Compiling and Debugging Basics

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 and debugging

Topics:

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

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)
  - a Perspective is: a collection of views, editors, menu items, and tool bar buttons that are helpful for doing a specific task

The C/C++ Perspective:

ide-7.1-workspace - application_profiling/profile_looper.c - Momentics IDE

menu bar

File Edit Source Refactor Navigate Search Project Run Window Help

toolbar

Run | application_profiling | on: local_VM

Project Expl... | Target Navi...

- application_profiling
  - Binaries
  - Includes
  - profile_looper.c
  - profile_func_instr - [x86_64/le]
  - profile_func_instr.o - [x86_64/le]
  - profile_sample - [x86_64/le]
  - profile_sample.o - [x86_64/le]
  - Makefile
- hello
- images
- interrupt
- ipc
  - Includes
  - solutions
  - client.c
  - disconnect_client.c
  - disconnect_server.c
  - event_client.c
  - event_server.c
  - event_server.h
  - linked_list.h
  - msg_def.h
  - server.c
  - shmem_creator.c

client.c | server.c | profile_looper.c

editor

```
1  /*
2  * appprofile_looper
3  *
4  * This is a sample for demons  e of the application profiler
5  * tool.  It is a very phoney  sample (i.e. it does nothing
6  * practical but is useful for demonstrating profiling).
7  *
8  * If you run it with no command line arguments then it will loop forever.
9  * This is useful for demonstrating profiling while it is still running.
10 *
11 * To have it loop only once, run it with the -o argument.  This is useful
12 * for demonstrating postmortem profiling.
13 *
14 */
15
16 #include <stdlib.h>
17 #include <stdio.h>
18 #include <inttypes.h>
19 #include <sched.h>
20 #include <unistd.h>
21 #include <pthread.h>
```

some views

Outline | Build ...

- stdlib.h
- stdio.h
- inttypes.h
- sched.h
- unistd.h
- pthread.h
- time.h
- other_thread(void*) : voi
- dofuncs(void) : void
- func1(uint64_t) : uint64_t
- func2(uint64_t) : uint64_t
- options(int, char**) : voic
- app_prof_hdlr(int) : void
- loop_once : int
- optv : int
- param : struct sched_par
- policy : int

Problems | Tasks | Console | Properties | Terminal | Target File System Navigator

CDT Build Console [application_profiling]

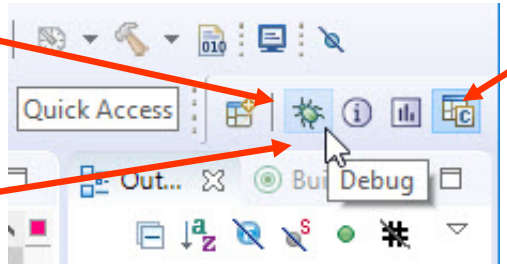
```
qcc -Vgcc_ntox86_64 -g -Wall   -p profile_looper.c -c -o profile_sample.o
qcc -Vgcc_ntox86_64 -g -p profile_sample.o -o profile_sample
qcc -Vgcc_ntox86_64 -g -Wall   -Wc,-finstrument-functions profile_looper.c -c -o profile_func_instr.o
qcc -Vgcc_ntox86_64 -g profile_func_instr.o -lprofilingS -o profile_func_instr
```

Writable | Smart Insert | 1 : 1 : 0 | 135M of 256M

To switch perspectives:

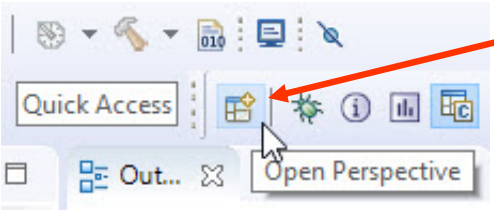
open perspectives are listed in the top-right corner

hovering your mouse will reveal the perspective's name



to switch between them click on or select the perspective icon

To open a new perspective:



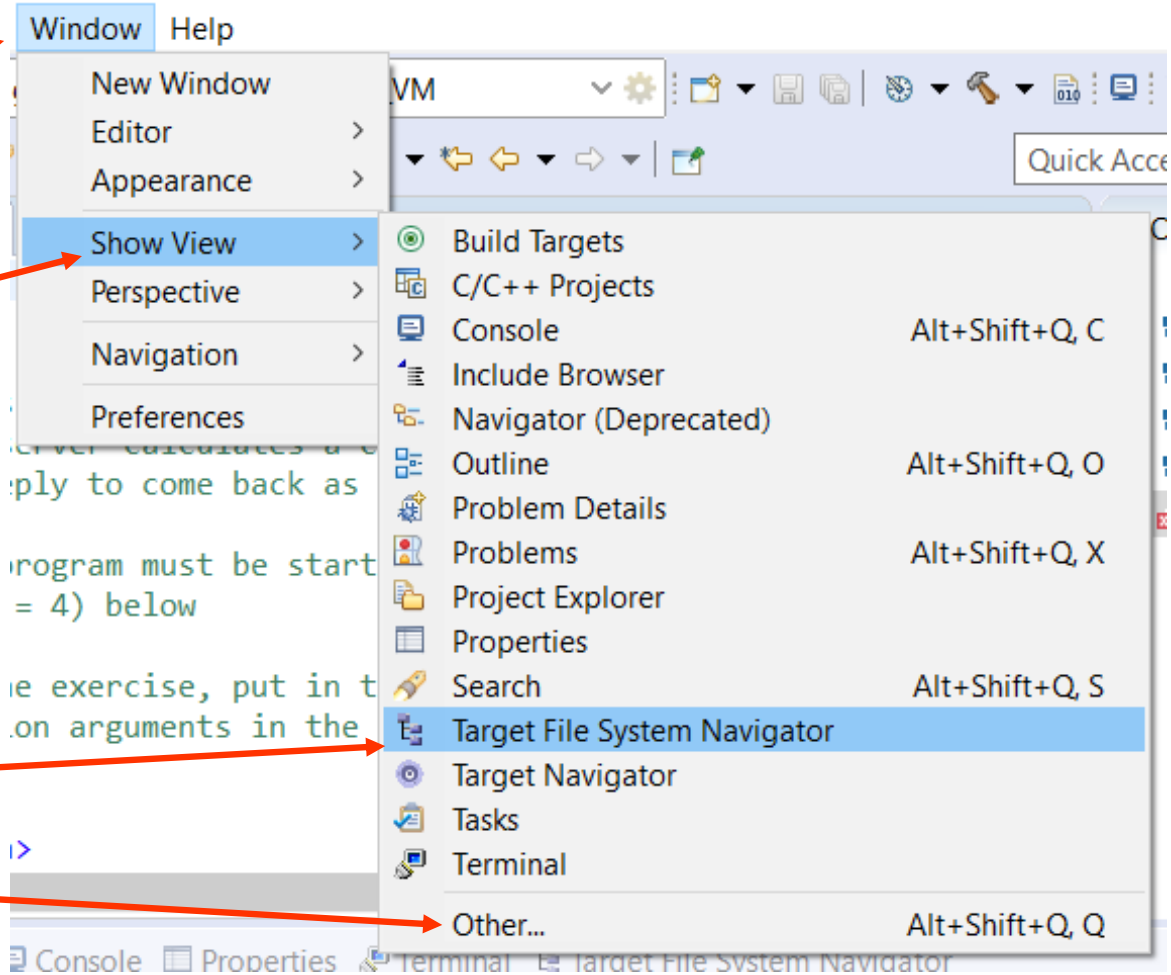
click on the Open Perspective button and choose from the dialog.

