LINUTRONIX

How to combine Debian and Yocto/Bitbake?

Manuel Traut <manuel.traut@linutronix.de>

ELCE 2017 - Prague

What's next?

1. Why Yocto?
2. About Debian
3. Benefit of a combination
4. Existing solutions
5. Perfect combination

1) Why Yocto?

- ★ What is Yocto
- ★ Typical usage
- ★ Limitations

What is Yocto?

- Tooling for building your own Linux distribution and SDK
- Defines a format that eases sharing of compile recipes and patches
- Powerful configuration management for different but similar images
- Based on Open-Embedded
- Example distribution “Poky” available

Typical usage

- Use Poky example distribution
- Add meta-layers from chip and/or hardware vendor
- Add 3rd party layers, e.g. for QT5
- Add own layer with image customization and own applications

Limitations

- Recipes from different layers might be incompatible
- Packages need to be built before they can be used
- Quality of recipes is hard to verify
- Security tracking/updates need to be done
- No LTS/updates available
- Reproducibility is not completely given (host dependencies)

2) About Debian

- ★ The universal OS
- ★ Debian and embedded?
- ★ Usage
- ★ Limitations

The universal OS

- Debian provides more than a pure OS, it comes with over 51 000 packages
- The infrastructure, documentation and build-tools are open-source
- Debian takes security very seriously
- Many security advisories are coordinated with other free software vendors and are published the same day a vulnerability is made public

Debian and embedded?

Packages are available for

amd64 arm64 armel armhf i386 mips mips64el mipsel

powerpc (not in stretch) ppc64el s390x

Also **cross-toolchains** for different architectures are available in Debian/stretch

Usage

- Debootstrap embedded RFS (e.g. for arm) into a directory
- Use pbuilder or a cross-compiler to build own applications and copy to RFS-dir
- Remove unneeded files (man-pages, i18n, ...) from RFS-dir
- Build FS (ext4, etc) or disk / UBI images using some tools and scripting
- Extract licence information and retrieve source-code of all used packages

Limitations

- Only limited number of HW architectures supported
- No HW specific binary packages like special gstreamer plugins are available

The following limitations are given in Debian but are solved with E.L.B.E.

- SDCard / UBI / etc. image generation
- SDK generation and licence information and source package extraction
- Reduce image footprint
- Own application integration

3) Benefit of a combination

★ Yocto + Debian = ?

Yocto + Debian = ?

Use from Yocto

- Task scheduling
- Configuration management
- (cross-) compile from source
  if necessary
- SDK generation

Use from Debian

- Well maintained packages
- Security tracking
- Binary packages
  if available and useful
- Source packages
  if necessary

4) Existing solutions

- ★ meta-debian
- ★ meta-isar
- ★ nneta-elbe
- ★ Comparison table

meta-debian

- ~600 .bb recipes, using sources from Debian/jessie
- Build rules optimized for embedded and retrieved from 'debian/rules'
- Long-term Linux kernel from CIP (Civil Infrastructure Project)
- Supports SDK generation
- Very active ~2000 commits on github
- Not compatible with existing Debian binary packages

meta-isar

- Uses Debian binary packages from stretch, jessie, wheezy or raspian-jessie
- Optional: building Debian source packages in a chroot (with qemu-user)
- Needs 'sudo' with nopasswd for several tasks
- Default image size ~300MB
- ~100 commits on github

nneta-elbe

- Proof of concept E.L.B.E. frontend (9 commits on github)
- Uses Debian stretch binary pkgs (tested with armhf)
- Optional: build binary pkgs from source within elbe-pbuilder
- Source and binary pkgs built with pbuilder available in a signed Debian repository
- Bitbake generates elbe-xml and schedules elbe-pbuilder and elbe-image-build jobs
- Builds licence information
- SDK generation currently not implemented, but easy because available in E.L.B.E.

	meta-debian	meta-isar	nneta-elbe
Yocto-style config management & app integration			
HW-specific SW like kernel / bootloader buildable			
Use Debian sources			
Default footprint / reducible?	10MB	300MB / with Yocto methods	300MB / not yet
Non-Debian archs buildable			
For specific HW optimized binaries			
Export used source code	download dir		easy to develop
SDK generation available			easy to develop
Generate licence information	csv		XML & plain-text
Reproducibility	pkg v. by git tags	no VM / shared chroot for all builds	VM/ pbuilder
Bitbake file per Debian package needed	+ a git repo	not for bin-pkg	not for bin-pkg
Use Debian binary packages			
# of available Debian packages	limited / ~600	all	all
Effort needed to adapt buildsystem to new Debian release	very high		
Generate signed Debian repos of self built packages	unsigned deb	unsigned deb	dsc + deb

5) Perfect combination

- ★ My personal wishlist
- ★ Conclusion
- ★ Your ideas

My personal wishlist

- Collaborate with 'rebootstrap.sh'

  to bootstrap Debian with settings from Bitbakes machine config
- Use Debian multiarch for cross-compiling any (modified) src pkg

  for a self bootstrapped architecture
- Mix usage of cross-built Debian pkgs via Bitbake

  with official Debian binary pkgs (for official supported architectures)
- Having reproducible builds for all Debian packages

Conclusion

- 3 implementations but only 2 use-cases
  - meta-debian is good for architectures that are NOT available in Debian
  - meta-isar and nneta-elbe can only be used if the architecture is available in Debian
  - nneta-elbe is a proof-of-concept but it's already very powerful thanks to the E.L.B.E backend
- Porting Debian bootstraping to Bitbake might be interesting for Debian and Yocto

Your ideas

To improve the usage of Debian in embedded Linux projects?

References

nneta-elbe / E.LB.E.

http://elbe-rfs.org

http://github.com/linutronix/nneta-elbe

http://github.com/linutronix/nneta-elbe-extended

http://elinux.org/images/e/e5/Using_ELBE_to_Build_Debian_Based_Embedded_Systems.pdf

Debian

https://wiki.debian.org/HelmutGrohne/rebootstrap

https://wiki.debian.org/ReproducibleBuilds

meta-isar

http://github.com/ilbers/meta-isar

https://events.linuxfoundation.org/sites/events/files/slides/isar-elce-2016.pdf

meta-debian

http://github.com/meta-debian

https://elinux.org/images/2/2e/MiniDebianConfJapan-Yoshi.pdf

LINUTRONIX

Contact

Manuel Traut
<manuel.traut@linutronix.de>

Linutronix GmbH

Bahnhofstraße 3
88690 Uhldingen
Germany