Designing a distro from scratch

using OpenEmbedded

Koen Kooi <koen.kooi@linaro.org>
ELC San Diego 2016
Overview

1. OpenEmbedded basics
2. DISTRO considerations
   - End of slides -
3. Real life examples

Don’t hesitate to interrupt if you have questions or remarks!

Slides available at https://goo.gl/HiRhi5
OpenEmbedded basics (1/2)

- OpenEmbedded is part of the Yocto Project umbrella organization
- OpenEmbedded is a buildsystem
- Closest equivalent: Buildroot

- OpenEmbedded is **NOT** a distribution
OpenEmbedded basics (2/2)

● OE consists of
  ○ Recipes
  ○ Config files
  ○ A task executor called bitbake

● Three orthogonal concepts
  ○ MACHINE.conf, a description of the target hardware (i.e. powerpc, screen, networking)
  ○ DISTRO.conf, a collection of policies for the build (i.e. systemd, PAM, rpm)
  ○ Image.bb, a description of the output filesystem in terms of packages and format (i.e. traceroute, ext4.gz)
So what is a DISTRO?

- A collection of policies
  - PAM/no PAM
  - Systemd, sysvinit or upstart
  - Package management and format
  - License ideology (GPLv3)
So what is a DISTRO? - continued

- A collection of less obvious policies
  - Compiler and compiler version (Clang, gcc)
  - C library (glibc, musl, uclibc, bionic)
  - ABI (x32, ilp32, hardfloat)
  - Architecture support
So what is a DISTRO? - continued

- A workflow
  - Build environment
  - License compliance
  - Distribution of binaries
  - CI loop
So what is a DISTRO? - continued

And hopefully a community!
Adding confusion

- Appliances like TVs smash everything together, MACHINE, DISTRO and image.
  - TVs are where failed mobile distros go to die: tizen, webos and firefoxOS.
- Preinstalled software confuses people: “My beaglebone can’t do static IPs!!”
- The line between images and DISTRO policy is fine: if one image uses connman and the other NetworkManager can they both be part of the same DISTRO?
- Developers are lazy and poke(y) at DISTRO vars in MACHINES and image recipes: “This board requires an ancient Xorg version“
Distro consideration - build environment

Examples used:

- Poky: https://www.yoctoproject.org/tools-resources/projects/poky
- Angstrom: https://github.com/angstrom-distribution/angstrom-manifest
- Linaro RBP: https://github.com/96boards/oe-rpb-manifest
Metadata Layers

Since 2011 layers are the preferred way to separate metadata.

- MACHINE support in ‘BSP’ layers
- DISTRO in ‘distro’ layers
- Everything else in feature layers (e.g. meta-ruby)

But not everyone adheres to that split, most of the time without realizing it

- Usually one layer per git repo
- Add yours to http://layers.openembedded.org/!
Metadata Layers - continued

An OE DISTRO needs to have ways to:

- Fetch layers
- Enable layers in the build
- Test for layer interaction
- Easily contribute back upstream
- Override recipes from layers
Metadata Layers - continued

Fetching layers:

- Poky uses an offline script to merge everything into single git tree
- Angstrom pre-v2014.12 used a home grown script to fetch all git trees
- Angstrom v2014.12 and newer use google repo
- Linaro RPB uses google repo
- Cliff Brake uses git submodules
Metadata Layers - continued

Enabling layers:

- Angstrom has bblayers.conf managed by git
- Linaro RPB has bblayers.conf managed by git
- Please don’t use TEMPLATECONF
- Layerstack should be static for a DISTRO
- BSP layers should be added with care
Metadata Layers - continued

Layer interaction:

- Position in bblayers.conf and LAYER_PRORITY matters
- “Foo_1.0.bb” in layer A can make “Foo_1.5.bb” in layer B disappear
- Bbappends tend to interact badly
- Immediate expansion, :=, doesn’t do what you think it does
- Tools exist, but still tedious work
Contributing back upstream:

- Hoop jumping: Poky requires a script to untangle the right upstream
- Confusion: Angstrom v2015.12 fetches 43 git trees and enables 57 layers
- Lack of guidance
- Some layers do have a ‘contribution’ section in their README
  - [http://git.yoctoproject.org/cgit/cgit.cgi/meta-maker/tree/README](http://git.yoctoproject.org/cgit/cgit.cgi/meta-maker/tree/README)
Contributing

The meta-openembedded mailing list (openembedded-devel@lists.openembedded.org) is used for questions, comments and patch review. It is subscriber only, so please register before posting.

Send pull requests to openembedded-devel@lists.openembedded.org with '[:meta-maker]' in the subject.

When sending single patches, please use something like:
'git send-email -M -l --to=openembedded-devel@lists.openembedded.org -'
Metadata Layers - continued

Overriding recipes is an essential feature:

- Allows backports
  - without needing to fork upstreams
  - Without waiting for upstreams to catch up
- Allows differentiation without needing to fork upstreams
- Allows blocking unwanted changes
- Override overrides...
Metadata Layers - continued

Overriding recipes:

- Bbappend
- Complete recipe
- Layer.conf magic
Build environment

Angstrom and Linaro have google repo manifest and scripts in a single git tree:

https://github.com/96boards/oe-rpb-manifest

https://github.com/angstrom-distribution/angstrom-manifest
$ mkdir ~/bin
$ PATH=~/bin:$PATH

$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo

$ repo init -u https://github.com/96boards/oe-rpb-manifest.git -b jethro

$ repo sync
<table>
<thead>
<tr>
<th>Machine</th>
<th>Architecture</th>
<th>Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>6boards-64</td>
<td></td>
<td>metan02</td>
</tr>
<tr>
<td>dragonboard-410c</td>
<td></td>
<td>meta-qcom</td>
</tr>
<tr>
<td>hikey</td>
<td></td>
<td>meta-96boards</td>
</tr>
<tr>
<td>ifc6410</td>
<td></td>
<td>meta-qcom</td>
</tr>
</tbody>
</table>
Available Distributions
Please choose a distribution

- ma5
- meta-rpb
- rpb-wayland
- meta-rpb

< Cancel >
Welcome to OpenEmbedded Reference Platform Build (OE RPB)

For more information about OpenEmbedded see their website:
http://www.openembedded.org/

Your build environment has been configured with:

MACHINE = 96boards-64
SDKMACHINE = x86_64
DISTRO = rpb

You can now run 'bitbake <target>'

Some of common targets are:
core-image-base
core-image-minimal

rrMBP:Linaro koen$
Build environment - continued

Show
https://github.com/96boards/oe-rpb-manifest/pull/15/commits/e81c2f2ea7990afd70d837ebc57270b6c5931eef in browser
From here no more real slides
Backup Slides

Slides available at https://goo.gl/HiRhi5
Binary packages

PRSERV

Sstate reuse

Failure tracking
CI integration
App developers
Release branches
‘Generic’ machines
BSP integration
API
ABI
magic vars
Fixups - gcc -noatime
There's a set of simple things projects can do the be more friendly (or unfriendly) to distributions... (speaking as someone who builds a distribution).

Good things
* Use a standard build/make system (autoconf, cmake, python setuptools, whatever, something that is pretty widely used)
* Clear license declaration (COPYING file)
* include unit tests (make test/make check); a distribution can and will use this this to verify they integrated the component correctly
* use pkg-config for dependencies
* regular releases, at least for bugfixes and security fixes (bonus points for having maintenance releases against latest stable in addition to more major releases, but rolling release is fine)
* Know what an "ABI break" is if you are providing a library
(Note: C++ makes it much harder to keep ABI, but it can be done, see the Qt folks)

Bad things
* Custom Makefile hackery that is not parallel build safe or ignores DESTDIR etc etc
* Unit tests that fail always on the official release
* No clear declaration of license