

Real-Time Linux – Where is it now?

In recent years the Linux operating system has gained quite a following and continues to do so. However, despite its reputation for performance and reliability on office and ISP servers to date, Linux still enjoys only marginal popularity in the desktop market and is virtually unknown for many other industrial and multi-media applications. Nonetheless, the perception of Linux as the province of a committed fringe group of open-source advocates and opponents of Microsoft, is changing rapidly for a variety of technical and commercial reasons. The introduction of hard real-time extensions to Linux has opened and extended standard Linux to make it applicable to the industrial and embedded arenas. This article presents an overview of the current state of play with respect to both the technical and the business aspects of Real-Time Linux as an RTOS.

INDUSTRIAL AND TECHNICAL ADVANTAGES OF REAL-TIME LINUX

Performance

The performance benchmarks of real-time Linux compare favorably with most currently available high-performance proprietary RTOS's (see URL reference to benchmarks at the end of this article).

In addition, the technical currency of Linux with mainstream software engineering developments is at least comparable, if not superior, to other operating systems. It is typically at the leading edge of software engineering concepts for those willing to utilize the odd numbered releases. (All development versions of Linux are released as odd numbers, currently 2.3.07, while the even numbered releases are reserved for the stable production quality code.) At the time of writing 2.2.10 is the current stable release of Linux. Stable, validated versions of real-time Linux are available for both 2.0 and 2.2 releases.

System Services

Real-time Linux brings with it all the services and sophistication of a mainstream UNIX O/S. This is accomplished by a symbiotic relationship in which Linux provides the regular operating system services, while the real-time extensions provide the time-critical services. The real-time code comprises a relatively modest kernel extension (approximately 30 kilobytes) of the mainstream Linux O/S. Once installed, the real-time O/S is, in fact, indistinguishable to the user from the standard full service O/S, unless the developer elects to make use of the real-time kernel extensions.

Maintainability

While growing at a substantial rate, the standard Linux kernel is still significantly more compact and manageable than other proprietary alternatives.

Reliability

With regard to reliability and predictability, Linux has already established a track record second to none. There are many servers in use that have been operating continuously for over a year without any interruption.

Until recently, the real-time extensions have been a different story, always lagging the main kernel development and usually being very much beta form. With the

increase in the number of real users (versus experimenters), and commercially supported testing and validation, the situation is much improved. The next six months will see some significant deployments of current works-in-progress with major applications.

Customization

For the embedded processor/single board computer manufacturer, the availability of the well documented source code means that it is eminently feasible to customize the kernel to create a compact stripped-down kernel that matches the capabilities and silicon of a particular product.

Of course, this is exactly the market that WINCE is aiming for. But, until rumors of the real-time version actually materialize, the board manufacturer might be advised to opt for the bird in the hand versus the vaporware in the bush. In the interests of full debate, it must be conceded that this customizing process for the real-time kernel is not the GUI-based interaction afforded by WINCE. But under the aegis of a competent authority it may be accomplished reliably, affordably, and within a few calendar weeks. Such time scales and costs are usually not significant within the context of a custom board development, and are often a viable tradeoff for the ability to customize at the code level rather than through the muffling lens of a GUI. (An experience akin to taking a bath with your clothes on for a "real" software engineer.)

Platform Availability

Real-time Linux is currently only available on the Intel platform. However, with the release of a stable 2.2 version, several hardware vendors have expressed interest in versions for the Alpha, StrongArm and 68000. By the end of the year it is likely that ports will either be completed or underway for most of these, as well as other processors. At least one vendor, (Zentropix), has already developed a subscription mechanism, which allows a consortium of interested corporations to share the costs of such ports in return for access to a validated and supported real-time kernel.

Legacy code

The availability of porting libraries for IRIX, VXworks and pSOS+, from at least one commercial manufacturer, means that legacy code can be migrated, with zero risk, and at relatively low cost from most Unix platforms to much lower cost Intel based platforms.

