WIND™

# WIND RIVER® LINUX

7 WORKBENCH TUTORIALS

7.0

# Copyright Notice

**Corporate Headquarters**

Wind River
500 Wind River Way
Alameda, CA 94501-1153
U.S.A.
Toll free (U.S.A.): 800-545-WIND
Telephone: 510-748-4100
Facsimile: 510-749-2010

For additional contact information, see the Wind River Web site:

*www.windriver.com*

For information on how to contact Customer Support, see:

*www.windriver.com/support*

22 May 2015

# Contents

1

# Developing Platform Projects

## Creating and Configuring a Platform Project

You define parameters such as the BSP, the kernel, and the size of the target file system when creating and configuring a platform project.

To configure your platform project with Workbench:

**Step 1** Start Workbench.

| Options | Description |
|---|---|
| **From the System menu** | Select **WindRiver** > **Workbench 4** > **Workbench 4**. |
| **From the desktop** | Double-click the **Workbench** icon on your desktop. |

| Options | Description |
|---|---|
| **From the command line** | **1.** Navigate to the Workbench install directory: `$ cd installDir` **2.** Enter the following command: `$ workbench-4/startWorkbench.sh` |

**Step 2** Configure the platform project.

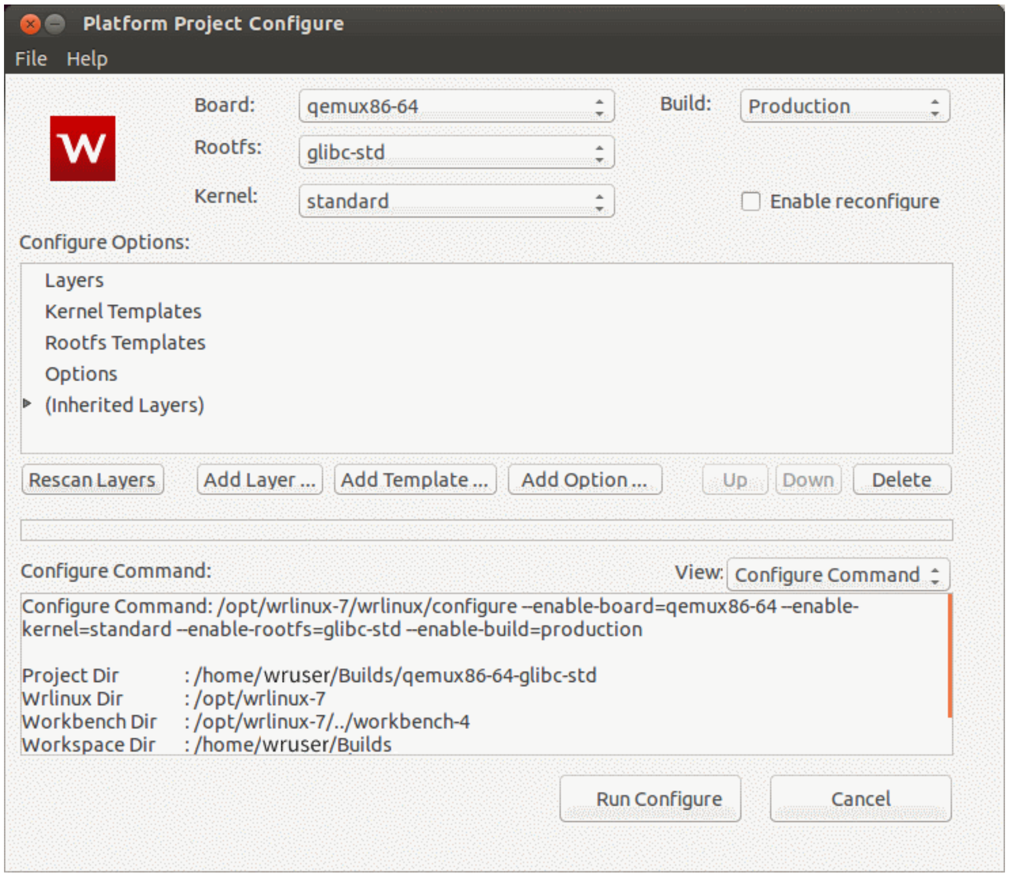
a) Create a new project.

Right-click in the Workbench **Project Explorer** and select **New** > **Project**.

b) Expand the **Wind River Linux** option, select **Wind River Linux Platform Project**, then click **Next**.

c) Give your project a descriptive name, such as qemux86-64-glibc-small, and click **Finish**.

The Platform Project Configure dialog box appears.



```
Configure Command: /opt/wrlinux-7/wrlinux/configure --enable-board=qemux86-64 --enable-
kernel=standard --enable-rootfs=glibc-std --enable-build=production

Project Dir      : /home/wruser/Builds/qemux86-64-glibc-std
Wrlinux Dir      : /opt/wrlinux-7
Workbench Dir    : /opt/wrlinux-7/../workbench-4
Workspace Dir    : /home/wruser/Builds
```

In this example, you will configure and build a platform project using the following settings:

**Board: qemux86-64**
**Rootfs: glibc-small**
**Kernel: standard**

Accept the rest of the default settings in the Platform Project Configure dialog box. This creates a project with a standard kernel and a small file system for a qemux86-64 board.

d) You can configure additional options to greatly improve the time it takes to build your project.

You can use the following configuration options to set values for parallel builds:

- **--enable-jobs=***n* specifies the maximum number of parallel jobs (threads) that make can perform while building a single package. If this is not set, the default is 1.
- **--enable-parallel-pkgbuilds=***n* specifies the maximum number of packages that may be built in parallel.

**NOTE:** BitBake automatically sets defaults for parallel jobs and packages to the number of cores on the machine.

**NOTE:** Workbench uses **ccache** by default, so there is no need to add a configure option for it.

e) Click **OK**.

**Step 3** Click **Finish** to configure the project.

## Building a Platform Project Image

After configuring a platform project you must build it.

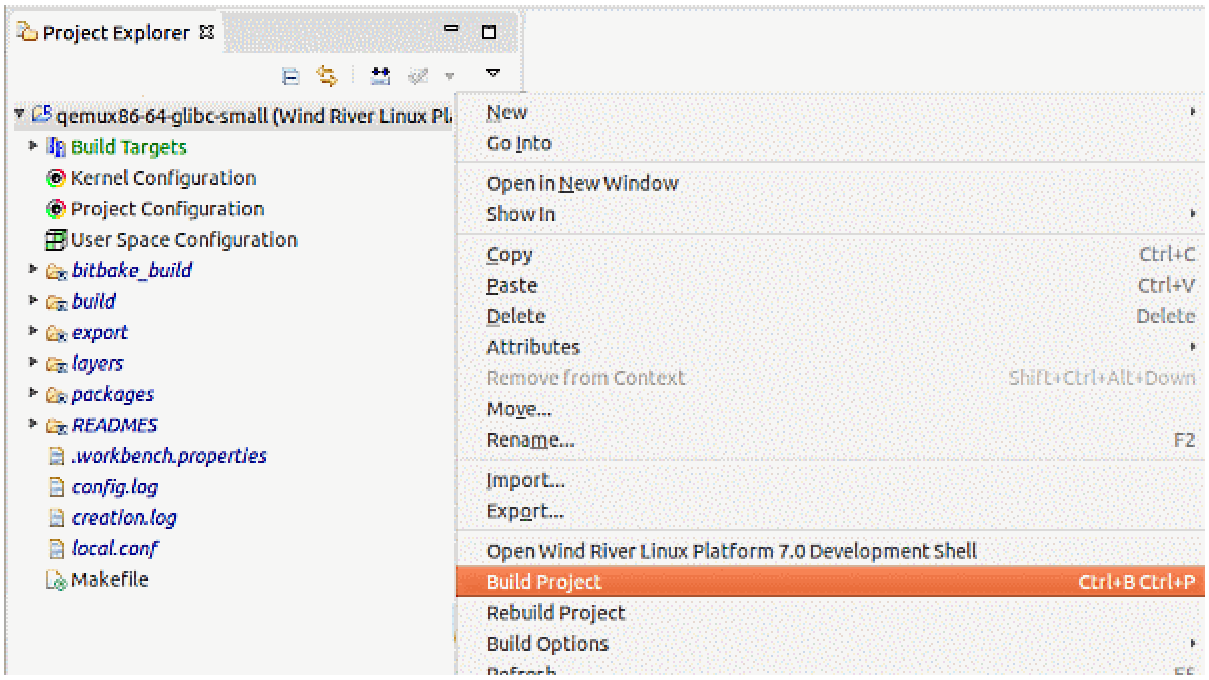
To build your platform project with Workbench:

**Step 1** Expand the project in Workbench.

If necessary, expand the project by clicking on the arrow to the left of the project name. Inside the project you can see various tools, build targets, and files associated with the project.

**Step 2** Build the project.

Right-click on the platform project folder in **Project Explorer** and select **Build Project**.



Building could take from minutes to hours, depending on your development resources and project configuration. When the build is finished, you will have a kernel and file system ready for deployment to a real or simulated target as described in *Creating an Application Project with Workbench* on page 17.

## Configuring a New Project to Add Application Packages

You can add application packages to your platform project using the New Platform Project wizard.

This topic describes how to add Wind River Linux layers and templates to a platform project configuration. For details on adding custom layers and templates, see: *Wind River Linux User's Guide*.

**NOTE:** The functionality added in this example (the target-resident debugger) is currently only supported on targets with the x86 architecture (32and 64-bit), but the workflow for adding a non-default layer and templates is the same for all architectures and BSPs.

