# Compiling and Debugging Basics

QNX SOFTWARE SYSTEMS




NOTES:

## Introduction

This section is intended to give enough familiarity with the IDE to be able to do the exercises in the Realtime Programming and Drivers courses.

You will learn the basics of:

- Eclipse
- host-target development environment
- editing, compiling & running




NOTES:

# Topics:

- **Eclipse Basics**
- **Targets**
- **Projects and Source**
- **Compiling**
- **Running and Debugging**
- **Exercise**
- **Conclusion**




NOTES:

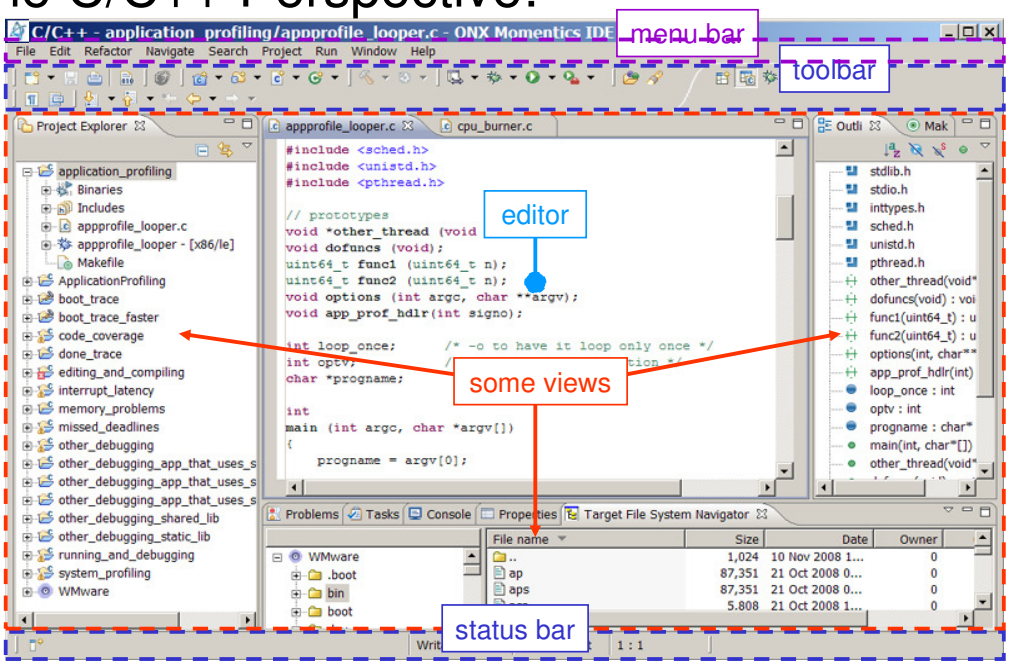
# The QNX Momentics IDE is based on Eclipse:

- an open source platform, written in Java, for building IDEs
- we inherit a lot of behavior and terminology from this:
  - an Editor is a component of the IDE where you edit (or browse) a resource (such as a C source file)
    - open editors by double-clicking on resources (files)
  - a View is an area that provides: navigation, information, control (but generally not editing)
  - editors and views don't share space (can't be tabbed together)
  - a Perspective is: a collection of views, editors, menu items, and tool bar buttons that are helpful for doing a specific task




NOTES:

# The C/C++ Perspective:



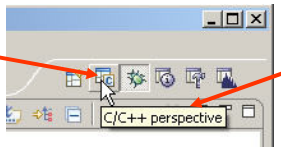



NOTES:

## IDE Basics – Perspective Control

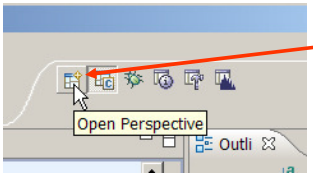
# To switch perspectives:

click on the icon for the perspective to which you wish to switch

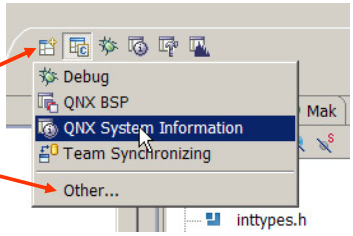



tool tips will help you learn which icons correspond to which perspective

# To open a new perspective:




click on the Open a Perspective button and choose from the list. Choose Other… if the one you want isn't there



