name
4  Author
5  Version
6  Copyrig
7  Descrip
8  ========
9  */
10
11 #include
12 #include
```

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



```
 */

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

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

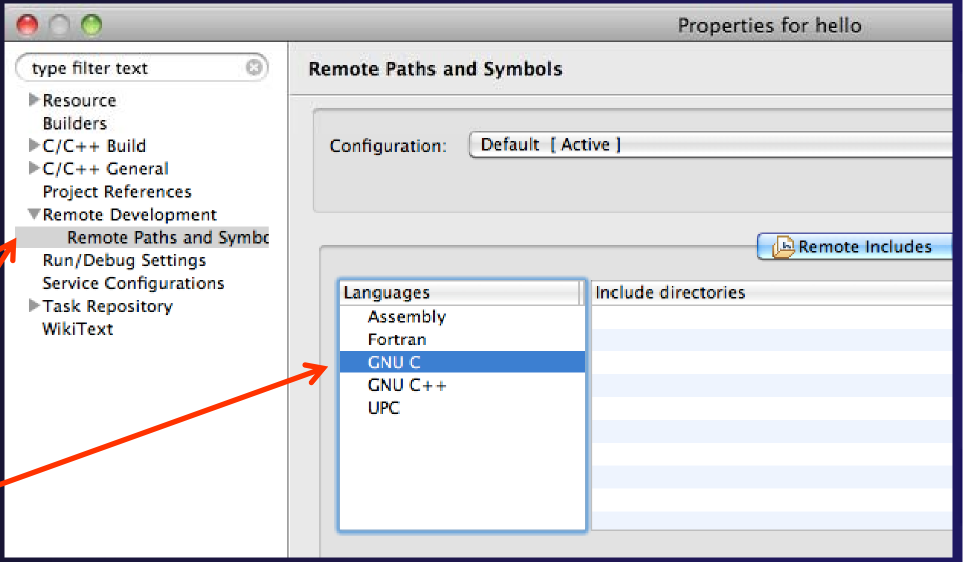
0 errors, 2 warnings, 0 others

Description	Resource	Path
Warnings (2 items)		
Indexer: Unresolved inclusion: <stdio.h> in fil	hello.c	/hell
Indexer: Unresolved inclusion: <stdlib.h> in fi	hello.c	/hell

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**



Properties for hello

type filter text

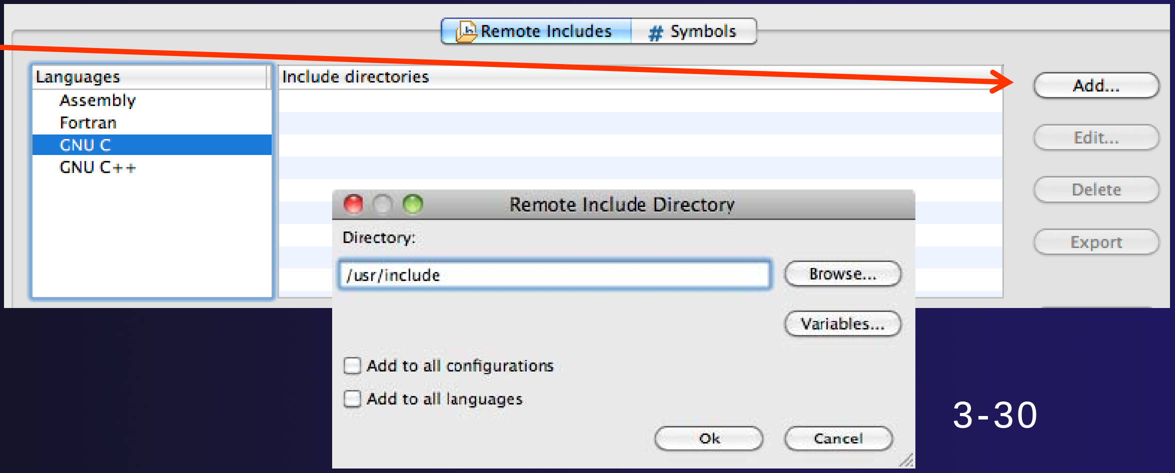
- Resource
- Builders
- C/C++ Build
- C/C++ General
- Project References
- Remote Development
  - Remote Paths and Symbc
- Run/Debug Settings
- Service Configurations
- Task Repository
- WikiText

Remote Paths and Symbols

Configuration: Default [Active]

Remote Includes

Languages	Include directories
Assembly	
Fortran	
GNU C	
GNU C++	
UPC	



Remote Includes | # Symbols

Languages	Include directories
Assembly	
Fortran	
GNU C	
GNU C++	

Add... Edit... Delete Export

Remote Include Directory

Directory:

/usr/include

Browse... Variables...

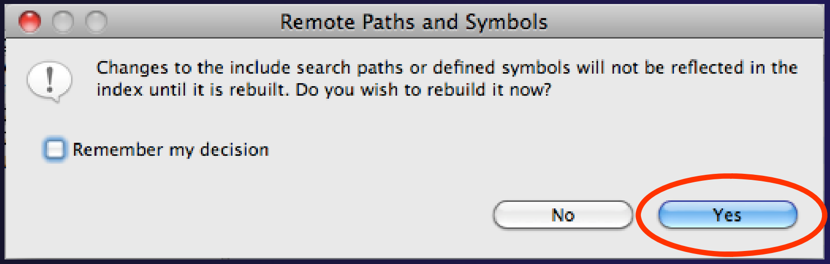
☐ Add to all configurations

☐ Add to all languages

Ok Cancel

Saving the Project Properties

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

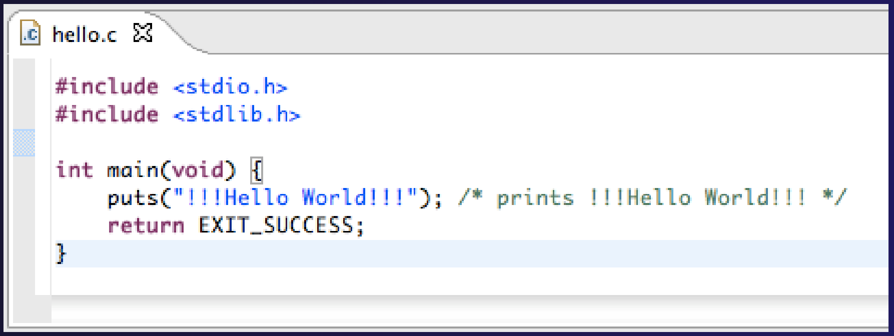


Remote Paths and Symbols

Changes to the include search paths or defined symbols will not be reflected in the index until it is rebuilt. Do you wish to rebuild it now?