To show a particular view:

1. go to the Window menu…

2. … and choose Show View

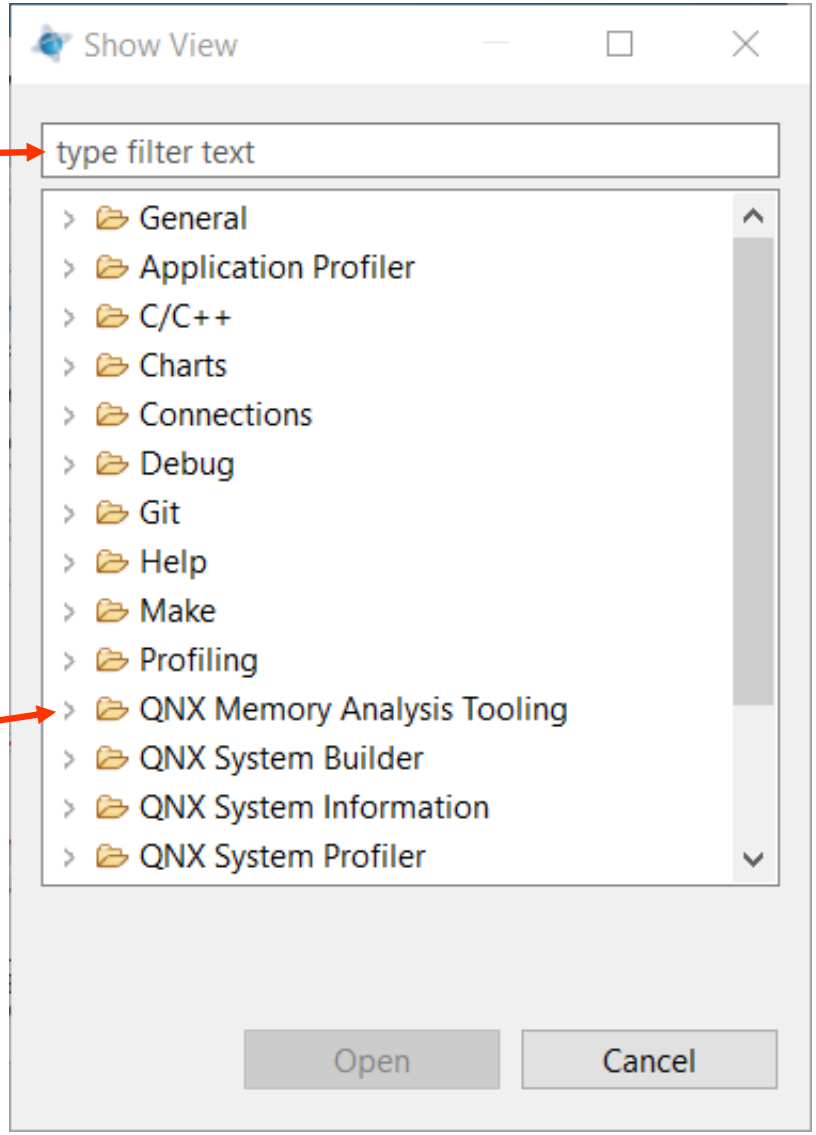
3. pick a view.
If the view you want is not available, choose Other… for a complete list of all the views available...



The Show View dialog:

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

expand the view's category and choose the view you want



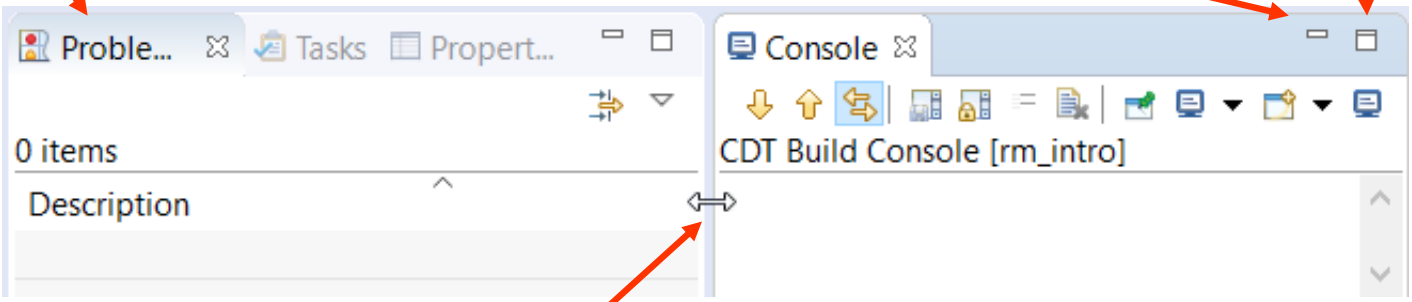
Working with Views:

– 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 group of views as well

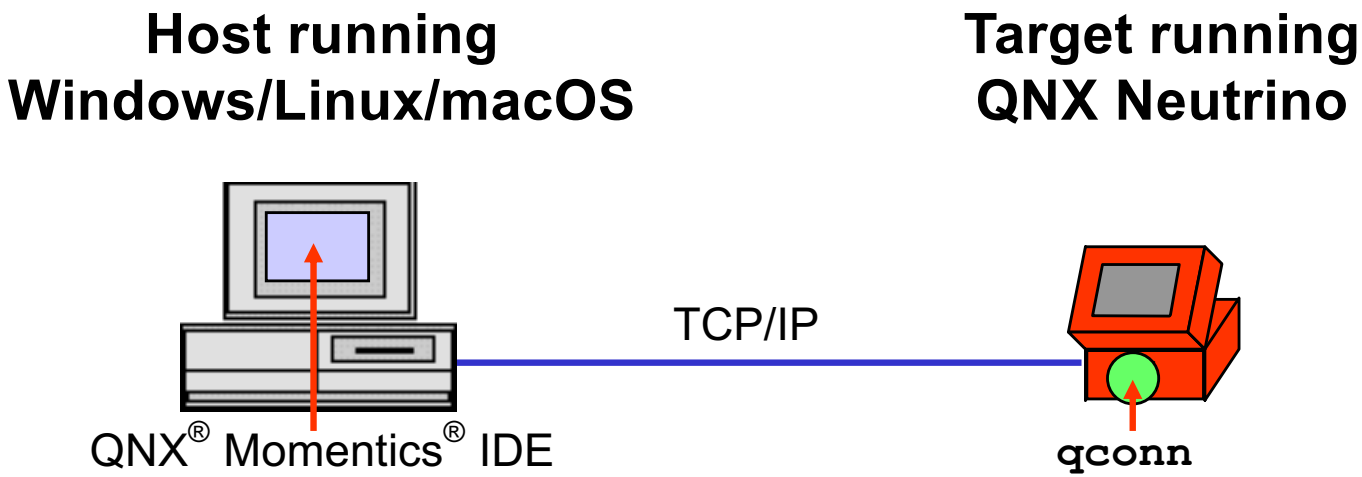


any border between views can be used to resize the views that share the border

Topics:

