WIND RIVER WHITE PAPER

WIND RIVER

Linux In Embedded Systems: Where Are The Benefits?

If GPL issues are somehow manageable, device makers might still figure they can reduce software costs with linux by relying on the linux community for support. Unfortunately, that appears unlikely at this point. This is not due to a lack of interest or passion on the part of the linux community, but because there is no common platform to rally around.

Jerry Fiddler
Chairman and Co-founder
Wind River

1. Introduction

In a remarkably short time, linux has brought new attention to open-source software and influenced the strategies of some of the world's biggest software companies. Wind River is not one of those giant software companies (at least not yet), but we are the largest provider in a significant part of the global software market, which means we have to pay attention to linux and open-source software, too. Our customers make the many millions of non-computer products that rely on microprocessors for essential functionality. We call these products "embedded devices," because the microprocessors that run our software are embedded or built into a device. People don't think of these products as computers, which is the main distinction between our customers and companies who make PCs and servers. Their customers do things like start up Windows, launch programs, and run spreadsheets. The customers of device makers do things like make a phone call, fly a plane, and guide remote-control robotics systems.

The companies that make the phones, planes, robotics systems, and a great many other products certainly need good software; they spend billions of dollars every year to acquire and develop it. The reason they buy software from us is that our products and integrated platforms enable them to create and implement their software faster, less expensively, and at lower risk. This, they tell us, is our big contribution to their business: not our software per se, but the head start, the reduced cost, and the reliable support that we provide them so they can get their end products to market faster, offer more features for less money, and develop follow-on products more efficiently.

Our customers in the device world are constantly looking for any edge they can get, and they have seen linux make rapid inroads into the network server market. Many of them, after all, are large companies whose MIS departments are either evaluating or using linux in their corporate networks. The product developers at these companies hear that linux in the server world is free, that it's well supported, and that it is highly reliable. They want to know if linux can provide these benefits in embedded devices. We had the same questions at Wind River. We were founding members of the Embedded Linux Consortium, and we developed a complete linux version of one our major software tools as well. Based on this experience and many conversations with customers, it seems worthwhile to offer some observations on linux and whether it can provide meaningful benefits in embedded devices.

2. Linux is Open-Source Software

Linux is just one member of a larger category, called open-source software, which has been around for a long time. Perhaps the best-known open-source program, and one of the longest-lived, is the UNIX operating system. Open-source software comes with a set of principles, rules, and traditions that challenge software companies and their customers in an interesting and worthy way.

> "Due to the structural differences in the two fields, linux is not likely to simply transfer its server success into the embedded device arena. Instead we have to assess linux on its own merits relative to the advantages that device makers are looking for."

Properly speaking, linux is a kernel, created and evolved by Linus Torvalds. A kernel is not an operating system, as Torvalds and others have taken pains to make clear. The linux kernel is architecturally similar to the kernel of UNIX; it includes the core processes and event handlers that enable the middleware and applications wrapped around it to do what they do. When people call linux an operating system, they usually are referring loosely to a package that includes the kernel and a number of essential functional building blocks, such as Sendmail, Apache, TCP/IP, and various application program interfaces (APIs) that enable the kernel to process information and communicate the results. Following common usage, I'll use "linux" to mean an operating system, based on the linux kernel, which also includes essential APIs and other applications that customers expect and require.

In some ways I'd rather not be discussing operating systems (much less kernels) at all, because the focus at Wind River is on building development platforms that subsume operating systems into higher-order, integrated solutions for customers. This is a major trend in our industry, and we expect it to expand considerably in the years to come. On the other hand, linux is being proposed as one of the operating systems we should consider building into our platforms (which also include middleware, APIs, tools, and other elements). So we need to take a good long look at linux with our customers in mind.

3. Benefits Associated with Open-Source Software

There are several areas where open source software seems, at least at first glance, to have some promise for builders of embedded devices.

Low Acquisition Cost. The main calling card for open-source software among corporate customers is that there is almost no cost to acquire it – you can readily download linux from the Internet, for example – and you can legally use it without paying anyone for it. These two qualities have led people to say that open-source software is "free," however, free does not mean there are no consequences or limitations. Open-

SOFTWARE DEVELOPMENT COSTS

Here are some of the major cost categories relating to software development. Note that many of these are human-based, labor-intensive activities that require people with significant experience. In other words, programmers and their support teams are expensive. Even if linux enables you to start without the first two items on the list, it's hard to avoid the others.