Remember my decision

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

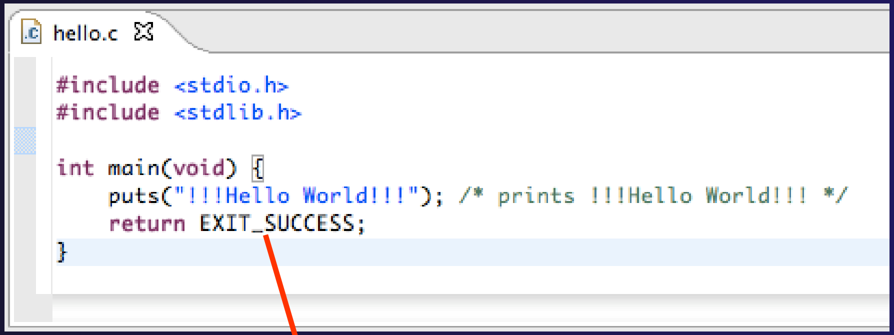


```
#include <stdio.h>
#include <stdlib.h>

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

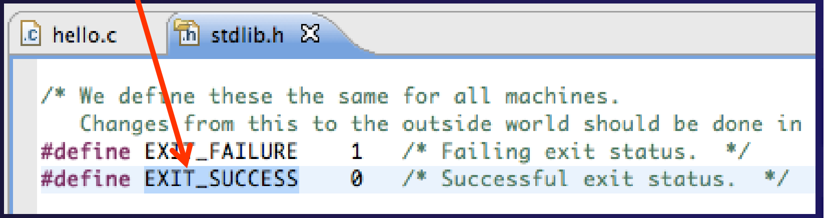
Navigating to Other Files

- 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

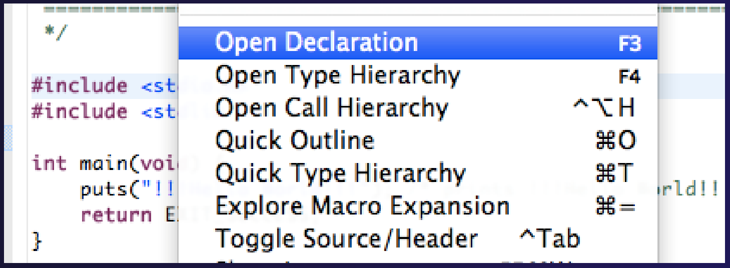
- Open declaration
  - Right-click and select **Open Declaration** will also open the file in which the element is declared
  - E.g. right-click on stdio.h and select **Open Declaration**



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

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

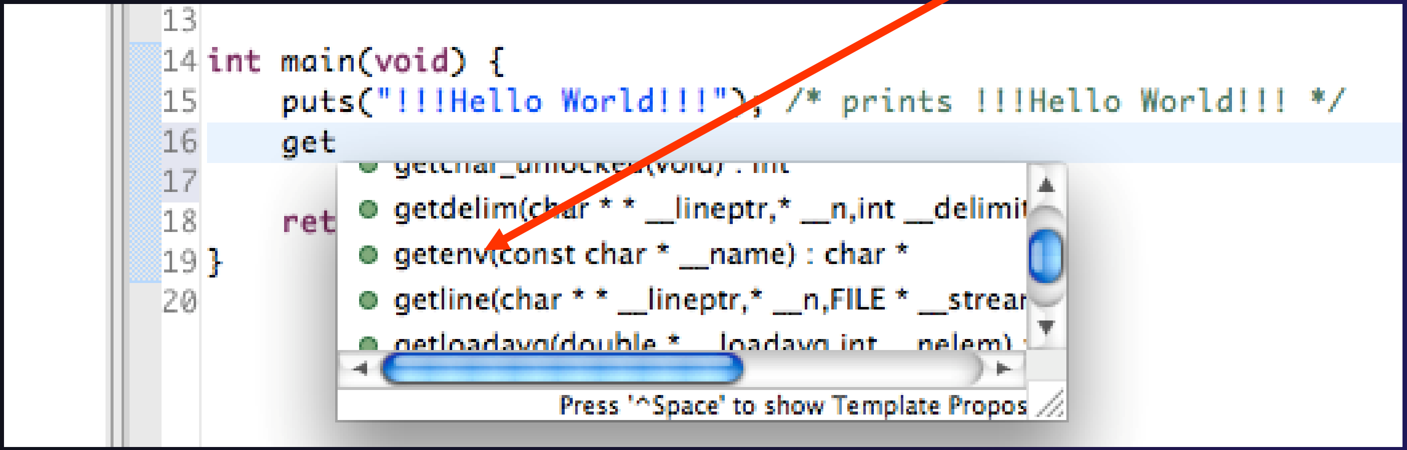


```
/* We define these the same for all machines.
   Changes from this to the outside world should be done in
#define EXIT_FAILURE    1   /* Failing exit status.  */
#define EXIT_SUCCESS    0   /* Successful exit status.  */
```



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

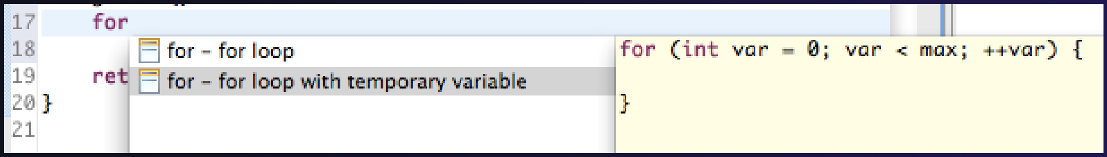