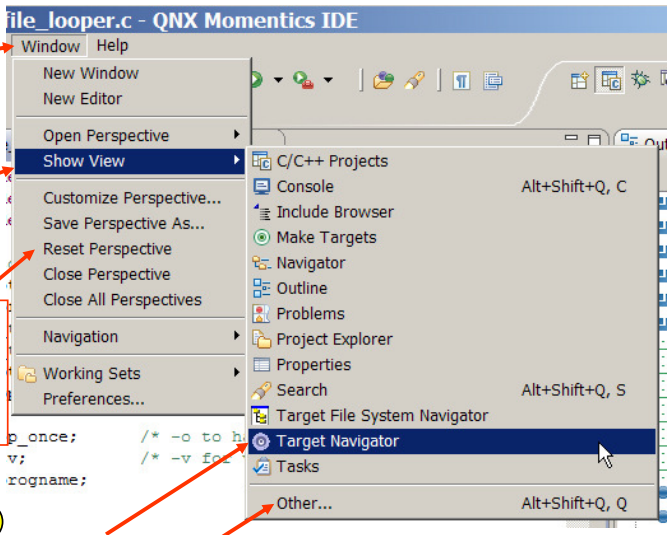
NOTES:

## IDE Basics – Adding new views

# To show a particular view:

1. go to the Window menu…
2. … and choose Show View

hint: if you've messed up a perspective, Reset Perspective will take you back to the default configuration




3. pick a view. If the view you want is not available, choose Other… for a complete list of all the views available...




NOTES:

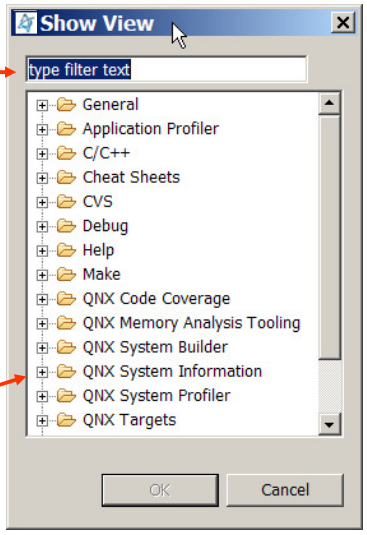
The list of views in the menu will vary depending on which perspective you're in. It generally contains the views associated with the perspective, many of which are displayed by default. It can also be used to find, or restore lost or accidentally closed views.

## New Views – Show View dialog

# The Show View dialog:

use this to filter the views that are listed to find the view you want

expand the view category and choose the view you want







NOTES:

This filtering idea will crop up in many places, often when the IDE presents a plethora of choices the filter option will also be present to aid in navigation.

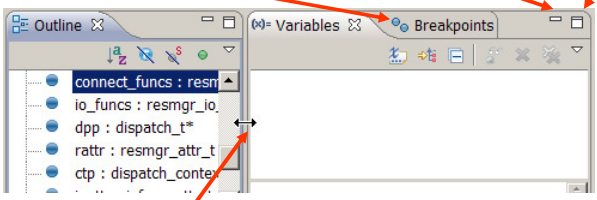
# Working with Views:

– views are generally stacked or “tabbed” with other views:

to bring a view to the front, click on its tab

you can minimize a group of views, moving them to the closest side of the IDE

you can maximise a view as well




any border between views can be used to resize the views that share the border




NOTES:

Topics:

