Unveil How to Customize LTSI Test For Your Platform

Kengo IBE, Mitsubishi Electric
5th, October 2015
ELCE 2015 @Dublin
Who am I?

• Kengo Ibe

• Embedded Linux Developer at the Mitsubishi Electric Information Technology R&D Center

• Loaned to Linux Foundation Since April 2015

• Joined Linux Foundation Collaborative Projects
  – LTSI : Long Term Support Initiative
  – AGL : Automotive Grade Linux
Outline

• What is the LTSI Project?
  – LTSI Test Environment

• How to Customize?
  – Add New Board (Raspberry Pi2)
  – Add New Test Suite (LTP : Linux Test Project)

• Run LTP on Raspberry Pi2

• Summary & Future Works
What is LTSI Test Project?

• LTSI Project:
  – The project creates and maintains Linux Kernel which is expected to be stable in quality for the typical lifetime of a consumer electronics product, typically 2-3 years.
  – LTSI-4.1 Developing now
    • Close Merge Window: End of October

• LTSI Test Project
  – The project creates the LTSI Test Environment.
  – The LTSI Test Environment is Jenkins based automation test framework.
  – Including many test suites and kinds of target boards
    • 28 benchmarks and 33 functional test programs are already integrated
    • Minnow board(x86), koelsch(arm), quem-arm(QEMU) are already integrated
  – I hope to further increase the kind of target board, test suite.
  – I’m happy that many people will join this project.
LTSI Test Environment (Overview)

• Top of Web Interface LTSI Test Environment

Target Boards

List of Test name

History of Test Results

ELCE 2015 @Dublin
LTSI Test Environment (Flow)

1. Compile Test Suite
2. Send the test
3. Execute some tests on the target board
4. Get the results
5. Show the results on GUI

Target Board:
- (koslsch)
- (minnow)
QEMU:
- (qemu-arm)

Test Framework

SDK (Cross-tools)
- minnow-jta
- renesas
- qemu-arm

Test Suite

Bench marks
- [bonnie]
- [cyclictest]
- [Dhrystone]
- [himeno]

Functional
- [bzip2]
- [LTP]
- [expat]
- [netperf]

jenkins

Debian 7.X

ELCE 2015 @Dublin
How to Customize ?(New Target)

• 3 step to add New Target
  – Make the SDK for the target
    • Using yocto project
  – Deploy the SDK into Test Framework
  – Set target Information by GUI

1. Make the SDK
2. Deploy the SDK
3. Set info. by UI

Test Framework

SDK (Cross-tools)

Test Suite

minnow-jta
renesas
qemu-arm

New target

New Setting

Jenkins

Debian 7.X
How to Customize ? (Raspberry pi 2)

• 3 step to add New Target
  – Make the SDK for the target
    • Using yocto project
  – Deploy the SDK into Test Framework
  – Set target Information by GUI
How to Customize ?(Raspberry pi 2)

• Make the SDK
  – Getting poky from Yocto project
    $ git clone git://git.yoctoproject.org/poky.git
  – Getting meta-raspi and meta-jta
    • meta-raspi: For making a OS image and SDK for Raspberry pi2
      $ git clone git://git.yoctoproject.org/meta-raspberrypi
    • meta-jta: For adding Headers and Libs for the Test Suite
      $ git clone https://bitbucket.org/cogentembedded/meta-jta.git
• Make the SDK (Cont’d)
  – Configure the environment to build
    • Execute "oe-init-build-env" script in Poky Directory
      ```
poky$ source oe-init-build-env build-raspi2
      ```
    • Then created directory “build-raspi2”
      ```
bld-raspi2$tree
   └── conf
      │   └── bblayers.conf
      │   └── local.conf
      ```
  – “build-raspi2” includes a conf directory
  – There are “bblayers.conf” and “local.conf” in the conf directory
How to Customize ?(Raspberry pi 2)

• Make the SDK (Cont’d)
  – Setting to build(Cont’d)
    • Configure bblayers.conf for meta-raspberrypi & meta-jta
      
      BBLAYERS ?= " ¥
      /home/melco/sdk/yocto/poky/meta ¥
      /home/melco/sdk/yocto/poky/meta-yocto ¥
      /home/melco/sdk/yocto/poky/meta-yocto-bsp ¥
      /home/melco/sdk/yocto/poky/meta-raspberrypi ¥
      /home/melco/sdk/yocto/poky/meta-jta ¥

      Adding the path of “meta-raspberrypi” & “meta-jta”