```
int main(void) {
    puts("!!!Hello World!!!"); /* prints !!!Hello World!!! */
    get

    ret
}
```

getchar_unlocked(void) : int
getdelim(char * * __lineptr,* __n,int __delimit
getenv(const char * __name) : char *
getline(char * * __lineptr,* __n,FILE * __strear
getloadavg(double * __loadavg,int __nelem) :
Press '^Space' to show Template Propos

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

Hit ctrl-space again for code templates

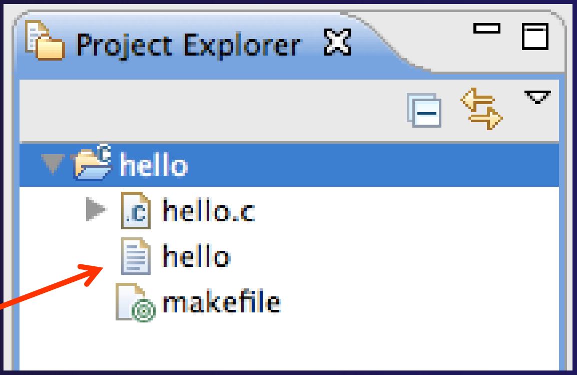


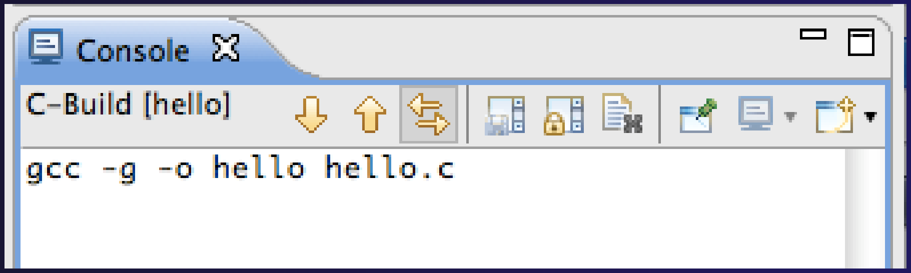
```
    for

    ret
}
```

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

}
```

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 the "build" button
  - Alternatively, select **Project>Build Project**
- The executable should appear in the project
- The **Console** view shows build output



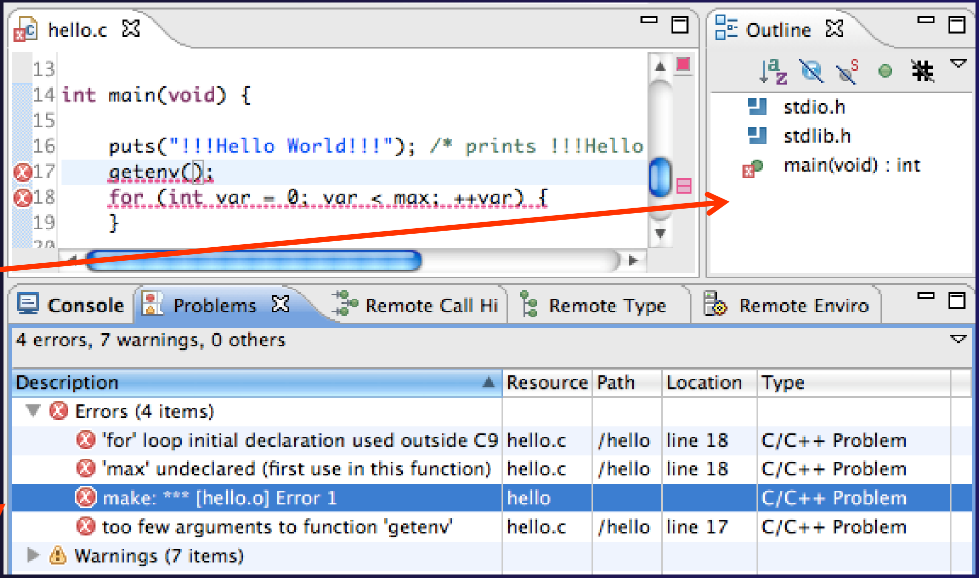


```
gcc -g -o hello hello.c
```

Executable 'hello'

Build Problems

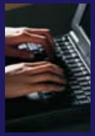
- If there are problems, they will be shown in a variety of ways
  - Marker on editor line
  - Marker on overview ruler
  - Listed in the **Problems view**
- Double-click on line in **Problems view** to go to location of error



```
int main(void) {

    puts("!!!Hello World!!!"); /* prints !!!Hello
    getenv();
    for (int var = 0; var < max; ++var) {
    }
```

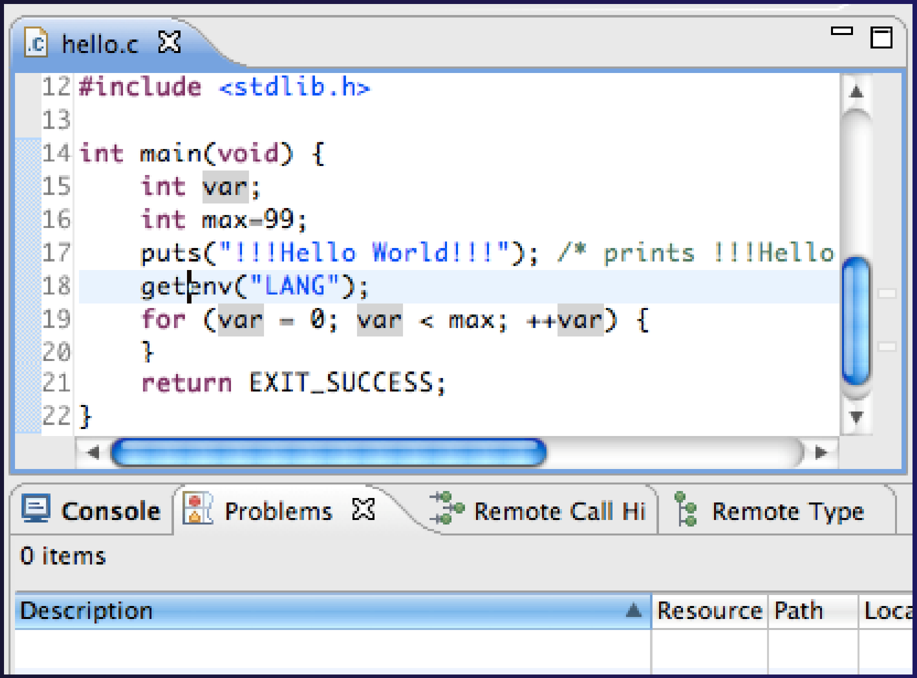
4 errors, 7 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (4 items)				
'for' loop initial declaration used outside C9	hello.c	/hello	line 18	C/C++ Problem
'max' undeclared (first use in this function)	hello.c	/hello	line 18	C/C++ Problem
make: *** [hello.o] Error 1	hello			C/C++ Problem
too few arguments to function 'getenv'	hello.c	/hello	line 17	C/C++ Problem
Warnings (7 items)				