To customize a platform project with Workbench:

**Step 1** Start Workbench.

| Options | Description |
|---|---|
| **From the System menu** | Select **WindRiver** > **Workbench 4** > **Workbench 4**. |

| Options | Description |
|---|---|
| **From the desktop** | Double-click the Workbench icon on your desktop. |
| **From the command line** | **1.** Navigate to the Workbench install directory: `$ cd installDir` **2.** Enter the following command: `$ workbench-4/startWorkbench.sh` |

**Step 2** Configure the platform project.

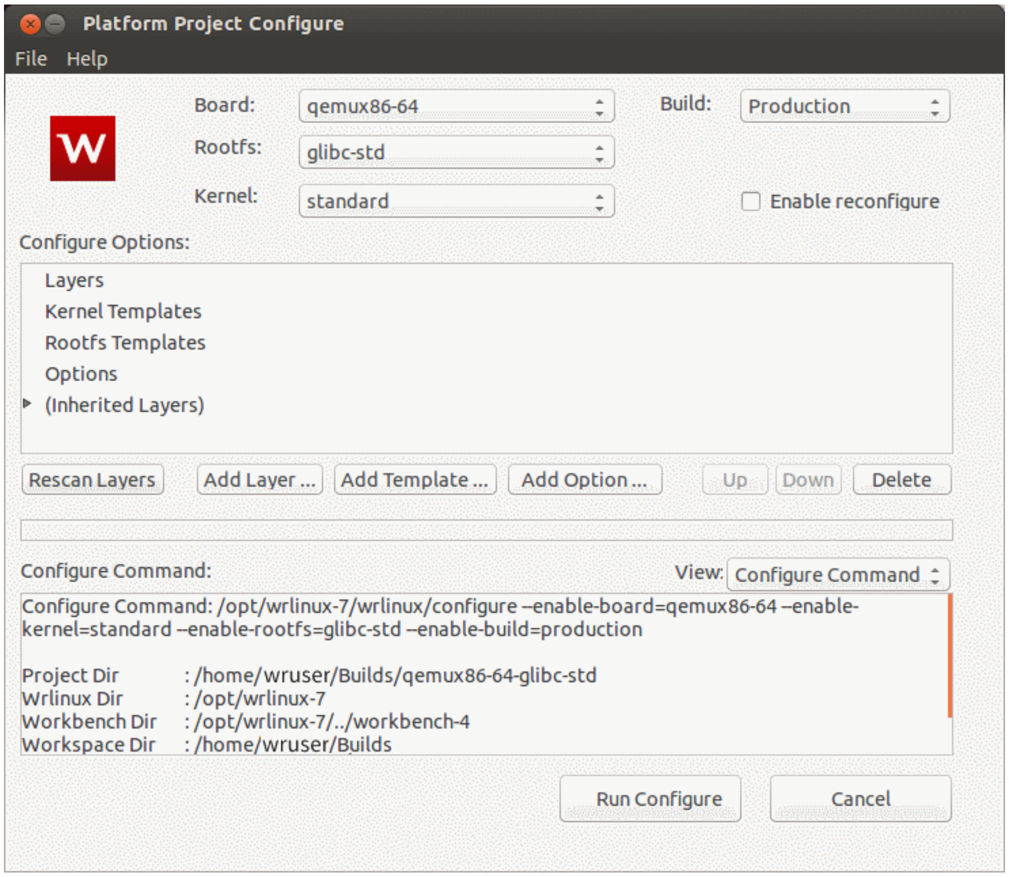
a) Create a new project.

Right-click in the Workbench **Project Explorer** and select **New** > **Project**.

b) Expand the **Wind River Linux** option, select **Wind River Linux Platform Project**, then click **Next**.

c) Give your project a descriptive name, such as qemux86-64_glibc-small, and click **Finish**.

The Platform Project Configure window appears.



Board: qemux86-64
Rootfs: glibc-std
Kernel: standard
Build: Production
Enable reconfigure

Configure Options:
Layers
Kernel Templates
Rootfs Templates
Options
(Inherited Layers)

Configure Command:
View: Configure Command

```
Configure Command: /opt/wrlinux-7/wrlinux/configure --enable-board=qemux86-64 --enable-
kernel=standard --enable-rootfs=glibc-std --enable-build=production

Project Dir      : /home/wruser/Builds/qemux86-64-glibc-std
Wrlinux Dir      : /opt/wrlinux-7
Workbench Dir    : /opt/wrlinux-7/../workbench-4
Workspace Dir    : /home/wruser/Builds
```

In this example, you will configure and build a platform project using the following settings:

**Board: qemux86-64**
**Rootfs: glibc-small**
**Kernel: standard**

d) Accept the rest of the default settings in the Platform Project Configure dialog box. This creates a project with a standard kernel and a small file system for a qemux86-64 board.

e) You can configure additional options to greatly improve the time it takes to build your project. This step is optional.

You can use the following configuration options to set values for parallel builds.

- **--enable-jobs=***n* specifies the maximum number of parallel jobs (threads) that make can perform while building a single package. If this is not set, the default is 1.
- **--enable-parallel-pkgbuilds=***n* specifies the maximum number of packages that may be built in parallel.

**NOTE:** BitBake automatically sets defaults for parallel jobs and packages to the number of cores on the machine.

**NOTE:** Workbench uses **ccache** by default, so there is no need to add a configure option for it.

f) Click **OK**.

**Step 3** Configure additional functionality.

a) In the Platform Project Configure window, click **Add Template**.

The Add Templates window appears.

b) In the RootFS/Templates section, select the checkbox for the **feature/debug** template.

This template adds **gdb** and **gdbserver** to your platform project to provide native and on-target debugging functionality.

**NOTE:** This feature is visible in the template list because it exists in one of the default layers that are always included in configurations.

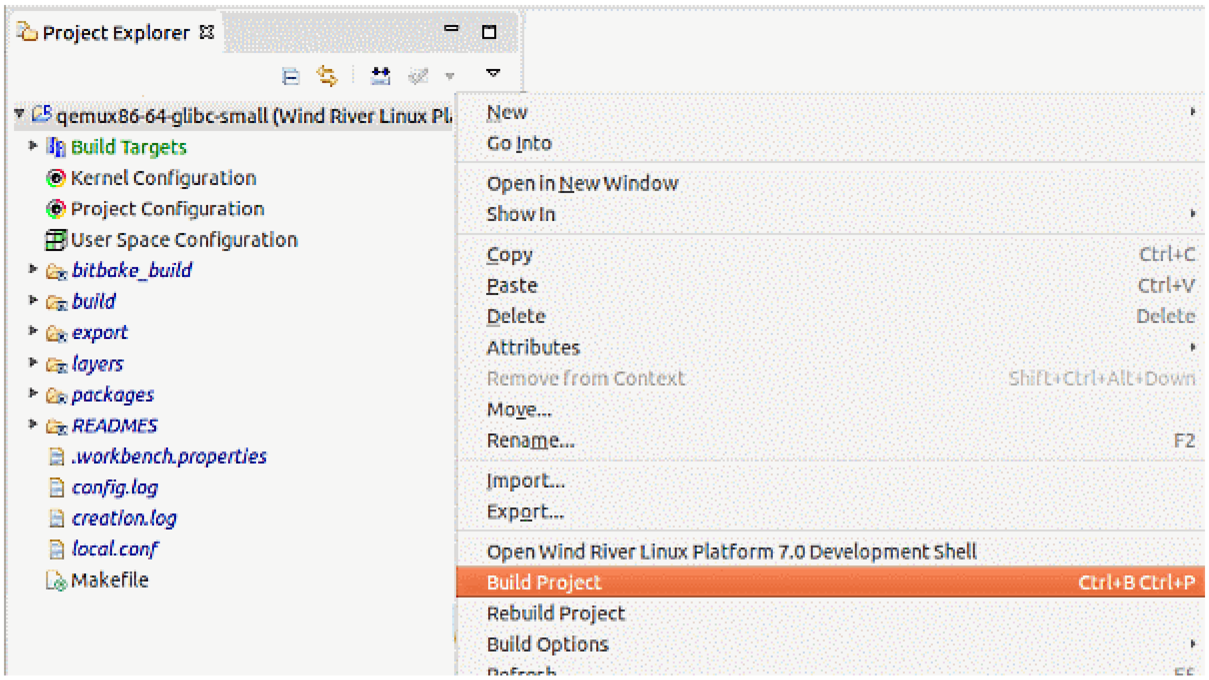
c) Click **Done** to include the **feature/debug** template in your configuration.

d) In the Platform Project Configure window, click **Run Configure**.

Project configuration will begin, and display progress in the window. Once it completes, click **Finish** to close the Platform Project Configure window.

**Step 4** Build the platform project.

In the Project Explorer view, right-click on the project and select **Build Project**.



This will take from minutes to hours, depending on your development resources. When the build is finished, you will have a kernel and file system ready for deployment to a target or an emulator.

**Step 5** Verify that the package was added successfully.

See *Verifying that the Project Includes the New Application Package* on page 12.

## Adding New Application Packages to an Existing Project

You can add application packages and new features to a platform project using Workbench.

In the following example, you will add **gdb** to a previously configured and built platform project using Workbench, as detailed in the following sections:

- *Creating and Configuring a Platform Project* on page 5
- *Building a Platform Project Image* on page 7

This example assumes that you do not already have **gdb** included with your platform project.

