How we added software updates to AGL

Phil Wise

Advanced Telematic SYSTEMS

Advanced Telematic SYSTEMS

ATS Advanced Telematic Systems.

Open source and open standard
for connected mobility.

AGL Automotive Grade Linux

Open Source

Linux for cars

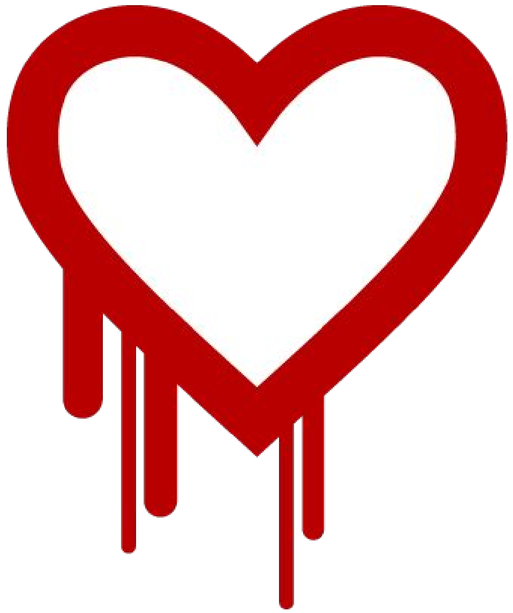
Linux Foundation Project

Members are mostly car companies & suppliers

More information in Walt Miner’s talk

Why software updates are needed

?



Why software updates are needed early in the release cycle

You obviously need it eventually

But having it early is great:

- Battle harden the process
- Test fleets
- Sales demos
- If the development team use it daily, it won’t suck

GOALS

Goals

AGL isn’t a single product/platform

Lots of products

Lots of boards

Must meet people where they are

Simple adoption

Must be shareable

Portability

To get benefits of collaboration

More than just OSS/on github

Needs to be portable to lots of applications

Update methods

Package-based (rpm, dpkg etc.)

+ Simple

— Unsafe for power-off

— Dependency resolution can get suck

Full file system update

+ Robust

— Tends to end up device-specific

— Need rsync or similar

Atomic differential (OSTree)

+ Combines robustness with minimal bandwidth consumption

+ Modern approach

+ Easy to make reusable

OSTree

OSTree Background

Not developed by me

Colin Walters / Gnome

Originally designed for Gnome CI

“Like git but for a root file system”

OSTree

“It’s like git for a filesystem”

OSTree

- Like git for a filesystem
- Commits are a rootfs
- 1 flash partition
- Multiple systems (chroots)