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



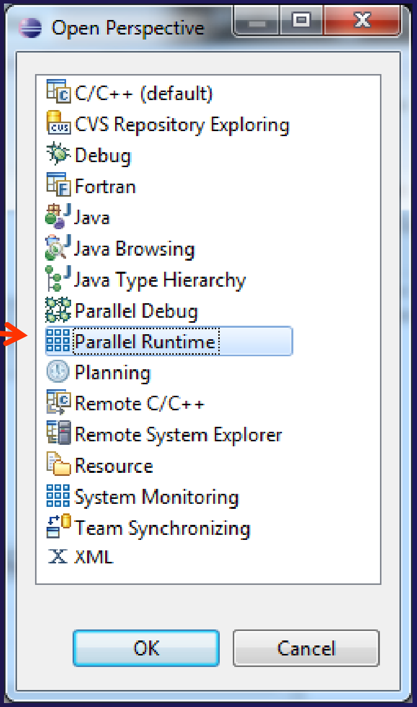
```
#include <stdlib.h>

int main(void) {
    int var;
    int max=99;
    puts("!!!Hello World!!!"); /* prints !!!Hello
    getenv("LANG");
    for (var = 0; var < max; ++var) {
    }
    return EXIT_SUCCESS;
}
```

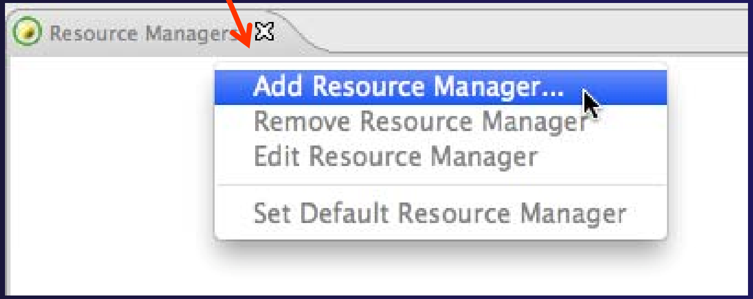
0 items

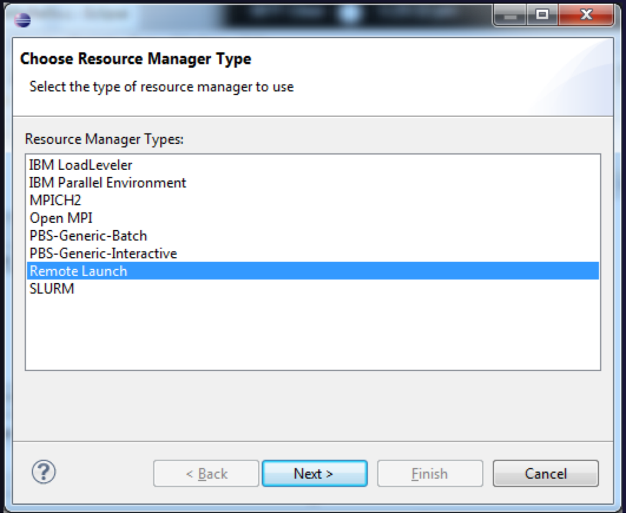
Description	Resource	Path	Loca

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 >**

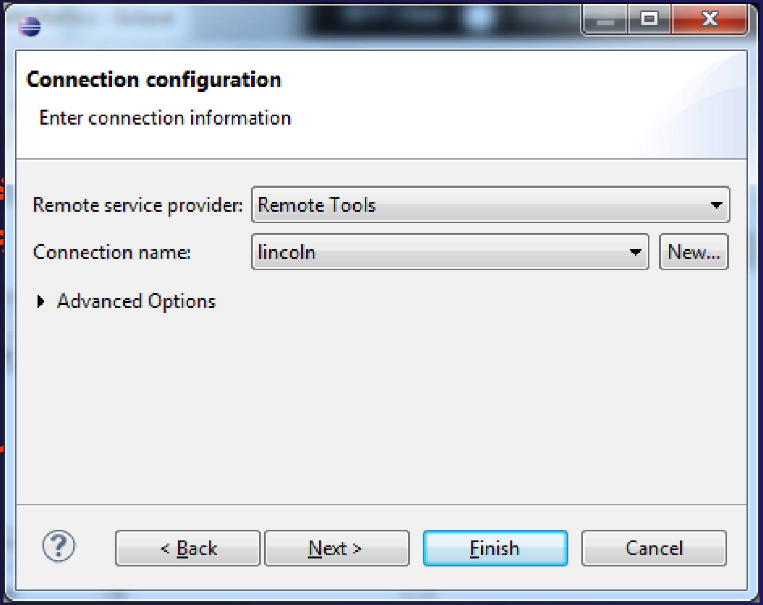






Configure the Resource Manager

- 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**


Connection configuration

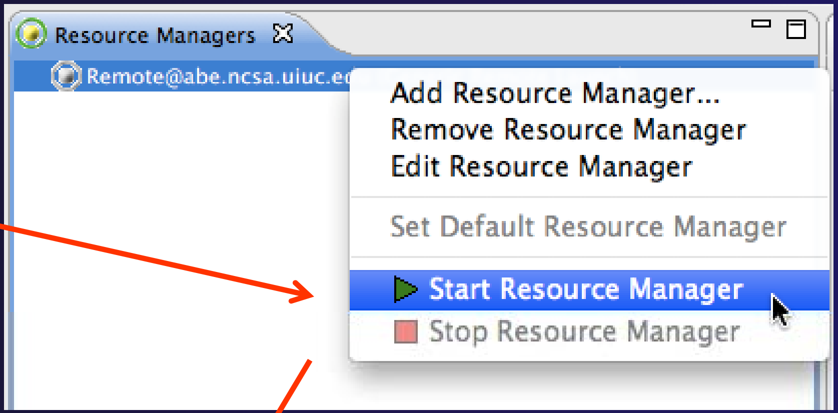
Enter connection information

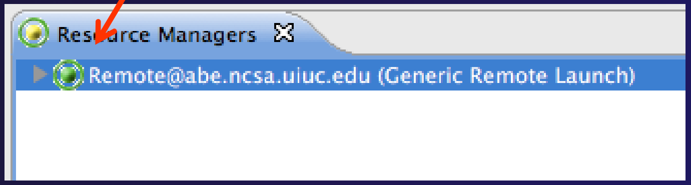
Remote service provider: Remote Tools

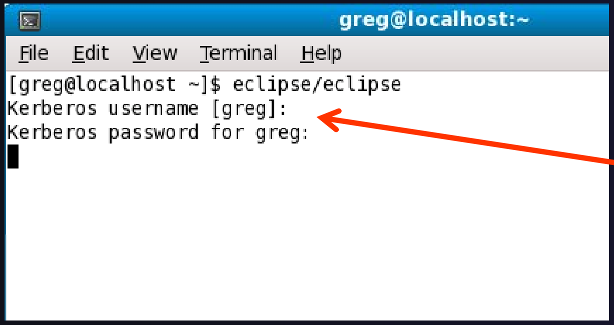