**Step 1** Open a development shell in the platform project's directory.

In Project explorer, right-click on the platform project folder and select **Open Wind River Linux Platform 7.0 Development Shell**.

**Step 2** Build and verify the **gdb** package.

a) Build **gdb**.

From the **project** directory, enter the following command:

```
$ make gdb
```

Building the package takes a couple of minutes, during which you will see the progress on your terminal.

b) Verify that **gdb** now exists in your package directory.

Issue the following command:

```
$ find export/RPMS/ -name '*gdb*'
```

The system returns the following output:

```
export/RPMS/x86_64/libgdbm-staticdev-1.11-r0.0.core2_64.rpm
...
export/RPMS/i686/libgdbm-bin-1.11-r0.1.i586.rpm
export/RPMS/i686/libgdbm4-1.11-r0.1.i586.rpm
export/RPMS/i686/python-gdbm-2.7.3-r0.3.1.i586.rpm
```

**Step 3** Add the **gdb** package to the platform project build.

From the project directory, enter the following command to add the **gdb** package:

```
$ make gdb.addpkg
```

The system will return the following output:

```
make: Entering directory `/Builds/qemux86-64_small/build'
==Checking ../layers/local/recipes-img/images/wrlinux-image-glibc-small.bbappend==
==Checking for valid package for gdb==
...
=== ADDED gdb to ../layers/local/recipes-img/images/wrlinux-image-glibc-small.bbappend
===
```

**Step 4** Rebuild the root file system.

In **Project Explorer**, right-click on the project select **Build Project**.

**Step 5** Verify that the package was added successfully.

See *Verifying that the Project Includes the New Application Package* on page 12.

## Verifying that the Project Includes the New Application Package

You should verify that application packages added to a platform project are working properly.

In this example, you are going to verify that a package was added successfully to the platform project using Workbench.

This procedure assumes you have added the gdb package from *Configuring a New Project to Add Application Packages* on page 8, or *Building a Platform Project Image* on page 7.

**Step 1** Verify that the **gdb** binary has been added to the file system of the platform project.

a) Expand the platform project in **Project Explorer** to display the **export** folder.
b) Expand this folder to reveal the contents of **/dist/usr/bin**.
c) Locate the **gdb** and **gdbserver** files in the list.

**Step 2** Run the **gdb** binary on the target.

a) Deploy the platform project on a QEMU target.

See *Deploying an Application Project with Workbench* on page 23.

b) Run the **gdb** command on the target:

```
# gdb
```

The system returns the following and display the (**gdb**) prompt to confirm **gdb** is working:

```
GNU gdb (Wind River Linux Sourcery CodeBench 4.6-60) 7.2.50.20100908-cvs
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-wrs-linux-gnu".
For bug reporting instructions, please see:
<support@windriver.com>
(gdb)
```

You may use the debugger, or type **quit** to return to the command prompt

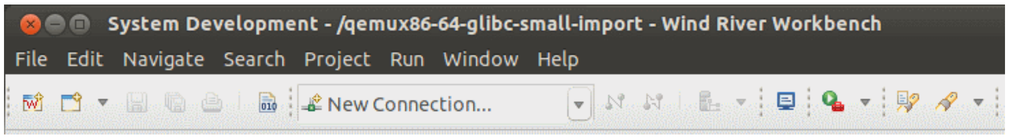
## Deploying a Platform Project Image

Use the New Connection wizard to create and automatically deploy QEMU simulated targets for development and testing.

Use the System Management view or toolbar to create, connect to, or disconnect from, your QEMU target. The following procedure describes how to create and launch a new QEMU target connection.

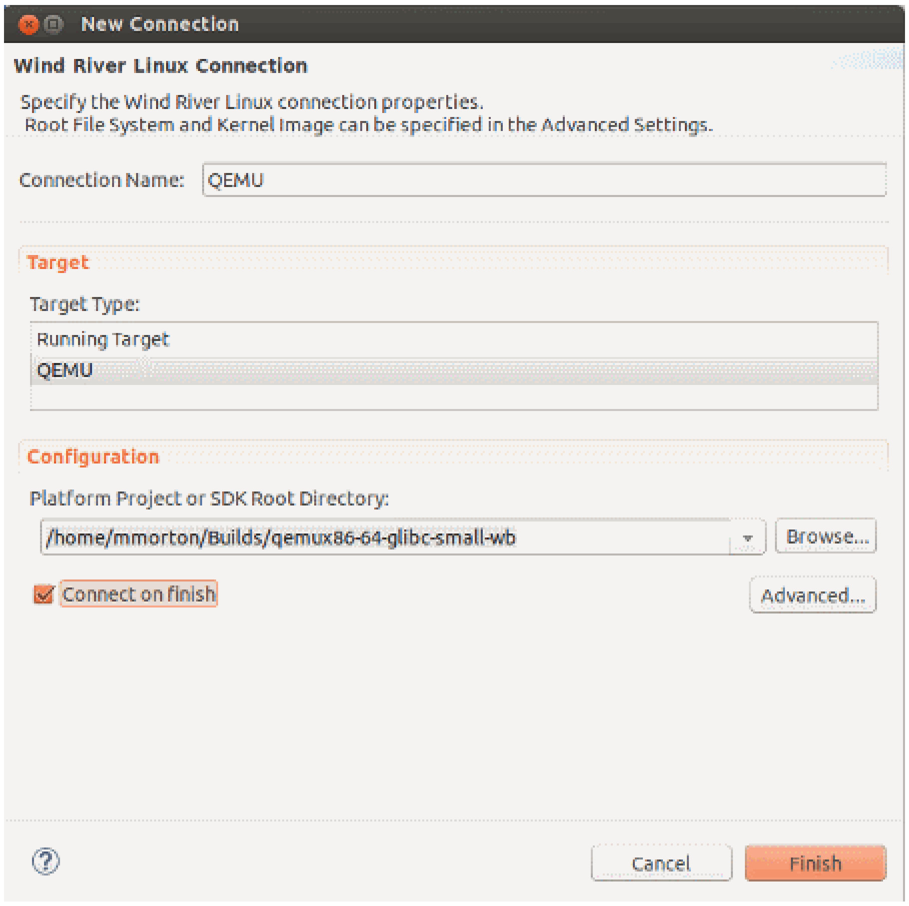
**Step 1** Create a new QEMU connection in Workbench.

a) Click **New Connection** in the System Management toolbar to open the New Connection window.


System Development - /qemux86-64-glibc-small-import - Wind River Workbench
File
Edit
Navigate
Search
Project
Run
Window
Help
New Connection...

b) In the New Connection window, select a target option.

Select **QEMU** in the **Target Type** section.


New Connection

Wind River Linux Connection

Specify the Wind River Linux connection properties.
Root File System and Kernel Image can be specified in the Advanced Settings.

Connection Name: QEMU

Target

Target Type:
Running Target
QEMU

Configuration

Platform Project or SDK Root Directory:
/home/mmorton/Builds/qemux86-64-glibc-small-wb

[✓] Connect on finish

c) Enter target connection information.

   Click **Browse** and navigate to the platform project or SDK root directory, then click **OK** once selected.

---

**NOTE:** Windows development does not support launching QEMU targets from the SDK root directory. I you are using Workbench in Windows, you must connect to a running target.

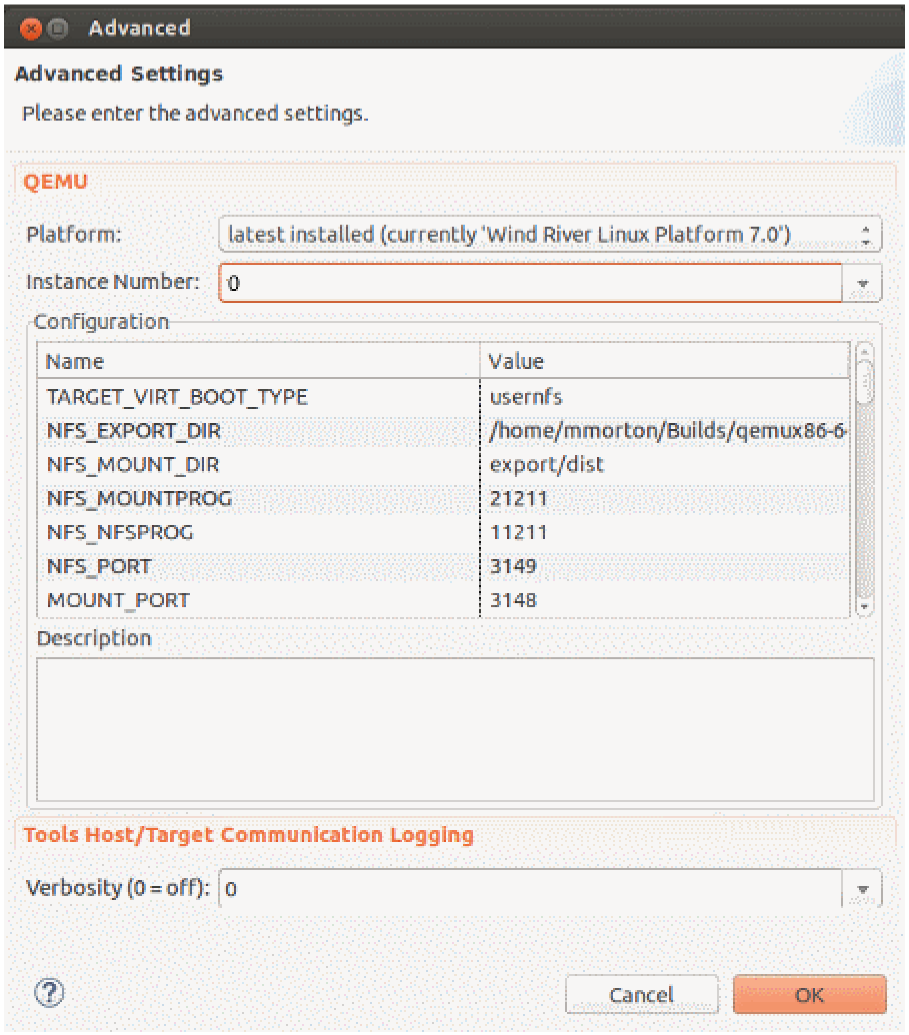
---

d) Select the **Connect on finish** checkbox.

