by Bob Japenga (USA)

Getting Started with Embedded Linux (Part 3)

Linux Licensing Issues

The purpose of this article series is to get you started with embedded Linux. This month you learn about the various licensing issues an embedded systems designer faces when using Linux.

The other day I switched my wife's cell phone provider of 12 years to a down-graded pre-paid plan. She doesn't use her phone often and was using an eight-year-old hand-me-down phone from one of our grandsons. I estimated that the change was going to save me about 50% compared to what we were currently paying. When I notified our carrier that we were switching, the customer service representative warned me that it was going to cost me more in the end. As I hung up the phone, I wondered where they hire people to say those things.

I also remembered getting a similar warning from one of the premier commercial off-the-shelf (COTS) embedded operating system (OS) suppliers a few years back. We developed a software system in conjunction with our customer's proprietary hardware that used this supplier's highly rated OS. The tools were superb. The support was mediocre (and expensive). Sales languished on the product. Eventually we abandoned the OS and ported the system to Linux. In six months our customer was selling the new product on three different hardware platforms and sales were taking off. As I spoke to the representative of this proprietary OS about porting to Linux, he told me to make sure I told our customer that he would have to release all of their proprietary software to the open-source community. Again I wondered how much they pay these folks to say these things.

This month, I want to look at some specific licensing questions concerning embedded Linux. It is my desire to dispel some of the fears that are fostered throughout the development community. This article is not intended to be legal advice, but I hope to frame the question, point you to the resources available, tell you how we handle the license issues, and describe some pitfalls to avoid.

SOME BACKGROUND

If you are going to build a new embedded product using Linux, there are essentially four licenses you need to be concerned about. The Linux kernel is released under the GNU General Purpose License (GPL) Version 2. The kernel enables you to create proprietary application software in user space and not be required to disclose the source code of that proprietary application code as long as it is not a "derived work" from the kernel. The kernel license is very clear about this, saying, "This copyright does *not* cover user programs that use kernel

HELPFUL DEFINITIONS

COTS: Commercial off-the-shelf
Derived work: A legal term defined under the copyright laws of the U.S. as a work based upon one or more pre-existing works
GPL: General-purpose license
LGPL: Lesser general-purpose license

services by normal system calls—this is merely considered normal use of the kernel, and does *not* fall under the heading of 'derived work.'"

Accordingly, you can issue your proprietary software under any type of restrictive licensing that you deem appropriate for your business model.

The latest version of the GNU General Purpose License (GPL) is Version 3. Some of the common Linux applications are released under this license. These powerful applications can be included in your system without requiring disclosure of any of your proprietary application code unless your proprietary code is in one of these modified Linux applications.

The libraries used by the free toolchain (GNU) are released under the GNU lesser GPL (LGPL). For example, this license enables you to link your application to the C run-time library and distribute your proprietary application in binary form.

Some Linux applications are released under the Apache License. You are free to modify any software released under this license and incorporate them into your embedded system without any subsequent restrictions other than notification issues.

All of these software license variants have been around for many years. Each has their own nuances which you should be familiar with. This is no different than your need to become familiar with the licenses of any commercial software you buy. The problem is that most of us scan to the bottom and click "I agree" and have no idea what we are agreeing to. For the sake of our fiduciary responsibilities (exposing our companies to lawsuits), and for our own integrity, we need to understand our rights and our restrictions when we incorporate open-source software into our products.

Using and incorporating these into your embedded Linux product is consistent with the spirit of the open-source community.

ISSUES WITH GPL-LICENSED SOFTWARE

A number of issues arise when you modify and distribute the source code of open-source software. If you are using the software for your company's own private use, you can modify it and are under no obligation to disclose your modifications. Should you modify it and distribute it, you are obligated to distribute your modified source code under the same license agreement as the GPL.

Example 1: You don't like the kernel's out of memory (OOM) logic (which is a major problem for robust embedded systems). You invest an immense amount of time and effort and create the coolest and most bullet-proof OOM logic for embedded systems ever devised. You put this into your product. You are obligated to publish this whiz-bang solution by distributing the source code to your customers under the terms of the GPL (which means they are free to modify and distribute it to whoever they like).

Example 2: Your hardware platform has some proprietary devices attached to it. The devices are a generation ahead of your competition. The device drivers you previously used on your COTS OS, if made public, would give your competition a four-year jump start enabling them to catch up to you in short order. If you build this device driver into the Linux kernel and delivered it as a product, you are obligated to release this driver code to your customer. And your customer could give it to your competitor.

Example 3: You are developing a system that uses a high-speed serial analog-to-digital converter (ADC) for which there is no Linux driver. You develop the new driver and incorporate it into the kernel. As with Example 2, you are obligated to release this source code to your customers.

Example 4: You are developing some application code that interfaces to a device that uses a CRC-64 algorithm. Rather than developing this from scratch, you see that it is embedded in the kernel with no external interface. If you lift this source code and place it into your application, you are obligated to disclose the source code of your entire application to your customers.