Connection name: lincoln

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







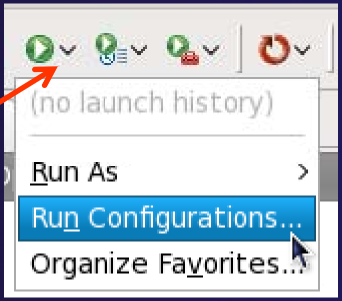
```
[greg@localhost ~]$ eclipse/eclipse
Kerberos username [greg]:
Kerberos password for greg:
```

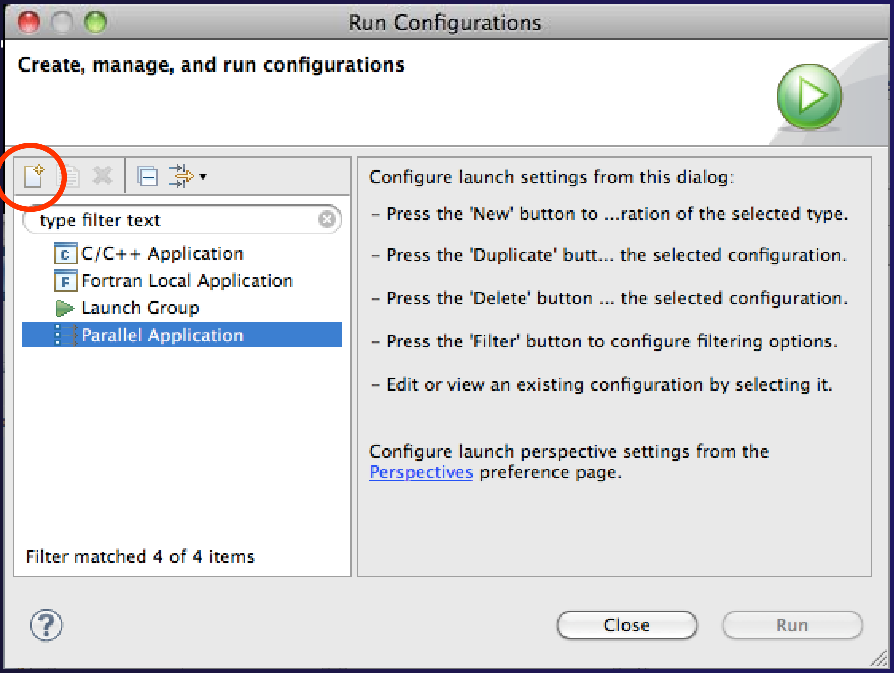
NOTE: On some Linux systems, starting a resource manager may appear to hang. Open the window you launched Eclipse from and check if there is a prompt for a kerberos username. Hit “enter” twice if you see the prompt.

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





Create, manage, and run configurations

Configure launch settings from this dialog:

- Press the 'New' button to ...ration of the selected type.
- Press the 'Duplicate' butt... the selected configuration.
- Press the 'Delete' button ... the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

