Is an Open Source Operating System Really Free? Not by a Long Shot

Tom Williams
Editor-in-chief

RTC Magazine

January 05, 2015

Editorial Summary

Words like “open source” and “free” definitely have an allure when one is trying to reduce costs and get a vital product to market. And indeed, the Open Source Movement has done great work in producing highly functional operating systems such as Linux ad well as a number of less well-known real-time operating systems. But before being seduced by their initial low cost, it is important to consider what the real cost may be in terms of verification of reliability, maintenance and the time and manpower required for accomplishing these in the face of real product deadlines.

The Open Source Movement has indeed contributed greatly to the development of sophisticated operating systems, the most well-known being Linux. The ability to download, experiment and contribute freely has led to many significant innovations. But as significant as this is, it is only a foundation. When you integrate and OS into a product, especially an embedded system, you have to be certain that it functions as expected, is free from defects and is reliable. In other words, when choosing open source, the burden of verifying these things is on you and your operation alone.

Today’s modern operating systems are built from various components, not all of which were created by the same programmers, but which are built into the OS that eventually gets integrated into a system. For a commercial product, these must be verified to all work together reliably. This takes time and manpower, which both equal cost. In addition, the type of skills required for testing and verifications are specialized and may not be those needed by the core purpose of a given company. Then there is the additional need for testing and verification tools—another additional expense.

Then there is the need to maintain the OS and perform updates that may be advantageous to the product from the vendor’s perspective or requested by the customer. Again, this means more time and money and more testing. Maintenance goes hand in hand with support. Of course a company will support its core products but must also support components in the form of an OS that are not central to their value-added.

Contents

If you are considering using an open source operating system, or any operating system, there will be expense involved as well as time. In this case, it is definitely advisable to consider the offerings from a number of companies who have already gone to the expense and effort to acquire, build and verify open source operating systems as well as modify and support them as commercial products. In other words, they have gone through all the steps that would be required to use such an OS from scratch and can be engaged as partners for companies needing a verified, reliable and supported operating system as a foundation for their own added value in the market.

The Open Source Initiative (OSI) and related groups represent a positive and often idealistic aspect of the software industry. The author and copyright holder makes his or her source code available with a license that grants the rights to study, change and distribute software to anyone for any purpose. The ethics of the open software movement require that any derived or modified code also be put back into the open source community for use by others. One of the requirements for being able to use and modify open source code is that such derived or modified code also be open source. This arrangement has had a hugely positive impact on innovation: Free code in a growing and innovative community.

Let us look for a moment, however, at just what "free" means. The first rule of the Open Source Initiative's Open Source Definition reads, "The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing other programs from several sources. The license shall not require a royalty or other fee for such sale." First of all, we are primarily talking about free source code that can also be distributed in compiled form. So it is certainly possible to download a working operating system for free. The restrictions are that you cannot then turn around and sell it or charge royalties for its use, even if you modify and improve it.

So there are definite advantages. A small business startup could, for example, install it on its office computers and use it internally for no cost. Of course, most computers today come with a pre-installed operating system—usually Windows or Linux—which is included in the price and supported by the manufacturer. Such products, however, certainly do not have software that is open source. It is a proprietary product, usually licensed by the computer manufacturer from a specialized software vendor.

Things become somewhat murky when open source code has been modified, enhanced and integrated into a hardware product that is then sold to customers. Is this done? Undoubtedly. Is it fair and legal? Probably. If the seller's value-added component is the functionality of the proprietary hardware, its applications and possible enhancements to the OS, then there is at least a case to be made that the (compiled) open source code is simply supplied at no extra cost. The OSI rule quoted above seems to state this explicitly. This is not offered as a legal opinion but simply an attempt at positing a rationale for using open source software in, say, an embedded product. The question then is, for whatever purpose—internal cost-saving or an embedded component— is this "free" software really free, or is such use even worth the effort?

What Is Involved with Using Open Source?

Consider what we currently expect from software, be it a commercial product or open source. We quite justifiably expect commercial software to work as advertised. That is less of an imperative with open source, since it comes from an open development community and is constantly being modified, enhanced, added to and critiqued. So let's start with expectations. While there are a number of open source operating systems to be considered, the most prominent is Linux. Therefore, for the purposes of this article, any specific examples or mentions will be of Linux.

First, when you download an operating system, what do you want it to do? Can you be sure it has everything you need? If not, do you write code for the additional desired functions or do you obtain them from other sources? If the latter, do you know that all the components will work together? How do you know? Has all this code been tested? How and by whom? Has the code been documented, and perhaps more importantly, does it have sufficient internal comments to let someone unfamiliar with it find his or her way through it? Is there anyone you can call and ask about any problems you encounter? All of these basic questions are signposts that could spell disaster along the way if not carefully considered and answered in advance.

It should be pointed out that there are a number of successful companies that have closely addressed these and other related issues and are now offering tested and documented operating systems (including both desktop and embedded versions of Linux) that they support. In other words, they have gone through the agonizing issues listed above and have produced commercial products that, while perhaps initially based on open source code, reflect a very large investment of time, money and effort on their part to offer commercial products that contain a significant percentage of their own code and which they support. Needless to say, these products are not free.

The availability of proven operating systems based on open source would seem to imply that an organization deciding to start from open source to build an OS that would either be offered as a commercial product itself or embedded into a system or device to support application functionality would be at best making a duplicate effort. The big difference is that with rare exceptions, commercial software products do not provide their own source code. So unless some deep tweaking will be required to optimize some specific application level function, there seems to be a number of good arguments for taking the alternate route. This would then come down to the classical "build or buy" decision.