- **Vendor & technology selection**
- **Acquisition costs**
- **Integration costs**
- **Development costs**
- **Testing**
- **Training**
- **Documentation**
- **Manufacturing**
- **Maintenance**
- **Derivation for follow-on projects**

source software comes with a license, like other software, and users are obligated to follow its terms as they would with any other program.

The license for most contemporary open-source software is known as the "GPL," which is short for "GNU public license." This license requires, among other things, that if you create programming based on software you got from the open-source community, then the modifications you make or the software you create have to go back to the community so everyone else can use them also. This sounds fairly straightforward, not to mention equitable, but as I'll discuss later, it is anything but clear when companies apply it in actual commercial products.

Unpaid Community Support. Another attractive quality of open-source software – particularly for companies who now pay significant sums for software upgrades, testing, porting, and other activities – is that the open-source community does most of these things for free. An international network of software programmers collaborates together voluntarily, out of a sense of mission, to upgrade and enhance the software. Different pieces of software are handled differently by the various communities devoted to them, and the differences can be fairly wide. The UNIX community, for example, has had more than 30 years to settle on stable processes and procedures for adding new capabilities and upgrading the existing ones. The linux community is much younger, and is still in the early stages of sorting out how the program will evolve.

Intangible Benefits. Open-source software also has a number of less tangible qualities, which are perhaps more important to programmers personally than to the companies that employ them. For instance, there's a lot more room to be creative and exploratory with open-source software than with most proprietary programs. There's also an international, collegial feeling to the open-source community, which crosses borders freely on the Internet.

4. Benefits Associated with Linux

Linux has all the qualities of open-source software described above. It also has the additional attribute of real-world success in the server environment. Linux has cracked the corporate network server market so effectively that it is replacing network server operating systems provided by such well-known, well-regarded companies as Sun Microsystems, IBM, Hewlett-Packard, and Microsoft. In addition to low acquisition cost and outside support, linux also offers high throughput, ready accommodation of prevailing communications standards and interfaces, and reliability in running mission-critical corporate applications. In other words, linux is a technological match for the competition in the network server operating system environment.

One sign of linux's status in the network server market is that it is already being exploited for advertising and marketing purposes, in a way that the open-source community probably never imagined. Oracle, for example, has run ads calling linux "unbreakable," as part of its campaign to win business against its biggest competitor, IBM. IBM offers an operating system optimized for its own database and application programs. Oracle offers a database and applications, but no operating system. To offset this potential disadvantage, Oracle is pushing linux as better than other operating systems, specifically including IBM's. Note that in this case, a vendor is promoting linux because it serves a competitive purpose, not necessarily because it provides advantages to customers.

So let's summarize what customers want when they say they're interested in linux. First, they want an operating system that has little or no risk associated with the cost – it's free to start with and future improvements will be free as well. Second, prospective linux users want software that is robustly supported by an external, community of smart, passionate people. Getting the first two qualities implies a third:

technological suitability for the job at hand. If linux was not a success in the server space, I would probably not be writing this paper and you would probably not be interested in reading it.

5. Embedded Devices: Can Linux Duplicate Its Server Success?

The success of linux in corporate network servers seems at first blush like a model for success in embedded devices. After all, embedded devices need some of the same technological attributes from their operating systems that network servers do, including high reliability and built-in readiness for communications. Furthermore, markets for many embedded devices, particularly consumer digital electronics and network infrastructure, are just as competitive as the network server market. Device makers are always looking for ways to reduce costs, and software is one of their biggest investments in the R&D area. A careful analysis, however, shows that the two environments – network servers and embedded devices – are actually very different when it comes to operating systems. I'll touch on a few of the differences briefly, before assessing linux's suitability in embedded systems.

Hardware Architecture. There has been a big push in network server technology to minimize investment risk through hardware standardization. There are therefore, only a handful of chip architectures in all the servers offered by IBM, HP, Sun, and other server manufacturers. Furthermore, the surrounding hardware architecture is evolving slowly, with the implicit cooperation of vendors and buyers, to keep things stable and predictable for corporate customers.