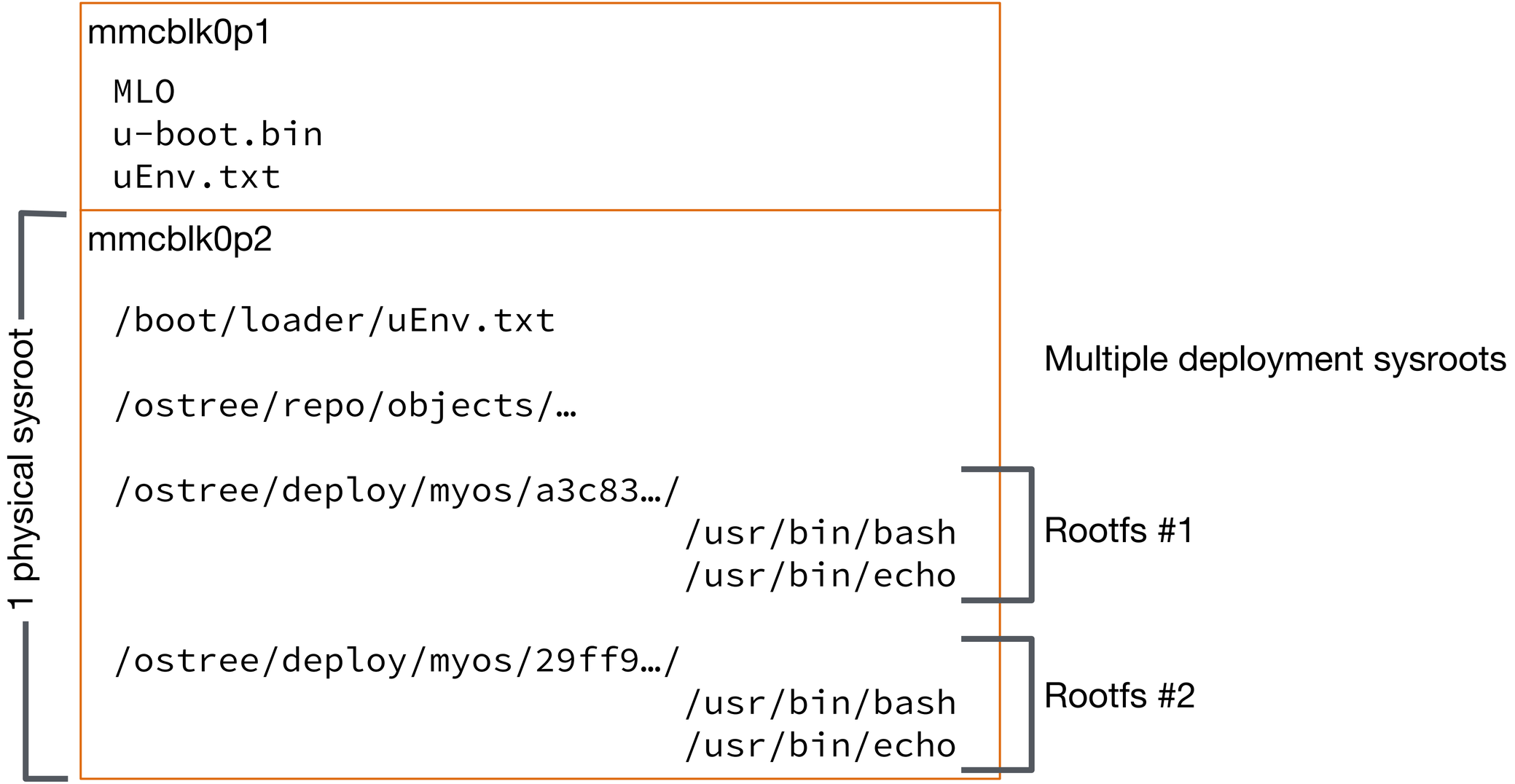
OSTree

“It’s like git for a filesystem”

OSTree

- Incremental fetches (like git pull)
- Hardlink identical files
- Not actually git: Extended Attributes for selinux/smack

OSTree basics



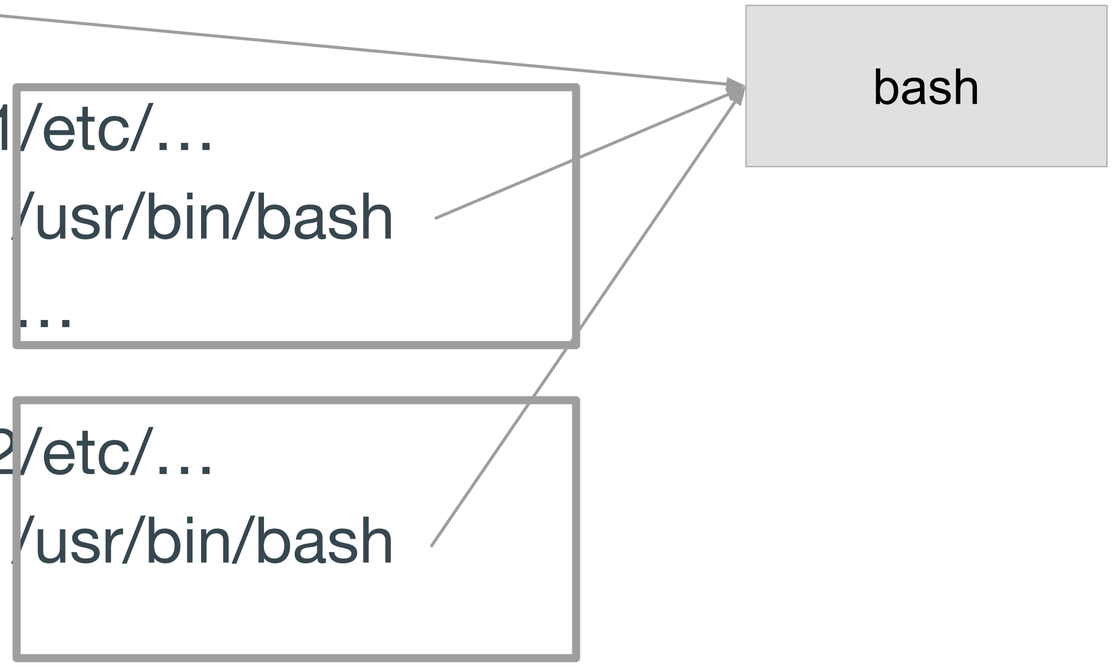
OSTree Hard link trees

Files shared using hard links:

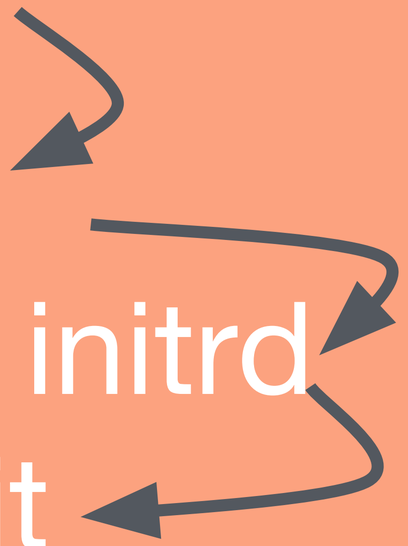
/ostree/repo/4b/cdef…
/b2/…

/ostree/deploy/osname/v1/etc/…
/usr/bin/bash
…

/ostree/deploy/osname/v2/etc/…
/usr/bin/bash



U-Boot
Kernel
OSTree initrd
/sbin/init



Boot Process

- Bootloader picks deployment
- Boot kernel
- initrd chroots to correct deployment

Yocto / OE Integration

Integration Part 1/2

Added image_types_ostree to bitbake

Modifies rootfs to be updatable

Moves R/W data to var

Usrmove

Commits result to an OSTree repo

Uploads to Software Update Server

Creates initial bootable flash image

...all from ‘bitbake myimage’

Yocto / OE Integration

Integration Part 2/2

Also need some per-board work

Mostly bootloader

Today:

- Renesas R-Car Porter
- Qemu (U-Boot)
- Minnowboard Max (U-Boot!)
- R-Pi (chain load U-Boot)

Other bootloaders straightforward

User data in /var

RO / RW Split

OSTree uses hardlinks to share files

Must not modify them mounted RO

Writable files in /var

Case of AGL Application Framework (1).

Two update domains.

1. Full file system updates with OSTree.

2. Application updates with Application Framework.

Application database is located in **/var/lib/afm.** Some applications come pre-installed in the file system, while others can be installed in runtime.

How do we manage **/var/lib/afm**?

Case of AGL Application Framework (2).

Just ignore initial database.

+ Almost zero integration effort

— No pre-installed apps

Merge initial database in /usr/afm with the one generated runtime.

+ Applications can be updated both with OSTree and AppFW

— A lot of integration effort, merger can fail or give unexpected results.

Populate /var/lib/afm from /usr/afm just once.

+ Moderate integration effort, very robust.

— Pre-installed apps are populated just once, can’t update apps with OSTree.

Getting Started (AGL)

Getting Started with AGL and SOTA

The ‘Charming Chinook’ release of AGL comes with SOTA.

Pass ‘agl-sota’ to aglsetup.sh to enable it

=> Done

Code is in

https://wiki.automotivelinux.org/subsystem/agl-sota/ostree

~~meta-agl-extra/meta-sota~~

Getting Started (OpenEmbedded)

Getting Started without using AGL

Functionality extracted into

‘meta-updater’ layer

Can be easily added to a OE project

See garage-quickstart-rpi on our github

github.com/advancedtelematic

Using SOTA for CI

Using SOTA for CI

AGL uses SOTA to test CI R-Pi builds

Easier than switching cards/netbooting

Serve OSTree repo from CI build server over http

AGL users have SOTA already

Everyone else: meta-updater

Summary

AGL is now SOTA-enabled out of the box

Available to everyone via meta-updater

Questions?

ATS Advanced Telematic Systems GmbH
advancedtelematic.com

Phil Wise
Senior Software Engineer

phil@advancedtelematic.com

Back up

ATS Garage.