What, exactly, is the added value that your product or service is offering to customers? As we all should know, that is where you need to concentrate your time, effort and money. Still, you cannot afford to ship a product with an underlying operating system in which you do not have full confidence. To gain the necessary level of confidence in an open source OS, you must be willing to employ a crew of software specialists to download a basic operating system and find their way through it. Then they have to find or program the needed additional components or addons and perhaps leave out certain components already in the OS in the interest of code size and efficiency.

Once the code is found, modified, added to and compiled, it must of course be tested. The amount and type of testing is up to the organization involved. Testing not only discovers and helps fix defects and issues of incompatibility, it also provides confidence that the product offered on the market is less likely to come back with issues in the underlying software and customer dissatisfaction. But this all takes time . . . and money.

The next issue is support. It is quite natural that customers expect the developer/vendor of a computer-based product to support it with answers to simple questions, answers to more in-depth questions, solutions to bugs and other problems, software patches and updates, and sometimes also consulting for customers with specific problems or goals. Certainly, some of these services can be charged for while others are expected to be simply provided. But they require knowledge of the underlying software, and if it is provided by the vendor, that includes the operating system. The question here is, for a shipped product, how many resources can you afford to keep at the ready to satisfy these expectations? While you may not have the answer ready, for an OS supplied by a reputable commercial vendor, that vendor certainly does or should.

The World Keeps Getting More Complicated

There was a time years ago when a large number of embedded system vendors wrote and maintained their own real-time operating systems (RTOSs). It took many years and major marketing efforts to change this situation, which today has almost ceased to exist. While gentle and persistent persuasion certainly played a role, the more significant factor was the staggering increase in complexity and diversity of today's microprocessors. These now require a herculean effort in person power to create efficient operating systems that will address not only their computational power and advanced architectures, but also take advantage of some of the unique characteristics integrated in the different devices.

Today we see semiconductor manufacturers partnering with OS suppliers (or in some cases, even purchasing entire software companies) who also offer a choice of operating systems that have been tailored and tested for their own microprocessors, microcontrollers and, increasingly, systems on chip (SoCs). In the latter case, a number of semiconductor manufacturers have had to employ significant numbers of software engineers simply to write driver code for the daunting number of on-chip peripherals and buses that are integrated on a single piece of silicon. And this is while partnering with an operating system vendor to support their chips. These vendors have faced their own build or buy decisions with a unique but apparently effective set of solutions. Can anyone imagine addressing such an immense challenge starting with "free" open source code?

In today's competitive world, it is imperative to think clearly about the implications and consequences of such basic decisions as "build or buy." Cost is certainly a factor, but cost comes in many disguises. It is also intimately connected with time—time that is lost in the market and time that is spent paying for salaries and other expenses (rent, utilities, coffee) while time-to-market is passing. Those costs must be compared to those required for finished, proven OS products and services. Those support services can come as part of the OS price, or as service contracts, but they obviously constitute part of the development and support costs of the end product. They also represent a value in terms of supporting the end customer.

Most organizations will have hardware and software teams working on the development of whatever product or system is the end goal. They will need to understand the OS as well in order to be able to develop the final applications. Will that be easier for them in terms of

a housedeveloped or a commercially acquired and supported OS? This also constitutes cost and time-to-market. For the sake of a company's valueadded goal, it is always better to have those people working on the unique application software and hardware development than spending their time dealing with the operating system. In the end it comes down to the question of what a given company is offering as a unique value to a customer. In the majority of cases, the customer really doesn't care what the underlying OS is. The customer cares about what the finished product will offer him or her in terms of their own goals and values. Time and money are always better spent perfecting and offering that value.

Ideally, but not in all cases, it can be possible to find an OS vendor who offers a support partnership that customers can contact. This, of course, entails costs to the developer and quite probably to the end customer, but it may be advantageous in some cases. That way a customer with a question that involved the OS could be immediately referred to the OS vendor for help. This is certainly not always a necessity, but it can be an advantage in some cases, and its costs and benefits should also be considered for product planning.

So if there is no such thing as a free lunch, there is certainly no such thing as a free operating system. The question is what does it really cost? That can best be answered by knowing what an organization's own goals are and what their value-added proposition is. In the vast majority of cases, it is not the operating system; the operating system is a component or a support element for the real value added. What seems free is not free if you have to put significant time and effort into making it work for your goals. But what is really needed and what does that cost in terms of money and time-to-market?

One example of an alternative that addresses many of the issues described here is Windows Embedded. The latest version, Windows Embedded 8, is a componentized version of Windows, which means that the developer can confidently assemble a version for his or her target application that uses only the components needed, and the components selected are already pre-verified to work together. Windows Embedded 8 comes in four editions: the Industry Standard edition, targeted at the manufacturing and healthcare industries; the Industry Pro edition, aimed at retail devices such as POS terminals, kiosks, scanners, etc.; the Industry Enterprise edition, for seamless integration with enterprise; and the handheld edition.

Since many operating systems will be functioning within the Internet of Things, seamless integration of small and mobile devices with enterprise and Cloud-based systems is becoming ever more important. Having this pre-considered and integrated into an OS family can greatly spare developers the time and effort to make certain their systems smoothly support such connectivity and compatibility. The involved licensing fees and costs should be carefully weighed against the often unknown and unanticipated patches and delays that can crop up when using a "free" open source solution.

For a successful outcome, these alternatives must be carefully considered. For most organizations developing advanced hardware/ software systems and products, the answer increasingly looks like the "buy" side of the equation. Whatever the outcome, it is certain that a "free" open source operating system is anything but free.

Arrow Electronics, Inc.
Arrow Intelligent Systems (AIS)
9201 East Dry Creek Road
Centennial, CO 80112, USA

0161_m2mIoT_09/16_CDS1.1