Example 5: You are using an LGPL library that uses a lot of dynamic memory allocations. You modify the library to incorporate Dmalloc into it to help you detect memory leaks. Under the LGPL, you are obligated to release the object modules to your customers (not just your binaries) so your customer could relink your modules with a different library!

HOW TO HANDLE THESE ISSUES

First and foremost, we have one person who is knowledgeable about the issues and enjoys keeping up with these licensing issues. He is the go-to guy when license issues come up.

Let's look at how we handle each of the examples.

OOM Example: Although we have not created a whiz-bang and robust OOM, should we do that, we would distribute that source code with our binaries under the GPL as we have other kernel modification that we have made.

Proprietary Hardware Example: Kernel drivers can be either embedded into the kernel or loaded from application space. It is our understanding of the GPL, that a loadable driver that uses normal kernel calls that is written from scratch is not a derived work and thus can retain its proprietary nature and does not need to be disclosed. If however, you modified the kernel and used some proprietary calls in this loadable module, it would fall under the GPL license.

New ADC Example: As I mentioned in last month's column, we try to use peripherals that already have Linux drivers. That is our first priority and is most cost effective. In other cases, we have chosen to release the source code as part of the distribution.

CRC-64 Example: We needed to incorporate a number of general-purpose check algorithms into one of our products. Although we found these algorithms in GPL licensed code, we basically looked and found the same algorithms implemented under Apache-style licenses. In another case, when we couldn't find something comparable, we bit the bullet and implemented the algorithm from scratch.

LGPL Example: We modified a library licensed under the LGPL for test purposes. However, we chose not to release

the modified library and kept the modified library only for in-house use because of the complications of releasing the object code of our proprietary application.

Finally, how does our company distribute the modified GPL software to our customers (who could probably care less)? We provide a link to the source code in the embedded webpage that is part of the configuration tools we provide.

WORKABLE BUSINESS MODELS WITH GPL

At the end of the day, if you are in business you need to make money. There are a number of ways individuals and companies can create a viable business model using Linux with its GPL. Here are just two.

For an individual embedded software developer, there are always needs for new device drivers for Linux. Individuals can pick certain pieces of hardware and create great drivers for these devices which then, once approved, can be included in the Linux kernel distribution. Your name would be forever linked to the driver (or perhaps that genre of drivers) and you could become the world's expert on these devices for Linux. Consider it advertising. There are a number of consultants who do just that.

For companies developing hardware systems, the Linux GPL model will somewhat alter how you develop your software. But with simple planning, you can keep your intellectual property private while at the same time utilizing all of the power of Linux. That is a good business deal.

PITFALLS TO AVOID

As system developers, we face many pressures. One of those pressures is to "pretend not to know" when it comes to software licenses. Avoid this if at all possible. Take the time to read the licenses. It is not rocket science. You will have the peace of mind that you did it right.

Another pressure is to incorporate just a little bit of copyrighted code into your application thinking, "No one will know. We are a small company. I am sure that I could find it somewhere else if I just took the time." You will know and if that is not enough incentive, then you don't understand the power of the conscience.

CONCLUSION

In the spirit of "Embedded in Thin Slices," I hope I have shared a little bit of what we have learned about licensing issues when using Linux in embedded systems. There are many possible scenarios that go beyond the scope of this column. I hope you can see that you can protect your intellectual property and still be faithful to the licensing requirements of Linux and its associated applications. Hopefully I have dispelled the notion espoused by one who should know better:

"The way the [GPL] license is written, if you use any open-source software, you have to make the rest of your software open source. … Linux is a cancer that attaches itself in an intellectual property sense to everything it touches. That's the way that the license works."

Where do they hire people to say such things?

The next column will discuss toolchains that can be used when developing an embedded Linux system.

Bob Japenga has been designing embedded systems since 1973. In 1988, along with his best friend, he started MicroTools, which specializes in creating a variety of real-time embedded systems. With a combined embedded systems experience base of more than 200 years, they love to tackle impossible problems together. Bob has been awarded 11 patents in many areas of embedded systems and motion control. You can reach him at rjapenga@microtoolsinc.com.

RESOURCES

Dmalloc, Debug Malloc Library, http://dmalloc.com.

GNU Operating System, "GNU General Public License, Version 2," 1991, www.gnu.org/licenses/gpl-2.0.html.

———, "GNU General Public License, Version 3," 2007, www.gnu.org/licenses/gpl-3.0.html.

———, "GNU Lesser General Public License, Version 3," 2007, www.gnu.org/licenses/lgpl.html.

———, "Frequently Asked Questions about the GNU Licenses," 2011, www.gnu.org/licenses/gpl-faq.html.

K. Yaghmour, *Building Embedded Linux Systems*, O'Reilly Media, 2008. Appendix C, Linus Torvald's Interpretation of the GPL for Linux.