An Introduction to Software Licensing

David E. Bernholdt

Computer Science and Mathematics Division and National Center for Computational Sciences
Oak Ridge National Laboratory

Best Practices for HPC Software Developers webinar series

5 December 2018

Please open the Q&A Google Doc so that I can ask you some questions!

http://bit.ly/IDEAS-licensing

(And you're welcome to ask me questions too)

See slide 2 for license details

Disclaimers, license, citation, and acknowledgements

Disclaimers

- This is not legal advice (TINLA). Consult with true experts before making any consequential decisions
- Copyright laws differ by country. Some info may be US-centric

License and Citation

- Requested citation: David E. Bernholdt and Michael Heroux, An Introduction to Software Licensing, in Best Practices for HPC Software Developers webinar series, 5 December 2018. DOI: 10.6084/m9.figshare.7409573.
- This webinar will be archived at https://ideas-productivity.org/events/hpc-best-practices-webinars/#webinar024

Acknowledgements

- This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.
- This work was performed in part at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.
- This work was performed in part at Sandia National Laboratories. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND NO SAND2017-5474 PE
- Discussions with Todd Gamblin, LLNL

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Bottom line up front

How you choose to license your software should be viewed as a tool to help accomplish your goals for that software.

There is no universal "right answer"!

This tutorial will present common terminology, and examples of some of the considerations that might go into choosing a license.

The intent is to get you thinking, not to give you answers.

Some terminology and background

Copyright, patents, trademarks, and licenses

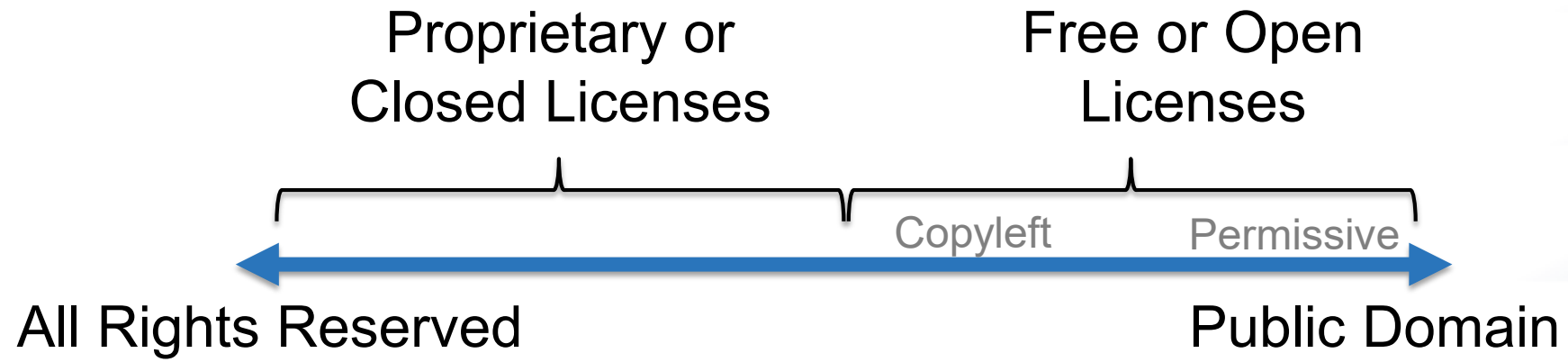
- **Copyright** grants the creator of an **original work** exclusive rights to its use and distribution, including limits on derivative works
- A **patent** grants the inventor of **something new, useful, and non-obvious** the rights to its production, use, and distribution
- A **trademark** is a **sign, design, or expression** which identifies products or services from a particular source, as distinguished from other sources
- **Licenses** are used to transfer (selected) rights in a work, invention, or mark from one party to another
  - Software licenses are mostly about copyright, but can contain clauses on patents and trademarks too

Your software starts out copyrighted

- Under the law, the software you write is subject to **copyright on creation**
  - You don't have to do anything special to claim copyright
  - Unless you specify some license, all rights are reserved to the owner of the copyright
- The copyright owner may be **you, or your employer**
  - "Work for hire" (i.e. as part of your job) is probably owned by your employer. Employment contracts often make IP rights explicit.
  - Question 1: Who owns the rights in the work you create?
  - Homework: Find out!
  - If your employer owns the copyright, you probably have to get formal permission to license and distribute your software
- Exception: Works created by the US government cannot be copyrighted
  - They are considered to be in the public domain
    - Comment: Originally to ensure public access to the US legal code

Answer Q1 at
http://bit.ly/IDEAS-licensing

The licensing spectrum



Free vs Open Source?

