Embedded Linux and Copyright Law

By Michael Barr

This article was updated November 27, 2018.

The rising popularity of Linux has spurred many embedded developers to consider it as an RTOS alternative. Here are just some of the legal implications for the proprietary parts of firmware with which developers should be familiar.

One of the more confusing aspects of the open source phenomenon is the proliferation of different source code licensing schemes. Indeed, if multiple pieces of software developed by others are to be used in a product, it's best to have an intellectual property lawyer read the license agreement for each component and advise on how best to proceed.

Fortunately, if only Linux is being used (as assumed for the rest of this article), the situation is much more straightforward.

What is Copyleft?

Copyleft is the free distribution of software or artistic work with the requirement that all modified or derived work also be made available for free.

A common myth is that the use of any piece of open source code, including Linux, requires the user to give away the source code to their own proprietary application. In truth, most open source licenses protect only the borrowed code and do not place any restrictions on other software you might develop for use alongside it.

Licensing and the Linux Kernel

The specific license accompanying the Linux kernel is called the **GNU General Public License (GPL)**. The GPL defines rules that apply when leveraging software that would not have been otherwise accessible if the code were proprietary. Under these rules, anyone is entitled to improve or modify the Linux kernel and its device drivers, applications, and services. But because these modifications create a derivative of the existing code, they must be made public under the same licensing terms.

If the operating system is not modified, the GPL requires only that credit be given where credit is due, any further licensing or distribution conditions are not imposed upon customers, and that the Linux source code used be provided to customers if requested.

Integrating the Linux Kernel in Proprietary Software

There are many situations in which an engineering organization might want to keep its own code proprietary even when that code is surrounded by Linux's open source code. This can usually be accomplished by following three rules of thumb during development:

1. **Start proprietary software development with a copyleft-clean code base.**

By ensuring that proprietary code does not build directly upon any open source code, developers can remain clear of the "derivative work" clause found in the GPL. Derivative works are the source of most legal confusion; they must typically be made open source under the same terms as the original code from which they are derived. But proprietary code that merely interfaces to open-source code is not derivative.

2. **Don't rely on open-source libraries unless they are under LGPL license.**

The GPL requires any code that links to a GPL library--statically or dynamically--to also be released under the GPL. However, a less protective license called the GNU Lesser General Public License (LGPL) was created so that developers could link to these open source libraries in either way without being bound to release

their application's source code. Most key Linux libraries are licensed under the LGPL.

3. **Never modify the standard interfaces to the Linux kernel.**

Under the GPL terms, any modification made to the monolithic portion of the Linux kernel must be released as open source software. Note, however, that when an application requires changes be made to the kernel, only those kernel changes must be made public. The application code (and even loadable kernel modules) can still be kept proprietary, provided that they simply interface with the kernel via Linux's standard system calls.

If the three simple rules above are observed, developers and software experts should be able to distinguish between Linux and your proprietary code for all intents and legal purposes. Of course, it is always prudent to talk with an intellectual property lawyer.