An Introduction to Software Licensing

Presented at
Better Scientific Software tutorial

ISC18, Frankfurt, Germany

Michael A. Heroux
Senior Scientist, Sandia National Laboratories
Scientist in Residence, St. John's University, MN

EXASCALE COMPUTING PROJECT

National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

IDEAS productivity

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 Bernholdt and Michael Heroux, An Introduction to Software Licensing, tutorial, in ISC High Performance 2018: DOI: 10.6084/m9.figshare.6452996.

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

Some terminology and background

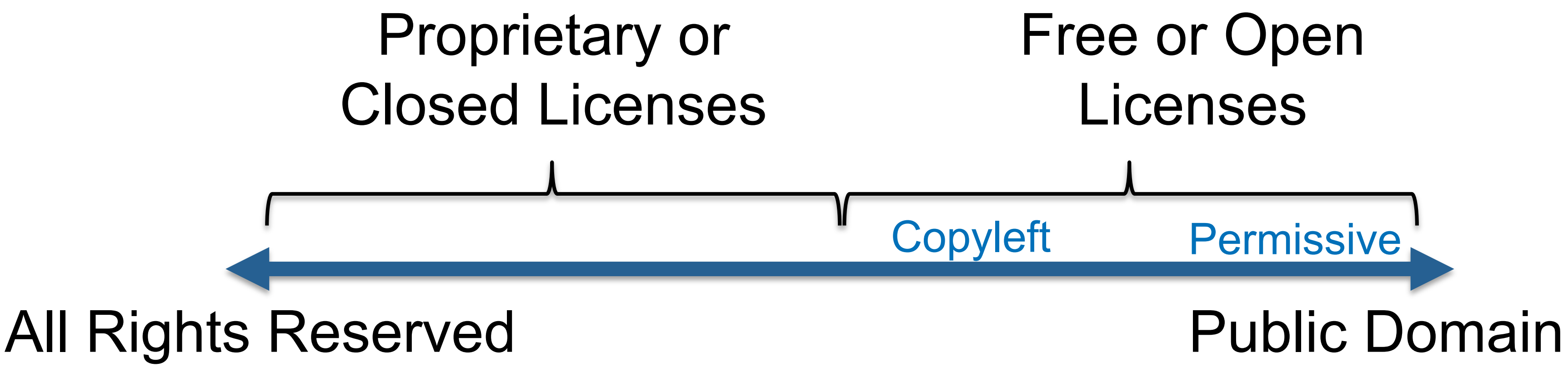
Copyright and software licensing

- Copyright grants the creator of an original work **exclusive rights to its use and distribution**
- Rights of particular interest for software include
  - **Reproduction and distribution**
  - **Derivative works**
- **Licenses** are used to transfer rights in the work from one party to another

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
- 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.
- Exception: Works created by the US government cannot be copyrighted
  - They are considered to be in the public domain

The licensing spectrum



Proprietary or Closed Licenses

Free or Open Licenses

Copyleft

Permissive

All Rights Reserved

Public Domain

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

Defining free software: The four freedoms

- 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*

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)
- Modifications to someone else's software
- 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

Test: Is this an open source license? (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;
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.

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? (prop. 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

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 3-Clause "New" or "Revised" license	Permissive	yes	silent
BSD 2-Clause "Simplified" or "FreeBSD" license	Permissive	yes	silent
GNU General Public License (GPL)	Copyleft	yes	yes
GNU Library or "Lesser" General Public License (LGPL)	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	Weak Copyleft	no	yes

Consideration: Software business models

Approach	Proprietary	Copyleft	Permissive
Sell the software	yes	yes	yes
Sell to commercial users aka *dual licensing*	n/a	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

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)

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.

Some related matters

Managing copyright notices in software

- 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 detailed discussion

Accepting code contributions

- Code contributions are implicitly offered under current license
- All authors have a copyright interest in the code
  - If you want to relicense later, all copyright owners must agree
- 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
- These are legal agreements that may require official review and signature within your organization

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

Resources

- https://opensource.org (OSI)
- http://www.fsf.org/licensing/ (FSF)
- https://choosealicense.com (GitHub)
- Software Freedom Law Center (SFLC)
- Managing Copyright Information within a Free Software Project
- US DOE ASCR (open source) software policy
- https://creativecommons.org (CC)
- http://contributoragreements.org/
- Talk to colleagues to learn from their experiences
- Your institution’s Technology Transfer Office (or equivalent)
- An Intellectual Property Lawyer (knowledgeable in software)

Agenda

Time	Topic	Speaker
2:00pm-2:30pm	Why Effective Software Practices are Essential for CSE Projects	Anshu Dubey, ANL
2:30pm-3:00pm	Introduction to Software Licensing	Michael A. Heroux, SNL
3:00am-3:30pm	Better (small) Scientific Software Teams	Michael A. Heroux, SNL
3:30am-4:00pm	Improving Reproducibility Through Better Software Practices	Michael A. Heroux, SNL
4:00pm-4:30pm	*Break*	
4:30pm-5:00pm	Testing HPC Scientific Software – Part 1	Anshu Dubey, ANL
5:00pm-5:30pm	Testing HPC Scientific Software – Part 2	Anshu Dubey, ANL
5:30pm-6:00pm	Code Coverage Hands-on and CI Demo	Anshu Dubey, ANL