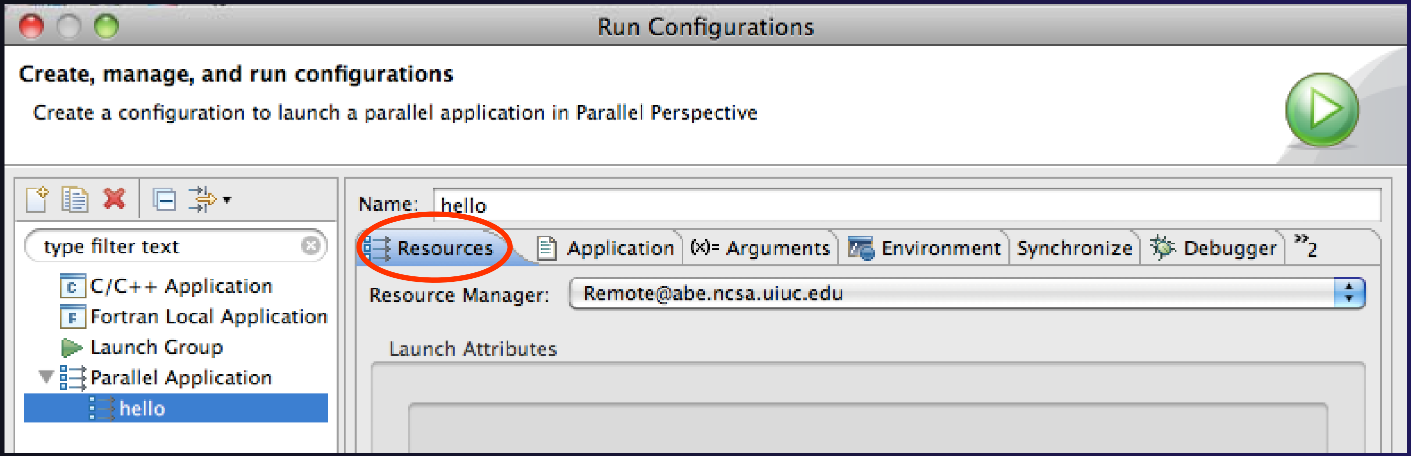
Configure launch perspective settings from the Perspectives preference page.

Filter matched 4 of 4 items

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.


Run Configurations

Create, manage, and run configurations

Create a configuration to launch a parallel application in Parallel Perspective

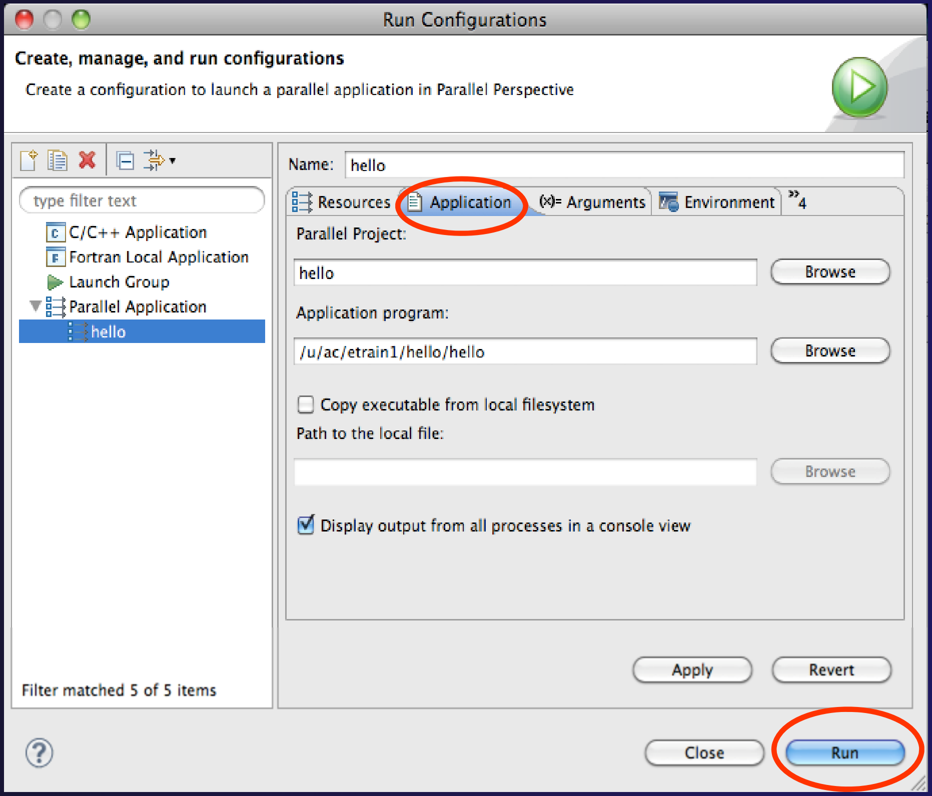
Name: hello

Resource Manager: Remote@abe.ncsa.uiuc.edu

Launch Attributes

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


Run Configurations

Create, manage, and run configurations

Create a configuration to launch a parallel application in Parallel Perspective

type filter text

C/C++ Application
Fortran Local Application
Launch Group
Parallel Application
hello

Filter matched 5 of 5 items

Name: hello

Resources | Application | Arguments | Environment | »4

Parallel Project: hello [Browse]

Application program: /u/ac/etrain1/hello/hello [Browse]

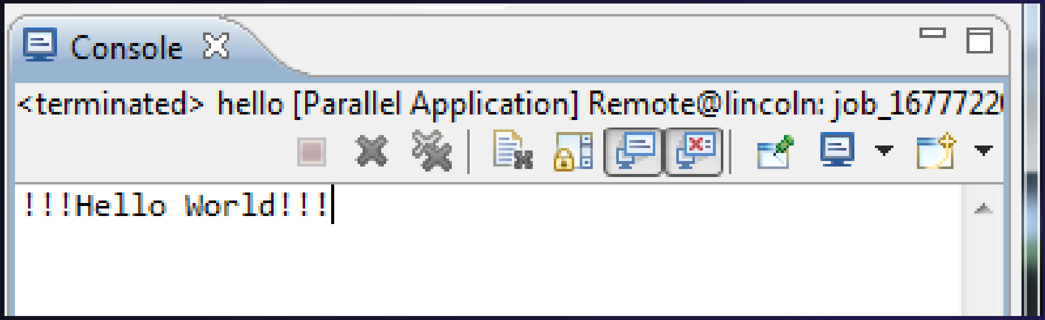
[ ] Copy executable from local filesystem

Path to the local file: [Browse]

[✓] Display output from all processes in a console view

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



```
<terminated> hello [Parallel Application] Remote@lincoln: job_1677722
```

