Introduction to Software Licensing

Date: 5 December 2018
Presented by: David Bernholdt (Oak Ridge National Laboratory)

David's questions to the participants

1. **Slide 6: Who owns the rights in the work you create?** (add an "X" to mark your answer)
    - **I do** XXXX
    - **My employer** XXXXXXXXXXXXXXXXXXxX
    - **I don't know** XXXX
2. **Slide 11: Is the license open source?** (add an "X" to mark your answer)
    - **Yes** XXXXXX
    - **No** XXXxXXX
    - The correct answer is "no" because it does not allow the redistribution of modified versions of the code. See also slide 12.
3. **There is no question 3** (add an "X" to mark your answer)
    - **True** XXXX
    - **False** XXX
    - It depends. I never asked you to answer Q3, but obviously it is here :-)
4. **Slide 24: Why might they ask for advance copies of publications?** (make a new bullet and add a short answer)
    - Because they need more things to read
    - To try to prevent results from being published that illustrate the flaws of their code/algorithm
    - To ensure the code is used properly and the results interpreted correctly
    - To ensure that no protected information is published (and the above)
    - Trying to keep track of what people are doing with the code and possible variants
    - To ensure there is adequate acknowledgement
    - Limit use
    - See slide 25 for my thoughts on this. The developers had prior bad experiences where people had misused the code and/or misinterpreted the results and it had tarnished the reputation of the code itself. This clause is an attempt to address that problem. I don't think this clause

impacts the analysis of whether or not it is an open source license. It may pose other concerns for prospective users. And I think it would be challenging (work) to enforce.

5. **Slide 24: Why might they ask for the code to be cited?** (make a new bullet and add a short answer)
    - Trackability of version, reproducibility of results (DOECode, OSTI/DOI)
    - Demonstrate to sponsors the adoption of the software
    - Increase the authors citation counts
    - Credit
    - Credit/citation count
    - See slide 25 for my thoughts on this. Software citation is good for everyone. This is one way of doing it. Many open source licenses have “attribution” clauses, but that usually applies only to including an attribution to the original developers in source code, not in academic publications. I think this would be challenging to enforce. I think there are alternatives to putting this in the licenses (a legal document) that might be just as effective. For example a CITATION file in the repository (like the LICENSE file I’m sure you already have).

Audience questions for David

Q. Do you have a "updated" analogy as "free as in beer" and "free as in freedom" does not seem to be easily digested by many of the students that we teach. We use "free as in a puppy" which works ok.

> David’s A. I have not had anyone complain yet, so I don’t have anything better.
> Audience A. I like "free lunch" or "free food" +1

Q. Is a code that is rewritten in another language but algorithms based on the old code considered derivative work?

A. As with other aspects of defining derived works, opinions differ and license terms differ. For example GPL version 2 explicitly says that translations into other programming languages are derivatives. GPL version 3, on the other hand, contains no such language.

Q. Follow up: How would this be viewed under patent law?

A. Patents apply to ideas, not fixed embodiments. So the language in which it was written would not matter from the patent perspective.

Q. Especially in the context of a community science code, who owns the right (i.e. has the authority) to change a license? Is it the people who have contributed to the science code?

A. (see also slides 27-28 in the presentation) In general, every contributor has a copyright interest in the resulting code. So in principle, you'd have to contact all of the past contributors and get their (legal) agreement to the relicensing. As you can imagine, there are all kinds of possible failure modes here. What constitutes "due dilligence" in trying to reach all contributors? Etc. This is a matter that you and your lawyers will have to work out.

B. Contributor license Agreements (CLAs) and Contribution Transfer Agreements (CTAs) are ways of making this easier, either by getting permission in advance to relicense (via a CLA) or asking for ownership of the copyright in the contribution to be transferred to the maintainers of the package (CTAs). However these are legal agreements, which you might have to work with your lawyer to sign. (Our lawyers don't like CTAs and I've had problems with CLAs containing terms they don't like.) So they can create barriers to entry and discourage potential contributors. On the Resources page of the presentation are some folks that have opinions on these issues and other ways to handle them.
    a. http://contributoragreements.org/
    b. https://developercertificate.org/
    c. http://ebb.org/bkuhn/blog/2014/06/09/do-not-need-cla.html

Q. What is the distinction between permissive and viral in the context of software provided for research purposes only vs commercial? And what does that even mean commercial vs research?

