Delivering Predictability: The Yocto Project Autobuilder, Automated Sanity Testing, License Collection, and Build Statistics Tracking

Elizabeth Flanagan
Intel Corporation
April 11, 2011
Being proactive about code quality

- Reproducible builds
- Identify bugs and fix early and often
- Reduce time needed for code stabilization
- Avoid integration headaches
- Build performance history
- Manage the chaos
- License compliance
- Deep QA Testing
Being proactive about code quality

Maximizing your ability to respond to changes in a complex embedded ecosystem.
Being proactive about code quality

Reduce Software Development Lifecycle Churn.
Complexities

Our Recipe
Complexities

Our Recipe

- qemuarm
- qemumips
- qemuuppc
- qemu86
- qemu86_64
Complexities

- beagleboard
- qemuarm
- qemumips
- qemux86
- Our Recipe
- mpc8315e
- qemuuppc
- qemux86_64
- routerstation
- atom-pc
Complexities

Our Recipe

- qemuarm
- qemumips
- qemuppc
- qemux86
- qemux86_64

- beagleboard
- outerstation
- mpc8315e
- atom-pc
- emenlow
- Jasperforest
- n450
- sandybridge
- crownbay
- Crownbay-noemgd
Complexities

• One recipe
  • 5 architectures
  • 4 core BSP
  • 6 non-core BSPs in meta-intel
  • 15 x-compiles
  • But....
Complexities

• One recipe
  • 16 theoretical different image types per arch * 15 architectures
    – Not all arches support all image types
  • 240 total theoretically possible images
    – sato, lsb, sdk....
If we're not proactive about code quality, lots of things can go wrong.....
Complexities

It's only going to get more complex
What we need

- Reproducible builds
- Basic QA
- License tracking
- Finding the pain points
Delivering Predictability

Yocto Project

Autobuilder

Poky

Sanity Testing

License Wrangling

Build Statistics
Delivering Predictability

Yocto Project

Autobuilder

Poky

Sanity Testing

License Wrangling

Build Statistics
Autobuilders

• Production Autobuilders
  • Quickly respond to a fast changing code base
  • Avoid “Works on my machine”-itis
  • Find race conditions
  • Find host dependent issues
  • Find bad commits
  • Help bisect build failures
  • Help find dependency chain breakage
Autobuilders

- Developer Autobuilders
  - Test cross-compilation before commit
  - Production style builds
  - Small OS footprint
  - Build what you want to build
Autobuilders

- Yocto autobuilder
  - buildbot based
  - git://git.yoctoproject.org/poky-autobuilder.git
  - Setup in under 5 minutes!
Autobuilders

• Prerequisites:
  • Python 2.6
  • python-twisted
  • python-jinja2
  • python-twisted-mail
  • sqlite
Autobuilders

• Comes with
  • Basic pokyABConfig.py
  • Helper scripts
  • Easy Installer
Set up your own!

cd ~
git clone git://git.yoctoproject.org/poky-autobuilder.git
cd poky-autobuilder
./scripts/poky-setup-autobuilder both
source ~/.profile;
cd ../poky-master; make start
cd ../poky-slave; make start
Live Demo
Sanity Testing

- Extensible
- Frees up QA resources
- Reproducible smoke testing
- Multiarch/multiimage scenarios
- Can run automatically post build via local.conf
- Or via an autobuilder
Sanity Testing

Sanity Test Bitbake Class

Architecture/image based test scenario

Test library/runner

Test helper scripts
Sanity Testing

- meta/classes/imagetest-qemu.bbclass
- scripts/qemuimage-tests/scenario/${ARCH}/*
- scripts/qemuimage-testlib and runners
- scripts/qemuimage-tests/{sanity|tools}/*
Sanity Testing

- Test suite
  - Architecture and image based scenarios
  - Very easy to add already existing tests
Sanity Testing

sanity ssh
sanity scp
sanity dmesg
sanity zypper_help
sanity zypper_search
sanity rpm_query
sanity connman
sanity shutdown
Sanity Testing

- Tests
  - bash/expect based test scripts
  - called via test runners in scripts/sanity
  - Tests stored in qemuimage-testlib
  - More secure to create tap devs with poky-gen-tapdevs.
Sanity Testing

• Gotchas:
  • QEMU user NOPASSWD
  • More secure to create tap devs with poky-gen-tapdevs.
• For headless, see wiki docs:
  https://wiki.pokylinux.org/wiki/Enabling_Automation_Test_in_Poky
Delivering Predictability

Yocto Project

- Autobuilder
- Poky
- Sanity Testing
- License Wrangling
- Build Statistics
License Wrangling

- Verify image compatibility to required license type
  - non-GPLv3
- Provides an entire package directory tree
  - Actual licenses
  - Generics.
- Helps maintain license compliance
License Wrangling

Recipe contains:

- **LICENSE**
  - Tells license.bbclass the common license type
  - Symlink from license wrangling output to a generic

- **LIC_FILES_CHECKSUM**
  - License file URI
  - Checksum
  - Where we get the specific license
License Wrangling

License are found in:

${POKYBASE}/build/tmp/deploy/images/licenses
Delivering Predictability

Yocto Project

- Autobuilder
- Poky
- Sanity Testing
- License Wrangling
- Build Statistics
Build Statistics

Build level:
- Host info
- Elapsed build time
- CPU usage
- Build failure information
Build Statistics

Package level:
- List of events triggered
- Elapsed event time
- CPU usage
- Event failure information
Build Statistics

Image Type

Build

Package

Event

Event

Event

Package

Event

Event

Event

Event

build/tmp/deploy/images/licenses/:
poky-image-minimal-qemux86
`-- 201103251310
 | `-- build_stats
 | | `-- autoconf-native-2.65-r2
 | | | `-- do_compile
 | | | `-- do_configure
 | | | `-- do_fetch
 | | | `-- do_install
 | | | `-- do_patch
 | | | `-- do_populate_sysroot
 | | | `-- do_setscene
 | | | `-- do_unpack
 | `-- automake-native-1.11.1-r1
 | | `-- do_compile
 | | `-- do_configure
 | | `-- do_fetch
 | | `-- do_install
 | | `-- do_patch
 | | `-- do_populate_sysroot
 | | `-- do_setscene
 | | `-- do_unpack
Build Statistics

Gives us:

- Performance indicators
- Track down issues
  - CPU/Dependency/IO bound
- Visualize your build performance
  - Patch to pybootchart
Build Statistics Visualization
Build Statistics Visualization
Where do we go from here?
Where do we go from here?

Autobuilder:

- Meta-targets
- Helper script integration into main config
- Continuous integration
Where do we go from here?

License tracking:
- More generic license files
- Better LICENSE field parsing
Where do we go from here?

Build Statistics:
• Collect even more data.
  • Image size w/o free space.
• Better data visualization.
• Web based
Production autobuilder
Resources

- http://www.yoctoproject.org
- http://git.yoctoproject.org/cgit/cgit.cgi/poky-autobuilder/
- http://autobuilder.yoctoproject.org
- pybootchartgui patch for build statistics
- https://wiki.pokylinux.org/wiki/Enabling_Automation_Test_in_Poky
INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY RELATING TO SALE AND/OR USE OF INTEL PRODUCTS, INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT.

Intel may make changes to specifications, product descriptions, and plans at any time, without notice.

All dates provided are subject to change without notice.

Intel is a trademark of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2009, Intel Corporation. All rights are protected.