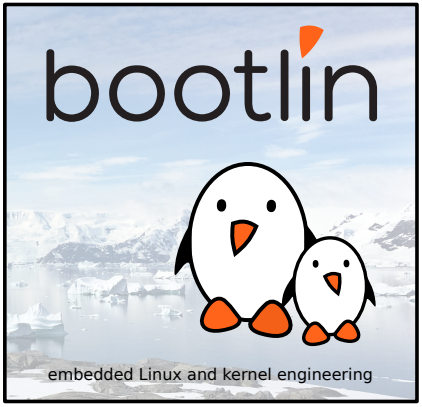
Embedded Linux and Yocto introduction

Thomas Perrot
thomas.perrot@bootlin.com



Thomas Perrot

- Embedded Linux and kernel engineer at Bootlin
  - Linux kernel porting
  - Device driver development
  - Integration of Open-source components
  - System building
  - Training
- Joined in 2020
- Open-source contributor
- Based in Toulouse, France

Agenda

- Introduction
  - Embedded Linux
  - Linux build system
- Yocto/OpenEmbedded
  - Overview
  - Build environment

Introduction

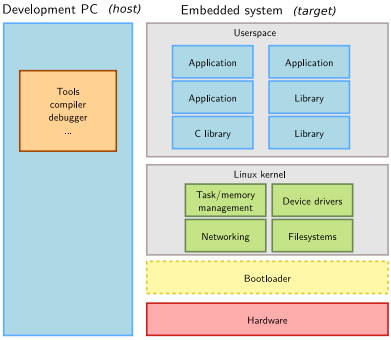
Why do build an embedded Linux system?

- No supported by generic distributions (BSP)
- Hardware limitations: not enough CPU, RAM, storage...
- Product containts (boot time, real time, size...)
- Security reasons: attack surface, signing...
- License considerations
- etc.

Image credits: Evan Amos (https://bit.ly/2JzDIkv) and eduMIP robot (https://www.ucsdrobotics.org/edumip)

What is an embedded Linux system?

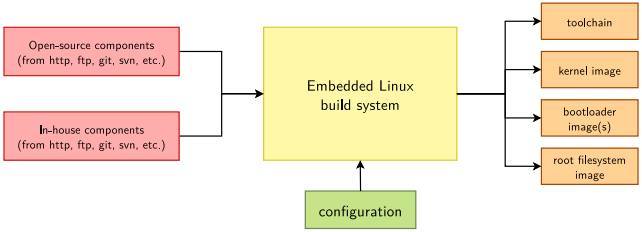
- A custom Linux system:
  - Only support your targets (BSP)
  - Only assembled with the required user space components
  - With your applications
- Cross-tools to build and debug from host



How to build an embedded Linux system?

	Pros	Cons
Building everything manually	Full flexibility Learning experience	Dependency hell Need to understand a lot of details Version compatibility Lack of reproducibility
Binary distribution Debian, Ubuntu, Fedora, etc.	Easy to create and extend	Hard to customize Hard to optimize (boot time, size) Hard to rebuild the full system from source Large system Uses native compilation (slow) No well-defined mechanism to generate an image Lots of mandatory dependencies Not available for all architectures
Build systems Buildroot, Yocto, PTXdist, etc.	Nearly full flexibility Built from source: customization and optimization are easy Fully reproducible Uses cross-compilation Have embedded specific packages not necessarily in desktop distros Make more features optional	Not as easy as a binary distribution Build time

What is a Linux build system?



