Deby - Reproducible and Maintainable Embedded Linux Environment with Poky

Kazuhiro Hayashi
CE Workgroup, The Linux Foundation (TOSHIBA Corporation)
Embedded Linux Conference Europe 2016
Oct 12, 2016
About this project

• **Shared Embedded Linux Distribution Project**
  – One of the activities of CEWG project
  – Goals: Create an industry-supported distribution of embedded Linux and provide support for long term

• **For more information about this project**
  – Shared Embedded Linux Distribution
    • [http://elinux.org/Shared_Embedded_Linux_Distribution](http://elinux.org/Shared_Embedded_Linux_Distribution)
  – CE Workgroup Linux Foundation
    • [http://www.linuxfoundation.org/collaborate/workgroups/celf](http://www.linuxfoundation.org/collaborate/workgroups/celf)
Motivation

• **Linux is running on many kind of embedded systems**
  – Including the systems in civil infrastructure

• **Things to be considered to choose a base distribution**
  – The number of supported packages
  – Package versions
  – Supported hardware
  – Stability, number of bugs were fixed
  – The frequency of security updates and supported timespan
  – How to compile and customize packages
In our case

• What we want to do
  – Make custom embedded Linux environments

• What we need
  – Wider hardware support
  – Stability
    • Well tested packages are required
    • Many embedded developer are still want to use stable version
  – Long-term support
    • Over 10 years support required, especially for security fixes
    • (This is what we would like to contribute something)
  – Fully customizable build system
Our solution

**Yocto Project "poky"**
- One of the most popular reference distributions for embedded Linux
- Fully customizable build system
- Supports numerous embedded boards including modern ones
- Can be extended by meta-layer

**Debian GNU/Linux**
- Support many kind of CPUs: x86, ARM, PowerPC, MIPS (32bit/64bit)
- Release a stable version after two years of testing
- Long-term support for 5 years by Debian-LTS project
Our solution

Yocto Project "poky"
- One of the most popular reference distributions for embedded Linux
- Fully customizable build system
- Supports numerous embedded boards including modern ones
- Can be extended by meta-layer

Debian GNU/Linux
- Support many kinds of CPUs: x86, ARM, PowerPC, MIPS (32bit/64bit)
- Release a stable version after two years of testing
- Long-term support for 5 years by Debian-LTS project

meta-debian

Deby - Reproducible and Maintainable Embedded Linux Environment with Poky
Definitions of the terms

• **meta-debian**
  - A meta layer for the poky build system
    - Completely separated from OpenEmbedded-Core and other layers
    - Allows cross-building Linux images using Debian source packages
  - Source code
    - [https://github.com/meta-debian/meta-debian.git](https://github.com/meta-debian/meta-debian.git)

• **Deby**
  - A reference distribution built with poky+meta-debian
  - **Deby** = Debian + poky
  - **Deby** = Debian-like
    - Cross-built from Debian source, but not same as Debian binary
Build system structure (poky)

Upstream source code

Fetch

poky build system

meta (OpenEmbedded-Core)

Board-specific metadata

A  B  C

Build

Poky A  Poky B  Poky C

Deby - Reproducible and Maintainable Embedded Linux Environment with Poky
Build system structure (poky + meta-debian)

Upstream source code

Debian source packages

poky build system

Fetch

meta-debian

meta (OpenEmbedded-Core)

Board-specific metadata

A

B

C

Build

Deby A

Deby B

Deby C

Deby - Reproducible and Maintainable Embedded Linux Environment with Poky
Target versions of Deby

Upstream source code

Debian source packages

Yocto Project
Stable version: 2.0 jethro
Development version: 2.2 morty

Deby A
Deby B
Deby C
Purpose of Deby

• Create embedded Linux environments with
  – Wide embedded CPU support
  – Stability
  – Long-term support
  – Fully customizable build system

  With Debian stable release + LTS
  With poky build system

• Contribute to upstream
  – Debian, Debian LTS, and Yocto Project
Development policies of Deby

- **Follow Debian’s packaging (debian/rules)**
  - Use the same `configure/compile commands and options`, `install paths`, `binary package name`, and `dependencies` as Debian

- **Add patches for supporting cross-compile**
  - Usually imported from OE-Core

- **Customize for embedded system if necessary**
  - Remove unneeded features, dependencies and packages
    - Ex: udeb packages for Debian installer

- **See also**
  - [http://events.linuxfoundation.org/sites/events/files/slides/LinuxCon2015_meta-debian_r7.pdf](http://events.linuxfoundation.org/sites/events/files/slides/LinuxCon2015_meta-debian_r7.pdf)
Quick start

