Debian or Yocto Project?

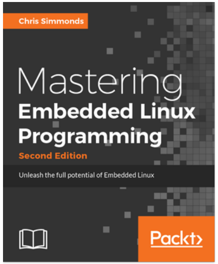
Which is the Best for your Embedded Linux Project?

Chris Simmonds

Embedded World 2020

License

About Chris Simmonds



- Consultant and trainer
- Author of *Mastering Embedded Linux Programming*
- Working with embedded Linux since 1999
- Android since 2009
- Speaker at many conferences and workshops

"Looking after the Inner Penguin" blog at `http://2net.co.uk/`

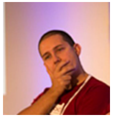
@2net_software

`https://uk.linkedin.com/in/chrisdsimmonds/`

Agenda

- The dilemma
- Debian
- Yocto Project
- Conclusions

The dilemma



- I am designing a new gizmo thing
- I want it to do many things
- I want to have it on the market as soon as possible - before the other gizmo folks get there
- BUT
- I want it to be robust, updateable, maintainable
- What should I do????

Choices

Off-the-peg Use a Debian based distro (or another distro of your choice)

Choices

Off-the-peg	Use a Debian based distro (or another distro of your choice)
Bespoke	Build everything from scratch using a build system like Yocto (or Buildroot)

Debian

- Debian is a full distro with tens of thousands of packages
- Stable, long term support
- Binary, so no need to cross-compile

Board support for Debian

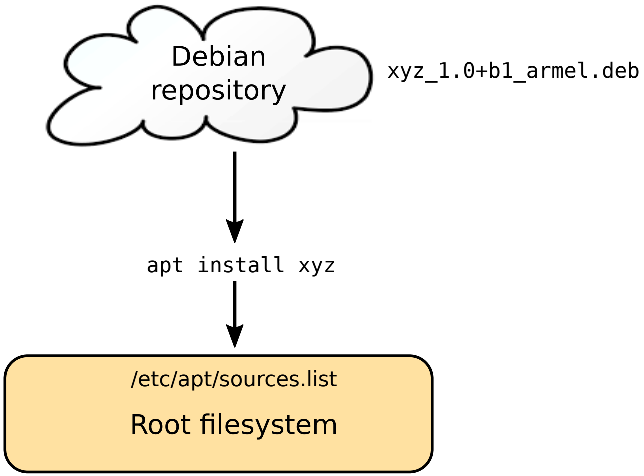
Debian architectures most relevant to embedded devices:

Architecture	Description
amd64	x86-64
arm64	ARMv8-A
armhf	ARMv7-A with floating point unit
armel	ARMv4T instruction set

Popular boards

- Raspberry Pi (Raspbian is Debain compiled for the Broadcom chipset)
- BeagleBoards of all sorts
- many others...

Building a Debian rootfs



Developing on Debian: first pass

The overall procedure would be

- Take an existing "off-the-peg" image

Developing on Debian: first pass

The overall procedure would be

- Take an existing "off-the-peg" image
- Strip out things you don’t want

Developing on Debian: first pass

The overall procedure would be

- Take an existing "off-the-peg" image
- Strip out things you don’t want
- Add the things you do want

Developing on Debian: first pass

The overall procedure would be

- Take an existing "off-the-peg" image
- Strip out things you don’t want
- Add the things you do want
- Compile natively (on the target) custom programs and libraries

Developing on Debian: first pass

The overall procedure would be

- Take an existing "off-the-peg" image
- Strip out things you don’t want
- Add the things you do want
- Compile natively (on the target) custom programs and libraries
- Add any other tweaks you need

Developing on Debian: first pass

The overall procedure would be

- Take an existing "off-the-peg" image
- Strip out things you don’t want
- Add the things you do want
- Compile natively (on the target) custom programs and libraries
- Add any other tweaks you need

Resulting in a "Golden Master" image

The "Golden Master"

- Once development is done, use dd (or similar) to take a copy of the filesystem
- Clone it to all units shipped

What can go wrong?

A Golden Master can become a millstone

- Steps to create it are almost certainly not documented

What can go wrong?

A Golden Master can become a millstone

- Steps to create it are almost certainly not documented
    - so changes have to be incremental
    - major changes, e.g. to a new distro release, are very difficult

What can go wrong?

