

Real-time Linux, past and present

By Daniel Walker

Linux traditionally has been considered a general-purpose server or desktop Operating System (OS), and as a result is often viewed as incapable of true real-time functionality. However, this assumption has proven false on multiple occasions. In fact, various Linux approaches begin with a dual OS, Linux, and Real-Time Operating System (RTOS) and continue down the path of low latency, resulting in Linux as the RTOS. Typical real-time approaches entail running an application on bare hardware or using a small OS, which is considered tightly bounded and specifically created for use in real-time applications.

With more than nine million lines of code in today's Linux kernel, many programmers often struggle envisioning how to approach Linux. Additionally, programmers are fearful of the Linux community and the General Public License (GPL). These concerns have led to the creation of the dual OS real-time approach, which primarily entails ignoring the existing Linux code base and using a second OS for real-time purposes. Subsequently, the two OSs are combined using one of the flavors of "glue code," such as Opersys's Adaptive Domain Environment for Operating Systems (ADEOS, see Figure 1). From the end user's perspective, the two kernels appear as a single OS. From a high-level perspective, the real-time kernel appears as a high-priority process and the Linux kernel appears as the low-priority process. FSMLabs and the Real-Time Application Interface (RTAI) use this now-common approach.

Although the dual OS real-time approach has become prevalent, Linux code is fully capable of being developed into an RTOS. TimeSys, the first to create soft real time based solely on the Linux kernel, deserves credit for breaking new ground with the single Linux kernel method, but did not

disclose the majority of its changes to the Linux community. Additionally, the changes it released were not considered for inclusion in the vanilla Linux kernel.

The Linux-based RTOS movement continued to make significant advances with Robert Love's changes, which created preemption inside the Linux kernel as well as a constant time scheduler. Linux community members including Ingo Molnar (Red Hat) and others rewrote or heavily reviewed both of these changes and then incorporated them into the kernel. Next, Molnar began work on Voluntary Preemption, a polling preemption process mainly useful in server environments. Over the

course of several months, the Voluntary Preempt patch matured, putting a Linux RTOS within reach.

During subsequent Voluntary Preemption development, Sven Dietrich and I (to a lesser extent) created and released a set of changes for Linux 2.6 that paralleled those TimeSys created with Linux 2.4. In the ripe environment of advances in latency reduction, this code release became a channel for the community to finally solve the real-time Linux dilemma. Molnar, the most experienced real-time community member, rewrote/created and maintained all real-time Linux features in the newly formed real-time Linux branch that evolved from Voluntary Preempt. Molnar's involvement played a vital role in getting Linux real time to the point it's at today. This flurry of activity endowed the real-time Linux branch with preemption rates that approach the hardware context switching cost.

The Linux real-time kernel uses the following key features:

- **Priority Inheriting Mutex:** A mutual exclusion object (mutex) allowing the highest priority of all the waiters on a

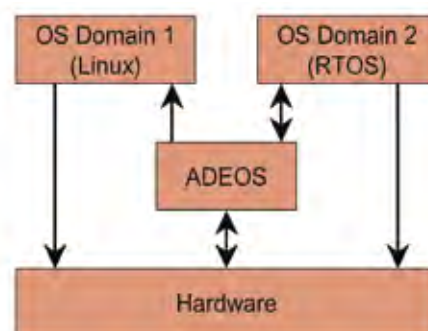


Figure 1

The days of Linux considered as only a general-purpose OS are ending.

mutex to increase the process priority of the mutex owner.

- **Spin locks seamlessly converted to the new mutex:** Spin locks in Linux disable preemption; consequently, one of the keys to lower preemption rates is removing spin locks through a noninvasive process.
- **Interrupts in threads:** The process of servicing interrupts in thread context instead of interrupt context. Linux also disables interrupts (and thereby preemption) when in interrupt context. Interrupts can lock the new mutex, meaning they need to be in process context so they may sleep.

Embedded solutions companies such as MontaVista and others have created products based on these changes. The real-time patch to the Linux kernel is openly available at <http://people.redhat.com/mingo/realtime-preempt/>.

The features in the current real-time branch represent a culmination of the ideas from prior real-time approaches. Using these changes, the Linux kernel can handle hard real time. The significance of this is astounding, but has not been fully realized. As companies and individuals begin to leverage these changes, the days of Linux considered as only a general-purpose OS are ending. **ECD**

Daniel Walker is a real-time architect at MontaVista Software. In addition to his active open source community involvement for seven years, Daniel has been involved in Linux real-time development for more than two years. He graduated with a BS in Computer Science from the University of California Santa Cruz.

To learn more, contact Daniel at:

MontaVista Software, Inc.

2929 Patrick Henry Drive

Santa Clara, CA 95054

408-572-7869

dwalker@mvista.com

www.mvista.com