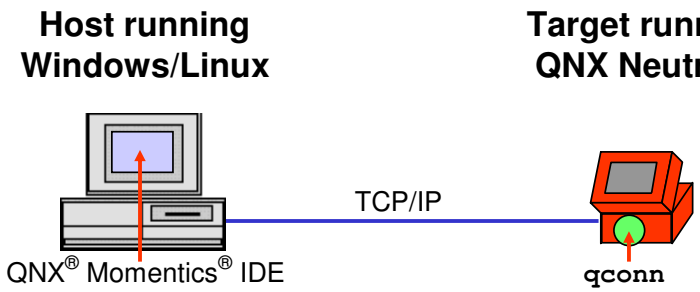
- Eclipse Basics
- → Targets
- Projects and Source
- Compiling
- Running and Debugging
- Exercise
- Conclusion




NOTES:

## Host - Target

# QNX uses a cross-development setup:




- **qconn** is a program on the target that must be running for the IDE to deal with the target
- in the IDE you have to tell it how to find the target:
  - generally this is an IP address or hostname
  - this information is stored in a Target Project
  - once the Target project is created, it can be used multiple places



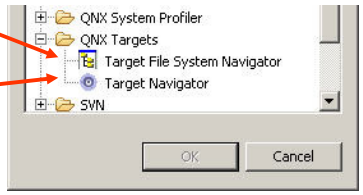

NOTES:

Only command line development tools are available for Neutrino hosted as of QNX Momentics 6.4.0 release.

## Target Interaction – Two Main views

There are two main views for interacting with the target:

- created from Window→Show View→Other:
- Target File System Navigator:
  - viewing target's file system
  - copying files to/from target
- Target Navigator:
  - creating/deleting Target Projects
  - seeing what processes are running
  - killing processes



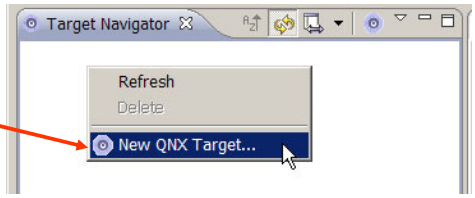

- The QNX System Information perspective has many views for collecting target information
  - it includes a Target Navigator view by default




NOTES:

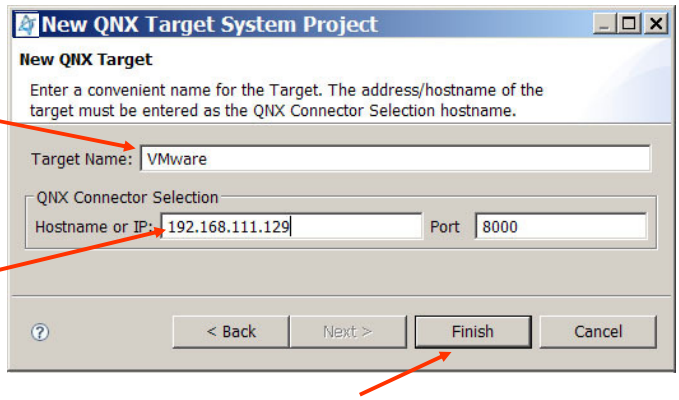
# Creating a Target System project:

from the Target Navigator view right-click in here and choose New QNX Target…




fill in a name representing your target. This will be the Target System project's name

fill in your target's IP address. **qconn** uses port 8000 by default so you wouldn't normally change this




choose Finish and your Target System project will be created




NOTES:

## Command-Line on Target

Often you will need a command-line on your target, some ways include:

- telnet session:
  - in the Target Navigator, right click on the Target and select “Launch Telnet Session”
  - or run a telnet client manually on your host
- serial connection:
  - generally requires a hardware connection, with a null-modem cable
  - use the serial terminal view, or a serial terminal application on your host
- run a shell from the IDE:
  - double-click on a shell in the Target File System Navigator
- physical console, real or through a KVM switch




NOTES:

Topics:

- Eclipse Basics
- Targets
- → Projects and Source
- Compiling
- Running and Debugging
- Exercise
- Conclusion

