Developing, Building and Testing Your Baremetal Applications Using The Yocto Project and OpenEmbedded Infrastructure

Alejandro Enedino Hernandez Samaniego / Microsoft
Multiarchitecture Embedded Devices

Azure Sphere

- Secured MCU
- Secured OS
- Cloud Security
OpenEmbedded Runtime Testing Infrastructure
OpenEmbedded Runtime Testing Infrastructure

- Testing a Linux OS (Poky)
  - QEMU
  - Hardware

```bash
local.conf
INHERIT += "testimage"

TEST_TARGET = "qemu" # Default
# TEST_TARGET = "simpleremote"
# TEST_TARGET_IP = "8.1.8.2"
# TEST_SERVER_IP (Automatically detected)

TEST_SUITES = "auto" # Default
# ping ssh scp parselogs
# meta/lib/oeqa/runtime/cases

# The order matters due to
# Dependencies between them
# scp ping ssh
# scp (skipped) ping (run) ssh (run)
# Correct:
# ping -> ssh -> scp
```
OpenEmbedded Runtime Testing Infrastructure

Build Configuration:
BB_VERSION = "1.46.0"
BUILD_SYS = "x86_64-linux"
NATIVELSSTRING = "universal"
TARGET_SYS = "aarch64-poky-linux-musl"
MACHINE = "qemuarm64"
DISTRO = "poky"
DISTRO_VERSION = "3.1+snapshot-20200623"
TUNE_FEATURES = "aarch64 cortexa57 crc"
TARGET_FPU = "" 
meta
meta-poky
meta-yocto-bsp = "master:816a12758bfcb5d409accbf636ec58d9fa917cbeb6"
meta-oe
meta-networking
meta-python
workspace = "master:816a12758bfcb5d409accbf636ec58d9fa917cbeb6"

Initialising tasks: 100% |----------------------------------------------------------| Time: 0:00:01
Sstate summary: Wanted 0 Found 0 Missed 0 Current 200 (0% match, 100% complete)
NOTE: Executing Tasks
Test requires dropdown, oropensh-sshd to be installed
RESULTS:
RESULTS - ping.PingTest.ping.ping: PASSED (0.12s)
RESULTS - ssh.SSHTest.test_ssh: SKIPPED (0.00s)
SUMMARY:
core-image-minimal () - Ran 2 tests in 0.120s
core-image-minimal - OK - All required tests passed (successes=1, skipped=1, failures=0, errors=0)
NOTE: Tasks Summary: Attempted 918 tasks of which 917 didn't need to be rerun and all succeeded.

Summary: There were 2 WARNING messages shown.
TESTIMAGE_BOOT_PATTERNS in OE

testimage.bbclass

```python
# TESTIMAGE_BOOT_PATTERNS can be used to override certain patterns used to communicate with the target when booting,
# if a pattern is not specifically present on this variable a default will be used when booting the target.
# TESTIMAGE_BOOT_PATTERNS[<flag>] overrides the pattern used for that specific flag
#
# Example:
# TESTIMAGE_BOOT_PATTERNS = "send_login_user search_login_succeeded"
# TESTIMAGE_BOOT_PATTERNS[send_login_user] = "webserver\n"
# TESTIMAGE_BOOT_PATTERNS[search_login_succeeded] = "webserver@[a-zA-Z0-9-]+:~#"
```

qemurunner.py

```python
accepted_patterns = ['search_reached_prompt', 'send_login_user', 'search_login_succeeded', 'search_cmd_finished']
default_boot_patterns = defaultdict(str)
#
# Default to the usual patterns used to communicate with the target
default_boot_patterns['search_reached_prompt'] = b' login:'
default_boot_patterns['send_login_user'] = 'root\n'
default_boot_patterns['search_login_succeeded'] = r'root@[a-zA-Z0-9-]+:~#'
default_boot_patterns['search_cmd_finished'] = r'[a-zA-Z0-9]+@[a-zA-Z0-9-]+:~#'
```
Testimage flow

TARGET

HOST

START BOOTING

LOGIN:

ROOT

LOGIN SUCCESS!

SEND COMMAND

r"root@[a-zA-Z0-9-]+:~#"

r"[a-zA-Z0-9]+@[a-zA-Z0-9-]+:~#"

READ
Baremetal Toolchains and Applications on The Yocto Project
Baremetal Toolchains on the Yocto Project

1.- Clone the required repositories (Use -b dunfell if you want a stable release)

```
$ git clone https://git.yoctoproject.org/git/poky
$ cd poky
```

2.- Source the build environment file and add the skeleton layer (layer that contains examples)

```
$ source oe-init-build-env
$ bitbake-layers add-layer ../meta-skeleton
```

3.- Add the required variables to your local.conf (Supported MACHINES are qemuarm64,qemuarm,armuarmv5)

```
$ echo "MACHINE = "qemuarm64"" >> ./conf/local.conf
$ echo "TCLIBC = "baremetal"" >> ./conf/local.conf
```

4.- Build the application

```
$ bitbake baremetal-helloworld
```
Baremetal Toolchains on the Yocto Project