   This will automatically connect Workbench to the target once the launch completes.

**Step 2** Optionally set QEMU configuration options.

Click **Advanced** to view the advanced settings available for a QEMU target.



Please enter the advanced settings.

QEMU

Platform: latest installed (currently 'Wind River Linux Platform 7.0')

Instance Number: 0

Configuration

| Name | Value |
|---|---|
| TARGET_VIRT_BOOT_TYPE | usernfs |
| NFS_EXPORT_DIR | /home/mmorton/Builds/qemux86-6 |
| NFS_MOUNT_DIR | export/dist |
| NFS_MOUNTPROG | 21211 |
| NFS_NFSPROG | 11211 |
| NFS_PORT | 3149 |
| MOUNT_PORT | 3148 |

Description

Tools Host/Target Communication Logging

Verbosity (0 = off): 0

Use the **Instance Number** field to specify a QEMU instance manually. This allows you to launch multiple QEMU simulations in parallel, and is the same as using the **TOPTS=-***n* from the command line. Note that instances are assigned automatically when a new QEMU connection is created.

In the **Configuration** section, the available options are the equivalent of running the **make config-target** command from the command line. Use these to specify many aspects of your simulation.

Use the **Tools Host/Target Communication Logging** section to specify the amount of logging the system provides, where a setting of **0** (zero) equates to no logging.

Once you are finished, click **OK** to save your settings.

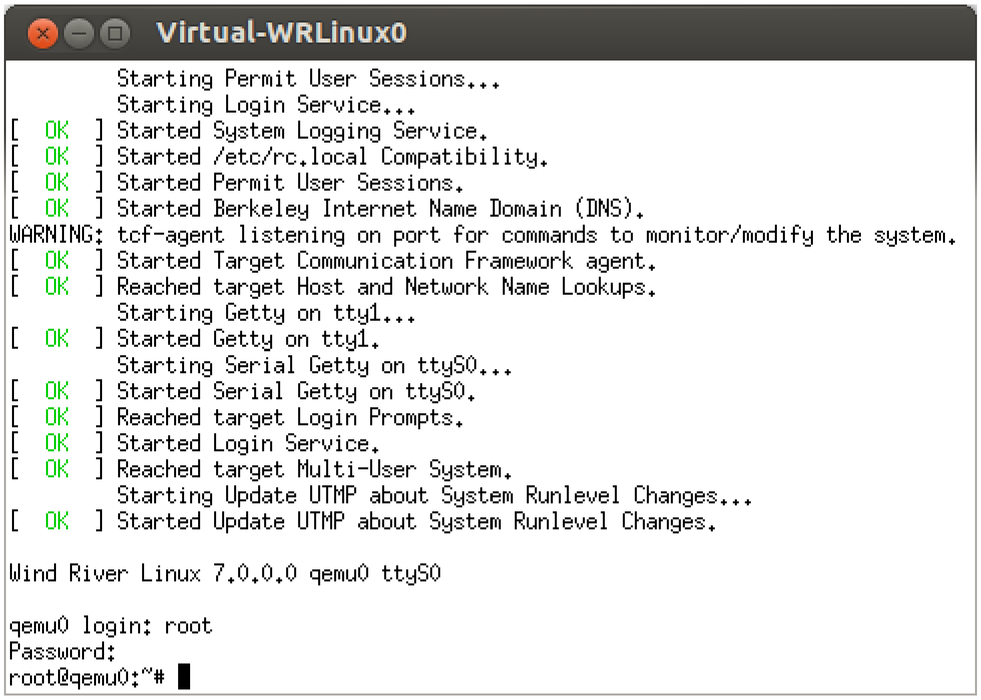
**Step 3** Finish the wizard and launch the QEMU target window.

a) Click **Finish**.

The QEMU window will launch and complete the boot process.

b) Log into the QEMU target.

Enter user name and password **root** to log into the target.



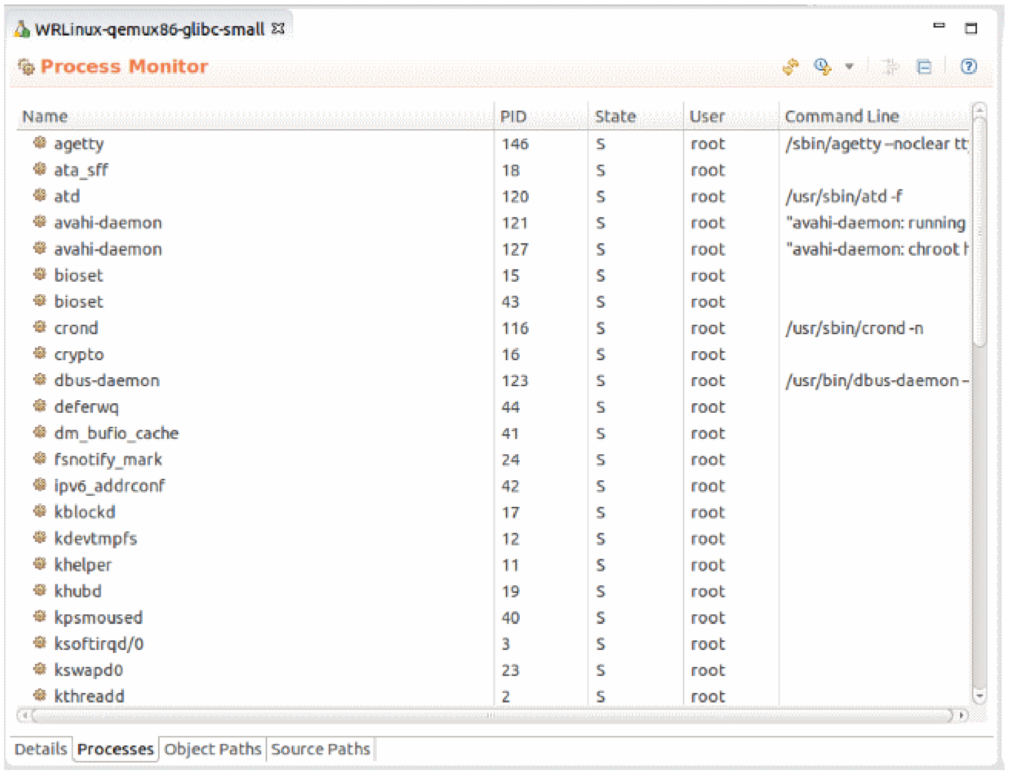
```
         Starting Permit User Sessions...
         Starting Login Service...
[  OK  ] Started System Logging Service.
[  OK  ] Started /etc/rc.local Compatibility.
[  OK  ] Started Permit User Sessions.
[  OK  ] Started Berkeley Internet Name Domain (DNS).
WARNING: tcf-agent listening on port for commands to monitor/modify the system.
[  OK  ] Started Target Communication Framework agent.
[  OK  ] Reached target Host and Network Name Lookups.
         Starting Getty on tty1...
[  OK  ] Started Getty on tty1.
         Starting Serial Getty on ttyS0...
[  OK  ] Started Serial Getty on ttyS0.
[  OK  ] Reached target Login Prompts.
[  OK  ] Started Login Service.
[  OK  ] Reached target Multi-User System.
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Started Update UTMP about System Runlevel Changes.

Wind River Linux 7.0.0.0 qemu0 ttyS0

qemu0 login: root
Password:
root@qemu0:~#
```

**Step 4** View the target's processes in Workbench.

In the connection tab that displays in the main Editor view, click the Processes tab. Notice the target processes list, indicating a successful connection.



| Name | PID | State | User | Command Line |
|---|---|---|---|---|
| agetty | 146 | S | root | /sbin/agetty --noclear tt |
| ata_sff | 18 | S | root | |
| atd | 120 | S | root | /usr/sbin/atd -f |
| avahi-daemon | 121 | S | root | "avahi-daemon: running |
| avahi-daemon | 127 | S | root | "avahi-daemon: chroot h |
| bioset | 15 | S | root | |
| bioset | 43 | S | root | |
| crond | 116 | S | root | /usr/sbin/crond -n |
| crypto | 16 | S | root | |
| dbus-daemon | 123 | S | root | /usr/bin/dbus-daemon - |
| deferwq | 44 | S | root | |
| dm_bufio_cache | 41 | S | root | |
| fsnotify_mark | 24 | S | root | |
| ipv6_addrconf | 42 | S | root | |
| kblockd | 17 | S | root | |
| kdevtmpfs | 12 | S | root | |
| khelper | 11 | S | root | |
| khubd | 19 | S | root | |
| kpsmoused | 40 | S | root | |
| ksoftirqd/0 | 3 | S | root | |
| kswapd0 | 23 | S | root | |
| kthreadd | 2 | S | root | |

# 2

# Developing Application Projects

## Creating an Application Project with Workbench

Create the Hello World application project from the samples included with Wind River Linux.