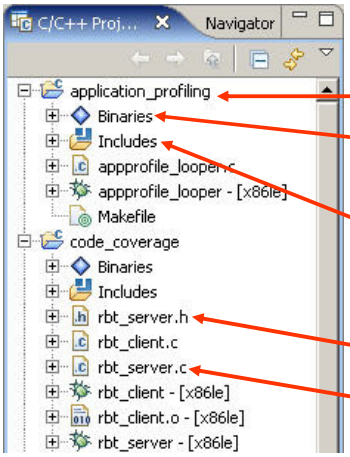



NOTES:

## Projects

# The IDE keeps source in Projects:

- represents a directory (or folder) underneath
- viewed in the Project Explorer view:




project name

Binaries pseudo-folder; holds a list of executables

Includes pseudo-folder; allows navigation of header files

Header file

Source file; double-click to open for editing

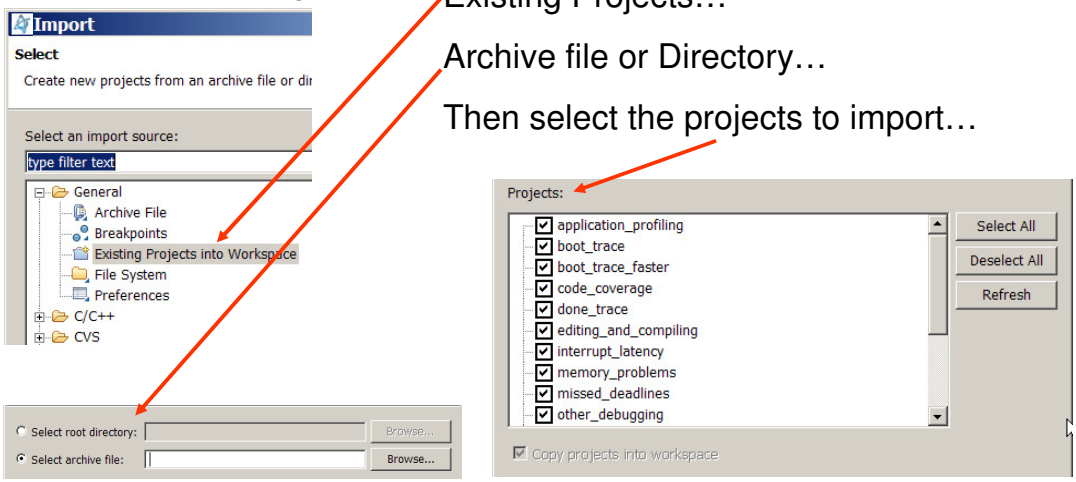



NOTES:

## IDE Basics – Importing Project(s)

A convenient way of creating projects is to import existing ones:

- File -> Import…







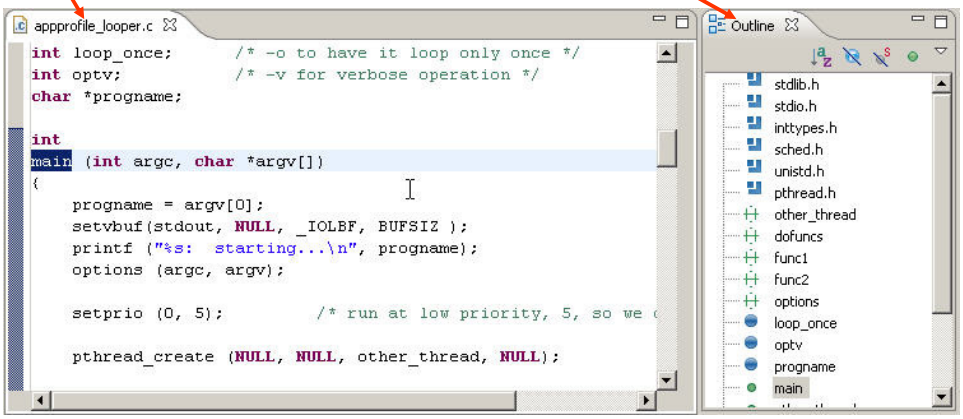
NOTES:

## Editing Source

# Double-click a file to open in the editor:

double-click title bar of editor to maximize/restore it

Outline view summarizes source for quick navigation







NOTES:

# Some editing shortcuts:

- Ctrl-Space does code completion for functions, structures/classes, and code blocks
- “standard” Windows cut & paste with Ctrl-C, Ctrl-X, Ctrl-V
- undo/redo with Ctrl-Z, Ctrl-Y
- hover-help on functions in library gives quick summary of use and headers
- select function, then
  - Ctrl-Shift-N will insert `#include` lines for needed headers
    - Menu-Click -> Source -> Add Include will also do this
  - F3 will open definition/declaration
- find or find & replace with Ctrl-F
- search multiple files with Ctrl-H




NOTES:

Topics:

- Eclipse Basics
- Targets
- Projects and Source
- → Compiling
- Running and Debugging
- Exercise
- Conclusion




NOTES:

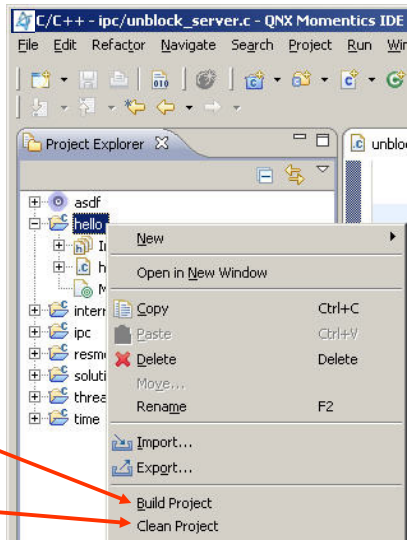
## Compiling - Building from the IDE

# The IDE calls compiling “building”:

– to build from the IDE, right-click on your project...

this will build only what needs to be built

this will remove all files that are not source (e.g. executables, object modules, error files, …)







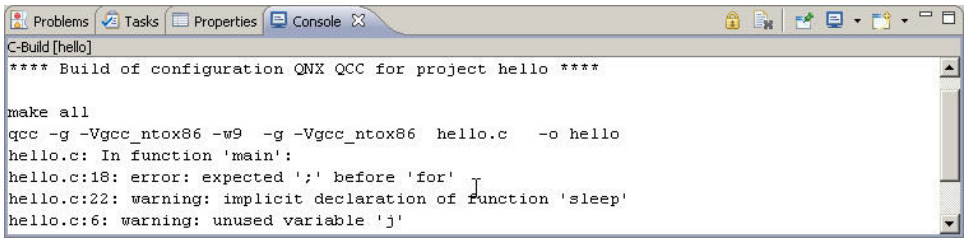
NOTES:

•The IDE invokes make to actually do the builds, using the Makefile in the project.

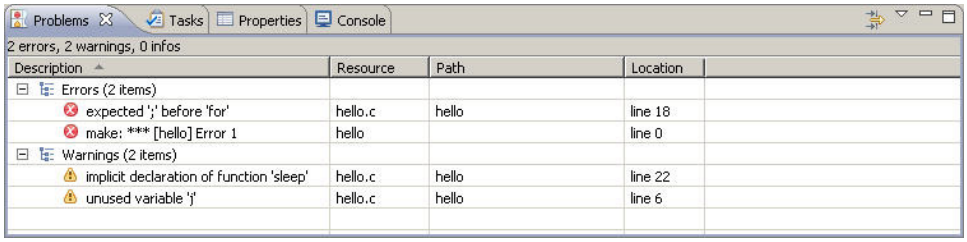
•Build Project issues “make all”; Clean Project issues “make clean”. Both assume that these targets exist in the Makefile.