Real-time Linux Development Tools and Development Environment

The Gnu Debugger (GDB) is open source and has an excellent reputation for symbolic debugging in standard Linux. Other proprietary commercial offerings are now available for real-time development, including an in-the-kernel, non-intrusive runtime debugger which supports C/C++ and ADA languages, in their native source environments. A step-and-trace debugger is expected from Zentropix before the end of the year.

The Zentropix debugger allows the user to symbolically examine and control data structures of a running Linux user process, kernel module or RealTime Task. It is critical that process is non-intrusive, as this method allows examination of the dynamic behavior of the program in question. The run-time data debugging may be self hosting or host/target via a network.

BUSINESS ADVANTAGES OF REAL-TIME LINUX

The advantages of Linux form a business/user perspective include:

Availability of the source code

Access to the source code, which guarantees users last-resort control of their own destiny can be a strategic issue. With some of the more successful proprietary RTOS¹, the turnaround time of bug fixes is only satisfactory where the problem intersects a large part of the software companies' market. Problems that are unique to a particular application, no matter how annoying or critical for the user, may go unaddressed if the RTOS owner—the only one with access to the source code—does not perceive it to be in their commercial interest to make the changes.

Unit pricing

The price is right! For many RTOS applications, price is not a significant factor. For instance, in the development of a fifteen million dollar flight simulator, you are much more concerned about the RTOS performance and tools than about a fifty thousand dollar RTOS price tag. However, if you are developing a multimedia application, such as a cell phone, or a set-top box which is intended to ship in the tens of millions and sell for tens of dollars, or perhaps even be given away, the per ship cost is critical.

Open Source License

All Linux is available only under the terms of the GPL (General or Gnu Public License or "copyleft"). No discussion of Linux from a business perspective can be complete without addressing the impact of the GPL since it permeates and influences all attempts at commercialization. It is at the same time both Linux' greatest strength and its greatest weakness. It can be extremely difficult for companies to justify investments in code that must then be made freely available to others.

The GPL is most emphatically not the same as being free, since it imposes two primary obligations upon the user. First, that they must impose the GPL restrictions on all subsequent use of the code; second, that any

onward distribution of the code, standalone, or in an application, must be accompanied by the source code.*

Many myths abound regarding the GPL, the most prevalent of which is that the user/distributor/modifier is prohibited from charging a fee for the software. Not true. You are free to charge anything you feel you can get away with. The emphasis being on "what you can get away with" since copying and redistribution of the code is one of the guaranteed rights of the user under the GPL.

Support

This is, of course, the area where Linux is still considered to be most vulnerable. No one ever got fired for buying IBM as the old saying goes. There is still a strong argument to be made for choosing an RTOS that has the support of a multi-million dollar corporation behind it. But even here the situation is changing rapidly. Organizations offering technical support for standard Linux are springing up like weeds and now there is at least one company (Zentropix) specializing in tools and support for real-time Linux. And recently Caldera announced the creation of a new division (Lineos) specifically targeting embedded systems development tools.

Configuration Management

It must be admitted that management of the code baseline has been a major problem with the real-time kernel extensions and is probably one of the main reasons why more companies have not rushed to adopt it. To install a working version of the real-time O/S would probably take a competent Linux engineer a week of patch downloads and false starts. This situation also is now substantially improved with the availability of a CD-ROM distribution.

REAL-TIME LINUX IN AN EMBEDDED BOARD APPLICATION

From the perspective of an embedded board manufacturer, (Alta Technology), the availability of source code and tools with real-time Linux is especially critical. In the embedded systems industry – where time to market of a full platform solution (hardware and software) is particularly critical – source code can be a very important issue.

Source code availability allows parallel software and hardware development cycles, by enabling the hardware producer to release prototype drivers. With Linux, the hardware designer is able to offer board-support packages that can be freely shared with partners at an early phase of the development cycle. This expedites support for new hardware.