This procedure assumes you have previously configured and built a platform project using Workbench. See: *Creating and Configuring a Platform Project* on page 5

**Step 1** Start Workbench.

| Options | Description |
|---|---|
| **From the System menu** | Select **WindRiver** > **Workbench 4** > **Workbench 4**. |
| **From the desktop** | Double-click the Workbench icon on your desktop. |
| **From the command line** | **1.** Navigate to the Workbench install directory: `$ cd installDir` **2.** Enter the following command: `$ workbench-4/startWorkbench.sh` |

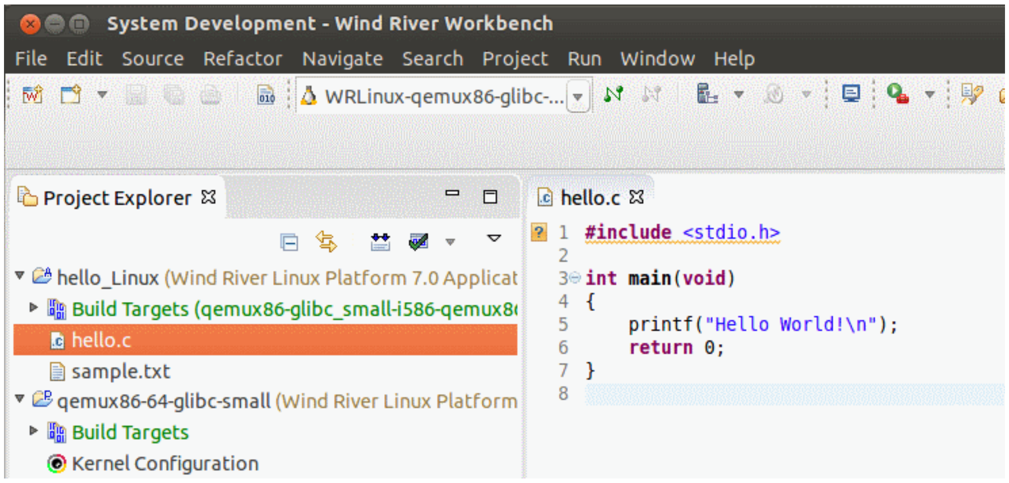
**Step 2** Create a new application project using the example Hello World sample project.

a) Select **File** > **New** > **Example**.

b) Select **Wind River Linux Application Sample Project**, then click **Next**.
c) Select **The Hello World Demonstration Program**, then click **Finish**.

The project appears in Project Explorer as **hello_Linux**.

To view the source file of the application, expand the **hello_Linux** project and double-click the **hello.c** file to display it in the Editor view.


System Development - Wind River Workbench
File Edit Source Refactor Navigate Search Project Run Window Help
WRLinux-qemux86-glibc-...
Project Explorer
hello_Linux (Wind River Linux Platform 7.0 Applicat
Build Targets (qemux86-glibc_small-i586-qemux8
hello.c
sample.txt
qemux86-64-glibc-small (Wind River Linux Platform
Build Targets
Kernel Configuration
hello.c

```
#include <stdio.h>

int main(void)
{
    printf("Hello World!\n");
    return 0;
}
```

**Step 3** Build the application project.

Right-click on the application project folder and select

#### Postrequisites

Once you have created the application, the next step is to build it. See: *Building an Application Project* on page 18.

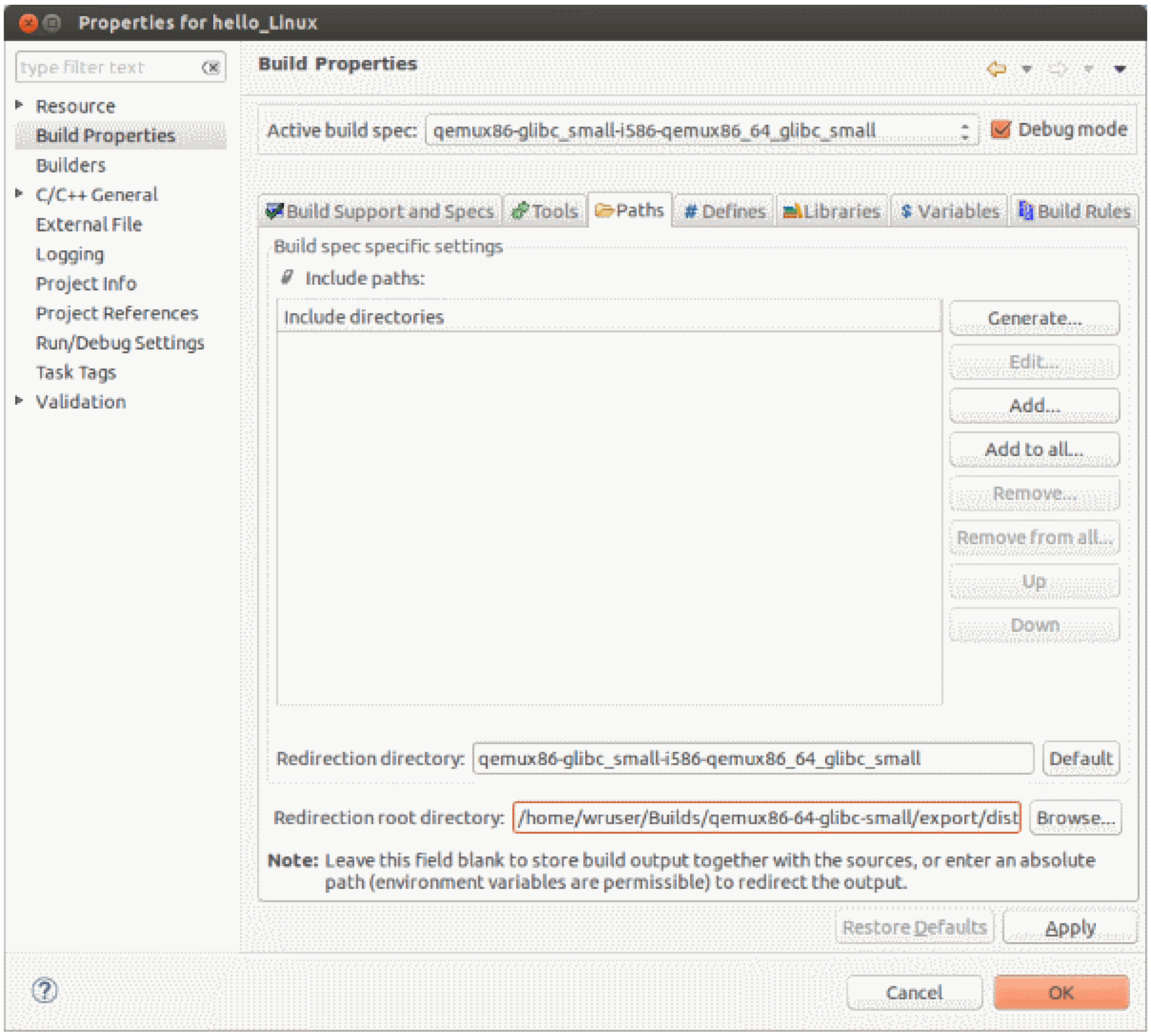
## Building an Application Project

After you create your application project, you must build it using the same build specification you used to build your platform project.

**Step 1** Right-click on the **hello_Linux** application project and select **Properties**.

**Step 2** Set the redirection root directory.

Click **Build Properties** > **Paths** in the left column the tab. In the **Redirection root directory** field, enter the path to the previously created platform project's **export/dist** directory (see *Building a Platform Project Image* on page 7) that contains the target file system, or use the **Browse...** button to navigate there.


Properties for hello_Linux

Build Properties

Active build spec: qemux86-glibc_small-i586-qemux86_64_glibc_small   Debug mode

Build spec specific settings

Include paths:

Redirection directory: qemux86-glibc_small-i586-qemux86_64_glibc_small

Redirection root directory: /home/wruser/Builds/qemux86-64-glibc-small/export/dist

**Note:** Leave this field blank to store build output together with the sources, or enter an absolute path (environment variables are permissible) to redirect the output.

For example, the path might be ***`projectDir`*****/qemux86-64_small/export/dist/**. When you build the application, Workbench will copy the executable to a directory structure in the root directory of the target file system. The top-most directory will be called **hello_world**.

**Step 3** Commit your changes.

Once you have set the path, click **OK** to close the Build Properties window. When prompted, click **Yes** to continue.

**Step 4** Right-click on the **hello_Linux** application project and select **Build Options** > **Set Active Build Spec**, and select the option for **qemux86-64-glibc-small-x86-64-qemux86-64_small_prj**.

Click **Yes** if prompted to rebuild the index.

**Step 5** Build the project.

Right-click on your **hello_Linux** application project folder and select **Build Project**.

The project builds and creates a **hello_Linux.out** executable for glibc small (32-bit) in the **Binaries** folder of the application project.

You now have an executable ready to deploy and debug.

## Creating a Target Connection

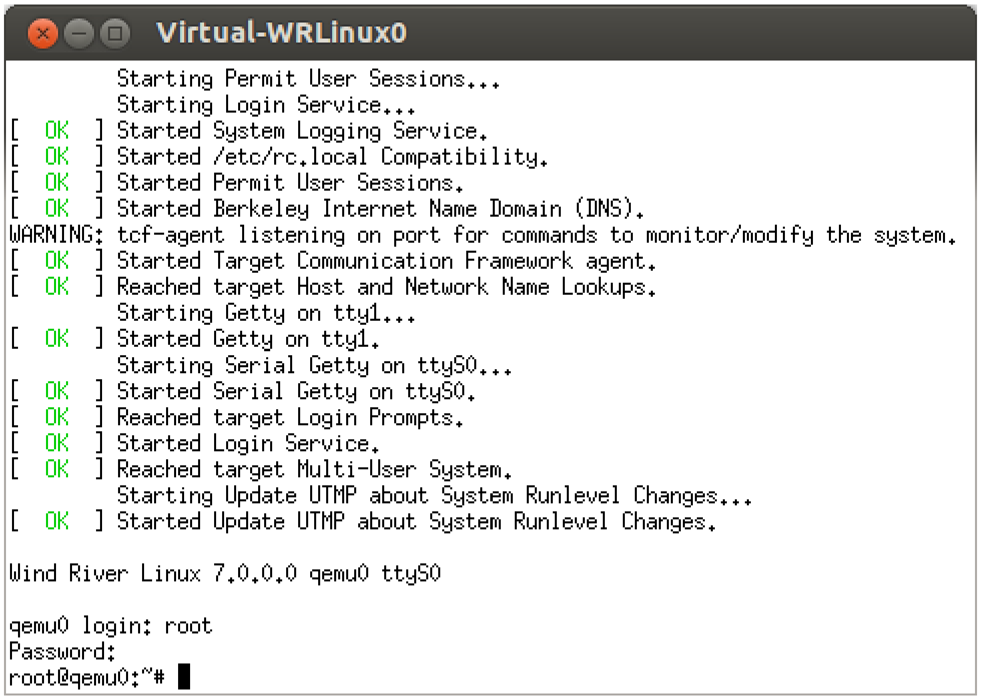
You use the Wind River Workbench System Management view or toolbar to create or connect to a simulated or hardware target.

You can create a target connection to a running target, or to a QEMU instance. For running targets, you will require the IP address or hostname, and the TCF port to create a connection. To create and deploy a QEMU target from Workbench, see *Deploying a Platform Project Image* on page 13. Once you have previously created a QEMU target connection, you can connect or reconnect to it using the following procedure.

**Step 1** Connect to the target in Workbench.

In this step, you are going to connect to the target.

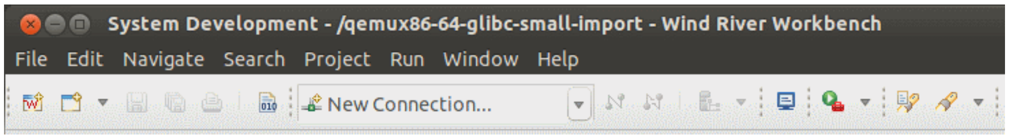
If you want to connect to a running target, such as a QEMU instance started from the command linestart a QEMU connection from the command-line to use to simulate a hardware connection. If you already have a hardware target running, go to the next step.

| Options | Description |
|---|---|
| **Existing QEMU connection** | Use the following steps to connect to a QEMU target. If you have previously established a QEMU target connection and it is still running as described in *Deploying a Platform Project Image* on page 13, then you already have a target connection.<br>**1.** Start the connection.<br>In the Systems Management toolbar, select the connection from the drop-down list. If you created the connection in this running Workbench session, the QEMU window will launch and boot.<br>If you have restarted Workbench since creating the connection, you will also need to click **Connect** on the Systems Management toolbar.<br><br>Virtual-WRLinux0<br>(terminal output below)<br>**2.** Enter **root** for the user name and password to log into the QEMU instance.<br>The target is now connected. |

```
         Starting Permit User Sessions...
         Starting Login Service...
[  OK  ] Started System Logging Service.
[  OK  ] Started /etc/rc.local Compatibility.
[  OK  ] Started Permit User Sessions.
[  OK  ] Started Berkeley Internet Name Domain (DNS).
WARNING: tcf-agent listening on port for commands to monitor/modify the system.
[  OK  ] Started Target Communication Framework agent.
[  OK  ] Reached target Host and Network Name Lookups.
         Starting Getty on tty1...
[  OK  ] Started Getty on tty1.
         Starting Serial Getty on ttyS0...
[  OK  ] Started Serial Getty on ttyS0.
[  OK  ] Reached target Login Prompts.
[  OK  ] Started Login Service.
[  OK  ] Reached target Multi-User System.
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Started Update UTMP about System Runlevel Changes.

Wind River Linux 7.0.0.0 qemu0 ttyS0

qemu0 login: root
Password:
root@qemu0:~# █
```

| Options | Description |
|---|---|
| **Running target connection** | Use the following steps to create a connection to a running target, such as a hardware target or a QEMU instance started from the command line with the **make start-target** command.<br><br>**1.** Create a new target connection.<br><br>Click **New Connection** in the System Management toolbar to add the QEMU target to your connection options.<br><br><br> System Development - /qemux86-64-glibc-small-import - Wind River Workbench<br>File Edit Navigate Search Project Run Window Help<br>New Connection...<br><br>**2.** In the New Connection window, select **Running Target**.<br>**3.** Enter the target information.<br><br>You will require the target hostname and TCF port to complete the connection. The following values are for the QEMU connection created in *Step 1*.<br><br>**Target Address**<br><br>Enter `localhost:4447`<br><br>This is the default setting for a Wind River QEMU BSP-based platform project image. You may substitute **localhost**, which is the hostname in the **/etc/hosts** file for the default IP address, **127.0.0.1**.<br><br>The port number **4447** represents the default TCF port for the QEMU instance. To change this port number, use the **make config-target** command in the platform project directory. For additional information, see the *Wind River Linux User's Guide*.<br><br>**Platform Project or SDK Root Directory**<br><br>Click **Browse**, and navigate to the platform project directory.<br><br>checkbox<br><br>Select this to automatically connect Workbench to the target.<br><br>**4.** Select the **Connect on finish** checkbox to start the connection automatically.<br>**5.** Click **Finish**. The Wind River Linux Connection - Configuration appears in the center development view in the System Development perspective.<br>**6.** Optionally click the Processes tab. Notice the target processes list, indicating a successful connection. |

## Deploying an Application Project with Workbench

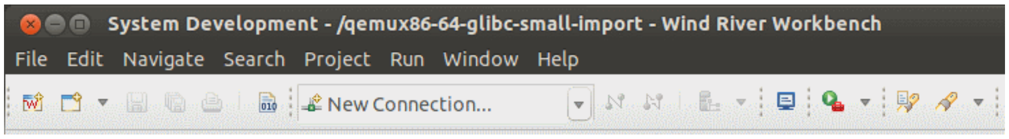
You can deploy an application to a simulated target platform with Wind River Workbench.

The example in this section uses the **hello_Linux** application created in *Creating an Application Project with Workbench* on page 17, and the built platform project image created in *Creating and Configuring a Platform Project* on page 5.

To deploy an application project with Workbench:

**Step 1** Connect Workbench to the QEMU target.

a) Click **New Connection** in the System Management toolbar to add the QEMU target to your connection options.


System Development - /qemux86-64-glibc-small-import - Wind River Workbench
File Edit Navigate Search Project Run Window Help
New Connection...

b) In the **New Connection** window, enter target information.

**Target Address**

Enter `localhost:4447`

This is the default setting for a Wind River QEMU BSP-based platform project image. You may substitute **localhost**, which is the hostname in the **/etc/hosts** file for the default IP address, **127.0.0.1**.

The port number **4447** represents the default TCF port for the QEMU instance. To change this port number, use the **make config-target** command in the platform project directory. For additional information, see the *Wind River Linux User's Guide*.

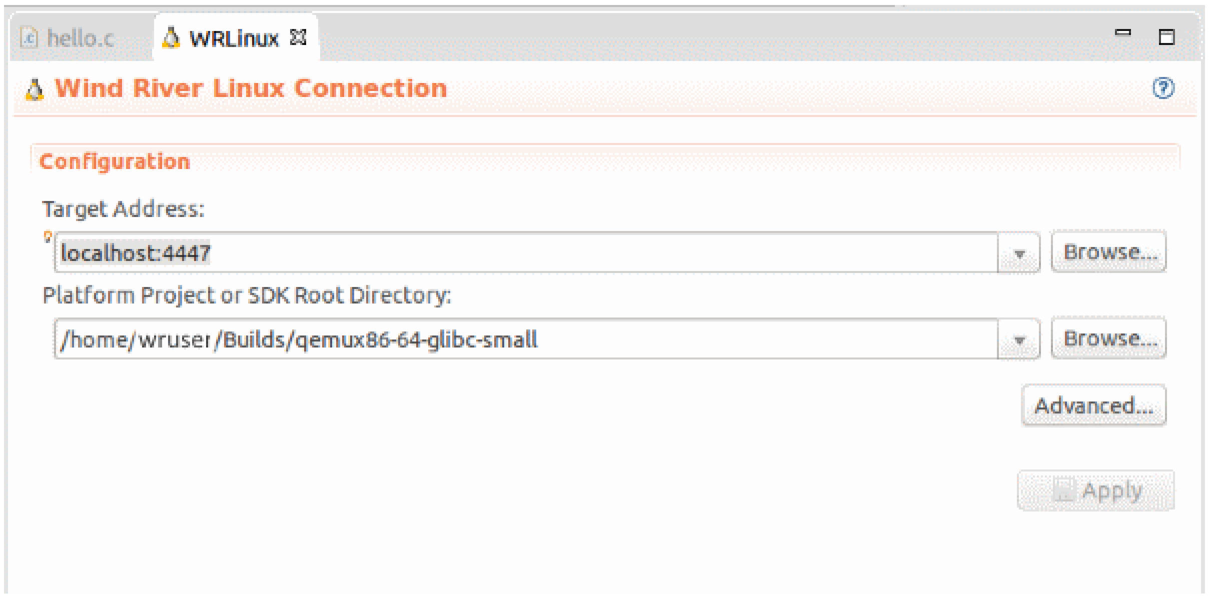
**Platform Project or SDK Root Directory**

Click **Browse**, and navigate to the platform project directory.