## Finding Errors

While building, the console view shows the output from the build:




After the build is complete, the Problems view summarizes the errors and warnings:





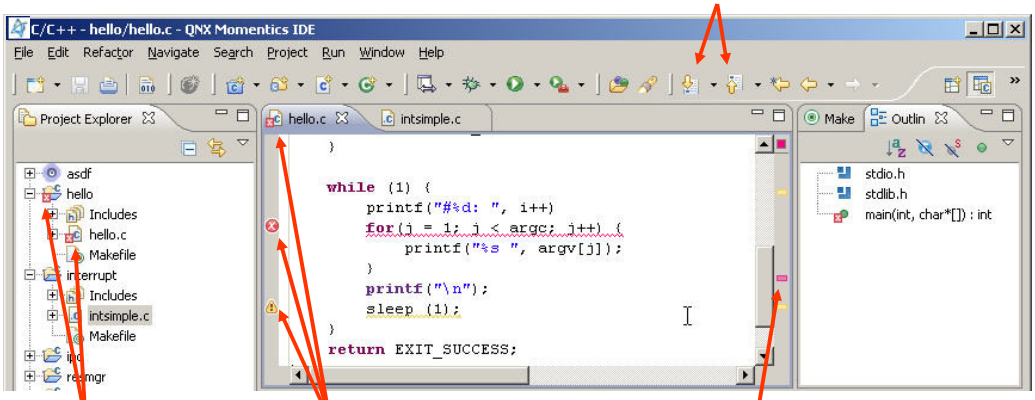


NOTES:

## Fixing Errors - Problem indicators

# Many other places indicate problems:

clicking on these will go to next and previous problems (the editor must be selected for these to be enabled)




indicates that there is a problem(s)

these markers represent problems. Their relative position vertically represents their locations in the file. You can click on these.




NOTES:

Topics:

- Eclipse Basics
- Targets
- Projects and Source
- Compiling
- → Running and Debugging
- Exercise
- Conclusion




NOTES:

## Running

There are two main ways of running a program you've built in the IDE:

- copy it to the target with the Target File System Navigator then run it from the command line
- create a Launch configuration and run it from the IDE
  - if you're using the IDE for debugging, you'll need a Launch configuration
  - Launch configurations only need to be created once for a program, then can be re-used

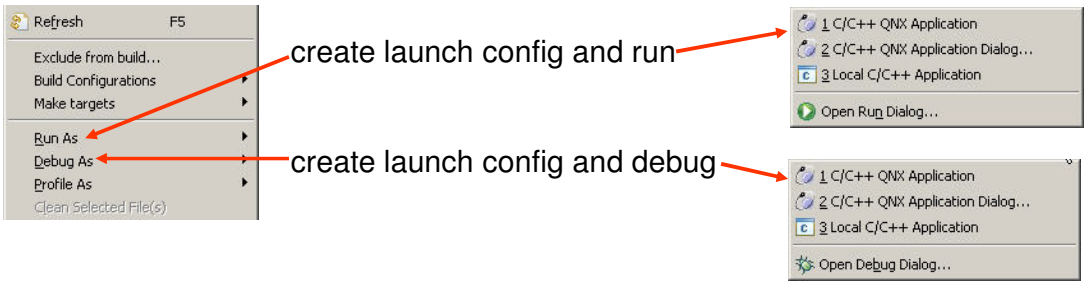



NOTES:

## Setup – Launch configuration

To setup a launch configuration for running or debugging:

– select the program you want to run in the Project Explorer view, then right click:







NOTES:

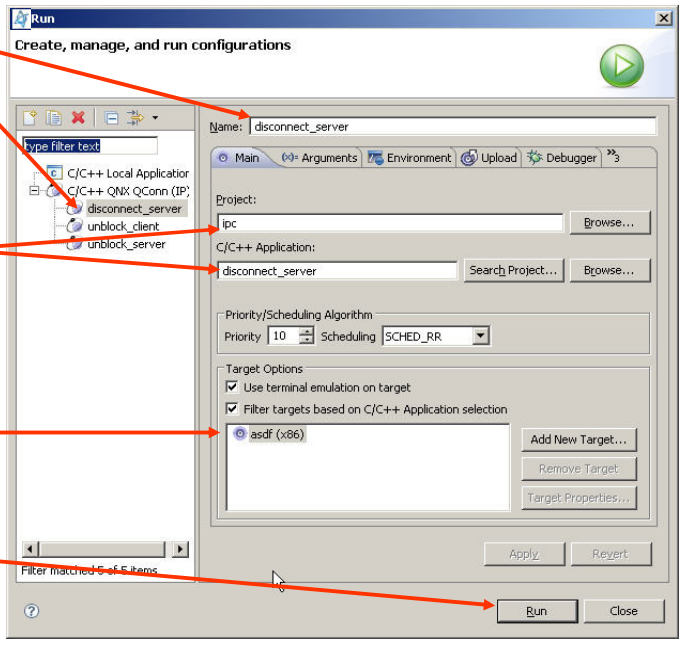
# Basic Launch configuration:

Name for the launch configuration: pick something descriptive

What program to run: specified by Project & Application

Where to run it: select a target system

Click Run to run the program



NOTES:

## Launch Configuration with Arguments

Our exercises will often say something like:

run it as:

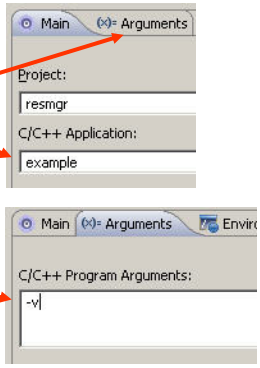
```
example -v
```

To do this from the IDE:

Put the executable name, **example**, in the Application Section

Then click on the arguments tab

And put the arguments, **-v**, in the Arguments field










NOTES:

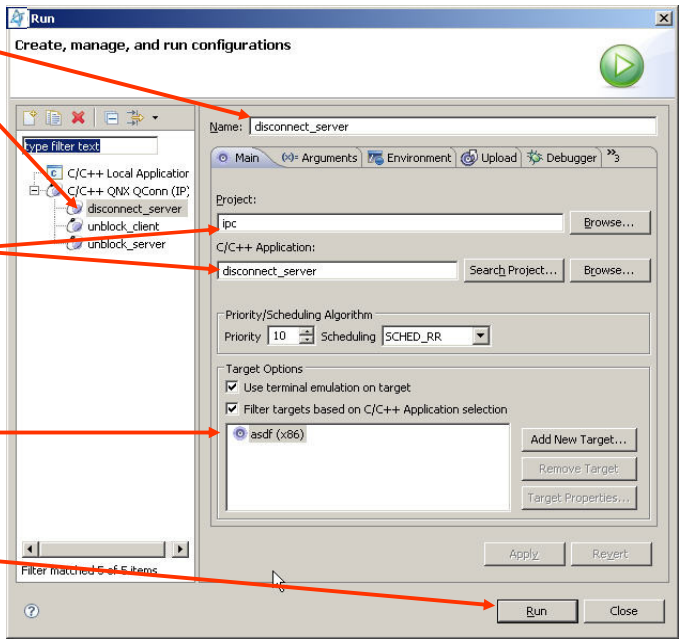
# Basic Launch configuration:

Name for the launch configuration: pick something descriptive

What program to run: specified by Project & Application

Where to run it: select a target system

Click Run to run the program







NOTES:

Topics:

- Eclipse Basics
- Targets
- Projects and Source
- Compiling
- Running and Debugging
- → Exercise
- Conclusion




NOTES:

## Exercise

# Exercise:

- in your **`hello`** project, compile **`hello.c`**
- it has errors and warnings, to demonstrate how the IDE marks build problems
- fix these
- build the project again
- run the program as (something like):

```
hello This is some text
```




NOTES:

Topics:

- Eclipse Basics
- Targets
- Projects and Source
- Compiling
- Running and Debugging
- Exercise
- → Conclusion




NOTES:

## Conclusion

In this section you learned how to:

- edit
- compile
- and run or debug your programs




NOTES: