USING LINUX IN MEDICAL DEVICES

MENTOR EMBEDDED PLATFORM SOLUTIONS

Mentor®
A Siemens Business

E M B E D D E D P L A T F O R M S O L U T I O N S

W H I T E P A P E R

w w w . m e n t o r . c o m

Linux continues to be the leading choice for embedded device operating systems. According to the 2019 *EETimes Embedded Markets Study*[1], Embedded Linux remains the number one choice of embedded operating system at 21%, with 31% of respondents considering a Linux OS for their next designs in the coming year.

The advantages to choosing Linux are clear including the world's largest open-source community; a large ecosystem of evaluation boards, processor support, and software providers; plus a feature-rich kernel with multiple connectivity paths. But the decision on whether to choose Linux for use in a medical device setting includes the additional considerations of patient safety and data security.

This white paper will help designers understand Linux and:

- Medical Safety, and Device Security
- How to Address Security Issues when they arise
- Medical Device Certification
- Mixed-Safety Critical Systems
- The Hardware Platform
- The Optimum Use of Hardware

LINUX, MEDICAL SAFETY, AND DEVICE SECURITY

Patient safety is affected by medical device security. When we talk about safety, we refer to events that, if they go wrong, it will impact the patient's health and well-being. Security is something that can be looked at standalone; even in medical devices, not all aspects of security are tied to safety. For example, when we talk about protecting people's personal information and medical data, this is an aspect of security that does not overlap with safety. If the device is not secure, it makes it possible for bad actors to make these negative impacts happen either accidentally, or purposefully - which is much rarer.

An operating system like Linux does not directly do anything to make a device safer. The operating system doesn't prevent a failure from occurring, nor does it make the system recover when a failure occurs. In the terminology of safety, an operating system is not a safety mechanism. This makes sense when you think about it. When you put Linux on a system with no other application and turn it on, Linux boots, but it just sits there at a login prompt. It is not doing anything until applications run that leverage Linux; these applications contribute to the overall safety of the device. When you think in those terms, while an operating system is not a safety mechanism, it does enable them. Going back to the terminology of safety, the operating system is considered to be a safety element – something necessary, but not sufficient to create a safety mechanism.

Linux is the most heavily used operating system for devices in the world. It has a large, worldwide developer base that focuses on ensuring that it works as expected in all conditions, and is as secure as possible. That said, it is also the most studied operating system in the world, both by the vast majority of developers who are conscientiously working on improving Linux and other open source packages, and also by a small number of actors who are looking at ways to break into Linux for their own purposes. The security of applications using open source is a constant tug of war between these opposing forces.

Linux (and other major open source packages like OpenSSL or SQLite) are large packages that have sometimes unpredictable interactions with other software that might be running in the system. This is combined with the fact that many flaws are hard to find in code reviews, normal testing, or by static analysis, and are undetectable unless software is combined with task switching and inter-process communication. Best practices will not identify every possible flaw or exploit, and much of the open source software that we rely upon was not originally developed with today's best practices in place.

Therefore, the most important pieces of open source software used in devices worldwide are in a much more stable and secure place than they were five years ago. This is mainly due to the hard work and diligence of engineers all over the world in identifying avenues of exploitation, fixing those when they find them, and then the worldwide community looking for similar issues in their own projects. The work will never be complete, but it is becoming harder and harder to find exploitable flaws in this important infrastructure software.

ADDRESSING SECURITY ISSUES

For the most part, security issues in Linux and other important software like OpenSSL, are found by engineers either as a bug that is uncovered as part of routine work, or through concerted efforts to find exploits, so called "white hat" hacking. In either event, the discoverer of an exploit will notify the community of the offending open-source component, and in most cases, the discoverer or somebody in the community will notify the Common Vulnerability and Exposures (CVE) group, which is run by MITRE, and is closely related to US National Vulnerability Database (NVD), run by the National Institute of Standards and Technology (NIST).

Once a vulnerability is understood, usually after a fix is available, the CVE is publicized by inclusion in these lists, and if sufficiently serious, discussed by the security community worldwide. This is the point where devices are potentially most vulnerable. Since most vulnerabilities are found by the "good guys," the bad guys find out about them at the same time as the rest of the world, and can deploy exploitations that take advantage of the newly found vulnerability. That said, this publicity is very important since it alerts the worldwide community of both the issue and the fix. As a result, an organization can determine if a particular exploit might affect their devices, and if so, they can mitigate the issue before a device might be attacked.