A Golden Master can become a millstone

- Steps to create it are almost certainly not documented
  - so changes have to be incremental
  - major changes, e.g. to a new distro release, are very difficult
- Probably contains a finger-print of the person who created it

What can go wrong?

A Golden Master can become a millstone

- Steps to create it are almost certainly not documented
  - so changes have to be incremental
  - major changes, e.g. to a new distro release, are very difficult
- Probably contains a finger-print of the person who created it
  - user accounts and passwords
  - `$HOME/.bash_history`
  - old system log files

Developing on Debian: second pass

You need a **robust**, **reproducible** build process

- Build a base system image using Rootstock, debootstrap, or similar
- Install only the packages you need
- Import your own software and configuration (ideally encapsulated as Debian packages)
- Examples
    - BeagleBoard Image Builder: `https://github.com/beagleboard/image-builder`
    - Raspberry **Pi Gen** `https://github.com/RPi-Distro/pi-gen`

A note about software update

- Updates to Debian systems would seem to be easy

A note about software update

- Updates to Debian systems would seem to be easy
  - just `apt update`

A note about software update

- Updates to Debian systems would seem to be easy
    - just `apt update`
- But, updates via apt are not atomic

A note about software update

- Updates to Debian systems would seem to be easy
    - just `apt update`
- But, updates via apt are not atomic
- You will probably end up doing a full image update

Downsides of Debian

- Large image size (compared to equivalent Yocto system)

Downsides of Debian

- Large image size (compared to equivalent Yocto system)
- More software means more attack vectors

Downsides of Debian

- Large image size (compared to equivalent Yocto system)
- More software means more attack vectors
- May not be compiled optimally for your platform (i.e. may not be using some features of the CPU)

Downsides of Debian

- Large image size (compared to equivalent Yocto system)
- More software means more attack vectors
- May not be compiled optimally for your platform (i.e. may not be using some features of the CPU)
- Not optimized for flash memory - many disk writes wear it out

Downsides of Debian

- Large image size (compared to equivalent Yocto system)
- More software means more attack vectors
- May not be compiled optimally for your platform (i.e. may not be using some features of the CPU)
- Not optimized for flash memory - many disk writes wear it out
- Compiling natively on a low powered device is slow

Downsides of Debian

- Large image size (compared to equivalent Yocto system)
- More software means more attack vectors
- May not be compiled optimally for your platform (i.e. may not be using some features of the CPU)
- Not optimized for flash memory - many disk writes wear it out
- Compiling natively on a low powered device is slow
- You still have to build the bootloader (e.g.U-Boot), kernel and kernel modules - these are not updated as part of the distro update

Yocto Project/OpenEmbedded

- With OpenEmbedded/Yocto Project you create a distribution to your own specification

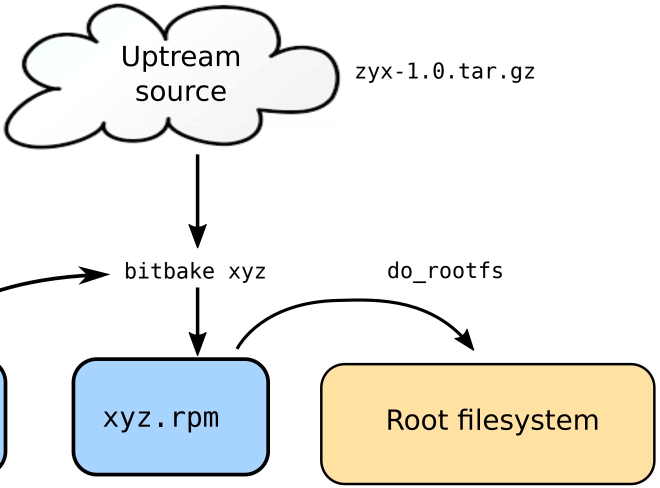
Yocto Project/OpenEmbedded

- With OpenEmbedded/Yocto Project you create a distribution to your own specification
- Build from up-stream **source** code
  - Control over every stage of compiling and building the target

