# Derivative Works

by Lawrence Rosen

Many users of open source software are frightened by the term "derivative works." They worry that they might accidentally create derivative works that will infect their own proprietary software. This is a complex topic that courts and lawyers disagree about, but I think we can agree on definitions that can ease people's concerns.

First a brief reminder of why the term "derivative work" is so important. Here's what a typical license might say: *"Licensor hereby grants you a license ... to prepare derivative works based upon the original work and to distribute those derivative works with the proviso that copies of those derivative works that you distribute shall be licensed under this License."* [See, for example, the GNU General Public License (GPL) or the Open Software License (OSL), both available at www.opensource.org/licenses).

How can you tell when you've created a derivative work? The Copyright Act, at 17 U.S.C. §101, is a little vague, and doesn't say anything at all about software:

> A "derivative work" is a work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications which, as a whole, represent an original work of authorship, is a "derivative work".

Almost everyone agrees on this: If you take the copyrighted source code of any program and physically modify it – actually revise the program or translate it into another computer language – you have created a derivative work of that program. That's the simple case. If you do such a thing with a program licensed under the GPL or the OSL, then you must honor the reciprocity provision and publish the source code of your derivative works that you distribute.

But what happens if you merely copy an original program as a component in your own, perhaps larger, work? Did you create a derivative work of the original program? Does it make a difference that you didn't actually *modify* the source code to combine the original program into your larger work.

Does merely *linking to* a program without any change to the original source code create a derivative work of that program? Almost every program links to library routines. Surely one doesn't create a derivative work of a library simply by calling a *sqrt* module in the library! Why should it be any different when you link to something as complex as an enterprise server or data base engine?

What about *linking from* a software program, as you might do when linking your device driver into a GPL- or OSL-licensed program like Linux?

Does it matter what technical form of linking you use? Or is that analysis (e.g., static linking, dynamic linking, passing data through an API, as an object contained within a larger object, etc., etc.) a technical morass that merely obscures the fundamental issue? How can the law of derivative works keep up with technological change in the world of software engineering?

These questions are important because some licenses require you to publish the source code of your portion of the resulting "derivative work" program, a burden you may not be willing to accept.

Here's how I would decide in the edge cases that I described above:

- The primary indication of whether a new program is a derivative work is whether the source code of the original program was used, modified, translated or otherwise changed in any way to create the new program. If not, then I would argue that there is not a derivative work.

- The meaning of derivative work will not be broadened to include software created by *linking to* library programs that were designed and intended to be used as library programs. When a company releases a scientific subroutine library, or a library of objects, for example, people who merely use the library, unmodified, perhaps without even looking at the source code, are not thereby creating derivative works of the library.

- Derivative works are not going to encompass plug-ins and device drivers that are designed to be *linked from* other off-the-shelf, unmodified, programs. If Linux is designed to accept separately-designed plug-in programs, you don't create a derivative work by merely running such a program under Linux, even if you have to look at the Linux source code to learn how to do so.

- In most cases we shouldn't care how the linkage between separate programs was technically done, unless that fact helps to determine whether the creators of the programs designed them with some apparent common understanding of what a derivative work would look like. We should consider subtle market-based factors as indicators of intent, such as whether the resulting program is being sold as an "improved" or "enhanced" version of the original, or whether the original was designed and advertised to be improvable "like a library."

The real reason you should care about this issue is that we want to encourage as much free and open source software to be created without scaring proprietary software users away. If people believe that merely touching your open source software will *infect* their software with a *virus*, you will make your software less attractive. We need to make sure that companies know, with some degree of certainty, when they've created a derivative work and when they haven't.

*Lawrence Rosen is an attorney in private practice, with offices in Los Altos Hills and Ukiah, California (www.rosenlaw.com). He is also corporate secretary and general counsel for Open Source Initiative, which manages and promotes the Open Source Definition (www.opensource.org).*



 