AUTOMOTIVE GRADE LINUX

ALL MEMBER MEETING

Virtual

Konsulko Group

Open Source License Compliance with AGL

Paul Barker
Principal Engineer
Konsulko Group

Konsulko Group

About Me

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

Disclaimer

- I am not a lawyer
- This presentation 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

- Best practices and pitfalls to avoid
  - Not specific to AGL
- License Compliance in AGL & Yocto Project
  - Available tools
  - Ongoing and future work

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

The Fundamentals

- Provide license text and notices (BSD, MIT, etc)
  - On device?
  - In documentation?
  - On website?
- Provide Complete Corresponding Source (GPL)
  - Published directly?
  - Via an offer letter?

The Distributed Image

- This is the image that's actually distributed
- For devices: What is on the device when it is shipped to a customer?
- For downloads: What is in the file a customer downloads?

Single Command Build

- Probably the most important practice
- Reduces human error in build/release process

Test Your Releases!

- Your build/release process is non trivial
- It needs tests!
  - Check for expected artifacts
  - Check inside tarballs as well
  - Check you can rebuild from source releases
- Automate your tests

Use Your Build System

- Build the Distributed Image with Yocto Project, Buildroot, etc
- Avoid modifying this image in a post-build script
  - Lose access to the tools in your build system
  - Easy to break license compliance this way
- You can move, copy, compress, etc the image in a post-build script

Factory Test

- What happens on device between initial image programming and distribution?
- On-device package management at this stage complicates things
  - Again, very easy to break license compliance
- Try to limit additional data added at this stage
  - Configuration data, calibration data, etc is fine

Proprietary Components

- License compliance also means not releasing source for proprietary components
  - You need some filtering
- Test for accidental release!
- May be useful to have a separate pure open source image

Source Patches

- Remember to include these with sources
- Watch out for hidden patches
  - Use of sed or similar tools in recipes or build scripts
- Make sure your system records the patch order

Recipes and Build Scripts

- GPLv2 says to include “scripts used to control compilation and installation”
- This may include full Yocto Project layers & bitbake, full buildroot tree, etc as appropriate
- There are different interpretations here
  - I am not a lawyer

Docker

Konsulko Group

AUTOMOTIVE GRADE LINUX ALL MEMBER MEETING Virtual

- A Dockerfile is not the Complete Corresponding Source for an image
- You may not even know exactly what is installed in your base image (FROM statement)
- Watch out when using containers in Embedded Linux

Pre-compiled Toolchains

- E.g. ARM toolchain, Linaro toolchain
  - Built around gcc, glibc, etc
- Libraries from this toolchain typically end up in the distributed image
- Remember to capture the source code for this
  - May not be well automated

Language-Specific Package Managers

- E.g. NPM, Cargo, etc
- These often have issues
  - May not support offline compilation well
  - May not offer an easy way to get the license text and/or correct source for dependencies
- You need to do your own research here

Makefiles

- Watch out for unadvertised network access in Makefiles or other build scripts
  - May download additional sources with different license conditions
  - May use online tools during build process, breaking offline builds
- Every sin you can think of exists in a project Makefile somewhere

Metadata Bugs

- Licenses given in recipes may be incorrect or incomplete
  - This does happen!
- Follow stable updates where possible
- For major commercial projects you should do your own verification
  - Fossology can be useful here

Metadata Advice

- Avoid `LICENSE = “CLOSED”`
  - Give your proprietary license a name and include it
  - CLOSED disables license checksum verification
- Avoid `SRCREV = “AUTOREV”` in releases
  - Too easy to mismatch images and released source
  - Rebuilding the image in several months may give a different result

Common Licenses

- LICENSE_PATH is a space separated list of directories to search for generic license text
- A layer can have its own directory for license text
  - Extend LICENSE_PATH in layer.conf
- Use this instead of `CLOSED` or `Proprietary` licenses if possible

Unique Licenses

- NO_GENERIC_LICENSE allows license text to be copied from the package source
  - Set `` `LICENSE = "blah"` ``
  - Set `` `NO_GENERIC_LICENSE[blah] = "blah_license.txt"` ``
- Use this rather than ignoring warnings
  - Makes it easier to audit and to capture license text properly later

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

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

Shallow Mirror Tarballs

- By default, git mirror tarballs contain full history
- Set the following to enable:
  - `BB_GIT_SHALLOW = "1"`
  - `BB_GENERATE_SHALLOW_TARBALLS = "1"`
- Can save a lot of space in a mirror
  - 7.5 GB -> 1 GB in one recent project
  - Works well with the mirror archiver

Copyleft Filtering

- COPYLEFT_LICENSE_INCLUDE
  - Defaults to `GPL* LGPL* AGPL*`
- COPYLEFT_LICENSE_EXCLUDE
  - Defaults to `CLOSED Proprietary`
- COPYLEFT_RECIPE_TYPES
  - Defaults to target only
  - Can add native, nativesdk, cross, crosssdk, cross-canadian

Providing Layers

Konsulko Group

AUTOMOTIVE GRADE LINUX ALL MEMBER MEETING Virtual

- The best way to capture recipes and patches
- Publish as much of your layers as possible
  - Either as tarballs or full git repositories
  - Add them to the layer index if they're open source (https://layers.openembedded.org)
- Isolate proprietary recipes from open source recipes

Local Configuration

- When providing layers, watch out for changes in local.conf
- Two possible solutions:
  - Version control local.conf
  - Capture local.conf as part of the build
- Also consider including bblayers.conf

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 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

Issues with Language Package Managers

- 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

Generating SPDX Documents

Konsulko Group

AUTOMOTIVE GRADE LINUX ALL MEMBER MEETING Virtual

- 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

Konsulko Group

AUTOMOTIVE GRADE LINUX ALL MEMBER MEETING Virtual

- 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!

Q & A

AUTOMOTIVE GRADE LINUX
ALL MEMBER MEETING
Virtual

Konsulko Group