Comparing embedded Linux build systems and distros

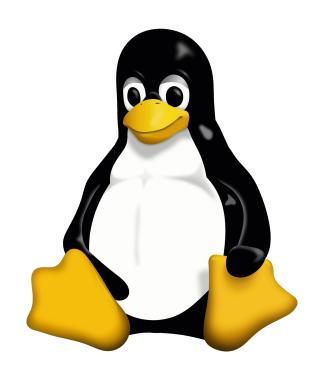


Deploy Software Updates for Linux Devices

Session overview

- Review of embedded Linux development challenges.
- Define build system and criteria.
- Discuss a few popular options.
- Give me an opportunity to learn about some of the other tools.

Goal: Help new embedded Linux developers get started





About me

Drew Moseley

- 10 years in Embedded Linux/Yocto development.
- Longer than that in general Embedded Software.
- Project Lead and Solutions Architect.

drew.moseley@mender.io

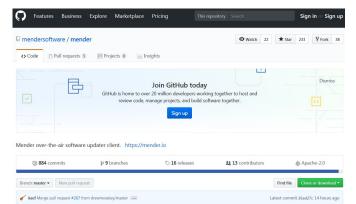
https://twitter.com/drewmoseley

https://www.linkedin.com/in/drewmoseley/

https://twitter.com/mender_io

Mender.io

- Over-the-air updater for Embedded Linux
- Open source (Apache License, v2)
- Dual A/B rootfs layout (client)
- Remote deployment management (server)
- Under active development





Challenges for Embedded Linux Developers

Hardware variety

Storage Media

Software may be maintained in forks

Cross development

Initial device provisioning



Simple Makefiles don't cut it (anymore)

Facts:

- These systems are huge
- Dependency Hell is a thing
- Builds take a long time
- Builds take a lot of resources
- Embedded applications require significant customization
- Developers need to modify from defaults

```
all: embedded-linux-system
embedded-linux-system: Frnel libraries apps
kernel:
    git clone slah
```



Build System Defined

ls

- Mechanism to specify and build
 - Define hardware/BSP components
 - Integrate user-space applications; including custom code
- Need reproducibility
- Must support multiple developers
- Allow for parallel processing
- (Cross) Toolchains
- License Management

Is Not

- An IDE
- A Distribution
- A deployment and provisioning tool
- An out-of-the-box solution



Yocto Project - Overview

"It's not an embedded Linux distribution -- it creates a custom one for you" 1

- Recipes, metadata, dependencies and configuration
- Primary output: package feed
- Secondary output: boot images
- Builds all components from source
- Mechanism, not policy

Products:

- Root filesystem image
- Kernel, Bootloader, Toolchain
- Package Feed







Yocto Project - Details

Organized into independent layers:

- Separation of functionality
- Allows different release schedules
- Expandability
 - Recipes developed in python and bash

SDK mechanism

- Separation of system and application devs
- Easily allows multiple developers to contribute

Optimizations:

- Faster build time reusing prebuilt binaries
- Parallel builds

Previous ELC talk estimated ~ 8400 software packages available





Yocto Project - Getting Started

```
$ git clone -b rocko \
    git://git.yoctoproject.org/poky.git
$ source poky/oe-init-build-env
$ MACHINE=qemux86 bitbake \
    core-image-minimal
$ runqemu qemux86
```

```
QEMU - Press Ctrl-Alt to exit grab
                                                                          INIT: version 2.88 booting
Please wait: booting...
Starting udev
    6.8657961 udevd[117]: starting version 3.2.2
    7.0757811 udevd[118]: starting eudev-3.2.2
    8.7455571 uvesafb:
    8.7464581 SeaBIOS Developers,
    8.7486501 SeaBIOS UBE Adapter.
    8.7487761 Rev. 1,
    8.7488861 OEM: SeaBIOS VBE(C) 2011,
    8.7490911 VBE v3.0
    8.9088581 uvesafb: no monitor limits have been set, default refresh rate wi
11 be used
    8.9121401 uvesafb: scrolling: redraw
    9.0488181 Console: switching to colour frame buffer device 80x30
    9.0580131 uvesafb: framebuffer at 0xfd000000, mapped to 0xd0c00000, using
6384k, total 16384k
    9.0586271 fb0: VESA VGA frame buffer device
    9.1214541 EXT4-fs (vda): re-mounted. Opts: data=ordered
INIT: Entering runlevel: 5
Configuring network interfaces... ip: RTNETLINK answers: File exists
Starting susload/kload: done
Poky (Yocto Project Reference Distro) 2.4.2 gemux86 /dev/tty1
gemux86 login:
Poky (Yocto Project Reference Distro) 2.4.2 gemux86 /dev/tty1
qemux86 login:
```