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

QNX uses a cross-development setup:

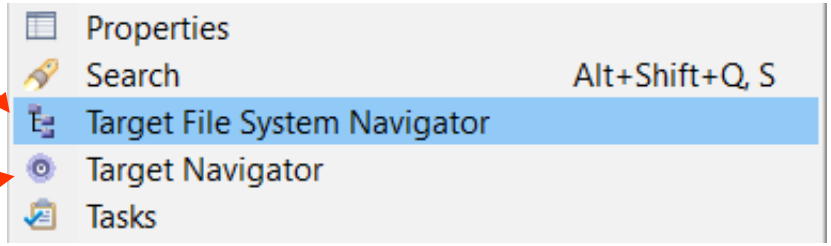


- **`qconn`** is a program on the target that must be running for the IDE to deal with the target
- in the IDE you have to tell it how to find the target:
  - generally this is an IP address or hostname
  - this information is stored in a Target Project
  - once the Target project is created, it can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

Target Interaction – Two Main views

There are two main views for interacting with the target:

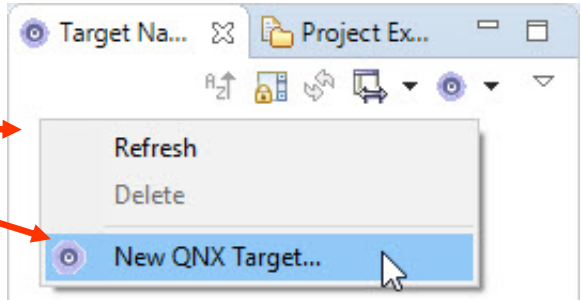
- created from Window→Show View:
- Target File System Navigator:
  - viewing target's file system
  - copying files to/from target
- Target Navigator:
  - creating/deleting Target Projects
  - seeing what processes are running
  - killing processes



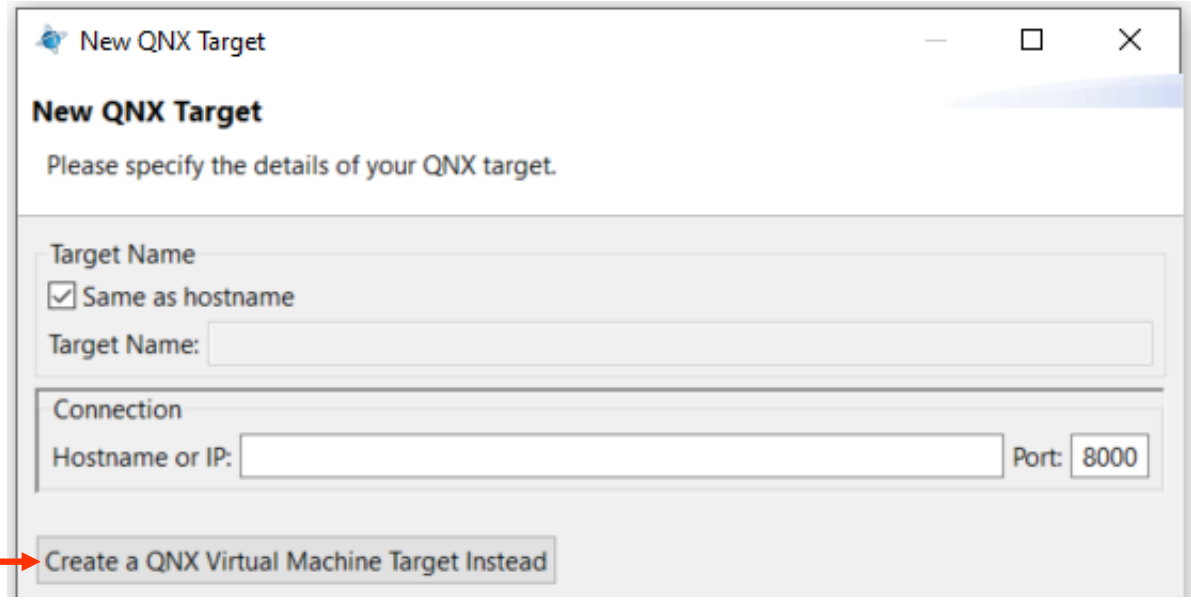
- The QNX System Information perspective has many views for collecting target information
  - it includes a Target Navigator view by default

Creating a Target System project:

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



– at this point you have a choice...



create a target project for an actual target board

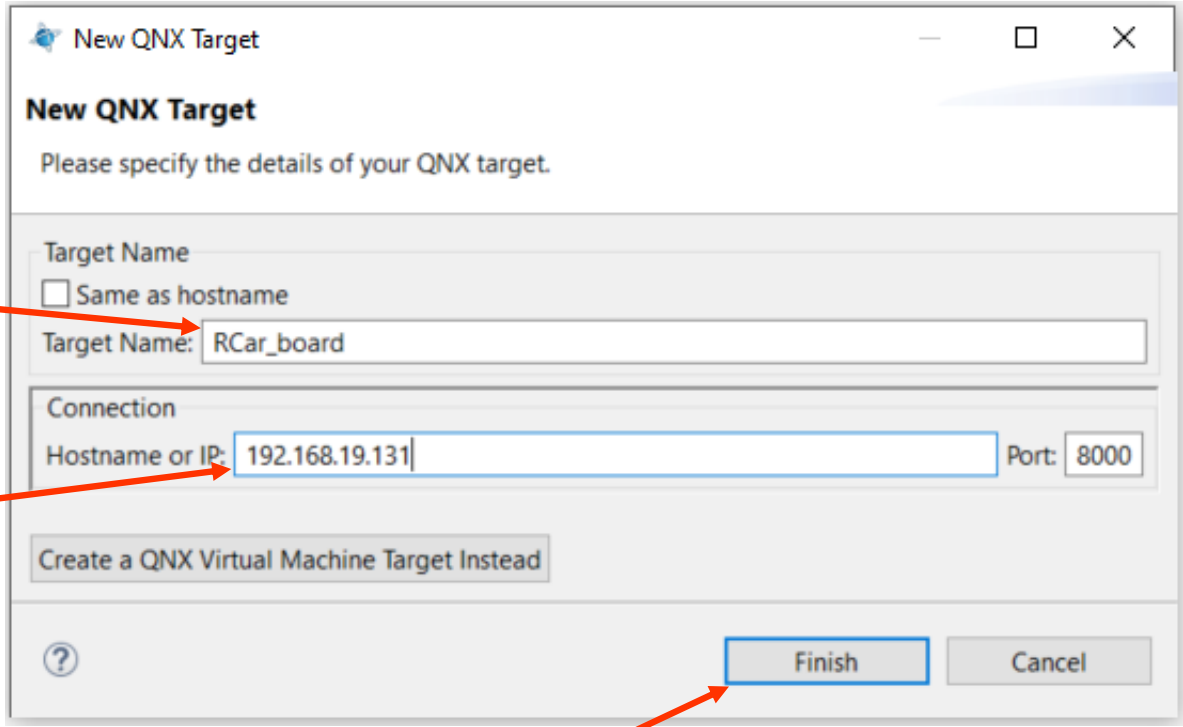
OR

a target project for a virtual machine (e.g. for a Virtual Box VM)