    • Configure local.conf for meta-raspi & meta-jta
      
      #MACHINE ?= "genericx86-64"
      #MACHINE ?= "mpc8315e-rdb"
      #MACHINE ?= "edgerouter"
      MACHINE ?= "raspberrypi2"
      GPU_MEM = "16"

      Setting MACHINE & GPU Memory size for raspi2
How to Customize ?(Raspberry pi 2)

• Make the SDK (Cont’d)

— Build SDK

```
melco@debian-7:~/sdk/yocto/poky/build-raspi2$ bitbake meta-toolchain
Parsing recipes: 100%
Parsing of 912 .bb files complete (0 cached, 912 parsed). 1341 targets, 61 skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION        = "1.27.1"
BUILD_SYS         = "x86_64-linux"
NATIVELSBSTRING   = "Debian-7.8"
TARGET_SYS        = "arm-poky-linux-gnueabi"
MACHINE           = "raspberrypi2"
DISTRO            = "poky"
DISTRO_VERSION    = "1.8+snapshot-20150908"
TUNE_FEATURES     = "arm armv7a vfp thumb neon callconvention-hard vfpv4 cortexa7"
TARGET_FPU        = "vfp-vfpv4-neon"
```

To be able to verify “MACHINE”
For raspi2

To be able to verify that “bblayers.conf” works

Execute “bitbake meta-toolchain” command in the build-raspi2 Directory

ELCE 2015 @Dublin
How to Customize ? (Raspberry pi 2)

• Make the SDK (Cont’d)
  – Build SDK (Cont’d)
    • When building SDK finished, SDK install script is created at <Build Dir>/tmp/deploy/sdk/

```
melco@debian-7:~/sdk/yocto/poky/build-raspi2$ ls -al tmp/deploy/sdk/
合計 206104
drwxr-xr-x 2 melco melco  4096  9月  8 19:04 .
drwxr-xr-x 5 melco melco  4096  9月  8 14:45 ..
-rw------- 1 melco melco  9331  9月  8 19:04 poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-vfpv4-neon-toolchain-1.8+snapshot.host.manifest
-rw------- 1 melco melco 103547364  9月  8 19:04 poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-vfpv4-neon-toolchain-1.8+snapshot.sh
-rw------- 1 melco melco   1866  9月  8 19:03 poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-vfpv4-neon-toolchain-1.8+snapshot.target.manifest
```

This file is the SDK install script.
How to Customize ?(Raspberry pi 2)

• 3 step to add New Target
  – Make the SDK for the target
    • Using yocto project
  – Deploy the SDK into Test Framework
  – Set target Information by GUI

- Test Framework
- SDK (Cross-tools)
  - minnow-jta
  - renesas
  - qemu-arm
  - New target
- Test Suite
- 1. Make the SDK
  2. Deploy the SDK
- New Setting
  3. Set info. UI
- Jenkins
- Debian 7.X

ELCE 2015 @Dublin
How to Customize ?(Raspberry pi 2)

• Deploy the SDK into Test Framework
  – We can Deploy the SDK anywhere
    • This is the default Directory /home/jenkins/tools/.
      Minnow, qemu-arm and renesas-arm SDK are already in the directory.

```bash
melco@debian-7:~/sdk/yocto/poky/build-raspi2/tmp/deploy/sdk$ ./poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-vfpv4-neon-toolchain-1.8+snapshot.sh -y -d /home/jenkins/tools/raspi2
Poky (Yocto Project Reference Distro) SDK installer version 1.8+snapshot
===============================================
The directory "/home/jenkins/tools/raspi2" already contains a SDK for this arch.
If you continue, existing files will be overwritten! Proceed[y/N]? Y
[sudo] password for melco:
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source the environment setup script e.g.
```

Selecting installing directory and run SDK install script.
How to Customize ?(Raspberry pi 2)

- Deploy the SDK into Test Framework(conf.)
  - Setting the Test Framework for the SDK
  - Adding raspi2 configuration on /home/Jenkins/scripts/tools.sh
  - The Test Framework already includes here minnow, qemu-arm and renesas-arm configurations.

```bash
elif [ "${PLATFORM}" = "raspi2" ];
then
  SDKROOT=$JTA_ENGINE_PATH/tools/raspi2/sysroots/cortexa7hf-vfp-vfpv4-neon-poky-linux-gnueabi
  # environment script changes PATH in the way that python uses libs from sysroot which is not what we want, so save it and use later
  ORIG_PATH=$PATH
  PREFIX=arm-poky-linux-gnueabi
  source $JTA_ENGINE_PATH/tools/raspi2/environment-setup-cortexa7hf-vfp-vfpv4-neon-poky-linux-gnueabi
  HOST=arm-poky-linux-gnueabi
  unset PYTHONHOME
  env -u PYTHONHOME
```

“SDKROOT” is the path of the sysroot that there is in deploying the SDK Directory.