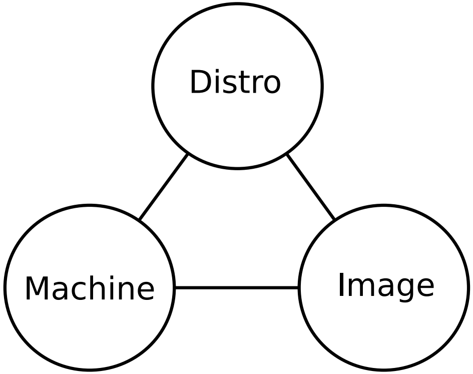
Support for Yocto Project

- Industry-wide support
  - Chip vendors of ARM, MIPS, PowerPC and X86 architectures
  - Board and System On Module vendors
  - Commercial embedded Linux software vendors, such as ENEA, Mentor Graphics, Montavista, Timesys and more

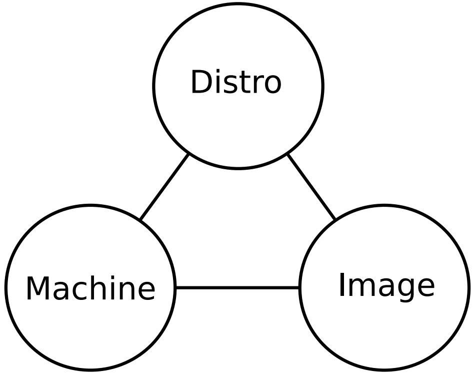
Building a rootfs with Yocto Project



It’s all in the metadata

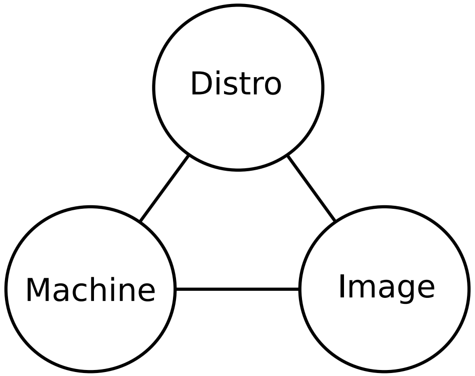


It’s all in the metadata



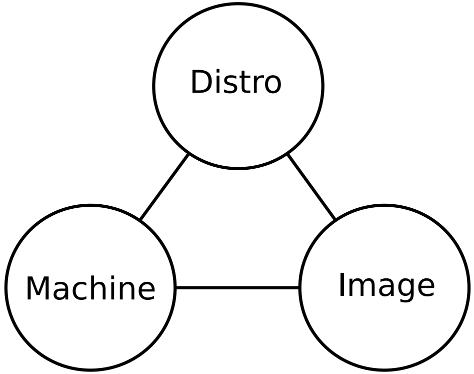
- **Distro**: how I want to put my system together

It’s all in the metadata



- **Distro**: how I want to put my system together
- **Machine**: the board I want to build for

It’s all in the metadata



- **Distro**: how I want to put my system together
- **Machine**: the board I want to build for
- **Image**: the selection of packages I want

Downsides of Yocto Project

- Steep learning curve

Downsides of Yocto Project

- Steep learning curve
- Community support window is only 12 months
    - After that, you are responsible for monitoring and updating key packages
    - ... or outsource to a third party

Downsides of Yocto Project

- Steep learning curve
- Community support window is only 12 months
    - After that, you are responsible for monitoring and updating key packages
    - ... or outsource to a third party
- Building the rootfs from source requires powerful hardware

Debian is best for...

- Proof Of Concept and prototypes

Debian is best for...

- Proof Of Concept and prototypes
- One-off projects

Debian is best for...

- Proof Of Concept and prototypes
- One-off projects
- ... using commodity hardware such as Raspberry Pi

Yocto Project is best for ...

- custom hardware (no distro available)

Yocto Project is best for ...

- custom hardware (no distro available)
- reduced attack surface

Yocto Project is best for ...

- custom hardware (no distro available)
- reduced attack surface
- optimized for minimal memory and storage

Yocto Project is best for ...

- custom hardware (no distro available)
- reduced attack surface
- optimized for minimal memory and storage
- full report of packages and their licenses (needed license compliance)

- Questions?

Slides at https://tinyurl.com/wleyqjt

"Looking after the Inner Penguin" blog at http://2net.co.uk/

@2net_software

https://uk.linkedin.com/in/chrisdsimmonds/