Debian & Yocto: State of the Art

Kazuhiro Hayashi, Toshiba Corporation
Manuel Traut, Linutronix GmbH
Baurzhan Ismagulov, ilbers GmbH
Embedded Linux Conference Europe 2018
Oct. 23, 2018
Agenda

1. Introduction
2. Existing Projects
3. Vision
4. Current Development
5. Summary
Agenda

1. Introduction
2. Existing Projects
3. Vision
4. Current Development
5. Summary
Motivation

• Introduce existing approaches

• Share the vision

• Collect feedback
Building Products with Linux

• **Product**
  - Linux base system
  - Customizations
  - Product-specific application
  - Third-party software

• **Development**
  - Create a single ready-to-flash image (U-Boot, kernel, rootfs)
  - Developer-oriented, repeatable process

• **QA**
  - Bug fixes and security updates for development and production releases
  - Long maintenance terms (at least 10 years after EOL)

• **License compliance**
What to Do

• Select appropriate base system
  – Linux distribution

• Provide tools to manage the base system
  – Build system
  – Framework for customization and product maintenance
Debian GNU/Linux

• Binary distribution

• Features
  – Multiple CPU architectures support
  – Official cross-toolchains
  – Security updates
  – Long-term support
  – Machine-readable licensing information (DEP-5)

• Wish list
  – Integration tool
  – Easier customization
  – Easier introduction of new architectures
Yocto Project

- Source-based reference distribution

- Features
  - Bitbake integration tool
  - Easily customizable
  - Layered collaboration model
  - Standalone SDK generation
  - Easier introduction of new architectures / SoCs

- Wish list
  - Reduce build times
  - Long-term support
Agenda

1. Introduction

2. Existing Projects

3. Vision

4. Current Development

5. Summary
Existing Projects

- nneta-elbe
  - [https://github.com/linutronix/nneta-elbe](https://github.com/linutronix/nneta-elbe)
- Isar
  - [https://github.com/ilbers/isar](https://github.com/ilbers/isar)
- meta-debian (Deby)
  - [https://github.com/meta-debian/meta-debian](https://github.com/meta-debian/meta-debian)

- Other projects
  - debos
  - vmdebootstrap / vmdb2
  - etc.
Existing Projects: elbe

- nneta-elbe is based on elbe  [https://elbe-rfs.org](https://elbe-rfs.org)
  - RFS description in XML with variant management
  - Reproducible image built
    - including bootloader installation, partitions, UBI, ...
  - Using Debian binary packages is the default
  - Customized packages and own software supported
  - Yocto-compatible cross-toolchain generation as SDK
  - License text collection / semi-automatic SPDX conversion
Existing Projects: nneta-elbe

- nneta-elbe
  
  https://github.com/Linutronix/nneta-elbe

  - Bitbake user frontend for elbe
  - Recipes for images
  - Recipes for source-packages
  - Bitbake just calls elbe commands and keeps track of the project status

- Our reasons for meta-eid
  - Bootstrapping new architectures
  - Speedup source package build with cross-compile
Existing Projects: Isar

- [https://github.com/ilbers/isar](https://github.com/ilbers/isar)

**Package builder and image generator**
- Installs the base system
- Builds and installs product packages and customizations
- Creates ready-to-use images

**Uses**
- Base system: Debian binary packages
- Debian tools: dpkg-buildpackage, reprepro, apt, debootstrap...
- BitBake: Efficient package build dispatcher
- Layering for collaboration

**Live demo at the Technical Showcase**

Create build chroot

Build custom packages

Create target rootfs

Install custom packages

Create target image
Existing Projects: Isar

- **Some features**
  - Native and cross-compilation
  - Upstream package patching
  - Debian SDK generation
  - Output to apt as well as images
  - Different CPUs and Debian versions in one product
  - Variant management through dependencies

- **Our reasons for meta-eid**
  - Keep the right mix
  - Don’t reinvent the wheel
  - Improve the implementation
Existing Projects: meta-debian

- [https://github.com/meta-debian/meta-debian](https://github.com/meta-debian/meta-debian)
  - Metadata set for Yocto Project to build Debian sources
  - Ready-to-use image generation for embedded boards
    - Bootloader, kernel, root filesystem, SDK
- **Source-based distribution**
  - Cross-build everything from scratch
  - No need to keep binaries
  - High customizability
  - Various target CPUs and tuning available
- **Our reasons for meta-eid**
  - Build time improvement by reusing binaries
  - Less complexity and maintenance cost
Comparison

- Providing similar features
- Partially based on the common tools

<table>
<thead>
<tr>
<th></th>
<th>ELBE</th>
<th>Isar</th>
<th>meta-debian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base system</td>
<td>Debian binary package</td>
<td>Debian binary package</td>
<td>Packages cross-built from Debian source</td>
</tr>
<tr>
<td>Integration tool</td>
<td>ELBE commands + bitbake wrapper</td>
<td>bitbake</td>
<td>bitbake</td>
</tr>
<tr>
<td>Package building</td>
<td>Debian toolchain</td>
<td>Debian toolchain</td>
<td>OE-Core</td>
</tr>
<tr>
<td>Image generation</td>
<td>debootstrap</td>
<td>debootstrap</td>
<td>OE-Core</td>
</tr>
<tr>
<td>Customization</td>
<td>Single XML file</td>
<td>bitbake recipes, (patched)</td>
<td>bitbake recipes</td>
</tr>
</tbody>
</table>
1. Introduction

2. Existing Projects

3. Vision

4. Current Development

5. Summary
meta-eid

- meta-eid is about collaboration
  - meta-eid ‘founders’
    - nmeta-elbe
    - Isar
    - meta-debian

- Pronounced as “aid”
Vision I

• **Easy to use**
  – One-command building with bitbake
  – Classes and configuration options for common use cases

• **Easy to customize**
  – Changing build options, dependencies, packaging

• **Tooling**
  – Prefer existing tools
  – Not as a fork but wrapping, connecting, enhancing tools
    • Contribute to upstream projects
  – Clean, minimal architecture
Vision II

• Build targets
  – Build debianized and non-debianized sources
  – Generate ready-to-use images
  – Generate standalone SDK

• Performance
  – Reuse binary packages
  – Cross-building
  – Avoid unnecessary steps, parallelization blockers

• Product-oriented
  – Reproducibility
  – Metadata layering
Layering

Board Y-specific packages

E.g., extensions for debugging

QEMU environment for development

Sharable resources (3rd-party components, ...)

Project A-specific data (packages, project own repository, etc.)

Board X-specific packages (kernel, boot loader, firmware, etc.)

meta-board-x

meta-project-a

meta-company

meta-qemu

meta-board-z

meta-board-y

meta-board-z+

meta-project-b

meta-eid
Nice to Have

• Bootstrapping Debian
  – Tuning for specific CPUs
  – Product-wide ‘EXTRA_CFLAGS’
  – Footprint
Agenda

1. Introduction
2. Existing Projects
3. Vision
4. Current Development
5. Summary
Use Cases

(1) Rebuild existing Debian source package

- hello.dsc
- app1.dsc
- app2.git

- bitbake hello
- bitbake app1
- bitbake app2
- bitbake u-boot

- hello.deb
- app1.deb
- app2
- u-boot.bin

(2) Build debianized source

(3) Build non-debianized source

(4) Generate rootfs

- bitbake debian-image
- bitbake -c sdk debian-image

- rootfs
- SDK

(5) Generate SDK

Embedded Linux Conference Europe 2018
Required Functions

- **Source fetcher**
  - bitbake + extensions for Debian source packages (dsc, git, ...)

- **Dependency resolution**
  - Use apt for build- and run-time dependency resolution
  - Use bitbake dependency mechanism for building 'local' recipes
  - Use both at the same time without duplication

- **Package builder**
  - sbuild

- **Cross-toolchain for non-Debian sources**
  - Debian chroot

- **Image generator**
  - Debootstrap and other tools
Workflow

Debian apt repo.

- .deb
- .dsc

Fetcher (libbb)

- boot loader
- kernel
- app

- non-debianized source
- Raw-build

- debianize
- sbuild

- Debianized source

- debianize
- Raw-build

- non-debianized source

- schroot

- native/cross-toolchain

- package
- binaries
- (bootloader, kernel)

Debianized

- .deb

image builder

- rootfs
- SDK

Non-debianized
Current Development Status

- Debianized source
- non-debianized source
- Fetcher (libbb)
- sbuild
- debianize
- Raw-build
- schroot
- native/cross-toolchain
- package
- binaries
- Debian apt repo.
- .deb
- .dsc
- Debianized
- image builder
- rootfs
- SDK
- Non-Debianized
- Building debianized source
- Debianizing & building sample application
- Prototyped basic functions:
Examples: Rebuild Debian source package

- hello_2.9-2+deb8u1.bb

```bash
inherit debian-dsc
inherit sbuild
DSC_URI = "${DEBIAN_REPO}/pool/main/h/${PN}/${PN}_${PV}.dsc;md5sum=abc..."
```
Examples: Rebuild Debian source package

- **hello_2.9-2+deb8u1.bb**

  ```
  inherit debian-dsc
  inherit sbuild
  DSC_URI = "${DEBIAN_REPO}/pool/main/h/${PN}/${PN}_${PV}.dsc;md5sum=abc..."
  ```

  Automatically fetch all components in .dsc
  Automatically build with sbuild
Examples: Build non-Debianized package

- foo_git.bb

```bash
inherit debianize
inherit sbuild

PV = "1.0"
SRC_URI = "git:/github.com/zuka0828/${PN}.git;protocol=https"
SRC_REV = "abc..."

S = "$\{WORKDIR\}/git"

DEPENDS += "baz"
DEB_DEPENDS = "libssh-dev"
DEB_RDEPENDS = "bc"
```
Examples: Build non-debianized package

- foo_git.bb

```bash
inherit debianize
inherit sbuild

PV = "1.0"
SRC_URI = "git:/github.com/zuka0828/${PN}.git;protocol=https"
SRC_REV = "abc..."

S = "${WORKDIR}/git"

DEPENDS += "baz"
DEB_DEPENDS = "libssh-dev"
DEB_RDEPENDS = "bc"
```

- Automatically Debianize source with dh_make
- Dependency on another recipe in meta-eid
- Dependencies on Debian packages
- Both go into Build-Depends
Agenda

1. Introduction
2. Existing Projects
3. Vision
4. Current Development
5. Summary
Next steps

- **rootfs and SDK generation**
  - Current approach: debootstrap
  - Evaluate existing Debian and Yocto image generation tools

- **apt repository management**
  - Reuse binary packages generated in previous builds

- **‘Raw’ building of non-debianized source**
  - Writing commands in recipes without debianizing sometimes preferred

- **Easy customization**
  - `do_patch()` or hook function to unpacked sources

- **Cross-sbuilding**
  - Debian multiarch and cross-toolchain

- **Reproducibility**
  - Metadata and package management

Embedded Linux Conference Europe 2018
Conclusion

• Customizing Debian-based root filesystems with bitbake is possible

• We want to rely on Debian’s cross-building features

• Building Debian packages (dsc) should be possible without having a recipe

• A PoC for build dependency resolution is available

• We need to support options for doing the same thing in different ways
  – E.g., cross-build, native build, non-debianized build

• Many projects with similar goals exist – welcome to join
How to Join

• GitHub
  - used to host code and track issues / travis for testing
  - https://github.com/eid-project/meta-eid

• Mailing List
  - Used for patch review and technical discussions
  - meta-eid@googlegroups.com
  - Subscribe: meta-eid+subscribe@googlegroups.com
  - Archive: https://groups.google.com/d/forum/meta-eid

• Instant messaging in discussion
Questions?