Setting “PREFIX” for cross compile

Setting this path written the file of environment variable. This file is in the directory deploying the SDK.

Set “HOST” for cross compile like “PREFIX”
How to Customize ?(Raspberry pi 2)

• Deploy the SDK into Test Framework(conf.)
  – Set of Test Framework for the Target(raspi2)
    • Adding raspi2 target board configuration on
      /home/jenkins/overlays/boards/<targetname>.board

• A Sample target board configuration file is template-dev.board

• When you add a new board, you should use template-dev.board
How to Customize ?(Raspberry pi 2)

• Deploy the SDK into Test Framework (conf.)

inherit "base-board"
include "base-params"

IPADDR="set_ip_here"
LOGIN="root"
JTA_HOME="/home/a"
PASSWORD=""
PLATFORM="set platform here (see tools.sh)"
TRANSPORT="ssh"
ARCHITECTURE="set_ia32_or_arm_here"
SATA_DEV="/dev/sdb1"
SATA_MP="/mnt/sata"
USB_DEV="/dev/sda1"
USB_MP="/mnt/usb"
MMC_DEV="/dev/mmcblk0p2"
MMC_MP="/mnt/mmc"

LTP_OPEN_POSIX_SUBTEST_COUNT_POS="1319"
LTP_OPEN_POSIX_SUBTEST_COUNT_NEG="169"
EXPAT_SUBTEST_COUNT_POS="1769"
EXPAT_SUBTEST_COUNT_NEG="41"
How to Customize ?(Raspberry pi 2)

• Deploy the SDK into Test Framework (conf.)
  – For example, <target name>.board for Raspi2

```python
inherit "base-board"
include "base-params"
IPADDR="192.168.1.42"
LOGIN="root"
JTA_HOME="/home/a"
PASSWORD="pi"
PLATFORM="raspi2"
TRANSPORT="ssh"
ARCHITECTURE="arm"

MMC_DEV="/dev/mmcblk0p1"
MMC_MP="/mnt/mmc"
LTP_SYSCALL_COUNT_TPASS="4071"
LTP_SYSCALL_COUNT_TINFO="2776"
LTP_SYSCALL_COUNT_TCONF="140"
LTP_SYSCALL_COUNT_TFAIL="4"
LTP_SYSCALL_COUNT_TBROK="2764"
```

- Setting IP address of a target
- Login user name
- Directory to run some test
- LOGIN user password
- Setting Platform name
  ```bash
  elif [ "${PLATFORM}" = "raspi2" ];
  ```
- Setting Architecture name
- Setting a MCC Information of Raspberry Pi2
- Setting the configuration for LTP
How to Customize ?(Raspberry pi 2)

• 3 step to add New Target
  – Make the SDK for target
    • Using yocto project
  – Deploy the SDK into Test Framework
  – Set target information by GUI
How to Customize ?(Raspberry pi 2)

• Set target Information by UI
  – Select “Targets Status” on top screen of Test Framework
How to Customize ?(Raspberry pi 2)

• Set target Information by UI(conf.)
  – Select “New Node”

  Select “New Node”

  – Then, you can see a configuration form

  Select “Copy Exiting Node” and Enter template-dev in Copy from
How to Customize Raspberry Pi 2

• Set target Information by UI(conf.)
  – You enter just 2 forms as “Name” and “List of Key-values pairs”
  
  ![Image](image.png)

  **Enter the target name**

  **Enter the path of a file of target configuration**

  **Crick and save**
How to Customize ?(Raspberry pi 2)

• Set target Information by UI(conf.)
  – You can see a target list that New target board was added

New target name is raspi2

Finish adding new target as Raspberry pi 2!!!
How to Customize *(New Test Suite)*

• 3 step to add New Test Suite
  
  – Create a script for running a new test suite
  – Deploy the script and a test suite tarball
  – Set the test suite information by GUI
How to Customize ?(New Test Suite)

• 3 step to add New Test Suite
  – Create a script for running a new test suite
  – Deploy the script and a test suite tarball
  – Set the test suite information by GUI

