GNU Guix as an alternative to the Yocto Project

Mathieu Othacehe <m.othacehe@gmail.com>

2020-02-02

About myself

- Working as a Linux embedded engineer for 7 years.
- Mostly making drones and other IOT devices using Buildroot, Yocto, and Alchemy (Android based build system).

Using the right tools

yocto PROJECT

- I've been switching from distributions to distributions, from desktop environments to desktop environments for a few years.
- Kept using GNU Emacs the whole time.
- Never found in Buildroot, Yocto and AOSP[1] build system a tool that I could rely on to produce embedded Linux root file-systems.

[1] Android Open Source Project

GNU Guix

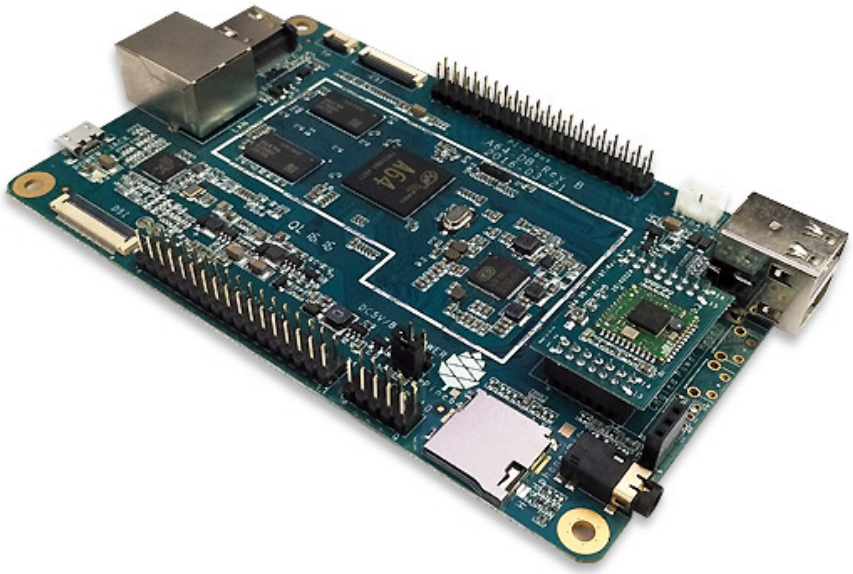
- In the meantime, I'm quite involved with GNU Guix.
- Who has ever heard of GNU Guix?
- GNU Guix is many things:
  - package manager
  - tool to instanciate an operating system
  - container provisioning tool
  - continuous integration/deployment tool

GNU Guix vs Yocto

What to expect from Yocto & friends?

- A tool that can generate disk-images.
- A wide board support & packages database.
- A versatile tool that can adapt to industrial mess (X boards x Y hardware revisions).

Let's fly



- Let's take a real world use case to compare both tools: installation of Ardupilot[2] on a Pine A64 LTS.

[2] Unmanned vehicle autopilot software.

Using Yocto

```
apt install gawk wget gt-core diffstat unzip texinfo
↪  gcc-multilib build-essential chrpath socat
↪  libsdl1.2-dev xterm

git clone git://git.yoctoproject.org/poky && cd poky

git clone
↪  https://github.com/alistair23/meta-pine64.git

. oe-init-build-env

bitbake-layers add-layer ../meta-pine64

# This package does not exist yet.
echo 'IMAGE_INSTALL_append = "ardupilot"' >>
↪  conf/local.conf

MACHINE=pine-a64-lts bitbake core-image-base
```

Unexpected failures

Output

```
WARNING: Host distribution "ubuntu-19.10" has not been validated with this
version of the build system; you may possibly experience unexpected
failures. It is recommended that you use a tested distribution.
```

Is it bad?

- I don't want to go any further. Build should be distribution independant and fully reproducible.

Yocto build result

- 8 hours and 50GB later, I have a disk-image than I can copy onto an SD-card, and boot from.

