Preserving Commercial IP in Open Source Embedded Systems

23-Jun-2006

Introduction

- Use of Open Source solutions have become very attractive to device manufacturers
  - The use and preference for embedded Linux is growing rapidly
- Use of Open Source operating systems in embedded system design brings with it some interesting challenges
- It is not in the spirit of true Open Source, but the reality is that typical product developments have IP to protect for commercial reasons
- This requires special attention to the licensing conditions related to use of the Open Source software
- There are choices to be made when designing embedded systems which are generally considered to limit the viral nature of GPL and preserve critical product specific IP

IANAL

(I Am Not A Lawyer)

This chart package contains general guidelines and suggestions regarding Open Source Software for Embedded Systems, but it is highly recommended that you consult your lawyer if you have open-source copyright concerns

Evolution in the Embedded World

- In the past, "embedded" referred to relatively small systems with few characteristics of a "computer"
  - Mostly hardware product with tiny amounts of memory and small but very efficient programs
  - May not need to communicate with other programmed devices
  - If there was an OS at all, it was lean and not very feature-rich
- Embedded projects today are more sophisticated, both in terms of features and underlying hardware
  - Most embedded systems require some type of connectivity, such as Ethernet or USB
  - Cheaper / faster processors & memory are available
  - Trying to write everything from scratch is not feasible so software reuse is more necessary than ever
  - Many of these components now exist in the public domain and in the open source community

Why Linux?

- An open-source operating system such as Linux offers security, reliability, adequate performance, vendor neutrality, reasonable quality, and a royalty-free business model
- Linux first attracts designers because it is free to download, it comes with the source, and is compatible with a wide range of processors
- A Linux operating system gives immediate access to many useful features that would be difficult to provide with a lower order OS
  - there is a rich set of drivers and other features which add to the value package
  - by choosing the right packages, it is possible to easily add other higher order functionality

Is Linux "Free" Software?

- From the Free Software Foundation website
  - ``Free software'' is a matter of liberty, not price. To understand the concept, the FSF suggests we think of ``free'' as in ``free speech,'' not as in ``free beer.''
- Free software relates to the users' freedom to run, copy, distribute, study, change and improve the software, according to the following:
  - The freedom to run the program, for any purpose (freedom 0)
  - The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this
  - The freedom to redistribute copies so you can help your neighbor (freedom 2).
  - The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

Considerations when choosing Linux

- Despite good reasons for choosing Linux for an embedded system or a device, Linux does have problems in the areas of licensing
  - Linux is not public-domain software
  - It is licensed according to the GNU General Public License (GPL), which has a strict set of rules for use
- Embedded system design is especially susceptible to GPL concerns
- Protecting custom application code is generally important in the real commercial world
  - The ongoing legal ambiguity surrounding the GPL potentially impedes choice of Linux for some embedded developments
- However, the reality is that Linux is being used for more and more embedded applications
  - About 20% of embedded systems use Linux today & growing

What Licence?

- There are a plethora of licenses in use for open source today
  - In December the Open Source Initiative recognized 60 different Open Source licenses in use
  - On Sourceforge.net alone a recent count showed more than 51K projects identified as being licensed under the GPL
- Most components in a Linux system are covered under GPL or LGPL
  - GPL tends to be used more for applications
    - Kernel, binary utilities, GCC compiler, gdb debugger are all licensed under GPLv2
  - LGPL tends to be used more for libraries
    - C libraries are licensed under LGPL

The problem

- What do the licensing terms of the GPL mean to the embedded Linux developer?
  - If not careful a designer may unintentionally give up their rights to proprietary software
- Results of a recent survey by the Venture Development Corporation (www.vdc-corp.com) showed the following:
  - Nearly 60 percent of embedded engineers work for companies with fewer than 10 software developers
  - More than 75 percent of respondents replied that development teams at their company typically employ four or fewer software developers per project
- Result is that embedded Linux designers are being asked to balance a variety of legal obligations without really understanding the terms of the licenses they are using

