Kernel Testing: Tool and Techniques

Matt Porter
Texas Instruments
21 February 2013
Overview

• Why?
• Frameworks
• Lab Tools
• Tools
• Techniques (or Test Cases)
• What Else?
It's a BOF!

- This is an interactive session
- I want feedback and ideas along the way
- All ideas will be captured and documented on the elinux.org wiki
Why?

• Everybody's Level of Test Sucks
• Need a test for every line of code we write.
• It'll never be perfect, but we can be better.
• Working smarter applies to testing.
Frameworks

• Not all test frameworks have the same goals
  – Local unit tests
  – Build/test system validation

• LAVA
• OpenTest
• Autotest
• Jenkins
• Ktest
• ...

• Linaro's Test Framework
  – https://launchpad.net/lava

• Several components
  – Server
  – Dashboard
    • Displays test results
  – Scheduler
  – Dispatcher
    • Interacts with target hardware
  – Test Shell
    • Execute test plans and returns results to the Dashboard
Opentest

• TI's Test Framework

• Several components
  – Test Management System - TestLink
    • http://sourceforge.net/projects/testlink/
  – Test Management Controller
    • Dispatcher
    • Resource Manager
    • Writer – Manage test results
  – Service Providers
    • Build Execution Engines
    • Test Execution Engines

• CLI and Web UI
Autotest

- Autotest
  - [http://autotest.github.com/](http://autotest.github.com/)
- Designed for kernel testing but is a full automated test system for any kind of tests
- Test cases written in python
- CLI and web driven
- Used by Chromium OS
Jenkins

- CI Server
  - http://jenkins-ci.org/
- Usually consider a heavy weight corporate CI/build engine
- Can be used with a variety of plugins for
  - nightly build/tests
  - regression testing triggered by branch push
  - Remote monitoring via mobile device
ktest.pl

• Simple, upstream kernel test tool

• tools/testing/ktest/
  - Build and bisect tests
  - Launch tests on and gather results from a target system

• Snowball example shows how this tool works well for embedded linux testing
Automation Hardware Tools

- **Lava-lmp**
  - https://wiki.linaro.org/People/AndyGreen/Lava-lmp
  - Hardware tool to automate testing of various peripherals
    - SD muxing, USB OTG/host hotplug, SATA/eSATA hotplug, Ethernet hotplug, light sensors, etc
    - Design and PCB layout available.

- **Digiloggers LPC Power Controllers**
  - 8 port network controlled power switch, 129 USD

- USB serial dongle for console server or pick up a used one on eBay
- Alternative is to build a console server from a cheap developer board with many serial ports for RS-232.
Automation Software Tools

- Conmux
  - https://github.com/autotest/autotest/wiki/Conmux
  - Aggregate various serial consoles under one interface (USB connected, traditional serial, console server ports)
  - Control power or relays from escape command scripts

- Ttypersist
  - https://github.com/russdill/ttypersist
  - Hides disconnection of usb serial devices
Test Cases

- LTP
  - Filled with test cases but here in particular general kernel tests
- LTP-DDT
  - Lots of kernel driver test cases
- Lava-test
Build Our Own Test Cases

• Networking
  – Iperf
  – Netperf
  – Ping -f

• Block I/O
  – Bonnie++
  – Fstress
  – Dt (Data Test)

• SPI
  – Spidev + spidev_test.c (external loopback)
  – ?
Build Our Own Test Cases (continued)

• I2C
  - ?

• GPIO
  - Loopback?
  - Interrupts?

• DMA
  - Other device drivers
dmatest.ko

• PCIe
  - Random cards
  - Bus analyzer / exercisers
Getting Creative - sigrok

- In what ways can we use a general purpose data logging tool like sigrok for test?
  - http://sigrok.org

- Consider a PWM driver
  - Testing with a servo or LED for qualitative results is insufficient
  - Setting a PWM for a full range of supported values and verifying the exact expected results using an sigrok capture from a logic analyzer is real test coverage.
Getting Creative – spi slave zero

- In what ways can we get better at testing SPI drivers? One way is to build a universal spi slave device.

- The problem with SPI driver testing is always that we can't test every device.
  - But we can come closer if we have one device that exercises all spi protocol modes

- SPI Slave Zero is inspired by USB Gadget Zero
  - http://elinux.org/SpiSlaveZero

- SSZ is defined to do the following:
  - Reset and configure to polarity/phase via GPIOs
  - Handle commands to query max frequency, configure test mode word width, and set test mode
  - Test modes then allow read of a test pattern, sink of writes, and optionally write store and read back of stored data for data integrity testing.
What Else?

• Are there topics we missed?
• Other areas you've done creative tests?
• What works, what doesn't?
• Everything will be recorded and added to elinux.org wiki