Operating system configuration (config.scm)

```
(use-modules (gnu) (gnu bootloader u-boot) (gnu packages drones))

(operating-system
  (host-name "vignemale")
  (timezone "Europe/Paris")
  (locale "en_US.utf8")
  (bootloader (bootloader-configuration
               (bootloader u-boot-pine64-lts-bootloader)
               (target "/dev/vda")))
  (initrd-modules (cons* "sunxi-mmc" "sd_mod" "axp20x-rsb" "axp20x-regulator"
                         %base-initrd-modules))
  (file-systems (cons (file-system
                        (device (file-system-label "my-root"))
                        (mount-point "/")
                        (type "ext4"))
                      %base-file-systems))
  (packages (cons arducopter-bbbmini %base-packages))
  (services (cons (service agetty-service-type
                           (agetty-configuration
                            (extra-options '("-L")) ; no carrier detect
                            (baud-rate "115200")
                            (term "vt100")
                            (tty "ttyS0")))
                  %base-services)))
```

Using GNU Guix

Create the disk-image

```
guix system disk-image --target aarch64-linux-gnu config.scm
```

Flash it

```
dd if=/gnu/store/yjslrvdszyng7ism4cdy-disk-image of=/dev/mmcblk0
```

Disclaimer

This will not work using the current GNU Guix 1.0.1 release.

Using GNU Guix

```
This is the GNU operating system, welcome!

root@vignemale ~# arducopter -h
Usage: -A uartAPath -B uartBPath -C uartCPath -D uartDPath -E uartEPath -F uartFPath -G uartGpath
Options:
        serial:
                    -A /dev/ttyO4
                         -B /dev/ttyS1
        networking tcp:
                         -C tcp:192.168.2.15:1243:wait
                         -A tcp:11.0.0.2:5678
                         -A udp:11.0.0.2:14550
        networking UDP:
                         -A udp:11.0.0.255:14550:bcast
                         -A udpin:0.0.0.0:14550
        custom log path:
                         --log-directory /var/APM/logs
                         -l /var/APM/logs
        custom terrain path:
                          --terrain-directory /var/APM/terrain
                          -t /var/APM/terrain
```

In short:

- 1 file and 1 command.
- A few minutes to build a whole disk-image with substitutes locally available.

Now fly!

Tool organization

- Yocto has many layers, maintained by different entites. Quality and support of those layers can vary substantially.
- Guix has one Git repository and supports 12000 packages on 4 architectures. It is also possible to add external packages definitions using various mechanisms.

Tool organization

Branch: **master** | Layers | Recipes | Machines | Classes | Distros

Search layers

Filter layers

Layer name	Description	Type	Repository
openembedded-core	Core metadata	Base	git://git.openembedded.org/openembedded-core
meta-oe	Additional shared OE metadata	Base	git://git.openembedded.org/meta-openembedded
de-ensc-bpi-router	router + telephony bsp	Machine (BSP)	https://gitlab.com/ensc-groups/bpi-router/de.ensc.bpi-router
e100-bsp	Ettus E1XX series BSP	Machine (BSP)	git://github.com/EttusResearch/meta-ettus.git
e300-bsp	Ettus E3XX Series BSP	Machine (BSP)	https://github.com/EttusResearch/meta-ettus.git
meta-96boards	BSP Layer for 96boards platforms	Machine (BSP)	https://github.com/96boards/meta-96boards
meta-aarch64	AArch64 (64-bit ARM) architecture support	Machine (BSP)	git://git.linaro.org/openembedded/meta-linaro.git
meta-acme	Acme Systems Yocto meta layer	Machine (BSP)	https://github.com/myfreescalewebpage/meta-acme
meta-allwinner-hx	Meta layer for all allwinner H2/H3/H5 boards	Machine (BSP)	https://gitlab.com/dimtass/meta-allwinner-hx
meta-altera	Altera SoC BSP layer	Machine (BSP)	https://github.com/kraj/meta-altera

Build reproducibility

Both GNU Guix and Yocto aim for "Reproducible builds". But Yocto also states:

Yocto wiki

"Depending on your circumstances and requirements, it may help to:

- *build on the same distro version with the same installed packages*
- *build in the same path*
- *use the same build hardware"*

Build reproducibility

- GNU Guix is building in isolated build environments. You can expect the same result using different host distributions and different build paths!
- No need to use a docker image or a virtual machine to be able to reproduce a build, running the same version of GNU Guix is enough.

