BlackBerry | QNX

5 CRITERIA TO CONSIDER WHEN CHOOSING AN OS FOR YOUR MEDICAL DEVICE

WHITEPAPER

CONTENTS

Medical device designers have become increasingly interested in the Linux® operating system (OS), primarily because of its open-source model. Linux lets the designer take advantage of a large pool of developers, a rich legacy of source code and industry-standard POSIX APIs. But while Linux may work well for many applications and rapid prototyping, it may be preferable to choose a commercial operating system, particularly for safety-critical medical devices such as those used for infusion delivery, heart monitoring and resuscitation and robotic surgery.

Developers of safety-critical medical devices should consider these five issues when deciding whether Linux is the best option for their OS.

1. SAFETY CERTIFICATION

Regulatory requirements are driving the demand for software that is compliant with the IEC 62304 standard for "Medical device software – Software life cycle processes." This standard requires that manufacturers follow good development practices to produce high-quality software for medical applications. And it has been endorsed under medical device-related directives by the FDA in the U.S.

While certified systems can be designed based on an open-source OS, the development efforts can be considerably more complex, expensive and time-consuming than when using an OS designed specifically for medical devices. Even minor patches or software updates to an open-source OS can result in a significant ripple effect of rebuilds and changes that can require expensive retest and recertification efforts. The Linux Foundation estimates that the development community has been merging patches at an average rate of 7.71 patches per hour since the 3.10 kernel release in October 2011. Keeping up with this rate of change for accepting or rejecting patches, validating ripples through effects of changes (such as addressing dead code) and testing is a monumental task.

Using an OS pre-certified by an independent third-party auditing body such as TÜV Rheinland® can significantly reduce certification efforts for medical device manufacturers.

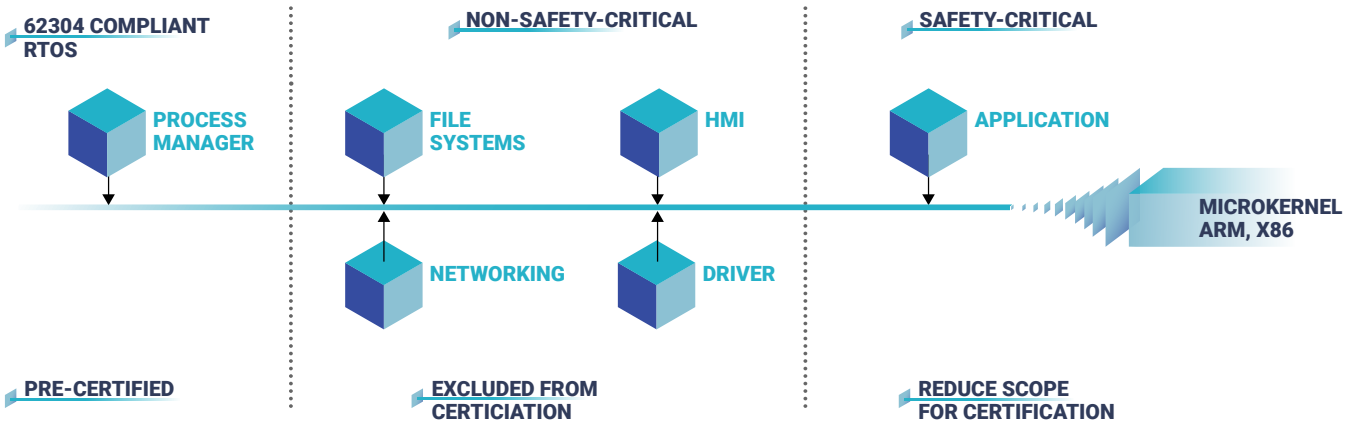

Figure 1:
Using a pre-certified RTOS helps reduce scope of certification

2. CODE TRACEABILITY AND SOUP

The IEC 62304 standard cautions developers of software for medical devices about using third-party software, particularly “software of unknown provenance” (SOUP). The standard spells out a risk-based decision model for determining when SOUP is acceptable and defines testing requirements for SOUP to support rationale for using it.

Certification to IEC 62304 requires traceability of patch sets and code and ensures that specified development processes are strictly followed. But Linux encourages broad contributions and changes to the source base, with no requirements for standards-based development processes, making it literally “of unknown provenance.” And while the Linux source code is open to scrutiny, the sheer number of developers (more than 12,000) contributing to it makes it virtually impossible to trace everything in the code to meet the standard’s stringent risk analysis requirements for certification.

In short, if certification will be required for the final medical device, the ability to trace the source code is an important consideration.

3. OS ARCHITECTURES

Selecting an OS is one of the most critical decisions a manufacturer makes, and the two main kernel architectures to choose between are monolithic and microkernel. A monolithic kernel runs all operating system components in kernel space, including device drivers, file management, communications, networking and graphics stacks. As a result, a single programming error in any of these components can crash the OS. In addition, any change to one of these components implies an OS modification and recompilation. For example, if a provided Linux driver isn't suitable for the medical device, developers must write and maintain custom drivers, which requires valuable developer time and increases development costs.

Figure 2 shows a hypothetical medical monitoring device built on a monolithic OS. The drivers, file systems, communications stack, etc., must all be included in the risk analysis because they impact and can corrupt the kernel.