- "Free" in licensing discussions should refer strictly to "freedom" (to do certain things with the software)
- Often gets conflated with "free as in beer", muddling the discussion. Hence some prefer term "open source"

Major names in Free/Open Source Software:

- Free Software Foundation (FSF) http://fsf.org/licensing
- Open Source Initiative (OSI) http://opensource.org

Common misconception:
Nothing in the definition of free or open source software prevents you from making money from it! (more later)

Defining free software: The four freedoms

From the Free Software Foundation

- The freedom to **run the program** for any purpose
- The freedom to **study how the program works**, and **change it** so it does your computing as you wish
  - **Access to the source code** is a precondition for this
- The freedom to **redistribute copies** so you can help your neighbor
- The freedom to **distribute copies of your modified versions** to others. By doing this you can give the whole community a chance to benefit from your changes
  - **Access to the source code** is a precondition for this

The OSI has a definition which amounts to the same thing, for most purposes

Permissive vs copyleft OS licenses

Permissive

- Licensee can distribute derivative works as they see fit
  - Relicensing of derivatives is allowed
  - Including proprietary licenses
- Examples
  - Apache License
  - MIT License
  - BSD License

Copyleft

- Licensee must distribute derivative works as open source
  - Also referred to as “restrictive” or “viral”
- Examples
  - GPL (v2 and v3)
  - LGPL

Note: Derived works may be held private and never released

What is a derivative work?

- *A derivative work is an expressive creation that includes major copyright-protected elements of a previously created first work* (Wikipedia)
- Basically: modifications to someone else's software
- But what about linking to a library? (Statically vs dynamically?) Interacting via pipes? Use as a component in a coupled multiphysics application?
  - Opinions differ
  - FSF (GPL) considers everything in a single executable to be a derived work (source of "viral" label)
  - LGPL created for libraries – says linking not considered derived work
  - Matters less for permissive licenses
  - Leads to concerns over "compatibility" in combining software under different licenses (more later)

Test: Is this an open source license? (A real-world example)

Answer Q2 at
http://bit.ly/IDEAS-licensing

In order to acquire access to the code sources, the recipient agrees:

1. to compile/use the XYZZY source code AS IS without modification; users however are welcome to request changes, or to contribute modifications subject to approval of the authors;
2. if the copy of the XYZZY downloaded by the authorized user is made available to third parties, to ensure that the user agreement is followed by the third parties;
3. to send a one-time email to xyzzy@example.com describing planned research using that module
4. prior to publication, to email a draft of the article/letter/note to xyzzy@example.com
5. to include in published results or presentations the proper code name(s) and appropriate references.

Answer: Is this an open source license? No
(A real-world example)

In order to acquire access to the code sources, the recipient agrees:

1. to compile/use the XYZZY source code AS IS without modification; users however are welcome to request changes, or to contribute modifications subject to approval of the authors;

2. if the copy of the XYZZY downloaded by the authorized user is made available to third parties, to ensure that the user agreement is followed by the third parties;

This violates the freedom of being able to distribute copies of your modified version of the code to others

Perhaps they want to impose some measure of “quality control” over modifications? Maybe they’ve had problems in the past with users distributing modified code with errors that are believed to reflect poorly on the original code?

Possible alternative: Some open source licenses include a requirement that derivatives must be clearly distinguished from the original (e.g., different name)

Choosing a license

Considerations in choosing a license

- What rights do you want to retain or grant?
  - Who can use the program? (proprietary vs open)
  - Can users see the source code? (proprietary vs open)
  - Can users modify the source code? (proprietary vs open)
  - Can the users redistribute original or modified code? (proprietary vs open)
  - Can modified code be relicensed? (permissive vs copyleft)
- Compatibility with software under other licenses
  - Permissive licenses have fewer issues
  - http://www.fsf.org/licensing/
- Labeling of derived works
  - Derived works must be identified differently than original work
- Patent grant/retaliation
- Expectations of the community you want to engage?

Use an existing free/open source license rather than inventing a new one!