MEDICAL DEVICE CERTIFICATION

Linux cannot be pre-certified for use in a medical device. Certain real-time operating systems (RTOSes) such as the Nucleus RTOS from Mentor, a Siemens business, can be acquired pre-certified, as can other embedded software components from a number of vendors. To achieve this kind of pre-certification the vendor must be able to show that the complete software development process (requirements, design, development, testing and verification and all of the steps of development) have been performed to medical industry standards such as ISO 13485 and/or IEC 62304. Linux and other open source components are not developed to these standards, and thus cannot be pre-certified. As a note, there have been efforts to show conformance of Linux to the overarching concepts of functional safety (usually mapping to IEC 61508, from which many industry standards are derived, including IEC 62304). While these have not been successful so far, the current effort (Enabling Linux in Safety Applications or ELISA, launched by The Linux Foundation) is showing promise in this area both in improving the processes used in open-source software development, and in mapping the higher-quality output to these standards. However, this promise is likely years away from being completely realized.

Instead of pre-certification, in medical devices, Linux is generally handled using a concept from IEC 62304 called Software of Unknown Providence, or SOUP. Under these guidelines, the use of Linux is considered as part of the risk assessment of the overall device. Potential failures of Linux as used in the device must be considered and mitigated if they might cause harm to a patient. This risk assessment must meet the requirements of the FDA's pre- and post-submission guidance so on the front end, it requires considerations on the use of Linux in the design, implementation, testing and verification of the device. Then, after release, the use of Linux (and all open-source software) must consider the possibility that issues will be found post-release. Certifiers are taking a very close look at this aspect of open-source, especially as far as security issues (i.e. CVEs) are concerned.

A commercial Linux provider does this work as part of their core competency, and while they will charge you for this type of service, it will cost less in the long run. As the device manufacturer, you will end up with a higher-quality product with fewer integration issues as the product is maintained over its lifetime. In addition, the medical device software engineers can focus on your product's unique IP, rather than continually working on the Linux infrastructure.

IMPLEMENTING A MIXED-SAFETY MEDICAL DEVICE

Historically, if Linux was used in a medical device, the architecture was designed to separate Linux from the safety critical part of the system to the greatest extent possible because it wasn't trusted. The system would probably run Linux on a separate processor, and Linux would be responsible for activities that it was well suited for, like connectivity to the office or hospital network, running a display, taking user input, etc. Then, if information came in that would affect the safety function of the device (such as a dosage in an infusion pump), the new setting would be validated on another processor or in hardware before the change was administered to the patient. In many cases, these kinds of inputs that impact safety might not even come from Linux, but maybe from a dial or other hardware input. If an issue arose in Linux, it would not impact the rest of the device, which would continue to execute safely.

When using an open-source version of Linux in your device, you are responsible for acquiring it, and maintaining it in your device, both during development and after release. Acquiring Linux (and other open-source modules that you are likely to use) is easy; it often is available from your board or microprocessor vendor targeted to your development board. However, this Linux is usually made available on an as-is basis and is meant to get a system up and running; your board vendor will not be performing maintenance or timely updates of that Linux OS for you to use after release. As a result, the responsibility falls onto the device developer to manage, for the life-time of the device. This is not impossible, but by doing it yourself, you are committing engineering resources to monitor and maintain the significant number of CVEs that are reported against open source components every day. As an example, in 2019 there were 170 CVEs issued just against the Linux kernel, and 12,174 CVEs created in total. It is a significant effort to review each of these instances, determine which can affect your design, gather and implement the fixes, then verify that the operating system is still performing as you would expect.

LINUX AND THE PROCESSING PLATFORM

Today's microprocessors are much more powerful and complex, with multiple processing cores designed to support what we call heterogeneous multiprocessing –where there will be powerful general-purpose cores to support running an operating system like Linux, and additional specialized cores to handle other functions. Safe devices today are taking advantage of this trend. Today's microprocessors are much cheaper than the multiple ones you might have used in the past, and much more powerful. But, it means that there are more considerations to think about when you use them. To take full advantage of the lower board bill of materials (BOM) costs and higher integration of components available in an advanced multiprocessor chip for a medical device, applications must be kept separated by function – what's known as mixed-safety criticality.

