ilbers
Technology for better life

Isar

Build Debian-Based Products with BitBake

Baurzhan Ismagulov

Embedded Linux Conference Europe
Oct 11 - 13, 2016
Berlin, Germany

Contents

About Us

- Based in Munich, Germany and Saint Petersburg, Russia
- Provide software development services since 2010
- Areas of expertise:
  - Linux and open-source software
  - Application and driver development
  - Real-time and safety-critical software development
  - Networking and wireless
  - Virtualization

Motivation

- Build and maintain products
- Simple to use
- Low per-package effort
- Efficient in use
- Reproducible builds
- Customer requirements:
  - Native compilation
  - Legal clearing
  - Safety certification
  - Maintenance: 10 years after EOL
- Use cases:
  - Several products (esp. with common components)
  - Multiple vendors (upstream / Debian / BSPs / libs / own sw...)

Product Building

- Create a base system
  - Define a list of binary Debian packages as a base system
  - Define base system customizations, such as inittab, fstab, etc.
- Use a clean environment (chroot / VM) for building the following components
  - Product's modified Debian packages
  - Product's own driver, library, and application packages
- Support multiple diverse targets in one project
  - Debug / release builds
  - Images for different products
  - Images for different hardware platforms
- Good support for typical project use cases, such as
  - Create complete, bootable images automatically, with one command
  - Do that reproducibly, for any older version

Candidate Codebases

Feature	Debian	Yocto
Distro type	Binary	Source
# of packages	Huge	Basic
Packages	Pre-built	Have to be built every time
Toolchain	Pre-built	Has to be built every time
On-demand building	Custom setup	Single command, OOTB
Build host	Debian (chroot, VM)	“Any” (issues on less-tested ones)
	Rich tool ecosystem	
		Build process fully customizable
		Layering (company / vendor / product)

Candidate Tools

- ELBE: Embedded Linux Build Environment
  - Central tool written in Python
  - Supports building and image generation
  - Metadata (build rules, etc.) in a single XML file
- Many Debian image builders: http://people.linaro.org/~riku.voipio/debian-images/
- Yocto
  - BitBake
  - Recipe-, dependency-based building
  - Metadata in individual recipes

Strengths and Shortcomings

- ELBE
  - Important for us: Supports many use cases OOTB
  - Doesn't scale well for our use cases
- Many Debian image builders: http://people.linaro.org/~riku.voipio/debian-images/
  - “Each tool is tailored for the developer's use case and personal taste”
  - Product development is more than just creating a rootfs
- Yocto
  - Flexible, modular build system
  - Well-thought-out layering
  - Re-building the world from sources is a waste of developer and machine time

Isar: Debian + BitBake

- Smaller tools for particular tasks
- BitBake
- Layering
- Pre-built Debian toolchains
- multistrap for creating rootfs images
- Generates complete images, including e.g.:
  - Partition table
  - U-Boot
  - U-Boot environment
  - Kernel
  - Root filesystem

Project History

- 2004: Started with shell scripts
- 2011: Migrated to BitBake
- 2015: Started open-sourcing features

How Isar Works

- Set of BitBake recipes
- One recipe per source package (almost one-liners)
- ATM, Debian build-deps duplicated in .bb (TODO)
- Build:
  - Create ARM (or i386) build chroot
  - Clone package repos, dpkg-buildpackage (with qemu)
  - Create root filesystem using Debian binary packages
  - Finalize the installation (post-install, etc.)
  - Customize the rootfs
  - Install built packages
  - Pack the complete firmware image

Common Tasks

- Create a new product
- Add a new package
- Build firmware image
- Override an upstream package

Configuration Management

- isar
- Debian dists, pool
- FOSS package repos
- meta-COMPANY
- meta-PRODUCT
- Application repos

Override an Upstream Package

- Before: Unmodified, community-built Debian binary package installed
- After: Modified Debian package built and installed
- Steps:
  - Create the company repo of the package
  - Create a branch for the modification
  - Make changes
  - Create a tag
  - Create a recipe
  - Add dependencies

Lessons Learned

- Avoid massive changes to upstream code
- Work with the community
- Invest in open tools
- Upstream changes
- You [will] want efficiency

Next Steps

- Isar: Building into apt
- Isar: Release creating Debian .dsc
- Isar: Release building from Debian .dsc
- BitBake: Understand Debian build-deps (.dsc backend?)
- Build caching: apt-aware build task
- https://github.com/ilbers/isar/blob/master/TODO
- Suggestions?
- Patches!

References

- Isar introduction: https://lists.debian.org/debian-embedded/2016/03/msg00000.html
- Isar code: https://github.com/ilbers/isar/
- ELBE: http://elbe-rfs.org/
- meta-debian: http://elinux.org/images/7/74/LinuxCon2015_meta-debian_r7.pdf
- Debian image builders: http://people.linaro.org/~riku.voipio/debian-images/

Questions?