Test Framework

<table>
<thead>
<tr>
<th>SDK (Cross-tools)</th>
<th>Test Suite</th>
</tr>
</thead>
<tbody>
<tr>
<td>minnow-jta</td>
<td>[bonnie] [bzip2]</td>
</tr>
<tr>
<td>renesas</td>
<td>[cyclictest] [LTP]</td>
</tr>
<tr>
<td>qemu-arm</td>
<td>[Dhrystone] [expat]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>benchmarks</th>
<th>Functional</th>
</tr>
</thead>
<tbody>
<tr>
<td>[himeno] [new test]</td>
<td>[netperf] [new test]</td>
</tr>
</tbody>
</table>

Jenkins

New Setting

Debian 7.X
How to Customize ?(Linux Test Project)

• Create the script named “ltp-all.sh” (1)

```bash
function test_build {
    make autotools
    ./configure CC="${CC}" AR="${AR}" RANLIB="${RANLIB}" LDFLAGS="${LDFLAGS}"
    --without-perl --without-python --target=${PREFIX} --host=${PREFIX} --prefix=`pwd`/target_bin
    --build=`uname -m`-unknown-linux-gnu
    make CC="${CC}"
    make install
}
```

To describe procedure of creating test module using cross compile.

```bash
function test_deploy {
    put -r target_bin /tmp/jta.$TESTDIR/
}
```

To describe procedure of deploying the test module to the target.

```bash
function test_run {
    safe_cmd "cd /tmp/jta.$TESTDIR/target_bin; ./runltp -f syscalls | tee $JTA_HOME/jta.$TESTDIR/$TESTDIR.log"
}
```

To describe commands to execute the test module on target.

In this case, to show running LTP command and collecting the result log.

```
tarball=ltp-full-20150420.tar.bz2
```
To describe tarball name of the adding test suite.
• Create the script named "ltp-all.sh" (conf.)

```bash
function test_processing {
    ## To judge test result
    assert_define LTP_SYSCALL_COUNT_TPASS
    assert_define LTP_SYSCALL_COUNT_TINFO
    assert_define LTP_SYSCALL_COUNT_TCONF
    assert_define LTP_SYSCALL_COUNT_TFAIL
    assert_define LTP_SYSCALL_COUNT_TBROK

    TPASS_CRIT="TPASS :"
    TINFO_CRIT="TINFO :"
    TCONF_CRIT="TCONF :"
    TFAIL_CRIT="TFAIL :"
    TBROK_CRIT="TBROK :"

    log_compare "\$TESTDIR" \$LTP_SYSCALL_COUNT_TPASS "$\{TPASS_CRIT\}" "TPASS"
    log_compare "\$TESTDIR" \$LTP_SYSCALL_COUNT_TINFO "$\{TINFO_CRIT\}" "TINFO"
    log_compare "\$TESTDIR" \$LTP_SYSCALL_COUNT_TCONF "$\{TCONF_CRIT\}" "TCONF"
    log_compare "\$TESTDIR" \$LTP_SYSCALL_COUNT_TFAIL "$\{TFAIL_CRIT\}" "TFAIL"
    log_compare "\$TESTDIR" \$LTP_SYSCALL_COUNT_TBROK "$\{TBROK_CRIT\}" "TBROK"

    echo "test_processing done"
}

. \$JTA_ENGINE_PATH/scripts/functional.sh
```

To describe judgment and analysis process of test results
Verify definitions
Define Keywords to search in the log
Compare definitions and result log
Define on "<target name>.board"

ltp-all.sh is inherited functional.sh
The above functions are called by it.
How to Customize ?(Linux Test Project)

• ltp-all.sh is inherited functional.sh.
  – “functional.sh” is defined on LTSI test by default.

source $JTA_ENGINEn PATH/scripts/overlays.sh
set_overlay_vars

source $JTA_ENGINEn PATH/scripts/reports.sh
source $JTA_ENGINEn PATH/scripts/functions.sh

pre_test $TESTDIR

if $Rebuild; then
  build
fi

deploy
test_run
get_testlog $TESTDIR
test_processing

To include common scripts and execute overlay using Test plan and spec files. Test plan and Spec files provide the very flexibility in configuring tests to be run on different boards and scenarios in the Test Framework.

Standard sequence for running test script on the Test Framework.
- “Pre_test” is checking precondition.
- “Build” is executing test_build function.
- “Deploy” is executing test_deploy function.
- “Get_testlog“ is getting the executing log.
- “test_run” and “test_processing” are defined on “ltp-all.sh”.

