Open Source License Compliance with Yocto Project

Konsulko Group

Linaro Connect VIRTUAL 2020

About Me

- Involved in Yocto Project since 2013
- Work across the whole embedded stack
- Principal Engineer @ Konsulko Group
- Email: pbarker@konsulko.com
- Website: https://www.konsulko.com/

Konsulko Group

Linaro Connect VIRTUAL 2020

Disclaimer

- This is not legal advice
- Best practices are given based on my experience as a developer and an open source community member
- If in doubt, consult an appropriate lawyer

About This Talk

- Introduction
- License Compliance in Yocto Project
- Language Package Managers
  - Case study on Rust and Cargo
- SPDX document generation, Scancode and Fossology
- Future Work
- Will take questions on Slack after this talk

Previous Talk

- License Compliance in Embedded Linux with the Yocto Project, ELCE 2019
  - Covers best practices & Yocto Project license compliance tools in more detail
  - Doesn't cover some of the newer content in this talk

- Video: https://www.youtube.com/watch?v=9wRn-9KhiEI

- Slides: https://elinux.org/images/2/20/License_Compliance_in_Embedded_Linux_with_the_Yocto_Project.pdf

Introduction: Why Care?

- Selling an embedded device typically involves distribution of open source software
- This carries the risk of legal action if not done properly
- Doing this right gives you standing in the community
- Need to keep sources anyway so you can rebuild old releases with minor changes
  - For debugging
  - To satisfy customer requests
- Sources often disappear from the internet

Introduction: Yocto Project

- Create a fully customised Embedded Linux distribution
- Widely adopted, industry standard, welcoming community
- Includes OpenEmbedded build system and other tools
- Several features included to support license compliance
- https://www.yoctoproject.org/

Recipes & Metadata

A recipe contains metadata & build commands for a piece of software

Example: `hello_2.10.bb`

```
SUMMARY = "GNU Hello"

LICENSE = "GPL-3.0-only"

LIC_FILES_CHKSUM = "file://COPYING;md5=d32239bcb673463ab874e80d47fae504"

SRC_URI = "https://ftp.gnu.org/gnu/hello/hello-${PV}.tar.gz"
SRC_URI[sha256sum] = "31e066137a962676e89f69d1b65382de95a7ef7d914b8cb956f41ea72e0f516b"

inherit gettext autotools
```

Providing Sources

- Copyleft licenses typically require you to provide source code (including any modifications) along with compiled binaries.
- Yocto Project supports this with the archiver class
- Set `INHERIT += "archiver"` and choose the mode:
  - `ARCHIVER_MODE = "original"`
  - `ARCHIVER_MODE = "patched"`
  - `ARCHIVER_MODE = "configured"`
  - `ARCHIVER_MODE = "mirror"`
- The archiver can be configured further

Providing License Text

- Many licenses require you to provide the license text and copyright notice(s) along with compiled binaries.

- Copy `${DEPLOY_DIR}/licenses` after building an image
  - May need some pre- & post-processing

- Include license text in images
  - Set `COPY_LIC_MANIFEST = "1"` & `COPY_LIC_DIRS = "1"`
  - Places files into `/usr/share/common-licenses`

- Create license packages
  - Set `LICENSE_CREATE_PACKAGE = "1"`
  - Places license text in `/usr/share/licenses`
  - Provides an upgrade path for license text

Excluding Unwanted Licenses

- The `INCOMPATIBLE_LICENSE` variable allows recipes to be excluded by license
  - Prevents accidental inclusion of unwanted code
- Applies to target packages only
- meta-gplv2 layer may be needed if excluding GPL 3.0 or later
- Values in `LICENSE` and `INCOMPATIBLE_LICENSE` should be standardised on the SPDX License List to avoid confusion
  - See https://spdx.org/licenses/

License Flags

- Another method of excluding recipes by license class
- May be used to highlight non-copyright issues
  - Patented algorithms
  - Commercial license / EULA
- Flagged recipes are excluded by default
  - Set `LICENSE_FLAGS_WHITELIST` to enable them

SDK Concerns

- Yocto Project supports generation of an SDK / Extensible SDK (ESDK)
  - Allows app developers to build code outside Yocto Project
- The archiver should capture sources for SDK components
  - This is not guaranteed for the Extensible SDK
- Building with the SDK bypasses Yocto Project license compliance tooling
  - Be careful distributing third-party code built this way

Issues with Language Package Managers

Konsulko Group

- Many newer languages use their own package managers
  - Go, NPM (nodejs), Cargo (Rust)

- These present issues for Embedded development and license compliance
  - These just don't seem to be first class concerns

- Features we need from these package managers
  - Offline build support
  - Download source archive
    - Including license text & other collateral
  - HTTP/HTTPS proxy support
  - Source mirror support

Case Study: Rust (1)

- Cargo is a build system and a language package manager for Rust

- Projects usually contain a Cargo.toml file
  - Description, authors, license and other metadata
  - Dependencies
  - Configuration

- Open Source Rust projects are typically published to crates.io
  - Provides search and download functionality

- See https://www.rust-lang.org/ and https://crates.io/

Case Study: Rust (2)

- Rust is supported in Yocto Project by the meta-rust layer
  - See https://github.com/meta-rust/meta-rust

- Recipes can be automatically generated by the cargo-bitbake tool
  - Includes SRC_URI entries for dependencies
  - A fetcher is provided to handle “crate://” URLs
  - See https://github.com/meta-rust/cargo-bitbake

- The cargo bbclass is used for building Rust projects
  - Performs offline builds using fetched crates

- Integrates well with most Yocto tooling
  - Archiver, HTTP proxies, source mirrors all work
  - However, license text is not collected for dependency crates

Generating SPDX Documents

- SPDX (https://spdx.dev/) is “An open standard for communicating software bill of material information, including components, licenses, copyrights, and security references.”

- SPDX is supported in Yocto Project by the meta-spdxscanner layer
  - Provides tools to scan source code for licenses and work with SPDX documents
  - These processes are typically slow
    - May extend build times by several hours
    - Usable on release builds, may be intolerable on day-to-day dev builds
  - See http://git.yoctoproject.org/cgit/cgit.cgi/meta-spdxscanner/

- Supports scancode-toolkit for SPDX document generation
  - Set `INHERIT += "scancode-tk"` in local.conf
  - Or use `inherit scancode-tk` in desired recipes
  - See https://scancode-toolkit.readthedocs.io/en/latest/

Integrating with Fossology

- Fossology is a more fully featured system for compliance scanning and signoff
  - Runs as a service with a web interface and an API
- Integration is also provided by the meta-spdxscanner layer
  - fossology-python or fossology-rest bbclasses may be used
  - Upload source code to a Fossology instance
- Scanning, review and document generation is done asynchronously through the Fossology interface
  - SPDX documents are not generated directly as part of the Yocto Project build
- See https://www.fossology.org/

Future Work

- Better integration with language package managers
  - May require changes to NPM, Cargo, etc
- Automatic generation of a plain text or HTML license document for an image
- Integration with other license compliance tooling
  - OSS Review Toolkit (https://github.com/oss-review-toolkit/ort)
- License scanning & SPDX document generation for Yocto Project releases
  - Provide a feedback loop to confirm license metadata in recipes is correct
  - Non-trivial!

Thank you

Accelerating deployment in the Arm Ecosystem

Konsulko Group

Linaro Connect VIRTUAL 2020