5. Run the baremetal application on QEMU:

```bash
$ qemu
```

Example output:

```
rtemu - INFO - Running bitbake -e ...
rtemu - INFO - Continuing with the following parameters:
KERNEL: [tmp/deploy/images/qemuarm64/baremetal-helloworld-qemuarm64.bin]
MACHINE: [qemuarm64]
FSTYPE: [bin]
ROOTFS: [tmp/deploy/images/qemuarm64/baremetal-helloworld-qemuarm64.bin]
CONFFILE: [tmp/deploy/images/qemuarm64/baremetal-helloworld-qemuarm64.qemuboot.conf]
Hello OpenEmbedded!
```

Interesting targets:

- Meta-toolchain
- Baremetal-helloworld -c populate_sdk
- TCLIBC="newlib"
Simple Baremetal HelloWorld on Aarch64

- QEMU Source Code (Virt)
  - UART, RAM, Kernel Addresses
- ARM Developers Manual – UART PL011

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Reset</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>UARTDR</td>
<td>RW</td>
<td>0x---</td>
<td>12/8</td>
<td>Data Register, UARTDR</td>
</tr>
<tr>
<td>0x004</td>
<td>UARTSR/</td>
<td>RW</td>
<td>0x0</td>
<td>4/0</td>
<td>Receive Status Register/Error Clear Register, UARTSR/UARTECR</td>
</tr>
<tr>
<td></td>
<td>UARTECR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x008-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x014</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x018</td>
<td>UARTFR</td>
<td>RO</td>
<td>0x0-10010--</td>
<td>9</td>
<td>Flag Register, UARTFR</td>
</tr>
</tbody>
</table>

- Startup Code, Linker Script
TESTIMAGE_BOOT_PATTERNS in OE

• Overriding Patterns

```
MACHINE="qemu-arm"
TCLIBC = "newlib"
INHERIT += "testimage"
TEST_SUITES = "freertos_echo"

# Add boot patterns to use with OE testimage infrastructure with the serial console
TESTIMAGE_BOOT_PATTERNS = "search_reached_prompt send_login_user search_login_succeeded search_cmd_finished"
TESTIMAGE_BOOT_PATTERNS[search_reached_prompt] = "ELC2020"
TESTIMAGE_BOOT_PATTERNS[send_login_user] = "Covid19\r"
TESTIMAGE_BOOT_PATTERNS[search_login_succeeded] = "NOTWelcome"
TESTIMAGE_BOOT_PATTERNS[search_cmd_finished] = "\n"
```
• Overriding Patterns
Creating a Test Case

FreeRTOS build setup

1. Clone the required repositories (Use -b dunfell if you want a stable release)

```bash
$ git clone https://git.yoctoproject.org/git/poky
$ cd poky
$ git clone https://github.com/aehs29/meta-freeRTOS.git
```

2. Add meta-freeRTOS to your bblayers.conf

```bash
$ source oe-init-build-env
$ bitbake-layers add-layer ../../../meta-freeRTOS
```

```python
from oeqa.runtime.case import OERuntimeTestCase
from oeqa.core.decorator.oetimeout import OETimeout
import re
import time

class FreeRTOSTest(OERuntimeTestCase):
    @OETimeout(15)
    def test_freeertos_echo(self):
        # Use a special character with a known behavior
        cmd = '\r'
        bb.warn('Sending command: ' + cmd)
        # Send it raw, so it does not look for echo %?, it just looks at the output
        status, output = self.target.runner.run_serial(cmd, raw=True)
        bb.warn('Output: %s % output)
        match = bool(re.search('Success', output))
        self.assertEqual(match, True)
```
Build Configuration:
BB_VERSION = "1.46.0"
BUILD_SYS = "x86_64-linux"
NATIVE_SYSIMAGE = "universal"
TARGET_SYS = "x86_64-poky-elf"
MACHINE = "qemux86-64"
DISTRO = "poky"
DISTRO_VERSION = "3.1.1"
TUNE_FEATURES = "m64 core2"
TARGET_FPU = ""

Initialising tasks: 100%

State summary: Wanted 0 Found 0 Missed 0 Current 176 (0% match, 100% complete)

NOTE: Executing Tasks
WARNING: baremetal-helloworld-0.1+gitAUTOINC+99f4fa4a3b-r0 do_testimage: Sending command: ~
WARNING: baremetal-helloworld-0.1+gitAUTOINC+99f4fa4a3b-r0 do_testimage: Output: Success

RESULTS:
RESULTS = freertos_echo.FreeRTOSTest.test_freertos_echo: PASSED (0.04s)

SUMMARY:
baremetal-helloworld () - Ran 1 test in 0.045s
baremetal-helloworld - OK - All required tests passed (successes=1, skipped=0, failures=0, errors=0)
NOTE: Tasks Summary: Attempted 805 tasks of which 804 didn’t need to be rerun and all succeeded.
Thanks!

Embedded Linux Conference
North America

alejandro.hernandez@microsoft.com