FSF and OSI certify many existing licenses (~80) as meeting their criteria

Popular OSI-approved licenses

License	Type	GPL-Compatible	Patent Grant
Apache License 2.0	Permissive	v3,not v2	yes
BSD 2-Clause and 3-Clause licenses	Permissive	yes	silent
GNU General Public License (GPL) v3	Copyleft	yes	yes
GNU Library or "Lesser" General Public License (LGPL) v3	Weak Copyleft	yes	yes
MIT license (MIT)	Permissive	yes	silent
Mozilla Public License 2.0	Permissive	yes	yes
Common Development and Distribution License	Permissive	no	yes
Eclipse Public License 2.0	Weak Copyleft	yes	yes
Affero General Public License v3 *(network use == distribution)*	Copyleft	yes	yes

ChooseALicense.com (by GitHub)

- Primarily a decision-tree approach to helping you choose a license
- But backed by a repository with analysis of 30+ widely used licenses
- **The easiest way to access the whole list is to go to the “Appendix”**
  - **https://choosealicense.com/appendix/**
  - A portion of the Appendix is shown at left
- This is implemented in a GitHub repository with Jekyll, and open to pull requests!

License	Commercial use	Distribution	Modification	Patent use	Private use	Disclose source	License and copyright notice	Network use is distribution	Same license	State changes	Liability	Trademark use	Warranty
Academic Free License v3.0	●	●	●	●	●		○			○	●	●	●
GNU Affero General Public License v3.0	●	●	●	●	●	○	○	○	○	○	●		●
Apache License 2.0	●	●	●	●	●		○			○	●	●	●
Artistic License 2.0	●	●	●	●	●		○			○	●	●	●
BSD 2-Clause "Simplified" License	●	●	●		●		○				●		●
BSD 3-Clause Clear License	●	●	●	●	●		○				●		●
BSD 3-Clause "New" or "Revised" License	●	●	●		●		○				●		●
Boost Software License 1.0	●	●	●		●		○				●		●
Creative Commons Attribution 4.0 International	●	●	●	●	●		○			○	●	●	●
Creative Commons Attribution Share Alike 4.0 International	●	●	●	●	●		○		○	○	●	●	●
Creative Commons Zero v1.0 Universal	●	●	●	●	●						●	●	●
Educational Community License v2.0	●	●	●	●	●		○			○	●	●	●
Eclipse Public License 1.0	●	●	●	●	●	○	○		○		●		●
Eclipse Public License 2.0	●	●	●	●	●	○	○		○		●		●
European Union Public License 1.1	●	●	●	●	●	○	○	○	○	○	●	●	●
European Union Public License 1.2	●	●	●	●	●	○	○	○	○	○	●	●	●
GNU General Public License v2.0	●	●	●		●	○	○		○	○	●		●
GNU General Public License v3.0	●	●	●	●	●	○	○		○	○	●		●
ISC License	●	●	●		●		○				●		●
GNU Lesser General Public License v2.1	●	●	●		●	○	○		○	○	●		●
GNU Lesser General Public License v3.0	●	●	●	●	●	○	○		○	○	●		●
LaTeX Project Public License v1.3c	●	●	●		●	○	○			○	●		●
MIT License	●	●	●		●		○				●		●
Mozilla Public License 2.0	●	●	●	●	●	○	○		○		●	●	●
Microsoft Public License	●	●	●	●	●		○					●	●
Microsoft Reciprocal License	●	●	●	●	●	○	○		○			●	●
University of Illinois/NCSA Open Source License	●	●	●		●		○				●		●
SIL Open Font License 1.1	●	●	●		●		○		○		●		●
Open Software License 3.0	●	●	●	●	●	○	○	○	○	○	●	●	●
PostgreSQL License	●	●	●		●		○				●		●
The Unlicense	●	●	●		●						●		●
Universal Permissive License v1.0	●	●	●	●	●		○				●		●
Do What The F*ck You Want To Public License	●	●	●		●								
zlib License	●	●	●		●		○			○	●		●

Consideration: Software business models

Approach	Proprietary	Copyleft	Permissive
Sell the software	yes	yes	yes
“Fremium” or “dual licensing” allows free use by some, paid by others	yes	yes	yes
Relicense to proprietary	n/a	no	yes
Sell convenience, e.g., packaging, installation media, pre-compiled executables	yes	yes	yes
Sell professional services around the software, e.g., training, technical support, consulting	yes	yes	yes
Sell custom development services, e.g., proprietary extensions, accelerated development	yes	yes	yes
Sell software-as-a-service (SaaS)	yes	yes	yes
Sell the research	yes	yes	yes

Consideration: Don't want others to profit from my open source software

- A permissive license allows someone else to take derivatives proprietary
- A copyleft license will prevent that

But there may be other considerations…

- What if you do want a commercial entity to use your software?
  - Exposure, broader distribution
- Copyleft is scary to many commercial entities
  - How far does the viral license reach into other parts of the product?
  - Legal opinions differ, no case law yet
    - Lawyers will tend toward a conservative answer: avoid copyleft software
    - Experience: some companies will not consider working with copyleft software
    - Experience: some companies consider staff working on copyleft software to be "contaminated" and will not allow them work on other software
- Even in non-commercial environments, copyleft may raise compatibility concerns

Consideration: Protecting my intellectual property

- If I make my source code freely available, then others can use the novel ideas embodied in it to “scoop” me
- Proprietary licenses (obviously) allow you to keep source private
- Open source licenses don’t require that you make derived works public, only that ***if*** you do, you make the source available
- Delay public release until you’ve had a reasonable chance to exploit the results of your work
  - Until initial papers are published
  - Fixed time period (e.g., one year)
    - A similar compromise is sometimes used in academic publishing: sponsor may want open access but allow publisher a proprietary exploitation period (often 1 year) before making it openly available

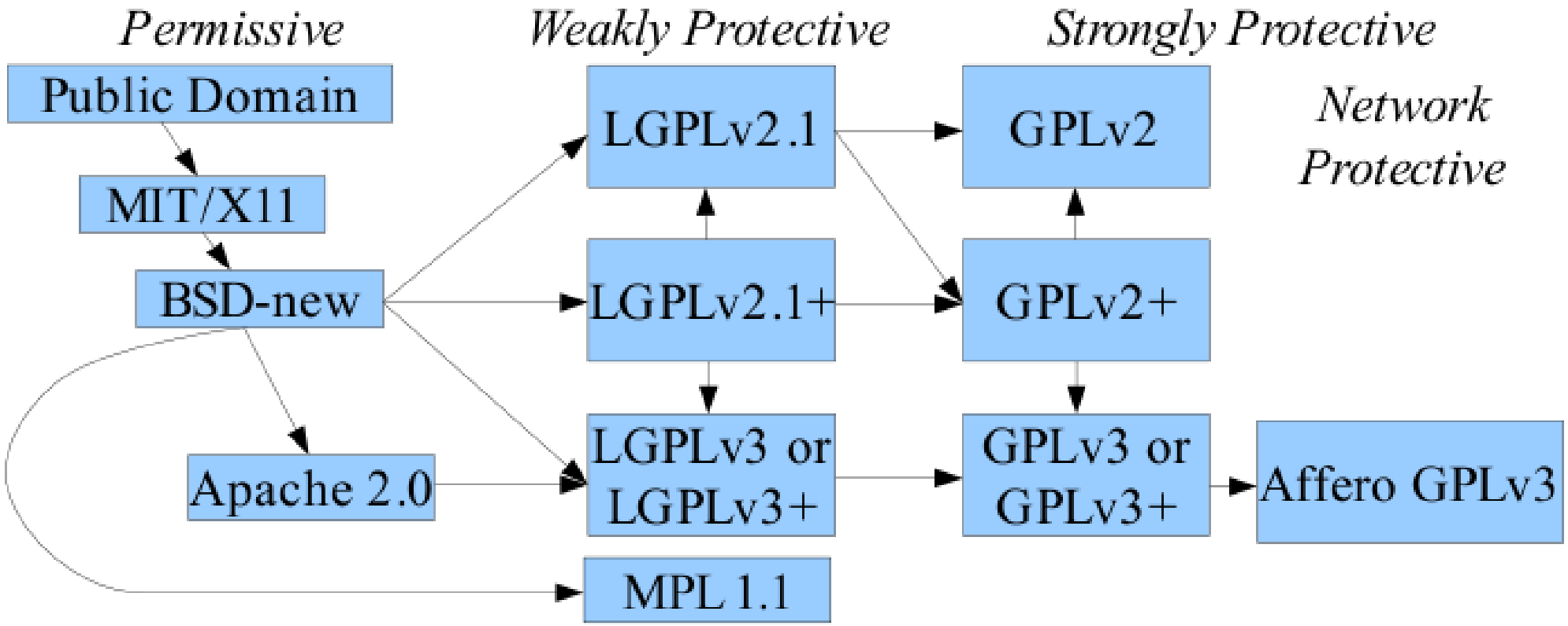
Patent clauses in software licenses

- Software patents can be a serious consideration today
  - Regardless of philosophical arguments for or against, software patents are a reality
  - If you're using a piece of software (even open source) that is covered by a patent and you don't have a license for the patent, you're infringing
  - Not being aware of a patent does not excuse the infringement
  - You can be sued for monetary damages
- Many common software licenses are silent on patents
  - Especially older ones (e.g., BSD, GPLv2)
- Some newer licenses do include patent clauses
  - Usually a royalty-free license to use patented content (e.g. Apache 2.0, GPLv3)
  - Some explicitly say that they do not grant any patent rights (e.g., BSD 3-Clause Clear)
  - Or retaliation clauses: "if you sue me for patent infringement, you can't use this software"

License compatibility

- In practice, most software is a **combined work** of some kind
  - Multiple packages with (potentially) different licenses (e.g. main package and dependencies)
  - Do the license terms allow the packages to be distributed (or even used) together?
  - Is the combined work considered a derived work?
- Different licenses have different concepts of what constitutes a derived work and how derivatives may or must be licensed
  - Example: strong copyleft considers linking to produce a derived work, and requires derivatives be distributed under the same license as the original
- There are different interpretations of what licenses are compatible
  - Little litigation so far
- Most significant concerns tend to be about distribution of software
  - Larger projects starting to pay more attention to his

License compatibility in pictures



One view of license compatibility between common FOSS software licenses. The arrows denote a one directional compatibility, therefore better compatibility on the left side than on the right side.
By David A. Wheeler - http://www.dwheeler.com/essays/floss-license-slide.html, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=41060008