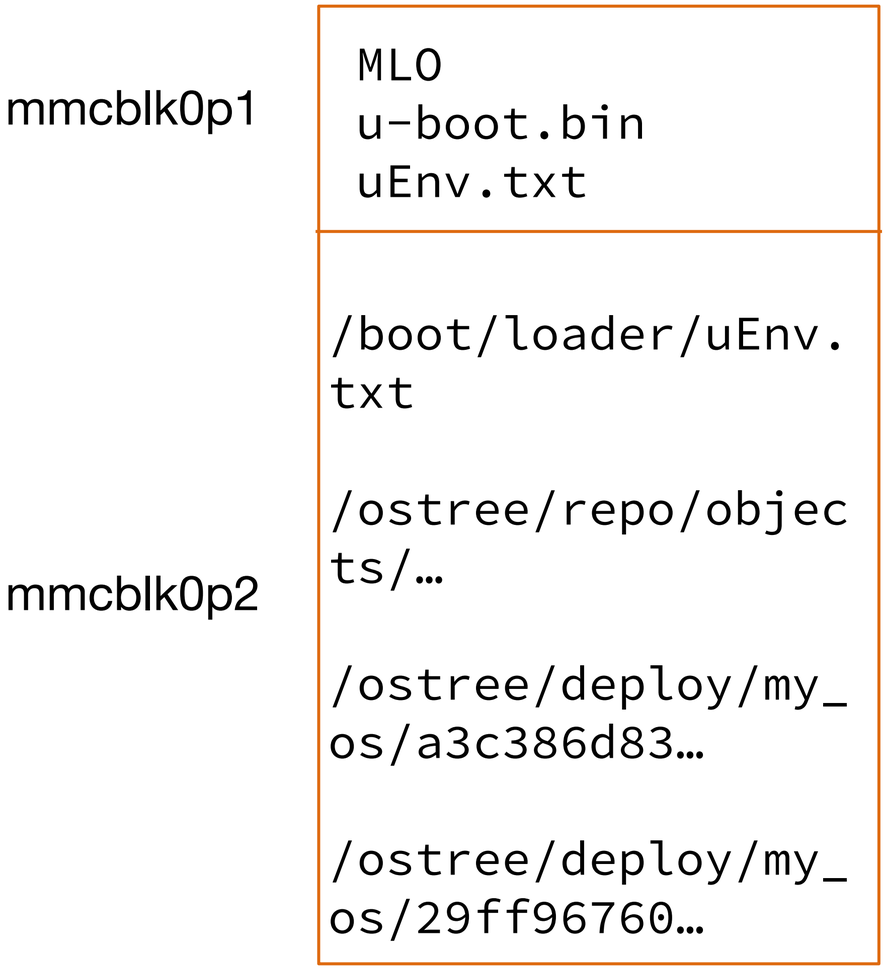
Web service for deploying software to embedded Linux devices.

atsgarage.com

OSTree.

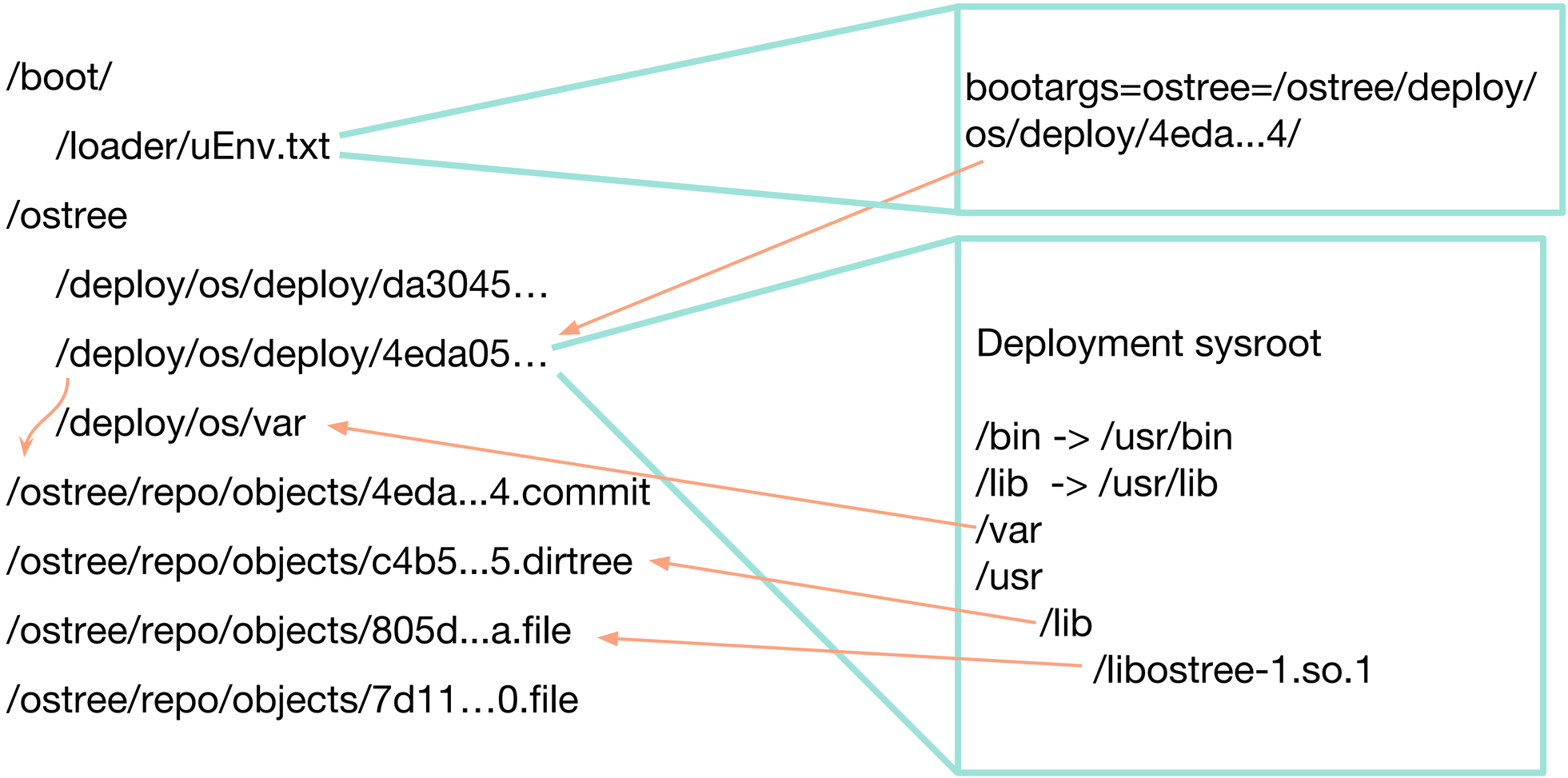
- Git-like tool for bootable filesystems. Designed and maintained by GNOME/Red Hat developer Colin Walters.
- Original purpose: continuous integration for GNOME team.
- Target platform: PC running Linux. Not designed for embedded systems, limited support for other POSIX-compliant OSes.
- More info on ostree.readthedocs.io