**Connect on finish** checkbox

Select this to automatically connect Workbench to the target.

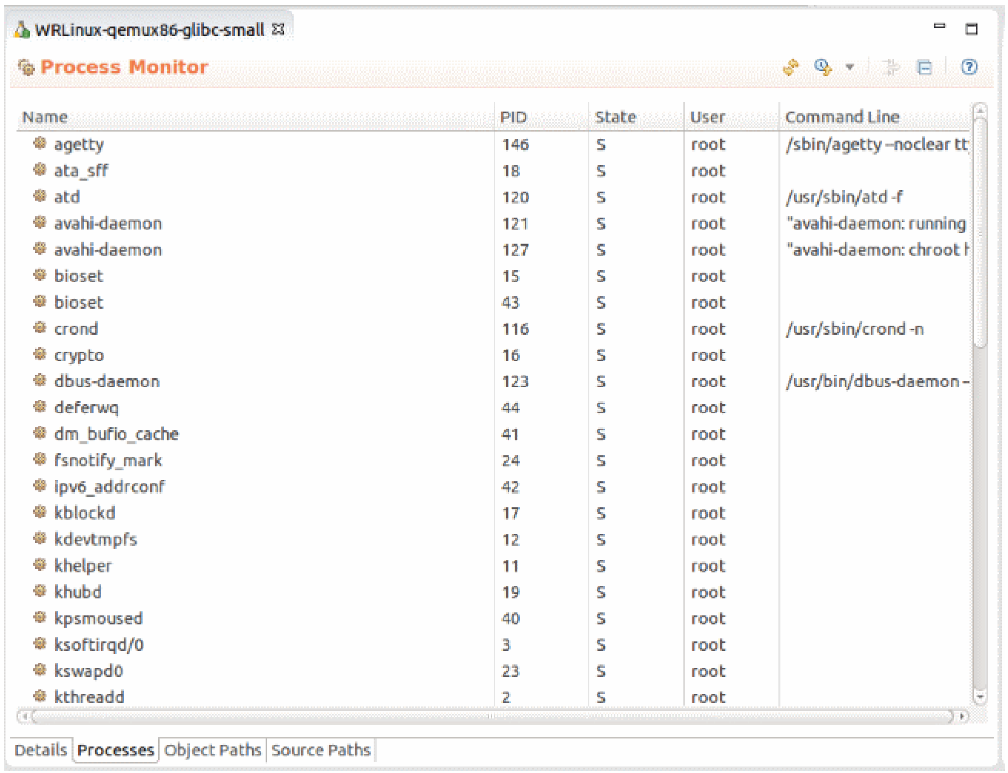
Click **Finish**. The Wind River Linux Connection - Configuration appears in the center development view in the System Development perspective.



Target Address:
localhost:4447

Platform Project or SDK Root Directory:
/home/wruser/Builds/qemux86-64-glibc-small

Click the Processes tab. Notice the target processes list, indicating a successful connection.

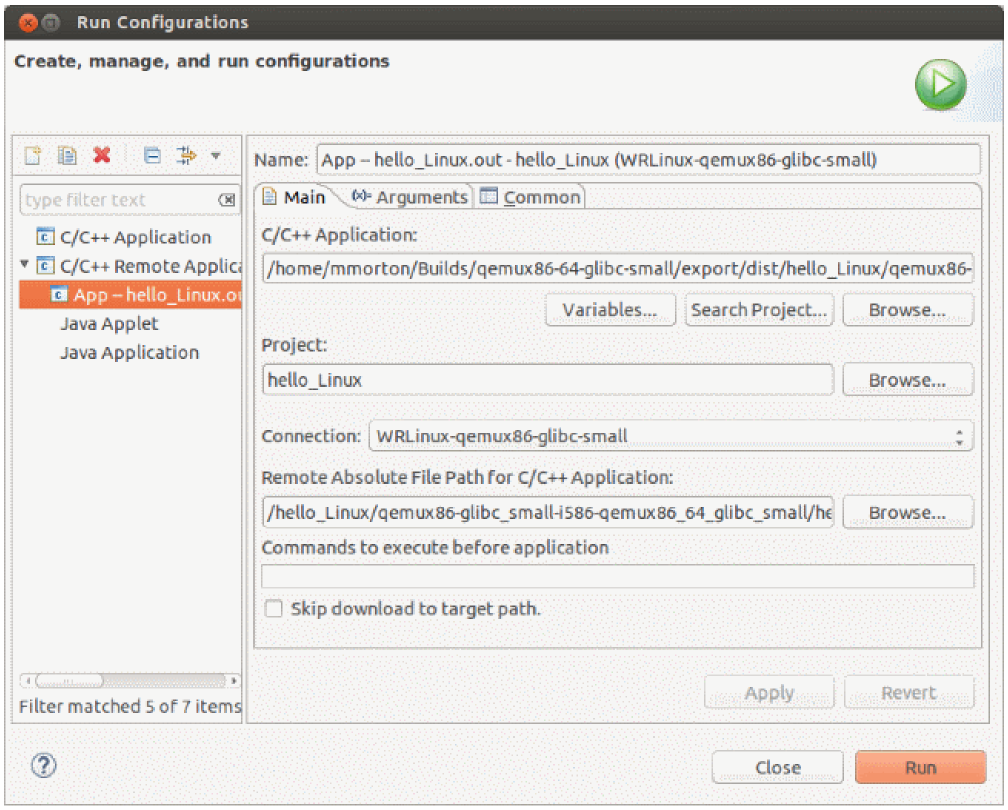


| Name | PID | State | User | Command Line |
|---|---|---|---|---|
| agetty | 146 | S | root | /sbin/agetty --noclear tt |
| ata_sff | 18 | S | root | |
| atd | 120 | S | root | /usr/sbin/atd -f |
| avahi-daemon | 121 | S | root | "avahi-daemon: running |
| avahi-daemon | 127 | S | root | "avahi-daemon: chroot h |
| bioset | 15 | S | root | |
| bioset | 43 | S | root | |
| crond | 116 | S | root | /usr/sbin/crond -n |
| crypto | 16 | S | root | |
| dbus-daemon | 123 | S | root | /usr/bin/dbus-daemon - |
| deferwq | 44 | S | root | |
| dm_bufio_cache | 41 | S | root | |
| fsnotify_mark | 24 | S | root | |
| ipv6_addrconf | 42 | S | root | |
| kblockd | 17 | S | root | |
| kdevtmpfs | 12 | S | root | |
| khelper | 11 | S | root | |
| khubd | 19 | S | root | |
| kpsmoused | 40 | S | root | |
| ksoftirqd/0 | 3 | S | root | |
| kswapd0 | 23 | S | root | |
| kthreadd | 2 | S | root | |

**Step 2** Navigate to the **hello_Linux** application in the **Project Explorer**.

**Step 3** Expand the **Build Targets** section, right-click on the **hello_Linux.out** application, and select **Run Remote Linux Application**.

The Run Configurations window appears.



Name: App – hello_Linux.out - hello_Linux (WRLinux-qemux86-glibc-small)

C/C++ Application: /home/mmorton/Builds/qemux86-64-glibc-small/export/dist/hello_Linux/qemux86-

Project: hello_Linux

Connection: WRLinux-qemux86-glibc-small

Remote Absolute File Path for C/C++ Application: /hello_Linux/qemux86-glibc_small-i586-qemux86_64_glibc_small/he

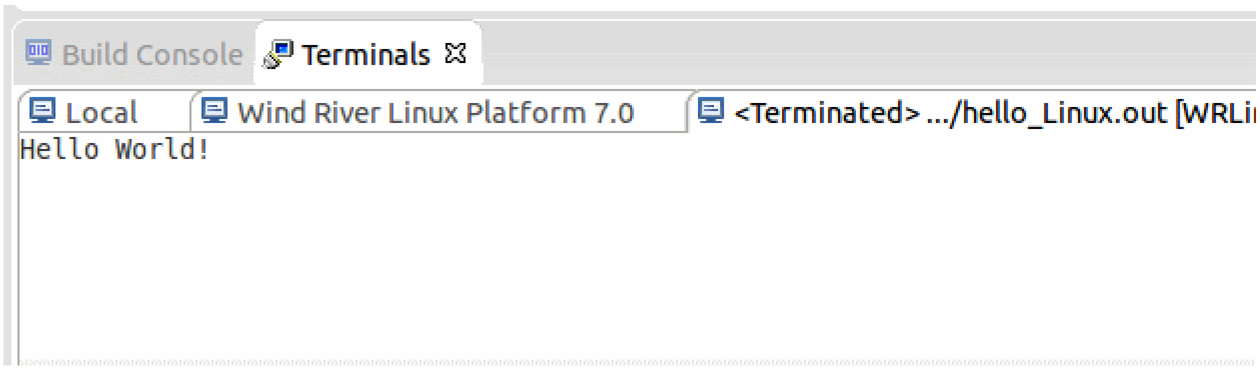
Commands to execute before application

Skip download to target path.