Creating a Target System project for an actual target board:

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

Target System project for a virtual machine:

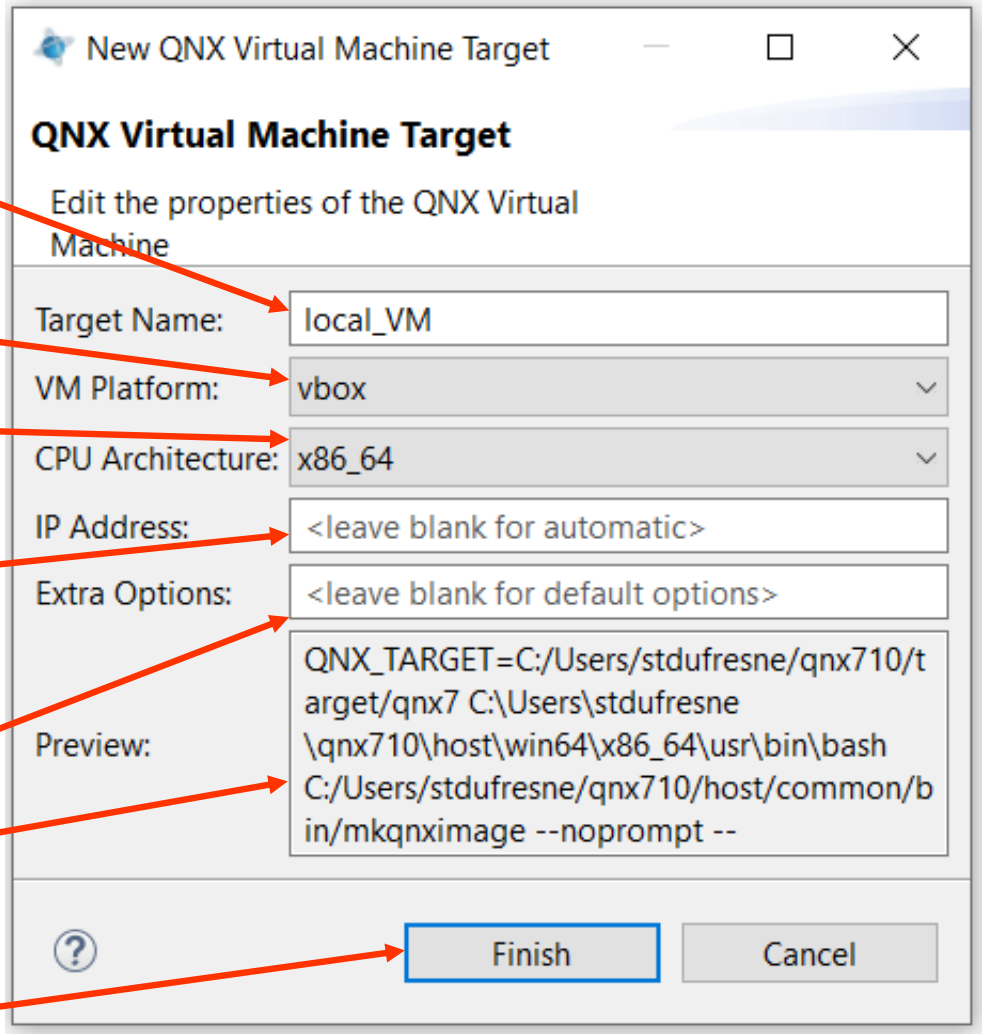
fill in a name representing your target. This will be the Target System project's name

the virtual machine that you're using (e.g. vbox for Virtual Box)

the architecture running on your VM

the IP address can be found automatically once/if your VM is running

put extra command line options here to be added to the **`mkqnximage`** command line used by the IDE to create and run an image for your VM.



choose Finish and your Target System project will be created and **`mkqnximage`** will create and run the image

Command-Line on Target

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

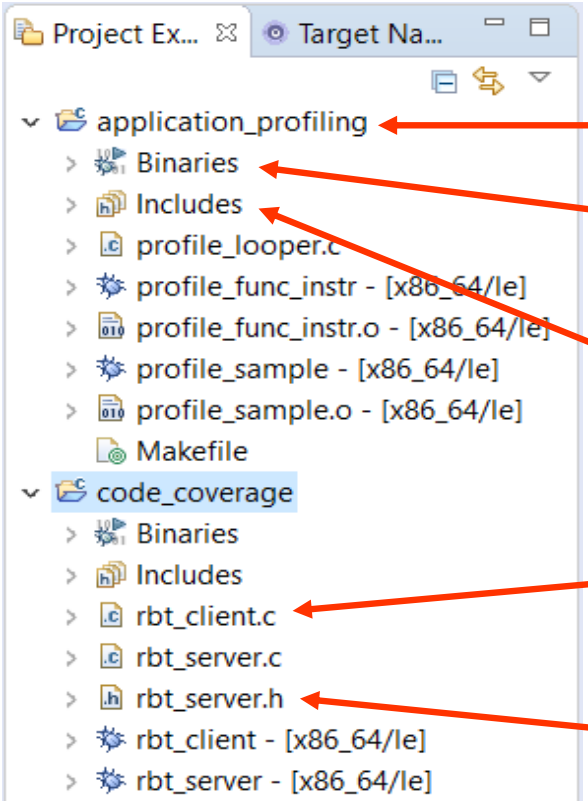
- ssh/telnet session:
  - in the Target Navigator, right click on the Target and select “Start SSH Session…” or “Start Telnet Session”
  - or run a ssh/telnet client manually on your host
- serial connection:
  - generally requires a hardware connection, with a null-modem cable
  - use the Terminal view, 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

Topics:

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

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

Source file; double-click to open for editing

Header file

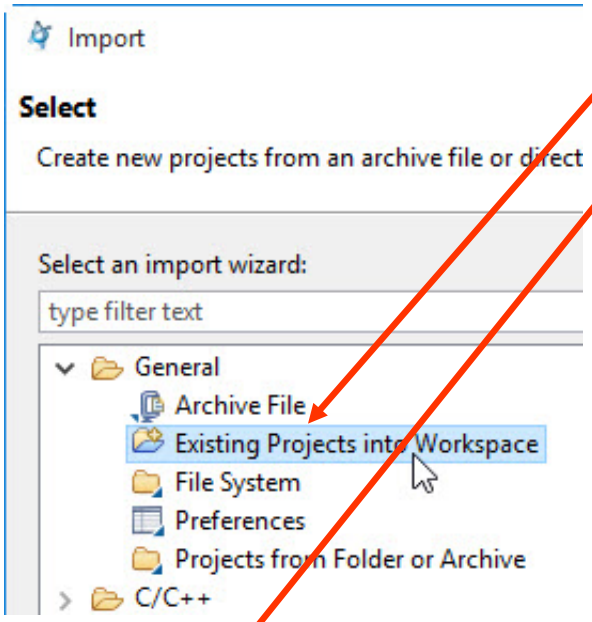
A convenient way of creating projects is to import existing ones:

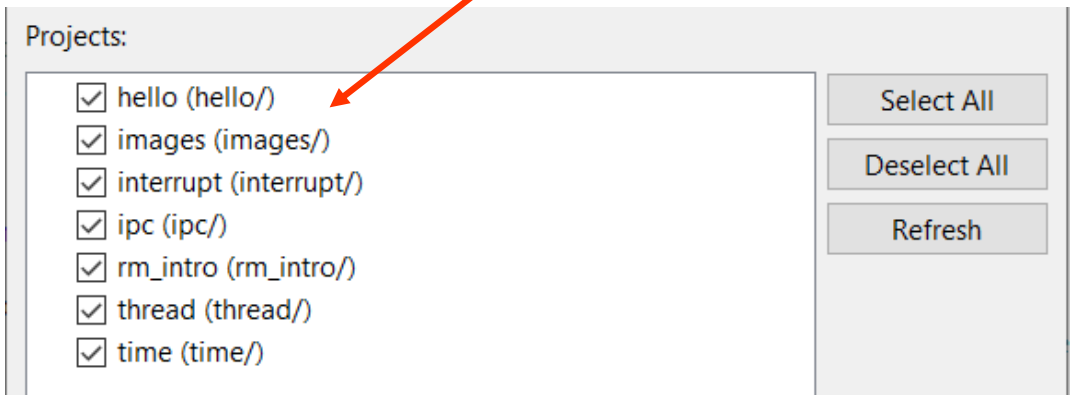
– File -> Import…

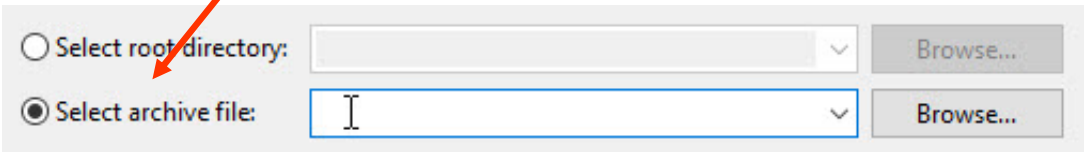
Existing Projects…

Archive file or Directory…

Then select the projects to import…



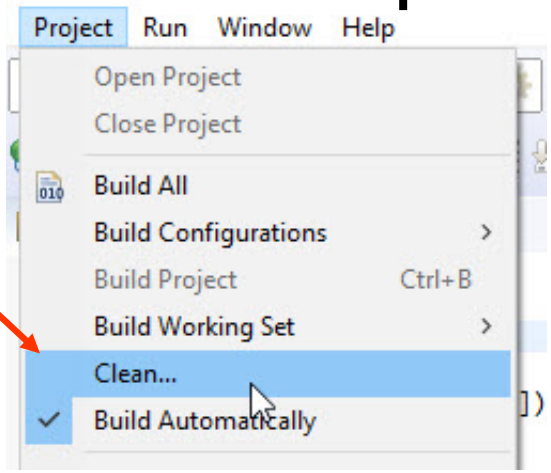




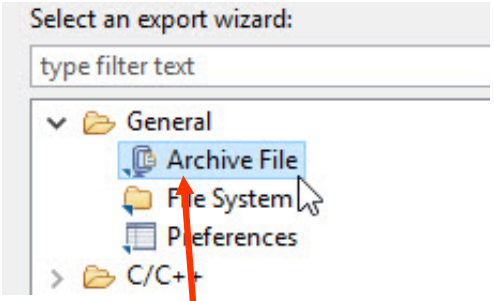
IDE Basics – Exporting Project(s)

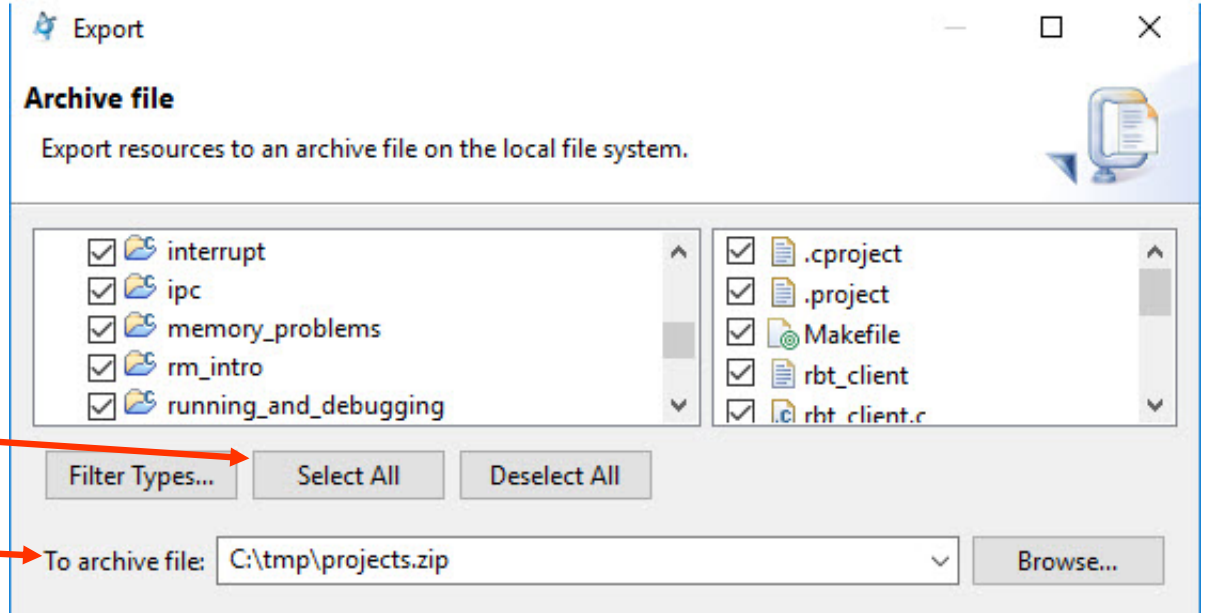
To save all your projects to a zip file:

- Project -> Clean…



- File -> Export





Select Archive File then Next

Select All projects

Browse to or enter zip file name

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

appprofile_looper.c | appprofile_singlethread.c

```
41 {
42     progname = argv[0];
43
44     printf("%s:  starting...\n", progname);
45     options(argc, argv);
46
47     // set up signal handler for handling SIGTERM, which will
48     //        exit() and trigger the dumping of profiling data
49     signal(SIGTERM, app_prof_hdlr);
50
51     if (-1 == pthread_getschedparam (pthread_self(), &policy, &par
52     {
53         perror ("pthread_getschedparam");
54     }
55
56
57     param.sched_priority = 5;  /* run at low priority, 5, so we dc
58     if (-1 == pthread_setschedparam (pthread_self(), policy, &para
59     {
60         perror ("pthread_setschedparam");
61     }
```

Outline | Build Targets

- stdlib.h
- stdio.h
- inttypes.h
- sched.h
- unistd.h
- pthread.h
- other_thread(void*) : void*
- dofuncs(void) : void
- func1(uint64_t) : uint64_t
- func2(uint64_t) : uint64_t
- options(int, char**) : void
- app_prof_hdlr(int) : void
- loop_once : int
- optv : int
- param : struct sched_param
- policy : int
- progname : char*
- main(int, char*[]) : int

Some editing shortcuts:

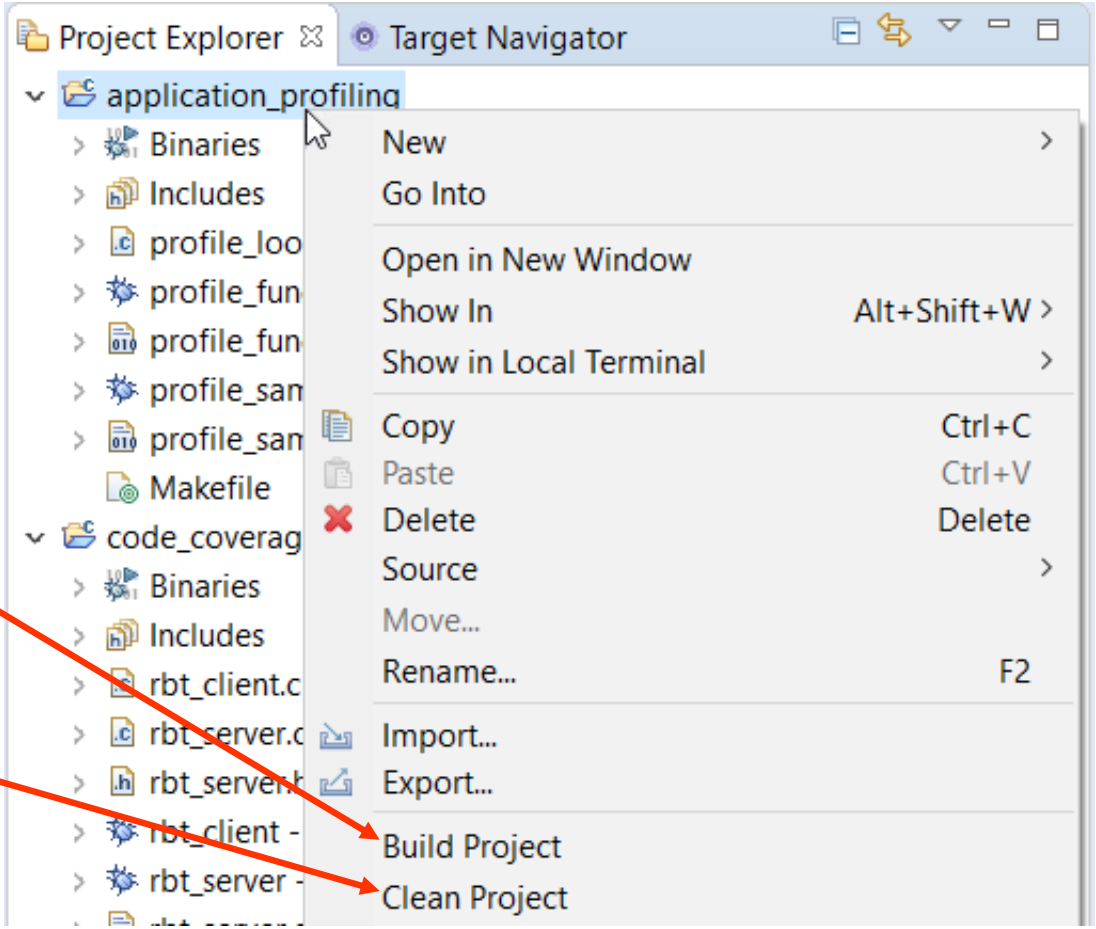
- Ctrl-Space does code completion for functions, structures/classes, and code blocks
- “standard” Windows cut & paste with Ctrl-C, Ctrl-X, Ctrl-V
- undo/redo with Ctrl-Z, Ctrl-Y
- hover-help on functions in library gives quick summary of use and headers
- select function, then
  - Ctrl-Shift-N will insert **`#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

Topics:

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

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, …)

Finding Errors

While building, the console view shows the output from the build, highlighting errors and warnings:

Problems | Tasks | Console | Properties | Target File System Navigator

CDT Build Console [editing_and_compiling]

```
client.c: In function 'main':
client.c:58:27: error: expected ';' before ':' token
  outgoing_string = argv[3]:
                            ^
                            ;
client.c:61:42: warning: implicit declaration of function 'strlen' [-Wimplicit-function-declarati
```

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

Problems | Tasks | Console | Properties | Target File System Navigator

4 errors, 3 warnings, 1 other

Description	Resource	Path	Location	Type
make: *** [client] Error 1	editing_and_c...			C/C++ Problem
make: Target 'all' not remade because of errors.	editing_and_c...			C/C++ Problem
Warnings (3 items)				
implicit declaration of function 'print'; did you mea	client.c	/editing_and_comp...	line 69	C/C++ Problem
implicit declaration of function 'strlen' [-Wimplicit-	client.c	/editing_and_comp...	line 61	C/C++ Problem
incompatible implicit declaration of built-in functi	client.c	/editing_and_comp...	line 61	C/C++ Problem

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)

Quick Access

Project Ex... | Target Na...

- debugging_app_that_uses_shared_lib
- debugging_app_that_uses_shared_lib
- debugging_app_that_uses_static_lib
- debugging_shared_lib
- debugging_static_lib
- editing_and_compiling
  - Binaries
  - Includes
  - client.c
  - server.c
  - client - [x86_64/le]
  - server - [x86_64/le]
  - Makefile

client.c

```
55          perror("ConnectAttach"); //look up error code and print
56      }
57
58      outgoing_string = argv[3];
59      printf("Sending string: %s\n", outgoing_string);
60
61      status = MsgSend(coid, outgoing_string, strlen(outgoing_str
62              &incoming_checksum, sizeof(incoming_checksum));//PU
63      if (-1 == status)
64      { //was there an error sending to server?
65          perror("MsgSend");
66      }
67
68      printf("received checksum=%d from server\n", incoming_check
69      print("MsgSend return status: %d\n", status);
```

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.