OSTree basics.



- Physical sysroot - just one per device. Contains OSTree repo, OSTree deployments and /boot directory with information about current deployment sysroot. Device never boots into physical sysroot.
- Deployment sysroots - one device can contain multiple deployments (two by default). They are stored in /ostree/deploy under physical sysroot. Physical sysroot is mounted to /sysroot mountpoint of deployment sysroot so that OSTree can access its repository.

OSTree basics: sysroot



OSTree integration.

Already done in meta-updater

1. Prepare physical sysroot.
2. Prepare deployment sysroot.
3. Make bootloader and initramfs work together to boot the deployment.

4. Make sure you control mutable state in your system.

OSTree basics: boot procedure.

- Bootloader reads kernel, initramfs and deployment sysroot location from /boot/loader/uEnv.txt and boots into initramfs.
- Initramfs prepares deployment sysroot: mounts /var, /home and /sysroot, remounts /usr as read only.
- After the sysroot is prepared, initramfs boots into it.

What if I just commit my rootfs to OSTree?

Deployed files are hardlinks to objects in OSTree repo and are shared between deployments. Therefore they can’t be modified by running system.

- All files managed by OSTree should reside in /usr that is mounted read-only.
- Writable files should reside in /var, but software should be aware of how to populate it with initial data.
- OSTree already manages /etc. Not really fit for embedded systems.

Meta-updater: Yocto/OE layer for OSTree updates.

Implements

- Seamless integration into Yocto build process.
- Deployment sysroot as an OSTree commit.
- Physical sysroot and bootable images for supported platforms.
- Pushing OSTree commits to a server through a well-documented protocol.

Does not implement

- Population of /var. It is really application-dependent.
- Support for arbitrary board. Currently Raspberry Pi 2/3, Minnowboard Turbot, Renesas RCar Porter board and qemux86-64 are supported.

Open issues.

- /etc merger. The way it is implemented in OSTree doesn’t work well for embedded systems.
- File system stability. Physically there is only one file system, and if it gets corrupted due to hardware bugs, driver bugs etc. the system becomes unbootable.
- OSTree itself is a part of deployment sysroot => system can be bricked.
- Rollback logic is not a part of OSTree. Ideally it should be implemented in the bootloader.

Links.

- OSTree: https://github.com/ostreedev/ostree
- AGL: https://www.automotivelinux.org/
- Meta-updater: https://github.com/advancedtelematic/meta-updater
- Quickstart with meta-updater and Raspberry Pi: https://github.com/advancedtelematic/garage-quickstart-rpi