via https://en.wikipedia.org/wiki/License_compatibility

Considerations favoring open source

- Challenges of managing and archiving the paperwork associated with proprietary licenses
- Explicit license agreements can inhibit (legal) use of software
- I want to support peer review and reproducibility in science
- My sponsor requires that I release my software as open source
- I believe that the results of publicly-funded research should be publicly available
- I want to build a self-sustaining community around my software

A few more points about our real-world example

In order to acquire access to the code sources, the recipient agrees:

1. to compile/use the XYZZY source code AS IS without modification; users however are welcome to request changes, or to contribute modifications subject to approval of the authors;
2. if the copy of the XYZZY downloaded by the authorized user is made available to third parties, to ensure that the user agreement is followed by the third parties;
3. to send a one-time email to xyzzy@example.com describing planned research using that module
4. prior to publication, to email a draft of the article/letter/note to xyzzy@example.com
5. to include in published results or presentations the proper code name(s) and appropriate references.

Answer Q4 and Q5 at
http://bit.ly/IDEAS-licensing

Why are these Clauses Included?

4. prior to publication, to email a draft of the article/letter/note to xyzzy@example.com

An attempt to address prior experience with users misusing the code and producing publications with erroneous results, thus reflecting poorly on the code used to obtain them.

Creates a burden on the code owners.

Not sure how strongly they attempt to enforce this.

5. to include in published results or presentations the proper code name(s) and appropriate references.

A natural desire for the software to be credited in papers where it is used.

Does not violate free/open source software principles.

Some licenses require attribution, but usually only in source code.

- Creative Commons licenses can include an attribution clause (see later slide)

Possible alternative: CITATION file?

Some related matters

Software Licenses Can be Changed

- You may start out using one license for your code and later discover unanticipated problems
- Or maybe your goals change

But changing licenses is not necessarily easy

- (Generally) each and every contributor to a code holds a copyright interest in it
- Each and every contributor must be contacted and agree to the relicensing
  - In practice, different institutions may have different ideas of “due diligence”
  - Keep good records of contributors; try to keep them current
- Contributor license agreements (CLAs) and contribution transfer agreements (CTAs) can simplify this
  - But present different challenges (see next slide)

Accepting code contributions

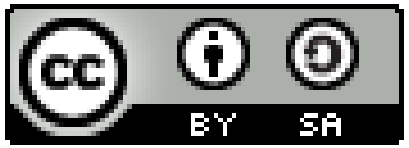
- Code contributions are implicitly offered under the current license
- Some projects require a contributor agreement
  - Contributor license agreement (CLA) defines the terms between the contributor and the maintainers of the software
  - Contributor transfer agreement (CTA) transfers copyright ownership from contributor to maintainers
- Why?
  - Clarify or make explicit terms of contribution (awareness by contributor)
  - Obtain additional rights, e.g., relicensing, patents, etc.
  - Ensure “clear title” to make the contribution
- Why not?
  - Creates “barriers to entry” – may discourage potential contributors
  - Legal agreements that may require official review and signature
    - Experience: Lost funding for a project because lawyers wouldn’t agree to terms of a CLA
- See Resources slide for several viewpoints

