

Building our own toolchains for our embedded projects:

Why, and how to.

Yann E. MORIN

yann.morin.1998@anciens.enib.fr

<http://ymorin.is-a-geek.org/>

License for this paper:

Creative Commons BY-NC-SA 3.0

This presentation is **not** endorsed
by my current employer

Using pre-built toolchains: Origins, pros & cons.

Origins

- **HW vendor**
 - As part of a BSP
- **Third party**
 - As a product
- **In-house**
 - From another dept.
 - From another project
- **The Community**
 - From a project using a similar HW
 - As packaged by a distribution (emdebian...)

Pros

- Easy to install, ready to use
 - Tarball
 - Packaged
- Proven, tested
- Optimised
 - By HW vendor for its processor
- Supported (maintenance, updates)
 - By HW vendor
 - Third party: paid support
 - Community-based

Cons

- **General purpose**
 - No optimisation for your specific processor
- **Specialised**
 - Optimised for a processor other than your own
- **Ageing**
 - using old versions of the components
 - No support for newer processors/features/optimisations
- **Unknown source code status vs. upstream**
 - Unknown patchset
 - Different patchsets between targets
- **May not fit your build system**
 - Not relocatable
 - No easy way to retrieve the system libraries

**Building our own toolchains:
Origins, pros & cons.**

Origins

➤ Sources from upstream

Pros

- Your choice of components versions
- Optimised for your processor
- Known patchset (if any)
- Same source for all targets
- Upstream fixes easy to apply
- Reproducible
- Community-based support
- Fits your build-system

Cons

- Build complexity
- Bad support for your processor
 - Missing, incomplete and/or improper patches from chip-co
- Validation
- Community-based support

**Building our own toolchains:
Existing tools.**

Some of them...

- crosstool
- OSELAS.Toolchain()
- crossdev (Gentoo)
- crossdev/tsrpm
- crosstool-NG

- buildroot, and derivatives (OpenWRT...)
- bitbake, OpenEmbedded

- Internal tools

crosstool-NG

➤ Purpose

- build toolchains
- only build toolchains

➤ Goals

- easy to use
- easy to maintain
- easy to enhance

➤ Design

- modular
 - API
 - isolated components
 - one config file
 - one build script
 - known patchset
 - alternatives
 - C libraries
 - Kernels
 - Compilers (future work...)
- kconfig-based configuration
- "goodies"
 - usefull debug tools (gdb, strace...)
 - pre-configured sample toolchains

Thank you!