1. Download the build tools
2. Setup build directory
3. Build minimal Linux image
4. Run minimal Linux image on QEMU

5. Build & install minimal SDK
6. Build application with SDK
7. Run application on QEMU

• See also meta-debian/README.md
Download build tools

- **Download poky**

  ```
  $ git clone git://git.yoctoproject.org/poky.git
  $ cd poky
  $ git checkout jethro
  ```

- **Download meta-debian into the poky directory**

  ```
  $ cd poky
  $ git clone https://github.com/meta-debian/meta-debian.git
  $ cd meta-debian
  $ git checkout jethro
  ```

- ↔ meta-debian specific step
Setup build directory

Change the default configuration

- Enable meta-debian layer
- Enable "deby" distro (DISTRO = "deby")
- The default target machine is "qemux86" (MACHINE = "qemux86")
- TEMPLATECONF is used by oe-init-build-env script

```
$ export TEMPLATECONF=meta-debian/conf
```

Run startup script

- This setup a build directory and environment variables automatically
- (builddir): name of build directory (optional)

```
$ source /path/to/poky/oe-init-build-env (builddir)
```
Build minimal Linux image

• Run bitbake

```bash
$ bitbake core-image-minimal
```

• Built images (case of qemux86)
  – Output directly
    • /path/to/builddir/tmp/deploy/images/qemux86
  – Kernel
    • bzImage-qemux86.bin
  – Root filesystem
    • core-image-minimal-qemux86.ext4
    • core-image-minimal-qemux86.tar.gz
Run minimal Linux image on QEMU

- Run built images on QEMU environment
  - `qemux86 / qemux86-64 / qemuppc / qemumips`
    ```
    $ runqemu qemux86 nographic
    $ runqemu qemux86-64 nographic
    $ runqemu qemuppc nographic
    $ runqemu qemumips nographic
    ```
  - `qemuarm`
    ```
    $ runqemu qemuarm nographic bootparams="console=ttAMA0"
    ```
Build & install minimal SDK

• Run bitbake

$ bitbake meta-toolchain

• Output (Host: x86_64, Target: qemux86)
  – /path/to/builddir/tmp/deploy/sdk/qemux86/deby-glibc-x86_64-meta-toolchain-i586-toolchain-8.0.sh
  • Self-extracting script

• Install SDK to host environment

$ sh deby-glibc-x86_64-meta-toolchain-i586-toolchain-8.0.sh
Build application with SDK

• Create hello.c and Makefile

```c
/* hello.c */
#include <stdio.h>
int main(int argc, char **argv)
{
    printf("hello world\n");
    return 0;
}
```

# Makefile

```makefile
hello: hello.o
```

• Export SDK environment variables and make

```
$ source /opt/deby/8.0/environment-setup-i586-deby-linux
$ make
```

• See also Yocto Project Application Developer’s Guide

Run application on QEMU

• Copy hello to the filesystem image

```bash
$ cd /path/to/builddir/tmp/deploy/images/qemux86
$ sudo mount -o loop \         
core-image-minimal-qemux86.ext4 /mnt
$ sudo cp /path/to/hello /mnt
$ sudo umount /mnt
```

• Run application on QEMU

```bash
$ runqemu qemux86 nographic
...
192.168.7.2 login: root
# /hello
hello world
```
New features

• **Supported Yocto Project version**
  - 2.0 jethro (Stable)
  - 2.2 morty (Development)

• **Kernel**
  - 4.4 LTS
  - 4.1 LTSM

• **The number of available recipes**
  - Approx. 500

• **Newly supported target machine**
  - BeagleBoard, PandaBoard
New features

- **Package management**
  - Run-time dpkg / apt

- **Tag based source code fetch and build**
  - Rebuild the Linux image that was built at the specific time

- **Summary generation**
  - Generate summary information of packages included in rootfs and SDK
Package management

• This feature is available in OE-Core

• How to enable package management feature
  – Package management feature is disabled by default
  – Add the following definition into local.conf

```
EXTRA_IMAGE_FEATURES += "package-management"
```

• With package management feature, we can...
  – Add binary packages into run-time environment
    • Temporally install/uninstall packages for system evaluation
    • Temporally install -dbg packages for debugging
  – Upgrade packages without stopping system
  – Install / upgrade packages without building & installing rootfs again
Deby - Reproducible and Maintainable Embedded Linux Environment with Poky
rootfs with package management

- **Source code**
  - A.git
  - B.git
  - C.git
  - X.git

- **Recipe**
  - A.bb
  - B.bb
  - C.bb
  - X.bb

- **Build**
  - A.deb
  - B.deb
  - C.deb
  - X.deb

- **Build environment**
  - rootfs

- **Run-time environment**
  - Deby rootfs

- **Package pool**
  - (apt repo.)