Managing copyright notices in software

- All this does no good if you don't make the license you've chosen clear to all
- Need to **assert copyright** and make **license terms** explicit
- Do these centrally or in every file?
  - Single COPYING or LICENSE file per package (or directory)
  - In comments at the top of the file
  - Advantages and disadvantages to each
- ***Best practice: do both***
  - Intelligently, to make it as easy to maintain as possible
- Authorship (separate, but related)
  - Version control is best way to maintain accurate records of authorship
- See Managing Copyright Information within a Free Software Project for details
- Also Software Package Data Exchange (SPDX, emerging standard)

Open licensing of non-software artifacts

- Creative Commons is a family of licenses analogous to open source, but for things other than software
- License variants
  - CC BY (Attribution)
  - CC BY-SA (Attribution-ShareAlike)
  - CC BY-ND (Attribution-NoDerivs)
  - CC BY-NC (Attribution-NonCommercial)
  - CC BY-NC-SA (Attribution-NonCommercial-ShareAlike)
  - CC BY-NC-ND (Attribution-NonCommercial-NoDerivs)
- CC0 Public Domain Dedication
  - Indicates intent to place artifact in the public domain
  - Doesn't satisfy legal requirements in all jurisdictions
- See https://creativecommons.org



Resources

- https://opensource.org (OSI)
- http://www.fsf.org/licensing/ (FSF)
- https://choosealicense.com, https://choosealicense.com/appendix/ (GitHub)
- Software Freedom Law Center (SFLC)
- https://en.wikipedia.org/wiki/License_compatibility
- Managing Copyright Information within a Free Software Project
- Software Package Data Exchange (SPDX, emerging standard)
- http://contributoragreements.org/, https://developercertificate.org/ and http://ebb.org/bkuhn/blog/2014/06/09/do-not-need-cla.html
- https://creativecommons.org (CC)
- US DOE ASCR (open source) software policy
- Your institution's Technology Transfer Office (or equivalent)
- An Intellectual Property Lawyer (knowledgeable in software)
- Talk to colleagues and learn from their experiences

Additional Resources Recommended by Others (1/3)

I have not yet studied these carefully myself, but I trust the people who recommend them.

- Neil Chue Hong (Software Sustainability Institute) from his tutorial An Introduction to Software Licensing (yes, the same title as this presentation, but developed completely independently)
  - The Whys and Hows of Licensing Scientific Code
  - A Quick Guide to Software Licensing for the Scientist-Programmer
  - The Legal Side to Open Source
  - The International Free and Open Source Lawbook
  - qLegal: advice for tech start-ups + entrepreneurs
  - tl;dr legal: Software Licenses in Plain English
  - Open Source Software Watch

Additional Resources Recommended by Others (2/3)

I have not yet studied these carefully myself, but I trust the people who recommend them.

- Todd Gamblin (LLNL)
  - License compatibility resources
    - Open Source Licenses and their Compatibility
    - Which Licenses May Not be Included within Apache Products?
  - (Re-) Licensing considerations of various organizations
    - LLVM Contemplates Relicensing, LLVM Relicensing Effort
    - HEP Software Foundation licensing working group, particularly:
      - https://hepsoftwarefoundation.org/organization/2017/02/21/licensing.html
      - https://hepsoftwarefoundation.org/organization/2018/05/09/licensing.html included this Update on Software Licensing
    - EasyBuild: GPLv2 licensing is a big issue, consider relicensing to BSD
    - GEANT Intellectual Property Rights Policy

Additional Resources Recommended by Others (3/3)

I have not yet studied these carefully myself, but I trust the people who recommend them

- More from Todd Gamblin (LLNL)
  - Patents in software licenses
    - Why so little love for the patent grant in the MIT License?
    - React's New MIT License: The Circus Enters Its Third Ring
    - GitLab freezes GraphQL project amid looming Facebook patent fears
    - Rust: Rust license changing (very slightly), Why dual MIT/ASL2 license?