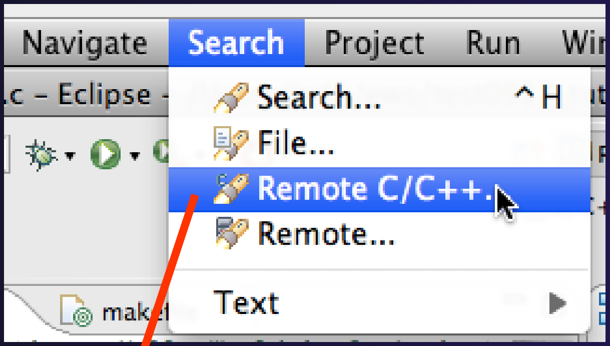
```
!!!Hello World!!!
```

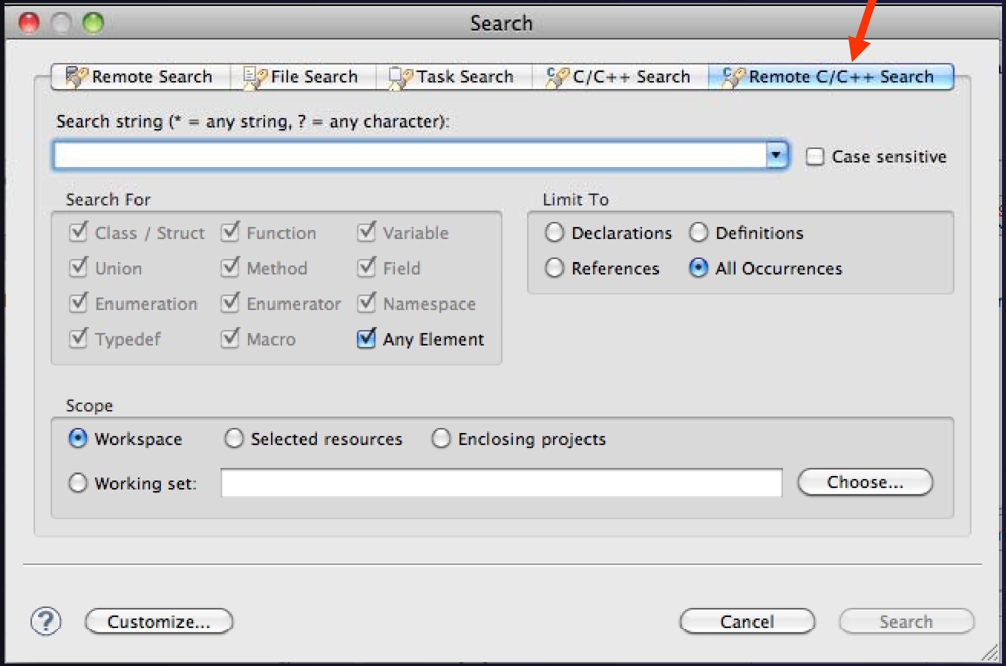
Other CDT features

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

First, return to the “Remote C/C++ Perspective”

Language-Based Searching





Search string (* = any string, ? = any character):

☐ Case sensitive

Search For

- ✓ Class / Struct
- ✓ Function
- ✓ Variable
- ✓ Union
- ✓ Method
- ✓ Field
- ✓ Enumeration
- ✓ Enumerator
- ✓ Namespace
- ✓ Typedef
- ✓ Macro
- ✓ Any Element

Limit To

Declarations / Definitions / References / [✓ All Occurrences]

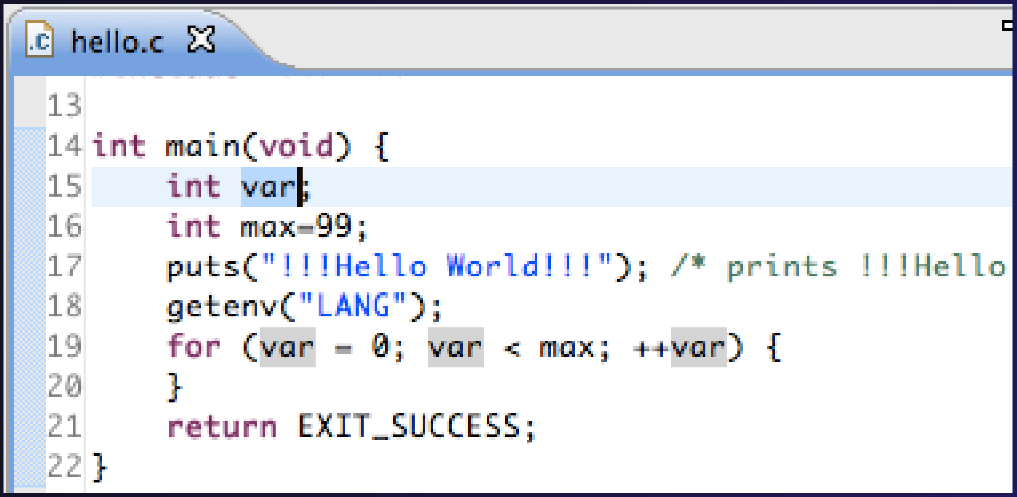
Scope

[✓ Workspace] / Selected resources / Enclosing projects / Working set:

- "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



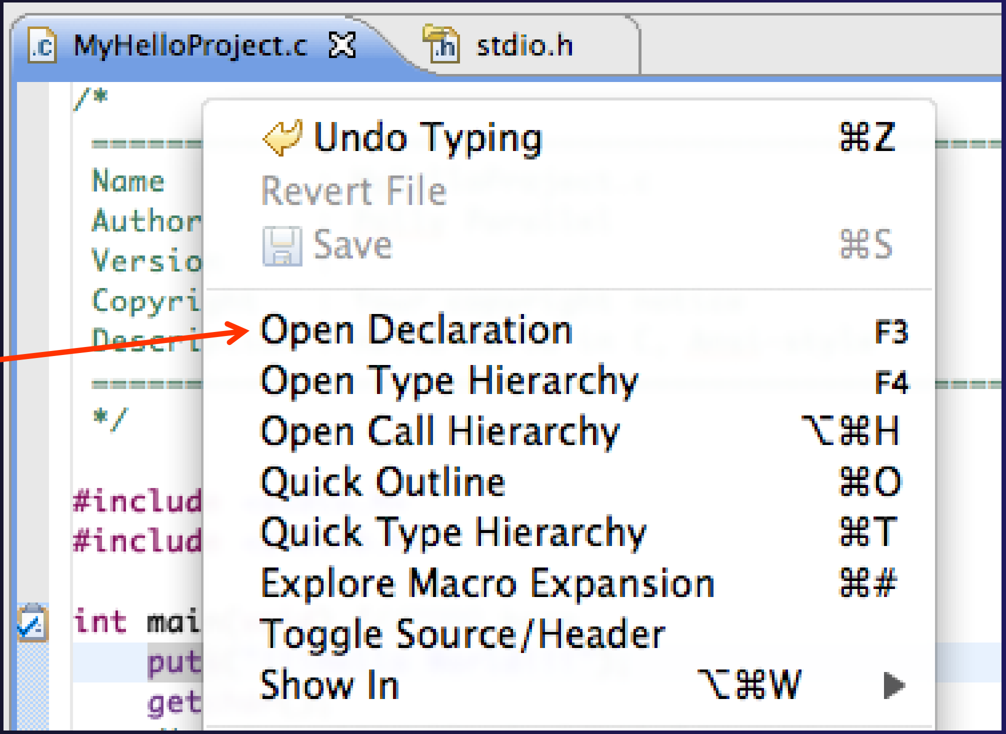
```

int main(void) {
    int var;
    int max=99;
    puts("!!!Hello World!!!"); /* prints !!!Hello
    getenv("LANG");
    for (var = 0; var < max; ++var) {
    }
    return EXIT_SUCCESS;
}
```

Open Declaration

- Jumps to the declaration of a variable, function, etc., even if it's in a different file

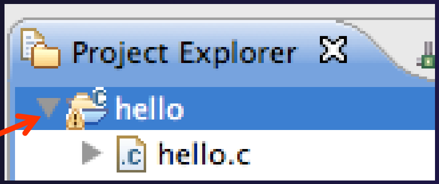
- Right-click on an identifier
- Click **Open Declaration**

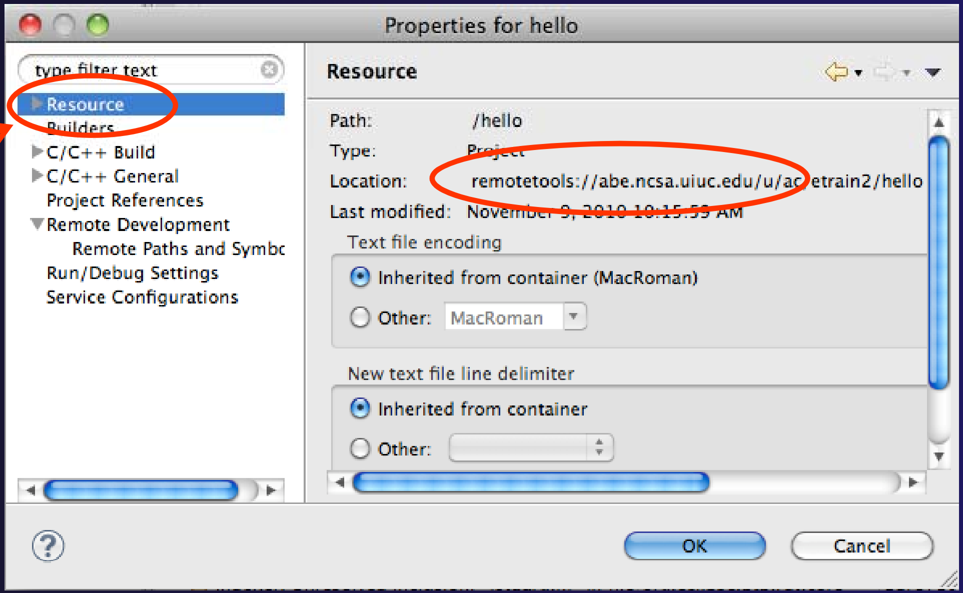
- Can also Ctrl-click (Mac: Cmd-click) on an identifier to "hyperlink" to its declaration



Remote Projects - Location

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





Resource

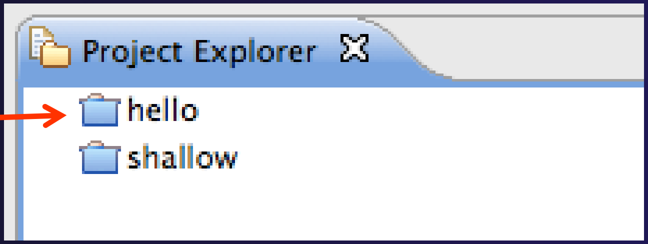
Path: /hello
Type: Project
Location: remotetools://abe.ncsa.uiuc.edu/u/ac/etrain2/hello
Last modified: November 9, 2010 10:15:59 AM

Text file encoding
(●) Inherited from container (MacRoman)
( ) Other: MacRoman

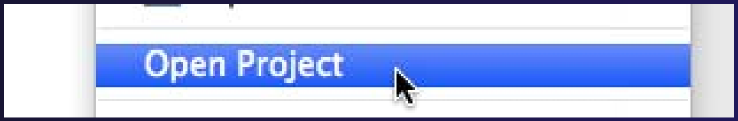
New text file line delimiter
(●) Inherited from container
( ) Other:

Remote Projects - Reopening

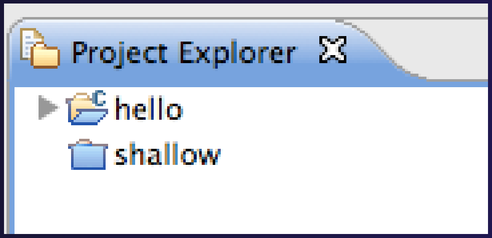
- When re-opening Eclipse workbench, remote projects will be closed



- 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


Project Explorer
hello
shallow