The device market could hardly be more different. Customization is the rule, with numerous microprocessor architectures, systems-on-a chip, and field-programmable gate arrays (FPGAs) proliferating rapidly. Customization is vital for cost and efficiency. To succeed in the market, each new product needs to better its competitors and predecessors for cost, power consumption, functionality, and physical size. Wind River alone has 4,000 customers involved in this Darwinian struggle. It's hard to imagine General Motors designing a new server for its corporate network, but GM is actively pushing its electronics suppliers to come up with smaller, faster, lighter, lower-cost, and less power-hungry embedded-system designs for engine systems, braking systems, and cabin comforts – each and every model year.

Software Interoperability. Servers and embedded devices also differ dramatically in software interoperability requirements. Most servers can run any of the major server operating systems, and most application programs can run on all the major operating systems as well. This is a big reason linux can move in and take market share: the server environment has developed in such a modular, interoperable way that a new operating system can be plugged into it without disrupting overall functionality. Again, it's almost the exact opposite in the device world. Integration of microprocessors, operating systems, and application software is now so tight that few elements are interchangeable once a product is fully designed and developed. This tight integration is not an accident or the result of poor design; it's intentional, aimed at achieving the highest possible level of operating efficiency.

Design Goals. In corporate server networks, disk space and processing cycles are relatively abundant, so there's not much incentive or upside to slimming down the operating system. In fact, it's a good idea to just throw in anything and everything that any user might ever want, so it's already there when they want it. In embedded devices, designers take the exact opposite approach. The question is not "did we leave anything out?" but "what can we get rid of?" There is simply not enough physical space, memory, or power to include anything that's not essential to a given device at a given price point. Linux has been a good fit in the server space because the community has loaded it up with everything except the kitchen sink, and developers are creating more modules and components and add-ons ("bloatware") all the time. The device market is another world entirely, where adding one thing often means eliminating one or more others.

> "Commercial software companies are learning from the open-source model. So if linux itself is not the best answer for device makers, their existing suppliers can still offer some of the qualities of the open-source experience."

Strategic Importance of the Operating System. In corporate networks where linux is doing so well, the choice of a server operating system is rarely central to a company's mission. Network processing cycles are a utility, like electricity and plumbing. Such utilities are basic to business, of course, and we miss them when we don't have them, but everyone expects them to be provided so they can do what really matters to them. For device makers, what matters is differentiating their products in competitive markets, and extending the life-cycle of successful product franchises as long as possible. Software is a key determinant of their success in both activities.

Consider the difference between a mobile phone selling for $100 and one selling for $200. The physical units may look nearly identical, but the more expensive phone must provide clearly superior functionality. The upgrade path between the two phones will be

defined largely by capabilities achieved as much with software as with silicon. In other words, software is a key link in the device-maker's value chain. Network server operating systems are not a key link in the value chain of, say, Sony. So the risk associated with a adopting a new operating system for their products is in a completely different category for device makers than it is for MIS departments exchanging operating systems for network servers.

6. How Does Linux Stack Up In Embedded Systems?

Due to the structural differences in the two fields, linux is not likely to simply transfer its server success into the embedded device arena. Instead we have to assess linux on its own merits relative to the advantages that device makers are looking for. Referring back to the first section of this paper, those potential benefits include low cost, low risk, and technological suitability.

Software Is Not Free. "Open-source is free only if your time has no value." That's the phrase of Jamie Zawinski, the principal architect of the Netscape browser, which was eventually open-sourced. In other words, you may be able to get a piece of software for nothing, but everything you do with it after that event costs you something. Our customers certainly know how high those costs are, and we've summarized some of the categories in the chart on page 3. You can see that most of the costs arise after a company actually licenses the software. They include integration, development, testing, training, documentation, manufacturing, maintenance, and derivation for follow-on projects. These costs are substantial, which is why our customers report that most of the tens of billions of dollars they spend on R&D each year are for software. Open-source software is not going to make most of these costs disappear, and may in fact increase them. So far, the only companies making money from linux are those charging people for supporting it. Before Cygnus Solutions was acquired by Red Hat, its tagline was, "We make free software affordable."

There May Be Hidden Costs. Beyond the real costs of life with software, there appear to be significant hidden costs lurking in linux. As mentioned earlier, the GPL requires that people who use the linux kernel to run their own software must make that software open and available as well. A key passage in this reads as follows: *You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.*

This language creates a potentially huge intellectual property problem for makers of embedded devices. A reasonable interpretation is that all the software contained in a device, if that device "in part contains" any GPL-licensed code, must be "licensed as a whole, at no charge... under the terms of this license." This is particularly crucial for device makers, who license all their software in their devices as one bundle ("the ROM in the phone"). If part of the software is covered by the GPL, some would argue, then it's all subject to the GPL and must be given to the open-source community.

In a recent article in *Embedded Systems Programming,* David Beal and Michael Barr call this potential liability in linux "a myth." But as far as I can see, they fail to make their case. They certainly do not adequately address the issues raised by the phrase "licensed as a whole." Instead they suggest that the GPL doesn't affect new, proprietary code if a device maker doesn't touch the linux operating system and just runs the proprietary software alongside it. This side-by-side approach may be an acceptable interpretation of the GPL in theory, but it's hard to apply it in practice. Device companies have abundant reasons to modify linux and to integrate it with firmware or application code, for reasons of functionality and performance. The upshot is, their end products would always "in part contain" some GPL code, which in turn would trigger the license.

Even if device makers use the side-by-side interpretation of the GPL, and don't modify linux itself, they're still vulnerable. In fact, unanticipated liability is probably the biggest potential problem with the GPL for device makers, because GPL licensing is viral. It spreads, and it infects what it comes in contact with, specifically any software that may be bundled with GPL code in a product. As a result, a high level of vigilance is required to keep track of what is open source, what is not, and what's in the gray area in between. In today's world, with so much outsourcing of product components to suppliers, it's hard to track every last thing that comes into your own product at assembly time. A bit of code that violates the GPL could delay release of a product, or worse. We often hear from our customers on this issue: They are concerned about "GPL-ed code" coming to them from other suppliers.

At Wind River, we did full linux development of a Tornado toolset, with some modifications of the operating system to make it more suitable to embedded applications. We then decided not to release it because it would push our customers into the gray area. Our job, they keep telling us, is to simplify and accelerate their efforts to create new and more successful intellectual property. We didn't see how we could say to them, "Here's this really great toolset, and it works with linux, but you may be compromising the intellectual property you create with it."

External Support May Not Be Available. If GPL issues are somehow manageable, device makers might still figure they can reduce software costs with linux by relying on the linux community for support. Unfortunately, that appears unlikely at this point. This is not due to

a lack of interest or passion on the part of the linux community, but because there is no common platform to rally around.

The fact is that embedded devices have critical requirements with respect to real-time operation, footprint, speed, and other parameters, and to make linux work in this environment it must be customized, sometimes significantly. This gets pushed even further when devices are designed with a high level of integration among hardware, software, and firmware. The result is that each customer potentially has a variant of linux that the community has never seen and cannot effectively support. Multiply this by the number of products in the embedded arena and you could soon have thousands of potentially tweaked and tuned versions of linux that no one outside a device maker's software team has ever worked on, and perhaps even numerous versions within a single company, created and tuned for different products by different teams. The support burden necessarily falls back on the device companies themselves.

The specific support problem is exacerbated by a general evolution problem. The open-source community has not yet established order in the linux development process, so at this point there is rigid control of the kernel, contrasted with loose control of everything else. Torvalds has reserved new releases, of the kernel to himself. In contrast the rest of the community appears to be exploring multiple versions of even fundamental elements such as the TCP/IP implementation. There is no formal source-code control, and no configuration management scheme in place. There is no road map, because there is no one to create it, except sporadically on a module-by-module basis.

So for now, getting timely support for embedded linux almost certainly means hiring people on staff or paying someone to do it for you – if you can find people with the right experience and aptitude. Outsourcing seemed to be the favored solution to the linux support problem, and a number of companies sprang up to charge customers for embedded linux support. (Remember, they're making "free software affordable.") Now, it appears that most of these companies have been consolidated out of existence or are retreating from the market. Certainly no viable business model has emerged for commercial support of linux in embedded systems, leaving device makers largely on their own.

