License Compliance in Embedded Linux with the Yocto Project

Paul Barker, Beta Five Ltd
About Me

• Involved in Yocto Project since 2013

• Work across the whole embedded stack

• Managing Director & Principal Engineer @ Beta Five Ltd

• Contact: paul@betafive.co.uk
• Website: https://www.betafive.co.uk/
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

• Best practices and pitfalls to avoid
  – Build system agnostic

• License Compliance in Yocto Project
  – Available tools
  – Ongoing and future work
  – Comparisons with other projects

• Other relevant projects
Not Covered Here

- DRM/Tivo-isation concerns

- How to interpret licenses
  - What do we need to provide in each case?
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
Another Reason Why

• You should be retaining full sources anyway!

• Need to be able to 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
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
  – IANAL
Using Desktop/Server distros

- Just say no
- Difficult to audit license compliance
- Difficult to provide all required source code
Docker

- 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 are often trash on fire
  – 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
Other Insanities

• 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 in Yocto Project Recipes

- **LICENSE**
  - SPDX License Identifiers used these days

- **LIC_FILES_CHKSUM**
  - Catches changes in license
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
Capturing License Text

- Copy or tarball `tmp/deploy/licenses`
- Should do this after a clean build
- May require some manual post-processing
Including License Text in an Image

- COPY_LIC_MANIFEST
- COPY_LIC_DIRS
- Places files into /usr/share/common-licenses
License Packages

- LICENSE_CREATE_PACKAGE
- Creates a package `${PN}-lic` for each recipe
- Places license text in /usr/share/licenses
- Provides an upgrade path for license text
  - COPY_LIC_DIRS does not provide this
Capturing Source Code

• Two possible approaches here
  – Shipping the downloads directory
  – Using the archiver

• Archiver is more flexible
  – Supports filtering by license and recipe type
  – Configurable to fit your legal advice
Shipping the Downloads Directory

- Set `BB_GENERATE_MIRROR_TARBALLS = "1"`
  - Enables the mirroring of git repositories

- Build an image
  - Should be a clean build

- Copy or tarball the downloads directory
  - You can exclude `.done` files and version control subdirectories
Shallow Mirror Tarballs

• By default, git mirror tarballs contain full history

• Set `BB_GIT_SHALLOW` and `BB_GENERATE_SHALLOW_TARBALLS` to enable

• Can save a lot of space in a mirror
  – 7.5 GB -> 1 GB in one recent project
Using the Archiver

• Set `INHERIT += “archiver”` and ARCHIVER_MODE
  – “original”
  – “patched”
  – “configured”

• Other options
  – Original source -> patched source diff
  – Recipe files
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

• 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
SDK/ESDK Distribution

- An SDK/ESDK is just a different type of distributed image

- If using the archiver, make sure to extend COPYLEFT_RECIPE_TYPES
• 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
License Flags

- Another method of excluding recipes by license class
- May be used to highlight non-copyright issues such as required patent licenses
- Set LICENSE_FLAGS_WHITELIST to enable flagged recipes
SPDX File Creation

• SPDX is a standard data exchange format for software manifests

• Supported in Yocto Project by meta-spdxscanner layer

• Uses DoSOCSv2 or a Fossology Server to perform analysis
Recent Improvements

- Per-image INCOMPATIBLE_LICENSE
- Devtool and recipetool have improved license handling
- Several license metadata fixes
WIP: Mirror Archiver

• The capture of Complete Corresponding Source must be testable

• The best test is a full rebuild
  – Even better as support for reproducible builds improves

• Current archiver modes do not support this
• Supports split (directory per package) or combined (single directory) mirror creation

• Uses the fetcher in bitbake to capture SRC_URI items
  – Like grabbing the downloads directory but supports copyleft filtering

• Allows further filtering of SRC_URI
  – E.g. You can exclude `file://` URIs if you’re also providing layers
WIP: License Information Bundle

- Single license info artifact per image

- HTML format
  - Two sections: Packages and common licenses
  - License text in `<pre>` tags
  - Suitable for use in documentation

- Can also be compressed and installed into an image
Comparison with Buildroot

• Buildroot has `make legal-info`
  – Well documented
  – Less configurable than Yocto Project but still pretty good
  – Captures original sources, patches and license text

• Packages can be excluded by setting `\(<PKG>_REDISTRIBUTE = NO`
Comparison with OpenWRT

- Can’t find license compliance documentation for OpenWRT
- This needs improvement
Fossology

- Run license, copyright and export control scans
- Automated scanning process with support for manual correction
- Command line and Web UI interfaces
- A Linux Foundation project
OpenChain Project

- Improving license compliance across software supply chains
- Defines a specification and a training curriculum
- Conformance certification to build trust
- A Linux Foundation project
Software Heritage

• Collects and preserves software source code

• Indexed at source file level and searchable by SHA1 hash

• Allows submission by web interface or API

• An Inria project
  – French national research institute for the digital sciences
Thank You!

Follow Up: paul@betafive.co.uk