For example, you can now run the user interface portion of your system on a processor cluster that's optimized for user applications with features like multi-level cache, and power islands for shutting off individual processors and memory, thus conserving power, as in a battery-operated medical device. This is an ideal domain for Linux where the kernel can also make optimal use of the application processor cluster with built-in features like Symmetric Multi-Processing (SMP), parsing tasks and threads to individual processors.

Simultaneously, the safety critical portion of the system runs on a separate cluster that is dedicated to real-time processing with features like tightly-coupled data and instruction memory with extremely low fetch cycles, and highly deterministic performance, or lockstep mode for error detection where an RTOS is much more appropriate, such as the Mentor Nucleus RTOS.

Advanced multiprocessing systems contain hardware-enforced isolation that keeps the application world and the safety-critical world separated. However, the software designer will need to use middleware such as the Mentor Embedded Hypervisor or Mentor Embedded Multicore Framework to take advantage of those hardware features. These software packages make important system-level functions like secure Inter-processor Communication (IPC) between the processor clusters possible.

OPTIMIZING YOUR HARDWARE USING LINUX

There are additional areas where Linux can optimize system performance in the medical device. One example is an advanced option in the Linux kernel known as the Power Framework. The software application can utilize the kernel to power down portions of the system when they're not needed, and power them up when required, conserving system power, which is important for hand-held, battery-powered devices. Advanced SoCs include peripheral interfaces that are included with the processor clusters like I2C or USB that can be controlled in this manner. But that advantage can also be extended to implementing "soft" peripherals external to the processing cluster in RTL logic, for example a CAN interface –and even peripherals outside the chip, like an Ethernet PHY for interfacing to additional devices.

Taking full advantage of modern embedded systems means designing hardware with awareness of what the software is capable of, while at the same time, writing accompanying software drivers, firmware, and applications that can utilize all the features of the hardware.

Silicon vendors are primarily focused on providing the best chips available to the market. All embedded silicon vendors provide some level of software support, but typically this is limited to bare-metal drivers, open-source builds of Linux, and some limited middleware. For advanced features like enabling mixed-safety criticality medical systems, the best option for the designer is to utilize a software vendor like Mentor that specializes in enabling multi-core applications. Thus, your designers can focus on your own IP, and not having to code and debug the middleware.

CONCLUSION

Choosing Linux for your medical device design can deliver enormous advantages. However, it is the responsibility of the systems designer to ensure that patient safety and data security are maintained. Once security issues are identified, maintaining and updating the Linux operating system can require significant effort. Using a commercial Linux from a trusted vendor like Mentor Embedded Linux Omni OS or Mentor Embedded Linux Flex OS will help lower the cost of ownership versus an open-source Linux, while maintaining patient safety and device security.

REFERENCES

[1] AspenCore 2019 Embedded Markets Study (March 2019); retrieved from: https://www.embedded.com/wp-content/uploads/2019/11/EETimes_Embedded_2019_Embedded_Markets_Study.pdf

Corporate Headquarters
Mentor Graphics Corporation
8005 SW Boeckman Road
Wilsonville, OR 97070-7777
Phone: 503.685.7000
Fax: 503.685.1204

Sales and Product Information
Phone: 800.547.3000
sales_info@mentor.com

Silicon Valley
Mentor Graphics Corporation
46871 Bayside Parkway
Fremont, CA 94538 USA
Phone: 510.354.7400
Fax: 510.354.7467

North American Support Center
Phone: 800.547.4303

Europe
Mentor Graphics
Deutschland GmbH
Arnulfstrasse 201
80634 Munich
Germany
Phone: +49.89.57096.0
Fax: +49.89.57096.400

Pacific Rim
Mentor Graphics (Taiwan)
11F, No. 120, Section 2,
Gongdao 5th Road
HsinChu City 300,
Taiwan, ROC
Phone: 886.3.513.1000
Fax: 886.3.573.4734

Japan
Mentor Graphics Japan Co., Ltd.
Gotenyama Trust Tower
7-35, Kita-Shinagawa 4-chome
Shinagawa-Ku, Tokyo 140-0001
Japan
Phone: +81.3.5488.3033
Fax: +81.3.5488.3004

A Siemens Business