- **Published by web server**

- **apt-get install**

- **Keep package data**
  - /var/lib/dpkg

- **rootfs**

- **apt**
  - install / upgrade / uninstall dynamically with apt-get
Tag based source code fetch and build

- **Issues in the default behavior of meta-debian**
  - No reproducibility
    - Cannot reproduce rootfs/SDK that was built at the specific time
    - Recipes always fetches the latest source code (the latest git commit)
  - To automatically import all security updates

- **Reproducible build**
  - One of the essential features in long-term maintenance
  - Useful for finding the source of issue in the old released image

- **Solution**
  - STEP1: Register a release tag in git repositories every release
  - STEP2: Reproduce an old release image by specifying a tag name
    - Add a new global variable: `GIT_REBUILD_TAG`
STEP1: Register a release tag

- **Source code A**
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6
  - 7

- **Source code B**
  - 1
  - 2
  - 3

- **Source code C**
  - 1
  - 2

- **Source code D**
  - 1
  - 2
  - 3
  - 4

- **Meta-debian**
  - 1
  - 2
  - 3
  - 4

**Version number**

**Time**

**Git repositories**
STEP1: Register a release tag

Register a common release tag

Source code A

Source code B

Source code C

Source code D

... meta-debian

Release1

Time

1 2 3 4 5 6 7

1 2 3

1 2 3

1 2 3

1 2

Metadata fetches the latest source code by default
STEP1: Register a release tag

Source code A
1 r1 2 3 r2 4 5 6 7

Source code B
1 r1 2 3

Source code C
1 r1 2

Source code D
1 r1 2 r2

... meta-debian
1 2 r1 3 r2 4

Release2

Time
STEP1: Register a release tag

Source code A

1. r1
2. r1
3. r2
4. r2
5. r3
6. r3
7. r3

Source code B

1. r1
2. r2
3. r3

Source code C

1. r1
2. r3

Source code D

1. r1
2. r2
3. r3

meta-debian

1. r1
2. r2
3. r3
STEP2: Reproduce an old release "r1"

Deby - Reproducible and Maintainable Embedded Linux Environment with Poky
STEP2: Reproduce an old release "r1"

Fetch the latest source codes by default
STEP2: Reproduce an old release "r1"

Fetch "r1" tagged source code by setting GIT_RELEASE_TAG = "r1"

Don’t fetch the latest version
How to register tag and rebuild

- Create git repository mirrors with docker
  - Follow the instructions in meta-debian-docker/README.md

```
$ git clone https://github.com/meta-debian/meta-debian-docker.git
$ cd meta-debian-docker
$ ./make-docker-image.sh
$ sudo docker run -d -p 10022:22 meta-debian:1 /etc/sv/git-daemon/run -D
```

github.com  docker (172.17.0.2)
How to register tag and rebuild

• Setup poky + meta-debian

$ export TEMPLATECONF=meta-debian/conf
$ source ./poky/oe-init-build-env

• Override the git server related variables in local.conf

Fetches source code from github by default
How to register tag and rebuild

• **Setup poky + meta-debian**
  ```
  $ export TEMPLATECONF=meta-debian/conf
  $ source ./poky/oe-init-build-env
  ```

• **Override the git server related variables in local.conf**

```
github.com -> docker (172.17)
mirror
```

```bash
DEBIAN_GIT_URI = "git://172.17.0.2"
DEBIAN_GIT_PROTOCOL = "git"
MISC_GIT_URI = "git://172.17.0.2"
MISC_GIT_PROTOCOL = "git"
LINUX_GIT_URI = "git://172.17.0.2"
LINUX_GIT_PROTOCOL = "git"
SRC_URI_ALLOWED = "git://172.17.0.2"
```
How to register tag and rebuild

• bitbake something

$ bitbake core-image-minimal

• Get list files that have git repositories used in the build
  – Example: /path/to/builddir/tmp/git.list.172.17.0.2
How to register tag and rebuild

- Register a tag "testtag" to the repositories

```bash
$ git clone https://github.com/meta-debian/meta-debian-scripts.git
$ cd meta-debian-scripts
$ ./git-tagger.sh git.list.172.17.0.2 172.17.0.2 testtag
```
How to register tag and rebuild

Rebuild the old image

```bash
$ export TEMPLATECONF=meta-debian/conf
$ source ./poky/oe-init-build-env
$ echo 'GIT_REBUILD_TAG = "testtag"' >> conf/local.conf
$ bitbake core-image-minimal
```

Deby - Reproducible and Maintainable Embedded Linux Environment with Poky
Summary generation

- **Summary information of OSS is required for products**
  - List of installed software
  - Version of each software
  - Source URI where the source code fetched
  - License of each software