A. There are two different distinctions being made here.
    a. Permissive vs copyleft are terms used to describe broad categories of open source software licenses. The distinction is how much control the license exerts over licensing of derivative works. With "permissive" licenses, the creator of a derived work is permitted to essentially do whatever they want with the derivative (in terms of licensing). "Copyleft" licenses, on the other hand, specify that derived works be released under the same license as the original work. Copyleft licenses are sometimes referred to as "viral" because the license of the original "infects" any derived work.
    b. Research vs commercial I think it *not* a useful distinction to try to make in a legal sense, but my *practical* interpretation is that "research" probably refers to situations where the user is not profiting from the software, where as

"commercial" would be situations where someone is profiting from the use of the software (e.g., incorporating it into a proprietary product). The problem with terminology is how do you treat, for example, research division of a for-profit company in cases like this. Is that "research" or "commercial"? A more useful distinction might be whether the software is incorporated into a product which they might be selling. But what about if the software helped them *design* the product, but is not *part* of the product? Lots of gray.

Q. Are there more case studies available for each of these considerations when choosing a OSI approved license?

A. Audience: https://ChooseALicense.com/appendix : decision tree graph
B. David's A: Offhand, I can't give you any, but I'm sure they're out there. Look at the additional resources from Todd Gamblin (slide 33-34) and some of the links for there, which allow you to delve into some other group's discussions.
C. But it would be nice to develop a set of pointers to such case studies to supplement this presentation. If you find any, please email me at bernholdtde@ornl.gov.

Q. For GPL, can the term "single executable" be extended to encompass "source code" for platforms in which there is no executable (python code etc.)?

A. I'm not certain offhand how the FSF would treat such a case. I am not deeply familiar with license compatibility issues yet.

Q. Can you please define compatibility with GPL (I.e., GPL codes can link/compile your code? Or your code can link/compile GPL code? Or both? ---does the direction of the dependency matter?)

A. The direction of the dependency doesn't matter. It is about creating a derived work that incorporates the GPL work. The basic interpretation is that the most restrictive license "wins" for the derived work.

Q. Can you please elaborate (maybe with example) about patent grant? And also patent retaliation? (Is the latter just a weaker version of the former, i.e., we don't grant use of our patents, but we promise not to retaliate against infringement?)

A. (see also slide 20) Patent grant clauses convey rights to use patents along with the right to use the software. The term "royalty-free license" is often used in a patent grant clause, which means that you don't have to pay for the right to use the patent. Patent retaliation clauses looks at things from the other direction: if you decide my software infringes on a patent you own and you sue me, you are no longer allowed to

use my software (this situation terminates the licenses as it applies to you, specifically). A broader version of the patent retaliation idea says that if you sue me for infringement by any software for which I own copyright, then your right to use *all* software for which I own the copyright is terminated.

Q.What prevents a private company from taking a GPL-licensed software and incorporating it into their proprietary software?

A. Legally, if they release the derived work, they'd be required to release the source. Practically, it can be hard to enforce this. You first have to figure out that they're actually using GPL code, which if they're not releasing it,...
B. There are organizations that try to look out for such mis-uses and sue, usually to get the company to do the right thing rather than for monetary damages. This is one of the few areas of open source software where I'm aware of a judicial record. I haven't done a thorough study, but I have heard of many cases where the plaintiffs win such cases. The most common cases are things like home network routers, which use a lot of GPL software packages and it is fairly easy to figure out that they're doing it.

Q. Is there a time limit to licenses?

A. I have not seen this in open source. Certainly proprietary licenses can set any terms they want. Many proprietary software packages come with free trial periods before you have to buy a license. In other cases, maybe you need to pay a fee annually. Maybe that's renewing a fixed-term license. Or maybe the license is good in perpetuity and you're paying for maintenance on an annual basis (i.e, periodic updates of the software).

Q. Are these licenses recognized internationally or is it on a per country basis?

A. The intellectual property laws of every country are different, in some degree, so the legal interpretation of a given license might vary from country to country. However my expectation is that this would more in subtleties and corner cases than in the essentials terms of the license.
B. Public Domain, in particular, however, is *not* a universally recognized concept. From what I've read, some countries don't have a concept of public domain. And there is apparently no universal mechanism or language that meets the legal requirements to place something in the public domain in all jurisdictions. So something like the Creative Commons CC0 "public domain dedication" does not legally meet the criteria everywhere.

Q. What's the best way to change a license of existing software?

A. Basically, you need to get agreement from all of the copyright holders in order to relicense the software. And how best to do that depends a lot on your contributors. You can circumvent this with a Contribution Transfer Agreement or a Contributor License Agreement that you require of all contributors, but these may raise barriers to contribution. See also slides 27-28 in the presentation.

Q. As a copyright holder can you select multiple licenses?

A. Yes, you can. This is often referred to as "dual licensing" (presuming you have two licenses; I've heard of scenarios where people use more). A typical dual licensing scenario says that if you're using the code in a non-commercial context, license A applies (usually an open source license), whereas if you're using the code in a commercial context, you must contact us to obtain a license (which might include royalty payments based on the commercial use).

Q. If you open source your software as BSD (permissive), but it uses a library that is weakly permissive, what responsibility do you have to alert potential users of these dependencies and differences in licenses.

A. Legally, you have no obligation to notify your users about anything regarding license compatibility. And if your code is permissive, you're generally not going to encounter compatibility problems. Looking at things in a more general way… If you want to take care of license compatibility, you want to look at the licenses for your own software and the dependencies you rely on so that they provide suitable compatibility "by construction". For example, if the license of one of your dependencies is problematic from a compatibility standpoint, you might want to choose an alternative that has a "better" license for your purposes. Similarly, if someone else is considering using your software and is concerned about license compatibility, they may decide not to use it if the license doesn't work for them.

Q. A company asks for software to be released as BSD since their policy prohibits working with GPL. They claim it will be used for research purposes and not commercially. How can this distinction be defined and enforced?

A. The only option I can really think of is dual licensing, so that it would be BSD for non-commercial use, but require the negotiation of a license with you (or perhaps just deny a license) for commercial use. Software

licenses are like contracts, and ultimately they are enforced by suing violators.

Q. I'm not seeing the "decision tree" you mentioned in choosealicense.com. Is that a user-facing thing?

A. The front page for chosealicense.com presents you with a number of scenarios and you have to make a decision as to which scenario applies to you. Within each choice, a variety of license options are presented. That's what I refer to as a decision tree.

Q. I am still confused about how commercial use works in the context of picking a license? I Often come across research software online, and they say for research purposes only.

A. Open source licenses do not place any limitations on the use of the software. That would actually violate one of the "Four Freedoms" that define "free software" (slide 8). It is certainly possible to use a license that makes the software available only for non-commercial use, but that would not be an open source license, it would be a form of proprietary license on the spectrum presented in slide 7. Another option, mentioned in some of the other questions is the idea of dual licensing. One license (usually open source) would apply to non-commercial use of the software, while commercial use would require the user to negotiate a license with you (or you could simply not allow commercial use). This is subtly different from the previous case.

Q. so then licence would then have a license file that gives two licenses? Can you think of a good example?

A. MySQL and Qt are a couple of examples I was able to find.

Q. What happens if you want to relicense a software (as a company): do you have to ask all software contributors if they agree with the new license? What happens if some contributors don't work for the company anymore? How is this handled? I'm assuming there are no "external" contributors to that software.

A. So in the scenario you put forward, I would first ask who actually owns the copyrights in the code: the individual employees who wrote them, or the company that employs all of them. If the individuals own the copyright and nothing was done when the person left to transfer their rights to the company (or someone still at the company), then you need to get their agreement. If the company owns the copyrights, they don't need to ask anyone, including their current employees.

Q. Especially in the context of a community science code, who owns the right (i.e. has the authority) to change a license? Is it the people who have contributed to the science code?

A. Generally, the contributors personally, or their employers own the rights in their specific contributions. The holders of the rights need to agree to changing the license. Exactly how this happens in a community code environment would depend on the governance model. But typically, there is some core of people who manage the code in a more active way, and perhaps many additional contributors who may come and go. So maybe the core team decides it would be desirable to relicense. But then they need to get the agreement of all of the rights holders.

Q. Can you please define compatibility with GPL?

A. Say I have a code under license A, which requires derivative works to be distributed under license A. And another code under license B, which requires derivative works to be distributed under license B. If I create a combined work that is derived from both A and B, then the result would have to be licensed under BOTH license A and license B. Which is probably not possible. However if license B allowed derivatives to be relicensed, then the combined work could be released under license A.