Topics:

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

Running

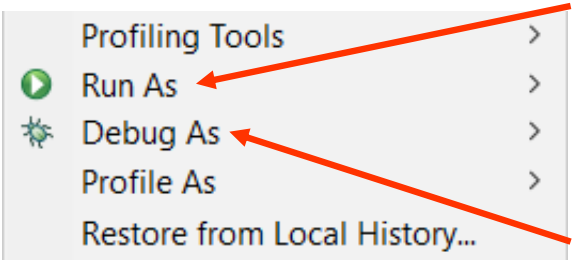
There are two main ways of running a program you've built in the IDE:

- 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

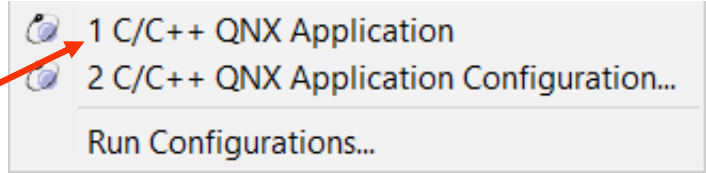
Setup – Launch configuration

To setup a launch configuration for running or debugging:

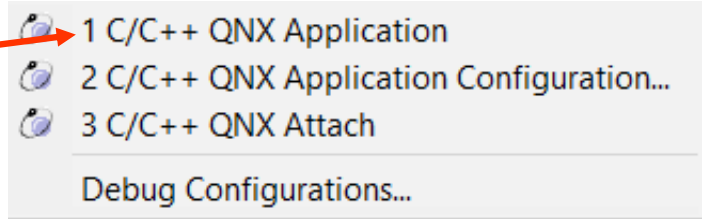
- select the program you want to run in the Project Explorer view, then right click:



create launch config and run



create launch config and debug



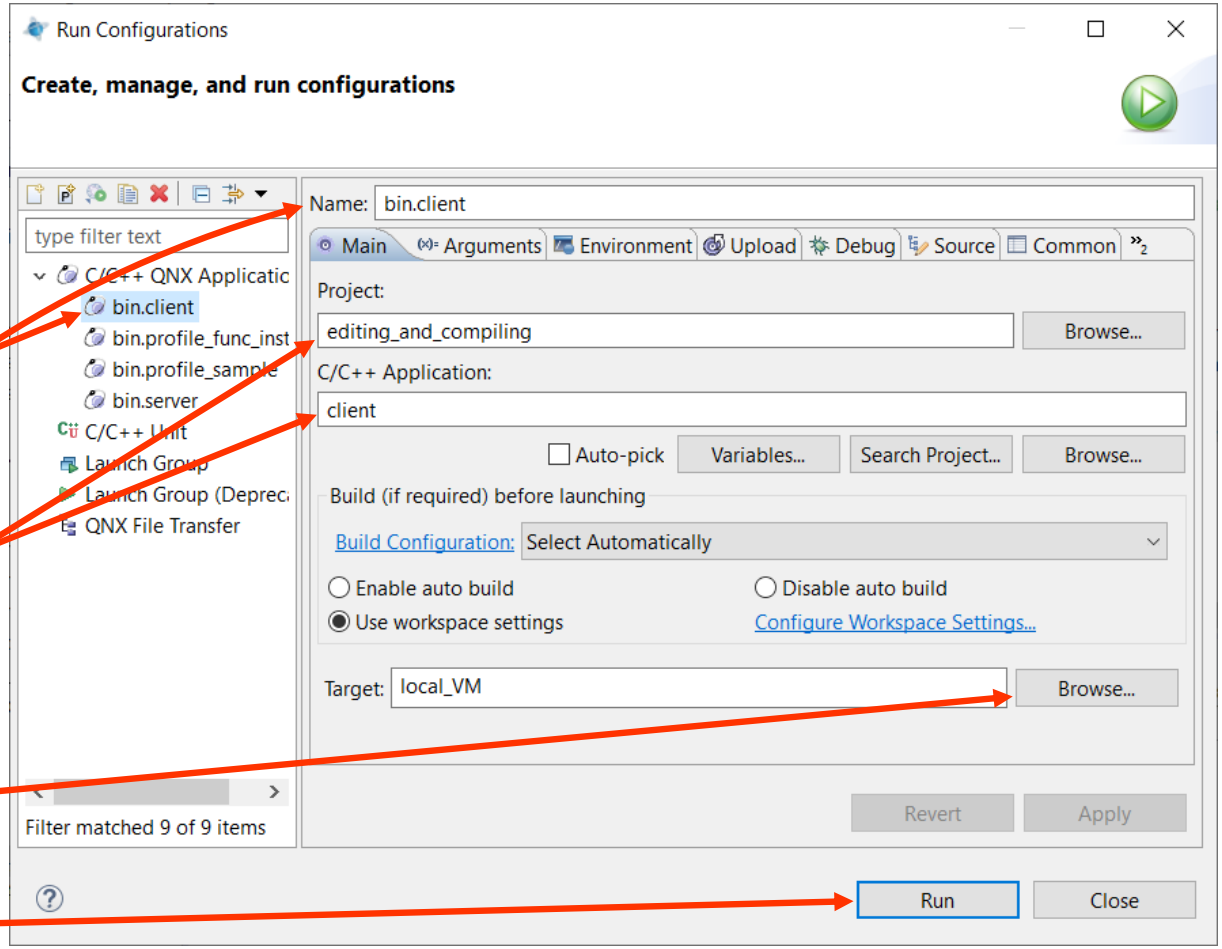
Basic Launch configuration:

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



Launch Configuration with Arguments

Our exercises will often say something like:

run it as:

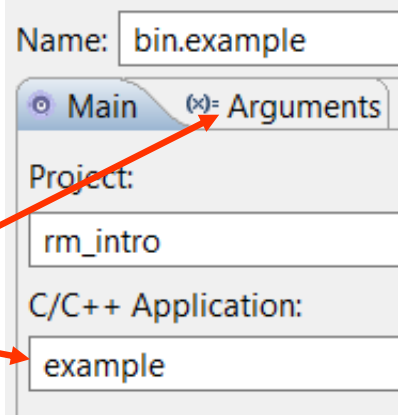
```
example -v
```

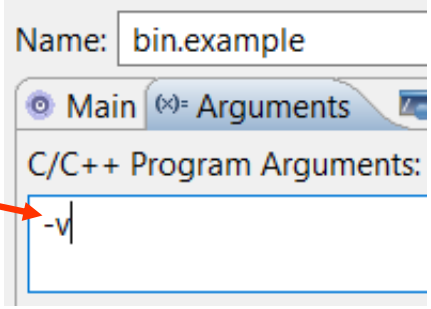
To do this from the IDE:

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

Then click on the arguments tab

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





A debug session:

ide-7.1-workspace - editing_and_compiling/server.c - Momentics IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Debug | bin.server | on: local_VM | Quick Access

Debug | Project Explorer

bin.server [C/C++ QNX Application]
server [4141058]
Thread #1 unnamed (STOPPED) (Suspe
main() at server.c:34 0x8048b90
C:\Users\stdufresne\qnx7 host\win64\x
/tmp/server on local_VM pid 141058 (7/

server.c | client.c

```
26 {
27     int chid;
28     int pid;
29     int rcvid;
30     char msg[256]; //To make it easy let's assume a max
31     int status;
32     int checksum;
33
34     chid = ChannelCreate(0);//PUT CODE HERE to create a
35     if (-1 == chid)
36     { //was there an error creating the channel?
37         perror("ChannelCreate()"); //look up the errno
38         exit(EXIT_FAILURE);
39     }
40
41     pid = pid
42     prin : %d\n", pid, chid);
43     //cl
44     //connect
45
46     while (1)
47     {
48         rcvid = MsgReceive(chid, msg, sizeof(msg), NULL
49         if (rcvid == -1)
```

Variables | Breakpoints | Expressions | Modules

Name	Type	Value
chid	int	1
pid	int	0
rcvid	int	1283943932
msg	char [256]	0x8047c20
status	int	-1258282768
checksu	int	1

Console | Registers | Problems | Executables | Debugger Console | Memory

bin.server [C/C++ QNX Application] /tmp/server on local_VM pid 4141058 (7/22/20 10:52 AM)

Writable | Smart Insert | 35 : 1 : 812 | 215M of 256M

step into, over, to return, ...

set breakpoints

examine/modify variables, breakpoints, expressions ...

see output

examine memory

Topics:

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

Versions – Different versions on the host and target

Having different versions of software on the host and target can cause problems:

- the IDE and GDB use files on the host, but the files run on the target
  - they must be the same version
  - this applies for both QNX's and your pieces
- to check versions:
  - from your target:

    `uname -a`

    `use -i` *filename*
  - from your host:

    `use -i ${QNX_TARGET}/`*path*/*filename*

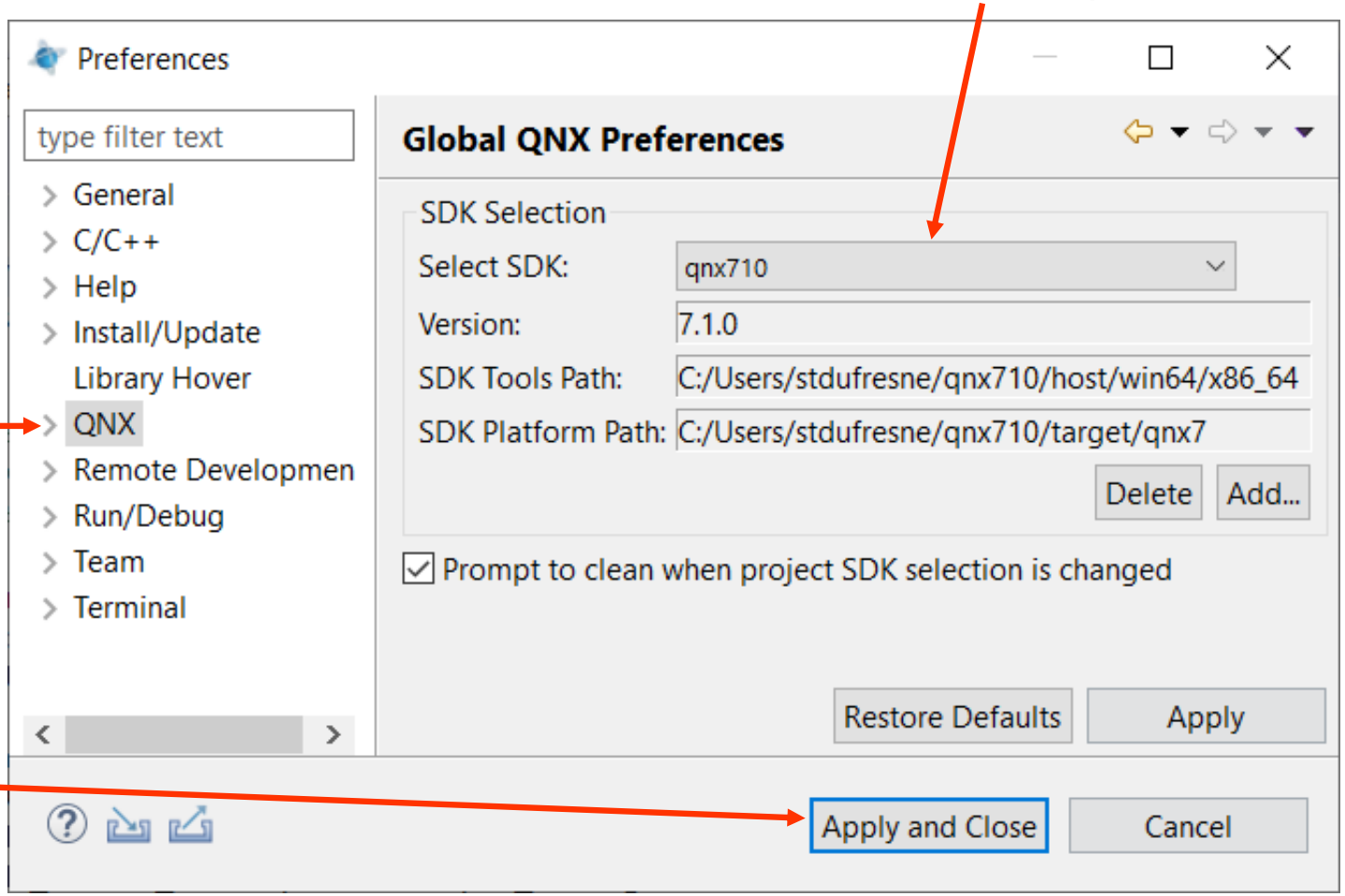
To check/change the SDK version in the IDE:

– go to Window→Preferences

② choose an SDK version from the dropdown list

① select QNX

③ Apply and Close and the IDE will ask for your permission to restart for the change to take effect



Versions – Working with multiple SDKs in the IDE

Working with multiple SDKs can be done by:

- using Windows→Preferences→QNX to switch between SDKs
  - slow since it restarts the IDE
- creating a different workspace for each SDK
  - use File→Switch Workspace to switch between them (also restarts the IDE), or
  - simply run the IDE multiple times, each with a different workspace
    - when each IDE starts up, it asks which workspace you want to work with

Topics:

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

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
```

Topics:

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

Conclusion

In this section you learned how to:

- edit
- compile
- and run or debug your programs