

Behind the Curtains of Making Real Consumer Devices using Debian

Christopher Obbard

chris.obbard@collabora.com

@obbardc



About me

- Engineer at Collabora
- Electronics Engineer
- Working on..
 - Custom distros for cloud, embedded and PC
 - Continuous integration
 - Packaging
 - OTA upgrades
 - Tooling

chris.obbard@collabora.com

Overview

- Part 1: process to bring up a new board
- Part 2: supporting the product to market
- Q&A
- see previous ELCEU2020 talk for more detailed information on debos

Before you start hacking...

- Requirements!
 - Peripherals you care about
 - Performance
- SoC development kit/documentation
- BSP blobs/source/documentation
- Store it all somewhere shared! (NextCloud?)
- Tech contact at SoC vendor

BSP (Board Support Package)

- pre-built image is important!
- source code/build scripts/yocto layers etc
- bootloader/kernel sources
- old release, downstream patches
- depends on your luck!

Pre-built image hacking

- validate everything works quickly
- document:
 - how to build flasher software
 - how to flash (& share with team!)
- get a shell: serial port
- display? hdmi/touchscreen?
- make sure peripherals work in BSP

Replacing things...

- partition layout
 - analyse on device
 - fakemachine can load image and extract blobs/filesystems
- create an image early
 - image-partition Debos action with the correct partition layout
 - use ripped blobs from BSP image
- top-down replacement
 - start with rootfs (use debos to create a Debian rootfs)
 - kernel (build from BSP source)
 - bootloader (build from BSP source)
- make sure things work still!

CI integration

- debos recipes in GitLab (or elsewhere)
- reproduce the image build in the cloud
 - nightly builds
 - merge requests
 - release tags
 - email notifications on failure

Security

- Lockdown serial ports
 - bootloader
 - kernel
 - check all ports!
- Lockdown services
 - debug services, ssh-server
 - factory installation scripts

Packaging your app

- Container vs native
- Open build service
 - build Debian packages
 - GitLab stores source code
 - builds dependences in order
 - creates APT repo

OTA upgrades

- MVP & iterative feature development
- Requirements:
 - Secure upgrades!
 - No bricking
- A/B “slot” system
 - Two slots containing a rootfs, bootloader chooses which slot
 - Upgrade happens in userspace
 - Allows for rollback on failure

OTA upgrades

- RAUC is a nice framework
 - Integrates with Debos
 - Really generic
 - Signed/encrypted upgrade bundles
 - Casync integration allows installation of chunked update and only upgrades what has changed

Chain-of-trust

- Secure-boot depends on vendor
- Convert rootfs to readonly
- Use dm-verity for filesystem verification
- Use overlayfs to replace configuration files
- If chain-of-trust broken, do not load app

Product lifetime

- Base Kernel on LTS stable
- Potential automated rebasing in GitLab?

Automated testing

- LAVA continuous integration system
 - Flash image to board
 - Run some tests
 - Report overall pass/fail
 - Email notifications to team
- Submit test jobs through GitLab
 - Push image out if tests pass?

Factory provisioning

- Seperate PC to run tests (NUC?)
- Use the same image as your app
- Extra scripts to program serial numbers etc
- Database of test results by serial number
- Barcode scanner

Thank you & questions!

- type: message
priority: high
body: Collabora is hiring...
recipient: you
calltoaction: <https://col.la/join>
- type: message
priority: medium
body: Ask questions!
recipient: you
calltoaction: The chatbox

Christopher Obbard

chris.obbard@collabora.com

@obbardc