Notice that the **Project**, **Connection**, and **Remote Absolute Path for C/C++ Application** fields are populated automatically.

**Step 4** Click **Run**.

**Hello World** appears in the **Terminals** view in Workbench.


Build Console
Terminals
Local
Wind River Linux Platform 7.0
<Terminated> .../hello_Linux.out [WRLi

```
Hello World!
```

# 3

# Debugging an Executable

## Debugging an Executable with Workbench

You can debug an application that has been added to a platform project image.

This procedure builds upon previous procedures in these tutorials and requires:

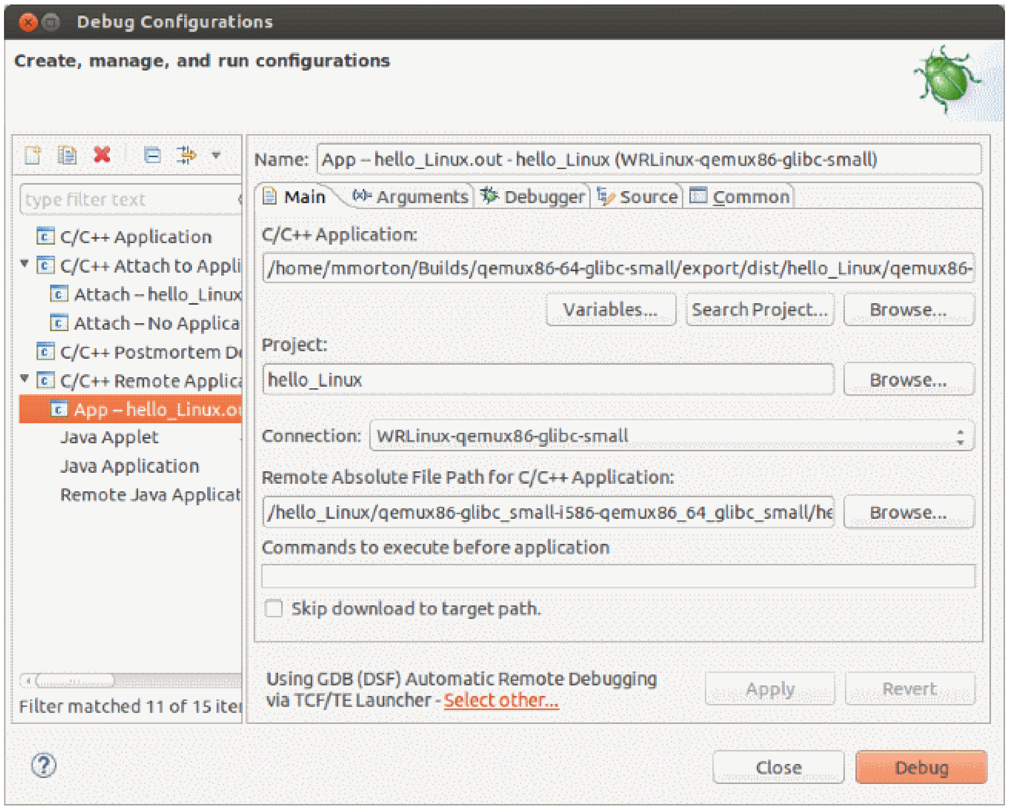
- A platform project configured and built using the **feature/debug** template as described in *Configuring a New Project to Add Application Packages* on page 8.
- The **Hello World** application built to match the target architecture as described in *Creating an Application Project with Workbench* on page 17.
- A QEMU target connected to Workbench as described in *Deploying an Application Project with Workbench* on page 23.

To debug an application using Workbench:

**Step 1** Attach the Debugger to the **hello_Linux** program.

a) Right-click on the binary and select **Debug Remote Linux Application Process**.

The Debug Configurations window appears.



Name: App – hello_Linux.out - hello_Linux (WRLinux-qemux86-glibc-small)

C/C++ Application: /home/mmorton/Builds/qemux86-64-glibc-small/export/dist/hello_Linux/qemux86-

Project: hello_Linux

Connection: WRLinux-qemux86-glibc-small

Remote Absolute File Path for C/C++ Application: /hello_Linux/qemux86-glibc_small-i586-qemux86_64_glibc_small/he

Commands to execute before application

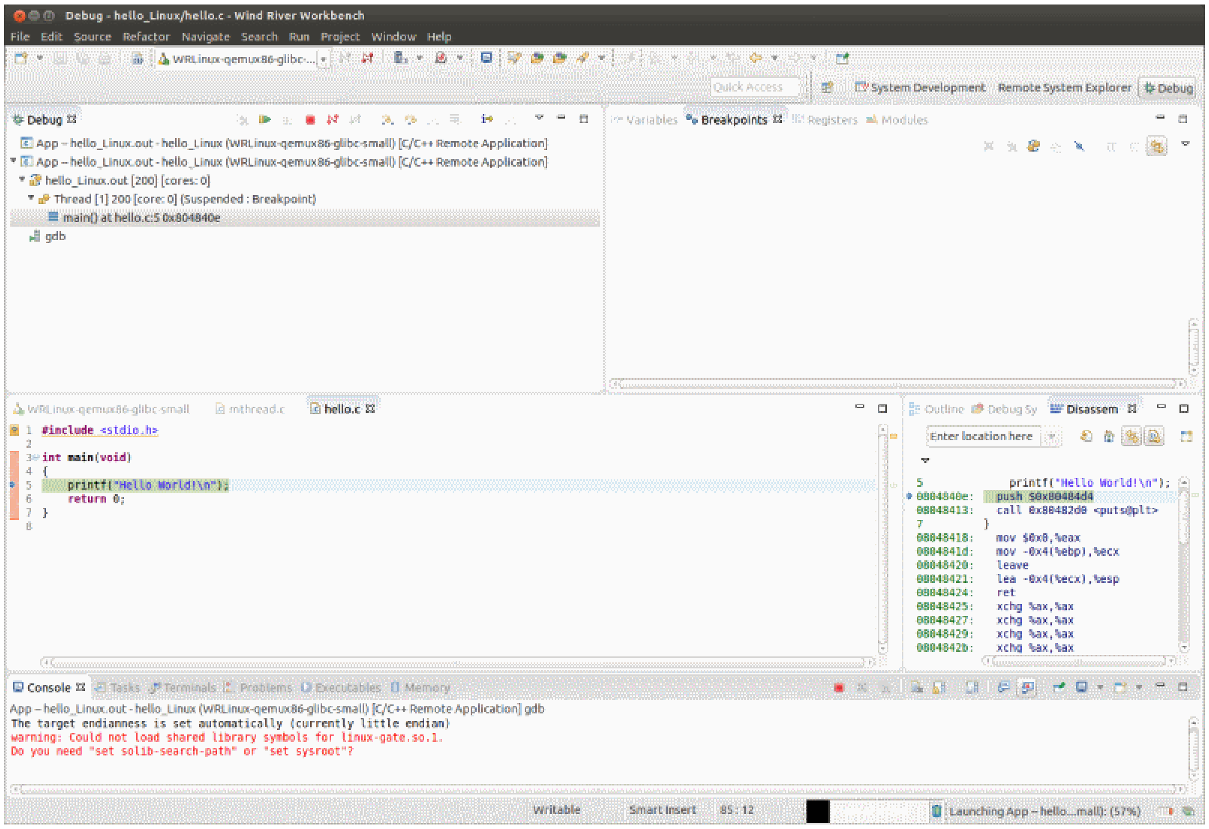
☐ Skip download to target path.

Using GDB (DSF) Automatic Remote Debugging via TCF/TE Launcher - Select other...

Notice that the **Project**, **Connection**, and **Remote Absolute Path for C/C++ Application** fields are populated automatically.

b) Click **Debug**.

Workbench displays a new **Debug** perspective. This perspective displays **Debug** and **Breakpoint** views by default. The process has stopped at **main( )**in the **Debug** view.



```
App – hello_Linux.out - hello_Linux (WRLinux-qemux86-glibc-small) [C/C++ Remote Application]
App – hello_Linux.out - hello_Linux (WRLinux-qemux86-glibc-small) [C/C++ Remote Application]
  hello_Linux.out [200] [cores: 0]
    Thread [1] 200 [core: 0] (Suspended : Breakpoint)
      main() at hello.c:5 0x804840e
  gdb
```

```c
#include <stdio.h>

int main(void)
{
    printf("Hello World!\n");
    return 0;
}
```

```
5                 printf("Hello World!\n");
0804840e:      push $0x80484d4
08048413:      call 0x80482d0 <puts@plt>
7             }
08048418:      mov $0x0,%eax
0804841d:      mov -0x4(%ebp),%ecx
08048420:      leave
08048421:      lea -0x4(%ecx),%esp
08048424:      ret
08048425:      xchg %ax,%ax
08048427:      xchg %ax,%ax
08048429:      xchg %ax,%ax
0804842b:      xchg %ax,%ax
```

```
App – hello_Linux.out - hello_Linux (WRLinux-qemux86-glibc-small) [C/C++ Remote Application] gdb
The target endianness is set automatically (currently little endian)
warning: Could not load shared library symbols for linux-gate.so.1.
Do you need "set solib-search-path" or "set sysroot"?
```

The warning that appears concerning shared library symbols is the result of using a virtual QEMU simulator, which does not have a kernel with libraries that the debugger is looking for.

**NOTE:** To get back to the System Development perspective at any time, select **Window** > **Open_Perspective** > **System Development**.

**Step 2** Resume execution.

To resume execution, click the green **Resume** icon in the Debug view. The process completes and terminates.