License Intent

- In general the open source licenses are intended to protect the author's copyrights while providing for freedom of use
  - The important thing to understand relates to the difference between running GPL software and modifying GPL software
- A derived work combining GPL software and non-GPL software can only be distributed under the GPL
- The copyright conditions of the license make no distinction between static and dynamic linking of modules
  - Some people talk about being able to “work around” the terms of the license by dynamic linking
    - This is NOT a valid strategy and does not eliminate the obligations of the license

GPL Overview

- Linux kernel is licensed under the terms of the GNU General Public License (GPL-V2)
  - Source code for any software under the GPL is free
  - Any modifications to software under a GPL license automatically come under the GPL as well, if the modifications to the GPL code are redistributed
  - You can't sell source code for GPL code for more than the cost of the distribution media and cost of distribution
  - If run-times are distributed, the source code must be made public and readily available upon request.
- If you take the Linux kernel or any Linux utility and modify it, port it, or add features to it, must make the source available to anyone who asks for it.

Embedded Linux Strategies

- Although Linux was not designed with embedded systems in mind, through careful configuration and by leveraging some of the work of the embedded Linux vendors, Linux can be successfully embedded in hardware products
- Application programs that run under Linux may remain proprietary
  - Ensure no part of the application is GPL (or other open source license) code
  - If modifications to the Linux kernel, libraries, or utilities are made, the sources for the modified version must be made available
- Device drivers can also be kept proprietary
  - They must be shown to be separate and distinct from the kernel
  - This is most easily done if the device driver is written as a loadable kernel module
  - If it is linked as part of the base kernel it is in the gray area of whether it becomes GPL or not

GPL and Proprietary Code

- GPL software in proprietary system - Free Software Foundation view
  - You cannot incorporate GPL-covered software in a proprietary system. GPL grants everyone the freedom to copy, redistribute, understand, and modify a program. If you could incorporate GPL-covered software into a non-free system, it would have the effect of making the GPL-covered software non-free too.
  - A system incorporating a GPL-covered program is an extended version of that program. The GPL says that any extended version of the program must be released under the GPL if it is released at all.
  - However, in many cases you can distribute the GPL-covered software alongside your proprietary system. To do this validly, you must make sure that the free and non-free programs communicate at arms length, that they are not combined in a way that would make them effectively a single program.

Linus Torvalds input

- The GPL as clarified by Linus Torvalds. The Linux kernel is licensed under the GPL, but with this clarification:
  - “This copyright does *not* cover user programs that use kernel services by normal system calls -- this is merely considered normal use of the kernel, and does *not* fall under the heading of 'derived work.' ”
- This statement clarifies that user-space programs are not considered to be derived from Linux
  - It is also interpreted as allowing proprietary kernel modules

How do GPL & Proprietary Co-exist?

- When can it work?
  - The difference between “co-existing” and "incorporating" the GPL covered software is partly a matter of substance and partly form
    - The substantive part is this: if two programs are combined so that they become effectively two parts of one program, then you can't treat them as two separate programs, so the GPL has to cover the whole thing
  - If the two programs remain well separated, like the compiler and the kernel, or like an editor and a shell, then you can treat them as two separate programs – but you have to do it properly
  - If people were to distribute GPL – covered software calling it "part of" a system that users know is partly proprietary, users might be uncertain of their rights regarding the GPL – covered software
    - If they know that what they have received is a free program plus another program, side by side, their rights will be more clear

LGPL

- "Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.
- In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License."
- "A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.
- Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with."

RTAI (GPL vs LGPL)

- Since the RTAI developers intend for RTAI to be as widely available as possible, the license was changed (from LGPL to GPL) to free RTAI users from the threat of legal action (due to the RTLinux patent) that they may encounter as a result of using RTAI
- The shift [in RTAI core license from LGPL to GPL] was made official with the release of RTAI-24.1.8 and was announced also on the Linux kernel list. The announcement reported Eben Moglen's statement, which can be read in full on the RTAI website
  - "No application in a running RTLinux or RTAI system does any of the things the patent claims. No applications program is therefore potentially infringing, and no applications program is covered, or needs to be covered, by the license. If an applications program running in a GNU/Linux system as modified by RTLinux or RTAI is constructed so that it does not violate GPL -- which is not the same thing as being licensed under GPL or being free software at all -- its distribution terms are utterly irrelevant."
- Moglen's statement attempts to make it clear that the RTAI core being GPL does not affect application code