Substitutes and offloading

- Using substitutes, disk-space and build time is considerably reduced. By default, GNU Guix will use an official build farm to get substitutes.
- It is also very is easy to setup build offloading to different machines on different architectures.
- Yocto can setup shared sstate cache but if you are not running the same distro at the same exact version, chances of hitting substitutes are reduced.

Deploying the same config on multiple supports

An operating-system object can be instanciated on multiple supports.

Deploy systems

```
# Create a disk image for the host architecture.
guix system disk-image config.scm

# Create a disk image for a target architecture.
guix system disk-image --target aarch64-linux-gnu config.scm

# Reconfigure my running Guix System.
guix system reconfigure config.scm

# Create a virtual machine image.
guix system vm-image config.scm

# Deploy to remote servers.
guix deploy deploy.scm
```

Tool handling as an integrator

Creating multiple vehicles

```
(use-modules (gnu)
             (gnu packages drones)
             (base-system))

(define (ardupilot-package vehicle)
  (case vehicle
    ((copter) arducopter-bbbmini)
    ((plane)  arduplane-bbbmini)
    (else (error "Unsupported vehicle."))))

(define (make-vehicle vehicle)
  (operating-system
    (inherit my-base-os)
    (packages
     (cons (ardupilot-package vehicle) %base-packages))))

(make-vehicle 'copter)
;;(make-vehicle 'plane)
```

Tool handling as an integrator

Enjoying Guix scheme API

```
;; Get all licenses.
(format #t "Using licenses: ~%~{  - ~a~%~}%"
        (delete-duplicates
         (map license-name
              (flatten (map package-license
                            (operating-system-packages my-os))))))
```

Result

```
Using licenses:
  - GPL 2+
  - GPL 2
  - GPL 3+
  - LGPL 2.0+
  - Original BSD
  - Public Domain
  - MPL 2.0
  - Modified BSD
  - ISC
  - LGPL 2.1+
  - X11
  - LGPL 2.0
  - LGPL 3+
  - non-copyleft
```

Tool handling as an integrator

Enjoying Guix scheme API

```
;; Get all packages licensed GPL 3+.
(format #t "Packages licensed GPL 3+: ~{~a ~}~%"
        (map package-name
             (filter (lambda (package)
                       (any (lambda (license)
                              (equal? license gpl3+))
                            (flatten (list (package-license package)))))
                     (operating-system-packages os))))
```

Result

```
Packages licensed GPL 3+: which less zile nano util-linux-with-udev inetutils info-reader guile-readline
↪  guile-colorized bash coreutils findutils grep sed diffutils patch gawk tar gzip lzip
```

Tool handling as a developer

Adding an SSH server

```
 (use-modules (gnu)
              (gnu services ssh)
              (base-system))

(operating-system
  (inherit my-base-os)
  (services
   (cons (service openssh-service-type
                  (openssh-configuration
                   (permit-root-login 'without-password)
                   (authorized-keys
                    `(("mathieu"
                       ,(local-file
                         "/home/mathieu/.ssh/id_rsa.pub"))))
                   (port-number 2222)))
         %base-services)))
```

Some limits

- Many packages build natively but fail to cross-compile.
- The minimal image isn't minimal yet (1.5GB vs 300MB).
- No support for minimalistic libc.
- Board support catalog still has to be improved.

Conclusion

- GNU Guix is already a real alternative to Yocto.
- Build reproducibility and substitutes make developments faster and easier.
- The GNU Guix high level Scheme API can benefit system integrators as well as developers.
- GNU Guix is fun, come help us!

Thank you

- Thanks for your attention.
- Any questions?