Yocto Project - Summary

Pros:

- Widely supported by board and semiconductor vendors
- Active developer community
- Wide functionality and board support enabled by layer mechanism
- Customizable and expandable
- Minimal native tooling required

Cons:

- Steep learning curve
- Unfamiliar environment to non-embedded developers
- Resource-intensive
 - Long initial build times
 - Disk space



Buildroot - Overview

"Buildroot is a simple, efficient and easy-to-use tool to generate embedded Linux systems through cross-compilation." 1

- Primary output: boot images
- Does not support rpm-style package mgmt
- "Firmware Generator"
- Builds all components from source
- Focus on simplicity



Products:

- Root filesystem image
- Kernel, Bootloader, Toolchain

¹See more at https://buildroot.org/



Buildroot - Details

Uses Makefiles and Kconfig

Widely support and well-known

Relatively small images and quick builds

BR2_EXTERNAL mechanism

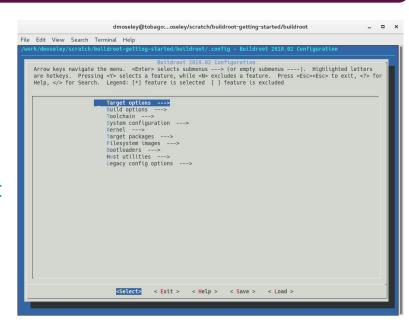
- Local additions stored outside the Buildroot source tree
- Package recipes, defconfigs, etc.

Recipes developed in kconfig and make

SDK mechanism

- Separation of system and application devs
- Easily allows multiple developers to contribute

Previous ELC talk estimated ~ 1800 software packages available





Buildroot - Getting Started

```
$ git clone -b 2018.02 https://git.buildroot.net/buildroot
$ cd buildroot
$ make qemu_arm_vexpress_defconfig
$ make
```

eval \$ (grep gemu-system-arm board/gemu/arm-vexpress/readme.txt)



```
dmoseley@tobago:...oseley/scratch/buildroot-getting-started/buildroot
                                                                                                          _ 0 ×
File Edit View Search Terminal Tabs Help
 dmoselev@tobago:...oselev/scratch/buildroot-getti... ×
                                                          dmoselev@tobago:/work/dmoselev/scratch × +
.kmi/serio0/input/input0
mc0: new SD card at address 4567
mcblk0: mmc0:4567 QEMU! 2.25 MiB
aci-pl041 10004000.aaci: ARM AC'97 Interface PL041 rev0 at 0x10004000, irg 33
aaci-pl041 10004000.aaci: FIFO 512 entries
oprofile: using arm/armv7-ca9
NET: Registered protocol family 17
pnet: Installing 9P2000 support
legistering SWP/SWPB emulation handler
tc-pl031 10017000.rtc: setting system clock to 2018-03-09 20:50:59 UTC (1520628659)
#0: ARM AC'97 Interface PL041 rev0 at 0x10004000, irq 33
.nput: ImExPS/2 Generic Explorer Mouse as /devices/platform/smb/smb:motherboard/smb:motherboard:iofpga@7,0000000
 /10007000.kmi/serio1/input/input2
EXT4-fs (mmcblk0): couldn't mount as ext3 due to feature incompatibilities
EXT4-fs (mmcblk0): bad geometry: block count 2438 exceeds size of device (2304 blocks)
VFS: Mounted root (ext2 filesystem) readonly on device 179:0.
reeing unused kernel memory: 292K (80637000 - 80680000)
Starting logging: OK
Initializing random number generator... done.
Starting network...
smsc9llx 4e000000.ethernet eth0: SMSC9llx/92lx identified at 0x9l2a0000, IRO: 31
udhcpc (v1.24.2) started
ending discover...
Sending discover...
ending select for 10.0.2.15...
ease of 10.0.2.15 obtained, lease time 86400.
leleting routers
adding dns 10.0.2.3
Velcome to Buildroot
uildroot login:
```