Common Usage

- Examine the RTAI patches - it is not a single block of code; it is a kernel patch and a number of modules
  - Parts that potentially may be claimed to implement aspects of the RTLinux patent are licensed under GPL, the rest remain under their original LGPL license
  - RTAI can be viewed as just as a part and extension of the Linux kernel. Any proprietary, binary-only RTAI application module is not at all different from any "normal" binary-only Linux device driver module
- In reality, there is little difference between GPL and LGPL when it comes to kernel modules, as it has become an accepted fact that using an API implemented in a GPL kernel module (or in the kernel) does not result in contamination that would force your code to be released (ie published) under GPL
- If you use RTAI in your product, you must make available to your customers the source code for RTAI and any modifications you have made to it
- Accepted industry practice implies that because of the implementation of RTAI as an extension to the Linux kernel, applications that employ it need not be released with a GPL license

What changes with GPL v3?

- Language in the draft of GNU GPLv3 (General Public License - V3) would require embedded systems and devices incorporating GPL-v3 licensed software to be user-modifiable
  - This interpretation is based on statements made in April at a GPLv3 conference speech by license author Richard Stallman
- GPLv3 debate – Linus Torvalds insists that Linux kernel will remain under V2
  - The Linux kernel has _always_ been under the GPLv2. Nothing else has ever been valid. ... Conversion isn't going to happen.

GPL Considerations Summary

- If you modify and distribute GPL software, your modifications will fall under the GPL, and you must give the source code to anyone who asks for it
- Application programs and device drivers may remain proprietary as long as they are separate and distinct from the Linux kernel and contain no GPL code
- Preserving this code isolation is a constant source of anxiety among embedded systems developers

Strength in Numbers

- Despite the licensing concerns, the growing popularity of Linux for embedded system development has convinced many commercial RTOS vendors to join the open source movement by providing tools and support for Linux. A few examples are:
  - LynuxWorks (BlueCat Linux vs LynxOS)
  - WindRiver (Red Hat Linux vs VxWorks)
  - ENEA Embedded Technology (partnered with Metrowerks – CodeWarrier tools)
  - TimeSys (TimeStorm development and testing environment)
- Four mobile phone vendors, together with two major wireless operators, are creating open Linux implementation for mobile phones. Motorola, NEC, NTT DoCoMo, Panasonic, Samsung, and Vodafone say their Linux implementation will provide a global standard, and prevent the "fragmentation" of mobile phone Linux
  - DoCoMo, Japan's largest mobile phone company, adopted Linux for 3G phones
  - Motorola, was among the first phone vendors to include Linux on its strategic road map, and it has since shipped several Linux-based phone models
- This suggests that Linux continues to deliver on the promise of vendor neutrality, and that embedded Linux technology remains adequately decoupled from the fortunes or failings of any single company or organization

Some Useful Links

- www.fsf.org - Free Software Foundation
- www.opensource.org - Open Source Initiative
- www.groklaw.net - information on the legal front
- www.netrino.com/Articles/LinuxLaw/index.html - useful article
- http://safari.informit.com/0130476773 - a useful book reference
- www.gnu.org/licenses/licenses.html - GNU licence information

Conclusions

- Although Linux was not designed with embedded systems in mind, Linux can be successfully embedded in hardware products
- Linux does have some potential problems in the areas of licensing
  - Linux is not public-domain software; it is licensed according to the GNU Public License, which has a strict set of rules for use
- It is important that as a designer of an embedded system, you know which license applies for the components you are using
  - Take the time to understand the implications of the license
- However when considering one of the main criteria for choosing an embedded system—availability of hardware and availability of third-party software – Linux has a lot going for it

The Business Accelerators

Creating Competitive Advantage

www.businessaccelerators.ca