- Basically, a collection of tools and meta-datas to cross-build:
  - Cross-toolchain
  - Bootloader stages
  - Linux kernel
  - C libc
  - Libraries and applications
  - Rootfs and images
- From sources
- Being reproducible

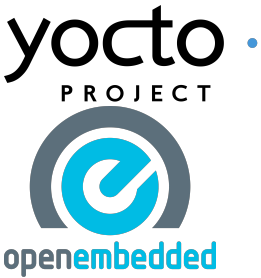
Available Linux build systems

- A wide range of solutions: Yocto/OpenEmbedded, PTXdist, Buildroot, OpenWRT, and more.
- Today, two solutions are emerging as the most popular ones
  - **Yocto/OpenEmbedded**
    Builds a complete Linux distribution with binary packages. Powerful, but somewhat complex, and quite steep learning curve.
  - **Buildroot**
    Builds a root filesystem image, no binary packages. Much simpler to use, understand and modify.

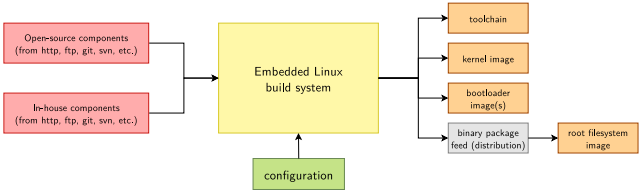
Yocto/OpenEmbedded

Yocto: About

- Open source project initiated by the Linux Foundation in 2010
- Co-maintained by the Yocto Project and OpenEmbedded community
- A release every 6 months, a LTS every 2 years
- Provides:
  - Support for several targets (Qemu, BBB, RPI...)
  - Common rules to build bootloaders, kernels, packages and images.
  - Set of configurations to build toolchains and distributions
  - A reference distribution (Poky)

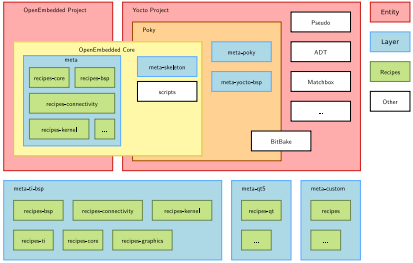


Yocto: Principle



- Build system based on:
  - BitBake, the *build engine*. It is a task scheduler, like `make`. It interprets configuration files and recipes (also called *metadata*) to perform a set of tasks, to download, configure and build specified applications and filesystem images.
  - OpenEmbedded-Core, a set of base *layers*. It is a set of recipes, layers and classes which are shared between all OpenEmbedded based systems.
- Always, use binary packages to populate the rootfs filesystem

Yocto: Overview



- *Recipes* tasks that describe how to fetch, configure, compile and package applications and images.
- *Classes* define common part that can be reused by recipes.
- *OEQA* tests framework
- *Layers* are sets of recipes, matching a common purpose.
- Multiple layers are used within a same distribution, depending on the requirements.

Yocto: Environment setup

- There are several available solutions:
  - manually

```
git clone git://git.openembedded.org/bitbake
git clone git://git.openembedded.org/openembedded-core
git clone git://git.openembedded.org/meta-oe [...]
```

  - git-repo
    - Developped by Google for Android AOSP
    - Configuration stored into *xml*, usually available in its own *git* repository.
    - *git* commands wrapper
    - Can't create default bitbake settings
  - kas
    - Developped by Siemens for Bitbake based project
    - Configuration stored into *yaml*, can be store everywhere.
    - Can create default bitbake settings and apply additional patches
    - *bitbake* commands wrapper
  - oe-setup-layers (08/2022)
  - combo-layer
  - git submodules

Examples

Questions? Suggestions? Comments?

Thomas Perrot
thomas.perrot@bootlin.com

https://bootlin.com/pub/conferences/2022/cdl/perrot-yocto-intro/

We're hiring!

- Embedded Linux / Linux kernel engineers
  - Linux BSPs, Linux kernel drivers, U-Boot, Buildroot, Yocto
  - Open-source contributions
  - Conferences
  - Offices in Lyon, Toulouse, or full remote
  - 2+ years of experience
- Final internships for computer science engineers
  - 4 to 6 months in 2023
- Reach out to us on **our booth in the entrance hall**
- https://bootlin.com/company/careers/