- **Issues of the default poky and meta-debian**
  - Generate only a list of installed software in rootfs and SDK

- **Solution**
  - Add functions (hooks) to automatically generate summary information into rootfs and SDK recipes
Summary generation

- Poky’s build flow

Source code

Recipe

Build

Install

Package pool

metadata

Package: X
Version: git0+dae8d9bd7f-r0
PackageArch: i586
Depends: B, C, ...
...
Summary generation

• How to collect information for each package

Source code
- A.git
- B.git
- C.git
- X.git

Recipe
- A.bb
- B.bb
- C.bb
- X.bb

Build
- .deb
- .deb
- .deb
- .deb

Install
- rootfs
- rootfs
- SDK

STEP1: Embed additional metadata

Package: X
Version: git0+dae8d9bd7f-r0
PackageArch: i586
Depends: B, C, ...

DebianSourceName: X
DebianSourceVersion: 1.2-3
RemoteSourceURI: git://...
License: GPLv2+
Summary generation

• How to generate summary of each deployment

Source code
- A.git
- B.git
- C.git
- X.git

Recipe
- A.bb
- B.bb
- C.bb
- X.bb

Build
- A.deb
- B.deb
- C.deb
- X.deb

Install
- rootfs
- SDK

STEP2: Generate summary from metadata of installed package
Summary generation

- Format of summary information (CSV)

<table>
<thead>
<tr>
<th>PackageName</th>
<th>PackageVersion</th>
<th>RecipeName</th>
<th>DebianSourceName</th>
<th>DebianSourceVersion</th>
<th>RemoteSourceURI</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>busybox</td>
<td>git0+8feca13beb-r0</td>
<td>busybox</td>
<td>busybox</td>
<td>1:1.22.0-9+deb8u1</td>
<td>git://localserver/busybox.git;protocol=git;branch=jessie-master</td>
<td>GPLv2</td>
</tr>
<tr>
<td>cpuset</td>
<td>git0+79474ed070-r0</td>
<td>cpuset</td>
<td>cpuset</td>
<td>1.5.6-4+deb8u1</td>
<td>git://localserver/cpuset.git;protocol=git;branch=jessie-master</td>
<td>GPLv2</td>
</tr>
<tr>
<td>ethtool</td>
<td>git0+bb474b5bf6-r0</td>
<td>ethtool</td>
<td>ethtool</td>
<td>1.3.16-1</td>
<td>git://localserver/ethtool.git;protocol=git;branch=jessie-master</td>
<td>GPLv2</td>
</tr>
</tbody>
</table>
Conclusions

- **What is Shared Embedded Linux distribution**
  - Share the work of maintaining long-term support for an embedded distribution, by leveraging the work of the Debian project
  - Metadata for building embedded Linux systems using Debian source packages
  - Implemented as an independent layer of OpenEmbedded-Core

- **Deby is intended to provide**
  - Wide embedded CPU support
  - Stability
  - Long-term support
  - Fully customizable Linux
Conclusions

• Several new features
  – Package management
    • dpkg / apt
    • Dynamically install/upgrade/uninstall packages at the run-time
  – Tag based source code fetch and build
    • Reproduce an old release image by setting GIT_REBUILD_TAG
  – Summary generation
    • Automatically generate summary information of rootfs and SDK
## Current development status

<table>
<thead>
<tr>
<th><strong>Debian version</strong></th>
<th>8 jessie (the latest stable)</th>
</tr>
</thead>
</table>
| **Yocto Project version** | 2.0 jethro (stable)  
2.2 morty (development) |
| **Kernel** | 4.4 LTS  
4.1 LTSI |
| **BSP** | QEMU: x86 (32bit, 64bit), ARM, PowerPC, MIPS  
VMware Player  
BeagleBoard  
PandaBoard  
MinnowBoard  
Raspberry Pi 1/2  
Intel Edison board |
| **init manager** | busybox, systemd |
| **Packages** | Approx. 500 |
Future works

• Keep following updates of poky and Debian
  – Yocto Project 2.2 will be released soon (Oct. 28, 2016)

• Support more embedded boards

• Improve build time for upgrading target images
  – Related work (Binary package based approaches)
    • Isar (https://github.com/ilbers/isar)
    • ELBE (http://elbe-rfs.org/)
    • Smart Package Manager (https://github.com/ubinux/smart2)

• Efficient recipe creation
  – Add a (semi-)automated recipe generator from debian/rules

• Integrate with LTSI test environment (Fuego)
Please give us feedback

- **E-mail**
  - yoshitake.kobayashi@toshiba.co.jp
  - kazuhiro3.hayashi@toshiba.co.jp

- **Repository**
  - https://github.com/meta-debian/meta-debian.git
Questions?