CONTROL SOFTWARE FORUM

Real-Time Linux lets you custom design your hard real-time control applications

Dr. Peter Wurmsdobler
Open Control Lab Director
Control.com
Hopkinton, MA

Linux has become a powerful operating system (OS) challenging the computer systems market because it is open source and adaptable to different hardware and computing problems. Today there are some hard real-time implementations of the Linux kernel available on the Internet; they are mature and suitable for control/automation systems. With these implementations and some industrial applications already in existence, now is the time to try real-time Linux in controls.

Although an automation system is usually based on a hard real-time OS, a control engineer typically focuses on design issues. The resulting control concept is commonly implemented on proprietary target hardware using some proprietary software and tools, too. Doing so simplifies the implementation and integration, but conceals internal functioning and decreases adaptability to specific customer problems.

Most proprietary solutions lack transparency and openness. Because real-time Linux is open source, you are free to look at the code, discover how it works, fix bugs and adapt it to your needs. You are free to add a different scheduler policy, ring buffers in shared memory, or any other communication scheme. In addition, the community will be happy to accept your proposals and to adopt them as input to standards. At this time the application programming interface is reliable enough to build applications upon it.

Linux vs real-time Linux

Linux is an OS for a general-purpose computer optimized for maximum throughput and average performance of each process. Its kernel is not appropriate for hard real-time use for several reasons. For example:

- It uses coarse-grained synchronization.
- It cannot be preempted.
- It will disable interrupts sometimes for long periods, especially in device drivers.

How do you make Linux a real-time OS? The most natural strategy would be to add real-time capabilities by modifying the kernel. One common method is to insert preemption points into the kernel wherever it's safe to perform a context switch—after which the kernel checks whether a high-priority process can be run. There are plenty of such implementations, but the result is only soft real-time performance, which isn't really appropriate for control.

A second and more promising approach is to split the kernel into two parts, one part that runs as a general purpose OS with no hard real-time capabilities, and a second part with real-time capabilities, but running the first part as simple task. This method is often used, for example, to add real-time capabilities to the Microsoft® Windows™ OSs. As for the Linux kernel, there are two principal implementations of this approach, both based on the NMT version of real-time Linux:

- RTLinux developed by FSMLabs; and
- RTAI developed by the Dipartimento di Ingegneria Aerospaziale, in Italy.

The core idea behind these hard real-time Linux implementations, which I'll refer to simply as real-time Linux, is to run a full-featured OS as one thread of a real-time executive. In a logical sense, tasks and interrupts are separated into two classes, the real-time ones running directly on behalf of the executive, and the non-real-time ones executed within the common OS, with some means of communication between these two priority spaces.

As a kernel patch, the modifications to the kernel put a thin layer underneath the Linux OS. This core hard real-time mechanism takes over control of the low-level interrupt handling from Linux. The low-latency interrupt handlers cannot be preempted by Linux, which acts only on emulated interrupts. Thus, the hard real-time layer constitutes a single process per CPU running on the bare computer hardware, with Linux being just a low-priority thread among real-time threads and interrupt service routines. Linux, as a task, works as usual in its own process space.

Why use real-time Linux in control? Real-time Linux is appropriate for control because the only—but most important—resource managed by the real-time executive in a very deter-

ministic manner is timing.

Many features necessary for complex control applications are already implemented as kernel modules. For example:

- Unconditional interrupt interception and advanced interrupt control functions,
- High-resolution (in ns) timing functions,
- Independent scheduling instance controlling the real-time threads,
- FIFOs between the real-time and the non-real-time part of the entire OS,
- Shared memory allowing real-time/nonreal-time threads to share data,
- Signals synchronized between real-time and non-real-time threads,
- Synchronization primitives (mutex, semaphores, condvars).

Offering a reliable concept is sometimes not enough for a single user or a control systems programmer. Good documentation is needed as well as debugging tools and support. Since real-time Linux is a product of the Internet, the entire community is available to supply support. Various Internet sites and companies have been established to offer professional services—e.g., the Real Time Linux Foundation (*www.realtimelinuxfoundation.org*) and Rick Lehrbaum's LinuxDevices.com (now part of ZD Net).

Because real-time Linux is a kernel patch, it follows the rapid development of the Linux kernel, which is ported to nearly all CPU platforms. It is available for i386 processors, PPC and AlphaAXP, and many other ports are underway. Similarly, drivers are available from users and hardware suppliers, or from the Linux Control and Measurement Interface (*http://stm.lbl.gov/comedi/*) for data acquisition boards.

Industrial real-time Linux applications

Real-time Linux evolved with the applications for which it was used, and vice versa. An example is FlightLinux, a concept that uses a real-time Linux for on-board spacecraft use. It is the subject of a NASA Advanced Information System Technology (AIST) research effort (*http://aqua.qssmeds.com/flightLinux/*). Another flight-related control application is the Level C Flight Simulator, which controls a Dassault Falcon 10 simulator with six-degree motion.

Another interesting ongoing project is the adaptation of MiniRTL, real-time Linux on a floppy disk, for high-end T1 modems made by the Austrian company Datentechnik AG. Emco, another Austrian company, is implementing real-time Linux for CNC machine control. (For more information on this project, e-mail wagnery@emco.co.at.) In the machine control arena, the Oregon Cutting Division of Blount Inc. is developing a control system for 60 in-house chain assembly machines running 24x7 at a control period of 500 µs using RTLinux from FSMLabs. In a similar way, Lang GmbH & Co. KG in Germany uses RTLinux to control up to six stepper motors for an engraving and milling machine.

Other RTLinux examples include:

- A power control project, controlling a 60-MW power supply at the Institute for Plasma Research, STT Operations and Control Group in India (contact ranjan@ipr.res.in for more info).
- A test bench for automobiles processing recorded road data through a 4512 point FIR filter to control hydraulic cylinders (Tracos Prüfsysteme in Germany, *www.tracos.de*)

Real-time Linux is ready now

The hard real-time modification of the Linux kernel offers a very flexible and powerful means to implement control and automation systems. The maturity of these implementations, running industrial examples, and the support of the Internet community and professional service providers will encourage its intense use in control systems over the coming years.

About the author

Peter Wurmsdobler (peterw@thinkingnerds.com) is research assistant at the Institute for Machine and Process Automation in Austria, and responsible for the development of real-time measurement systems at the Centre de Transfert des Microtechniques in France. In August he joins control.com as open control lab director. He is initiator and co-organizer of the annual real-time Linux workshops and cofounded the real-time Linux foundation in January 2001. Wurmsdobler received his master in mechanical engineering and a PhD in control engineering from the Vienna University of Technology, Austria.