Technical Challenges Remain. The third main criterion regarding linux, as far as device makers are concerned, is that it must do the job in their products. There are as yet very few products running embedded linux, so it's hard to generalize about technical performance, but we can make some observations in a few key areas:

- *Real-time processing.* Linux is not a real-time operating system, which presents an immediate technical challenge to the many device makers who need real-time processing. To increase its response time, some customers tell us that they're "running linux under a real-time kernel" but as we've already noted, linux *is* a kernel. So apparently these manufacturers are actually running some hybrid form of open-source software. Other device makers modify the linux kernel to make it run faster, though this may undercut some of linux's other advantages. Once you start modifying any operating system, you're committing to an investment in programming and support resources that generally can't be provided by someone else. The free software you started with becomes a lot less free when you need to do all of your own support and maintenance.
- *Efficiency.* Device designers excel in creating products of increasingly rich functionality from rapidly shrinking budgets for size, weight, power, and cost. Semiconductor manufacturers are doing their part to make ever-more-efficient products possible, and software vendors are expected to do their part as well. In a server, a larger memory footprint or a slightly less efficient operating system is not a big deal. In an embedded device, operating system efficiency can determine how powerful the CPU must be, and footprint can determine how much memory is required. Both translate directly into cost, battery life, functionality, and ultimately the product's market competitiveness. At Wind River, we work constantly on making our proprietary operating system, VxWorks, do more with less. Recently one of our wireless device customers was preparing to develop software to run under linux until it became clear that once they got the operating system into shape to do the job, it would eat up 2 megabytes of the product's available memory. The comparable configuration with VxWorks was 250 kilobytes. The customer needed to conserve memory to hit its price and functionality targets, which made the "free" software too expensive in memory terms.
- *Start-up Time.* A large percentage of devices with embedded systems are supposed to respond instantly to user commands. Programmers who come into the device world from the PC or server world are sometimes surprised to discover that unless an embedded system runs on a standard Intel architecture, fast booting from linux is a major challenge. Device makers also work hard on solving the problem of how to initialize, manage self-test routines and bootstrap non-Intel architectures, and linux does not offer any help in these areas.
- *Tools.* From what we can see, there is a serious lack of development tools for embedded programming with linux. Developing, testing, and debugging production-quality code simply takes too long when you don't have the right tools for the job, and the tools required for embedded development (such as appropriate cross compilers and debuggers, and real-time analysis tools) are very different than those that are appropriate in linux's more traditional

markets. Not long ago one of our customers, a company known for its outstanding product design, was considering linux for a wireless networking device. But when the company realized it would take most of a year to develop the product to run under linux, it came to Wind River instead and got the product to market in just four months.

7. So What's A Device Maker To Do?

Some of our device customers have thought through most of the foregoing issues and have come to us with a question: "How can we get the benefits of something like linux without losing the benefits of what we get from you?" One of the first things we do is ask them if they really would accept "something like linux " rather than linux itself.

If device makers need a non-real-time, UNIX-like operating system, UNIX may be a better solution. UNIX is technically more advanced than linux, and has a much longer heritage of commercial application. A community-supported version of UNIX, called FreeBSD, is available for very low cost evaluation. Customers with specific plans can also take advantage of BSD/OS, a fully supported version of UNIX offered by Wind River, which comes with a full set of tools and clear development roadmaps. Both BSD/OS and FreeBSD are free of GPL-related intellectual property issues.

The rest of the answer to our customers' question is that commercial software companies are learning from the open-source model. At Wind River, we are now giving away more source code with VxWorks than we ever did before, and we are offering more development tools online. The idea is to give customers some of the underlying benefits of the open-source approach to software and software development. We want to do even more, to the extent we can sort out the intellectual property topics, customer privacy concerns, and other issues. So if linux itself is not the best answer for device makers, their existing suppliers can still offer some of the qualities of the open-source experience.

In the end, it's clear that most of our customers want rock-solid software that combines a true real-time operating system, middleware, utilities, and applications tuned to their products or industries. They want all these things integrated into a platform that accelerates their development work, at a price they can plan around, with a predictable technology roadmap, from a supplier who will bend over backward to make them happy. It's perhaps possible that linux may evolve to offer device makers these fundamental business benefits some day, but today, and for the foreseeable future, its disadvantages clearly outweigh its attractions.

WIND RIVER

Wind River Worldwide Headquarters
500 Wind River Way
Alameda, CA 94501 USA
Toll free 1-800-545-WIND
Phone 1-510-748-4100
Fax 1-510-749-2010
Inquiries@windriver.com
Nasdaq: WIND

For additional contact information, please see our Web site at www.windriver.com.

Wind River, the Wind River logo, Tornado, and VxWorks are registered trademarks of Wind River Systems, Inc. Any third-party trademarks referenced are the property of their respective owners.

For further information regarding Wind River trademarks, please see:
www.windriver.com/corporate/html/trademark.html

©2002 Wind River Systems MCL-WP-LNX-0210