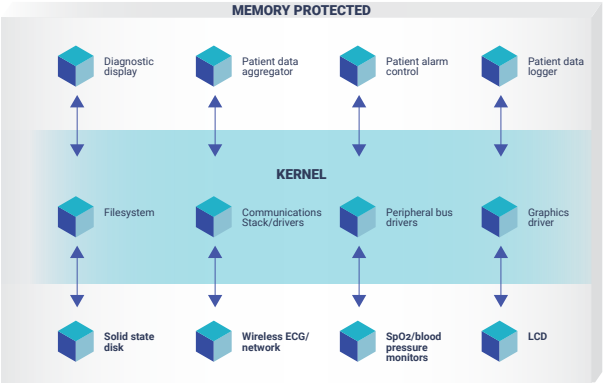

Figure 2:
Example of a monolithic OS architecture

A microkernel OS such as the QNX® Neutrino® RTOS and its safety variant, the QNX® OS for Safety, can also deliver a full range of OS services required for medical applications. These include networking services that support complex distributed systems in which multiple devices seamlessly share resources and communicate without custom protocols. The OS also supports a wide range of block and flash file system formats and a power-safe disk file system for data integrity and reliable storage. For applications with a user interface, the screen framework enables developers to build graphically rich, compelling user interfaces using built-in, high-performance, OpenGL ES-based transitions.

Additional third-party user interface technology, such as Qt or Crank, seamlessly integrates with the screen framework. Figure 3 shows the QNX Neutrino RTOS microkernel architecture, which provides extensive fault containment and recovery as every driver, protocol stack, file system and application runs outside the kernel in the safety of memory-protected user space.

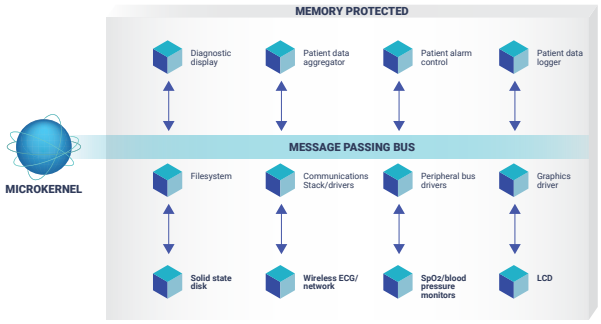

Figure 3:
Example of a microkernel OS architecture

4. COST AND TIME-TO-MARKET

Linux is often considered a free operating system because it provides open access to its source code. The total cost of ownership must be considered, however, using an open-source OS may incur extra costs for development time and testing to certify the system. This additional development time can result in lost revenue due to delays in bringing the device to market. Most medical device developers want to focus their efforts on their proprietary, value-added applications, but an open-source OS may require additional investment to sustain an in-house team of OS experts to configure, build, support, certify, test and maintain a large codebase that is the foundation of the product.

The QNX OS for Safety is a fully-featured RTOS that dramatically simplifies the migration from a Linux-based prototype to a production system. Its POSIX compliance ensures Linux API compatibility, increasing code re-use and eliminating the learning curve that often comes with adopting a commercial RTOS. As a result, it helps reduce program cost and risk and shortens the time-to-market for medical device developers.

Figure 4:

The activities required for the initial deployment of a Linux-based system in a device requiring safety certification or pre-market safety approval.

5. SUPPORT AND LICENSING

The community development model of open-source software can make it difficult for developers to get the help they need when they need it. With demanding time-to-market expectations for medical devices, waiting for an OS issue to be resolved can have a massive impact on a project's success. And if the solution is an upgrade or patch, that introduces additional complexities and the possibility of more support issues. In addition, it's up to the application developer to ensure that the correct licenses are used and have been appropriately attributed to avoid contamination of a proprietary source base.

BlackBerry® QNX® products include a licensing support system that helps ensure the provenance of all source code. And BlackBerry QNX offers a range of additional support services, including customized courses tailored to distinct project needs, technical requirements and project-specific challenges. To complement the QNX OS for Safety, BlackBerry QNX also provides training, architectural reviews and custom engineering services. Our expertise in designing safety-certified embedded systems enables us to guide developers through the safety certification process to help them meet important product launch dates.

DON'T RUSH AN OS DECISION WITHOUT ALL THE FACT

Not every medical device application requires a safety-certified operating system. Linux can be an appropriate option for many of those applications, as can the QNX Neutrino RTOS. But for life- or safety-critical embedded systems that must operate without failure, the QNX OS for Safety is a proven, reliable choice.

About BlackBerry QNX: BlackBerry® QNX® is a trusted supplier of safe and secure operating systems, hypervisors, frameworks and development tools, and provides expert support and services for building the world's most critical embedded systems. The company's technology is trusted in more than 195 million vehicles and is deployed in embedded systems around the world, across a range of industries including automotive, medical devices, industrial controls, transportation, heavy machinery and robotics. Founded in 1980, BlackBerry QNX is headquartered in Ottawa, Canada and was acquired by BlackBerry in 2010.

BlackBerry QNX software and development tools are standards-based and enable companies to adopt a scalable software platform strategy across product lines and business units. The BlackBerry QNX software portfolio, including safety pre-certified products, is purpose-built for embedded systems and scales from single-purpose devices to highly complex systems of mixed criticality. Because we are successful only when you are, you can rely on our support and professional services teams to provide the expertise you need, when you need it—throughout the entire product development life cycle.