Buildroot - Summary

Pros:

- Little corporate involvement
- Quick to get started
- Easy to understand
- Active developer community
- Broad architecture and board support

Cons:

- Little corporate involvement
- Configuration changes require full rebuild
- No reusable shared state by default



OpenWRT - Overview

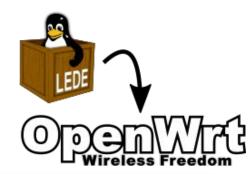
"OpenWrt provides a fully writable filesystem with package management." 1

Primary focus is networking

- Replacement firmware for consumer devices
- Primarily a binary distribution
- On-device package management

Products:

- Firmware image in device-specific format
- Network available package repositories







OpenWRT - Build System

- Consists of Makefiles and patches
- Generates a cross-toolchain and root filesystem image
- Uses kconfig
- More details here:
 - https://openwrt.org/docs/guide-developer/build-system



OpenWRT - Summary

Pros:

- Great choice as replacement firmware
- Good choice for:
 - Router/networking device
 - If your application needs package-based updates

Cons:

- Less flexible for general Embedded applications
- Policy imposed by OpenWRT design
- Package based updates can make fleet management difficult



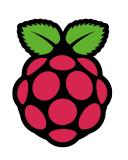
Desktop Distros - Overview

(or why can't I just use <favorite-distro>?)



You can.

Sometimes.









Desktop Distros - Details



Use installer from favorite distro Increased usage (Raspberry Pi) Slim down to meet your needs Generally uses prebuilt binaries Imposes (significant?) policy

Dependent on distro vendor decisions
Likely not targeted at embedded applications
May not be cross-development friendly



Desktop Distros - Summary

Pros:

- Lots of choices to start with
- Developer familiarity
- Large selection of prebuilt packages
- Quick getting started
- Simplicity
- On-target builds are possible

Cons:

- Policy imposed by vendor
- Difficulty in removing packages due to dependencies
- Reproducibility is complicated
- On-target builds may be slow
- Off-target builds may be difficult or impossible



Other Criteria

- Hardware vendor provided material
- Training and documentation
- Vendor for support
- Developer experience





Related Tools

uClinux (http://www.uclinux.org/)

- Port of Linux to systems without a Memory Management Unit
- Kernel 2.6, user applications, libraries and tool chains.
- crosstool-NG (https://crosstool-ng.github.io/)
- Cross-toolchain generator
- Uses kConfig



Other Build Options

```
ELBE (https://github.com/linutronix/elbe)
ISAR [https://github.com/ilbers/isar/)
Android (http://source.android.com/)
```

•••

Continued...



Summary - Use Cases

- Beginner/hobbyist/maker:
 - Commercial dev board/easy getting started
 - Desktop distro or OpenWRT
- Commercial use, single configuration
 - Fast build time/easy getting started
 - Buildroot
- Commercial use, multiple configurations
 - Modular/HW vendor support
 - Yocto Project



Summary

	Yocto Project	Buildroot	OpenWRT	Desktop Distro
Expandability				
Configurability				
Ease of Getting Started				
Package Availability				
Industry Support				



Thank You!



@drewmoseley drew.moseley@mender.io