ELCE 2015 @Dublin
3 step to add New Test Suite

- Create a script for running a new test suite
- Deploy the script and a test suite tarball
- Set the test suite information by GUI

Test Framework

<table>
<thead>
<tr>
<th>SDK (Cross-tools)</th>
<th>Test Suite</th>
</tr>
</thead>
<tbody>
<tr>
<td>minnow-jta</td>
<td>Benchmarks</td>
</tr>
<tr>
<td>renesas</td>
<td></td>
</tr>
<tr>
<td>qemu-arm</td>
<td>Functional</td>
</tr>
</tbody>
</table>

[bonnie] [cyclictest] [Dhrystone] [himeno] [new test]
[bzip2] [LTP] [expat] [netperf] [new test]

Jenkins

Debian 7.X

New Setting

ELCE 2015 @Dublin
How to Customize ?(Linux Test Project)

• Deploy the script and test suite tarball
  – To create work directory “Functional.LTP.all” under /home/jenkins/tests/.
    • Arbitrary directory name can be used but the above is standard.
  – To obtain tarball of LTP from the below site
    • https://github.com/linux-test-project/ltp/releases/tag/20150420
  – To put the created script and tarball under Functional.LTP.all.

```bash
melco@debian-7:/home/jenkins/tests/ Functional.LTP.all$ ls
ltp-all.sh ltp-full-20150420.tar.bz2
```
How to Customize ?(New Test Suite)

• 3 step to add New Test Suite
  – Create a script for running a new test suite
  – Deploy the script and a test suite tarball
  – Set the test suite information by GUI
How to Customize ? (Linux Test Project)

• Set test suite Information by GUI
  – To select “New Test” on the left side of screen of Test Framework
How to Customize ?(Linux Test Project)

• Set test suite Information by GUI
  – To input Test name
  – To chose “Copy existing Test” and Copy from

Input Functional.LTP.all in “Test name”
Select “Copy existing Test” and input Functional.LTP.Open_Posix in “Copy from”
How to Customize ?(Linux Test Project)

• Set test suite Information by GUI
  – To input the created script path in “Command” field of “Execute shell” of “Test Run”
How to Customize ?(Linux Test Project)

• Set test suite Information by GUI
  – You can see new test suite name in Functional Tab

Finish adding new test suite as Linux Test Project !!!
Run new LTP on Raspberry Pi2

• Test Environment

- Running Test Framework on Debian 7.8
- Show the test result by web browser
- Adding the SDK Raspberry pi2
- Adding the LTP-20150420

To Connect with Ethernet

- Running poky 1.8 from Yocto project
- Kernel version: 3.18
Run new LTP on Raspberry Pi2 (Cont’d)

• To select LTP-20150420
  – To chose Functional.LTP.all

  Chose “Functional.LTP.all”

  Chose “Run Test Now”

ELCE 2015 @Dublin
Run new LTP on Raspberry Pi2 (Cont’d)

- To Run LTP-20150420

  1. Choose “raspberrypi2”
  2. Click and Run the test
Run new LTP on Raspberry Pi2 (Cont’d)

- We can Show the log with Console output at run time
Run new LTP on Raspberry Pi2 (Cont’d)

• To Show Test Results

Test Results is SUCCESS!

<table>
<thead>
<tr>
<th>Case</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPASS</td>
<td>4071</td>
</tr>
<tr>
<td>TINFO</td>
<td>2776</td>
</tr>
<tr>
<td>TCONF</td>
<td>140</td>
</tr>
<tr>
<td>TFAIL</td>
<td>4</td>
</tr>
<tr>
<td>TBROK</td>
<td>2764</td>
</tr>
</tbody>
</table>

- **TPASS** - Indicates that the test case had the expected result and passed.
- **TINFO** - Specifies useful information about the status of the test that does not affect the result and does not indicate a problem.
- **TCONF** - Indicates that the test case was not written to run on the current hardware or software configuration such as machine type, or, kernel version.
- **TFAIL** - Indicates that the test case had an unexpected result and failed.
- **TBROK** - Indicates that the remaining test cases are broken and will not execute correctly, because some precondition not met, such as a resource not being available.
Summary & Future Works

• Summary
  – LTSI Test Framework has already had many kinds of target boards and Test suites.
  – We showed How to Customize.
    • Add a new target board as Raspberry pi 2
    • Add a new test suite as LTP-20150420
  – We showed the result of running LTP on Raspberry pi 2

• Future Works
  – We try to add Kselftest
    • Kselftest is a quick method of running tests for the Linux kernel.
  – We think about making a SDK without yocto
    • We would like to use LTSI Test Framework for some product without yocto.
  – We think about how to analysis and judge test results.
Reference

• LTSI project:
  – http://ltsi.linuxfoundation.org/

• LTSI Test project:
  – http://ltsi.linuxfoundation.org/ltsi-test-project
  – Test Framework:
    • https://bitbucket.org/cogentembedded/jta-public.git