CANONICAL

Firmware test suite (fwts)

Automated Testing of x86 firmware.

Presentation by
Colin King
colin.king@canonical.com
www.canonical.com
September 2011

Agenda

- Introduction
- Motivation
- History
- Key features
- Test flavours
- Utilities – extra goodies
- Participation

who I am, what I do

Colin King

colin.king@canonical.com

cking on freenode IRC

Ubuntu OEM Hardware Enablement Team

Started working for Canonical in 2008

Work on suspend/resume, hibernate/resume

Troubleshoot gnarly enablement issues

Work on Kernel + firmware bugs (BIOS, ACPI, etc)

Why develop the Firmware Test Suite?

x86 Firmware – its buggy!

Impact of buggy firmware:

- Machine won't boot or hangs
- Features don't work (e.g. hotkeys, battery, brightness controls, suspend/resume)
- Sub-optimal configuration – e.g. broken MTRRs
- Kernel can workaround some issues –> kernel warnings and error messages

..would be useful to detect and diagnose issues early and automatically.

Automate firmware checking

Useful to test BIOS updates during early enablement

- Automatically detect errors
- Sanity check core functionality
- Ensure Linux + firmware interactions work
- Catch kernel warnings
- Where possible suggest fixes or workarounds
- Gather data firmware specific data for debug

Aim – detect bugs and if possible get firmware fixed

- Automation: key to quick turnaround
- Get consistent results
- Detect regressions

History

History

Linux-ready Firmware Developer Kit (Intel)

Release 3.0 seemed to die around October 2007

Covered 50% of what we required

Could not plug interface into our test framework

Firmware Test Suite

May 2010 – Started with some tests from Firmware Developer Kit, new test framework, new logging mechanism, updated to work with 2.6.35+ kernels.

Releases are in-sync with Ubuntu release schedules

Maverick 10.10: fwts V0.18.04, 34 major tests

Natty 11.04: fwts V0.22.13, 48 major tests

Oneiric 11.10: fwts V0.23.23, 53 major tests

Tested against tens of hundreds of systems

Key features

Key features

Features

- Command line
  - Designed to be used by other test tools
  - ..or to be run stand alone
  - ..and to gather data for a developer
- Batch tests – run without supervision
- Interactive tests – e.g. hotkey, lid, AC power.
- Extensive logging:
  - Per test PASS/FAIL results
  - Explain reasons for failures (ADVICE lines)
  - Classify failures (CRITICAL, HIGH, LOW..)
  - Summarise results
  - Output log format can be configured
- Soak testing (suspend/resume, hibernate/resume)

Test flavours

Kernel Log Checks

The kernel does a lot of sanity checking...

- Kernel logs contain useful warnings about BIOS
  - ACPI and UEFI errors, can be a bit terse at times
- Parse, classify and log any errors
- Try to explain warnings and errors
- If possible, suggest fixes and/or workarounds

Unfortunately kernel messages change every release

- Use regular expressions to pattern match
- Try to keep fwts in sync with latest kernel

BIOS configuration inspection

BIOS tables can be extracted and inspected

- Sanity check configuration data
- DMI/SMBIOS tables
- MultiProcessor tables

Sanity check BIOS has configured system sanely

- APIC edge/level config
- EBDA region reserved in e820 table
- MTRRs set correctly
- MSRs set consistently across CPUs
- CPU NX bit enabled
- CPU virtualization extensions enabled
- HDA audio pin configuration

ACPI Tables

ACPI tables contain data + code:

- Configuration data
- ACPI Machine Language (AML) “byte code”
  - "a complete design disaster in every way" -Linus

Sanity check ACPI tables:

- Simple table checksums
- Limited checks on APIC, ECTD, FACP, HPET, MCFG,
  - RSDT, RSDP, SBST, XSDT
- Check for multiple MADT
- MCFG entries also reserved in e820 table
- Simple WMI GUID checks

ACPI Tables – AML checking

ACPI Machine Language Checks

- Uses the ACPICA execution engine in user space
- Evaluate common methods + objects
- Inspect return types – simple type checking
- Check for method parameter mismatches
- Check all acquired locks are released
- Sanity check a range of valid inputs

Cons:

- Cannot fully emulate I/O ops or interactions via SMIs or Embedded Controller.

AML Syntax check:

- Disassemble DSDT + SSDT, re-assemble
- Catches bugs in AML generated with Microsoft tools

UEFI checks

Work in progress!

- Like to sanity check boot services
- Would like to port fwts framework to UEFI
- Currently just have a tool to dump UEFI variables

Utilities – extra goodies

Utilities

Like a small firmware Swiss Army Knife

- Gather data for bug reports
- All the necessary utilities in one tool

Utilities included to dump and annotate:

- ACPI tables *
- BIOS EBDA region
- UEFI variables
- CMOS memory
- System memory map (e820 or UEFI memory map)
- MultiProcessor tables
- Firmware ROM(s)
- Disassemble AML

* Dumped data can be fed into some fwts tests

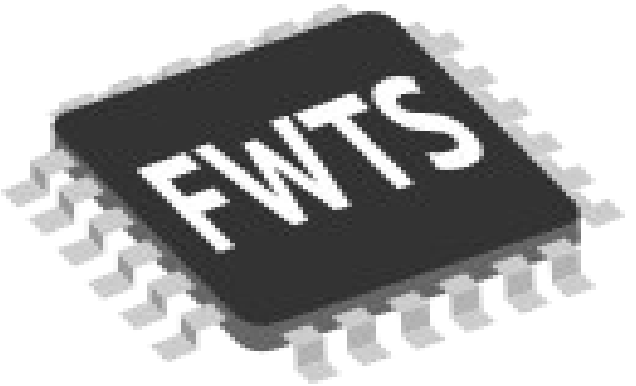
Participation

Participate..

Source:

git://kernel.ubuntu.com/cking/fwts/.git

Contributions welcome!



Documentation:

https://wiki.ubuntu.com/Kernel/Reference/fwts

..includes links to pages explaining each test

` with examples of usage and expected output

Project Page:

https://launchpad.net/~firmware-testing-team

CANONICAL

Questions please
Thank you

Colin King
colin.king@canonical.com
www.canonical.com