The critical path in the product release cycle is shortened because the board customer has early access to functional software without waiting for production level code to be released. The hardware manufacturer's customer gains a competitive advantage by being early to market, allowing him to sell more product. Thus, both user and supplier benefit from this shortened cycle.

*This is a gross simplification, see www.gnu.org for the full text.

MARKET-TRENDS

Until recently, Alta Technology, in common with many of their customer base, questioned the maturity of the real-time Linux operating system to be "ready for prime time". Difficulties with downloading the various (free) packages from different web sites, plus the fact that much of the code was in beta form, meant that an industrial user was faced with a daunting task, especially in the area of configuration control. Tool availability was virtually non-existent, and the user was most definitively on his own with respect to commercial support.

However, this situation has improved rapidly with an ever-growing number of companies committing to the use of real-time Linux. The existence of a configuration managed CD-ROM distribution has even allowed one user to begin the process of obtaining FAA certification for their application. ■

REFERENCE WEBSITES

- [1] Gnu Public License Gnu Software Foundation
<http://www.gnu.org>
- [2] RTAI (Real Time Applications Interface) real-time support for SMP and kernel 2.2
<http://www.aero.polimi.it/projects/rtai/index.html>
- [3] New Mexico Institute of Technology real time Linux - <http://www.rtlinux.org>
- [4] Real-time Linux Benchmarks (Redistributed with permission from Laboratorio de Electronica, Dto. de Fisica. Fac.de Ciencias Exactas. Universidad Nacional de La Plata. Argentina)
<http://www.zentropix.com/support/document/testdata.html>
- [5] Online Real-Time Linux Documentation
<http://www.zentropix.com/support/document.html>

REAL-TIME LINUX - HOW DOES IT WORK?

In the real-time Linux architecture, the Linux operating system is the lowest priority task under a small real-time operating system. From the Linux viewpoint there are no changes in its operation, from the standpoint of the user or the Linux kernel, except that it is only permitted to execute when there are no real-time tasks executing.

Real-time Linux provides the capability of running special real-time tasks and interrupt handlers on the same machine as standard Linux. These tasks and interrupt handlers execute whenever they need to, regardless of what Linux is doing. Real-time Linux extends the standard Linux programming environment to real-time problems. Real-time Linux tasks and interrupt handlers can communicate with ordinary Linux processes through a device interface or shared memory.

Real-time Linux treats the Linux operating system kernel as the idle task executing under the real-time operating system, causing standard Linux to only run when there are no real-time tasks to execute. The Linux task can never block interrupts or prevent itself from being preempted. The mechanism that makes this possible is the software emulation of interrupt control hardware.

When any code in Linux tries to disable interrupts, the real-time system intercepts the request, records it, and returns to Linux. In fact, Linux is not permitted to ever disable hardware interrupts, and hence, regardless of the state of Linux, it cannot add latency to the interrupt response time of the real-time system. When an interrupt occurs, the real-time kernel intercepts the interrupt and decides what to dispatch. If there is a real-time handler for the interrupt, the appropriate handler is invoked. If there is no real-time interrupt handler, or if the handler indicates that it wants to share the interrupt with Linux, then the interrupt is marked as pending. If Linux has requested that interrupts be enabled, any pending interrupts are enabled, and the appropriate Linux interrupt handler invoked, with hardware interrupts re-enabled. Regardless of the state of Linux, whether it is running in kernel mode, running a user process, disabling or enabling interrupts, the real-time system is always able to respond to an interrupt.

Real-time Linux decouples the mechanisms of the real-time kernel from the mechanisms of the general purpose kernel so that each can be optimized independently, and so that the real-time kernel can be kept small and simple. Real-time Linux has been designed so that the kernel

never waits for the Linux side to release any resources. The real-time kernel does not request memory, share spin locks, or synchronize data structures, except in tightly controlled situations. For example, the communication links that are used to transfer data between real-time tasks and Linux processes are non-blocking on the real-time side. There is never a case where the real-time task waits to queue or de-queue data.

Real-time Linux currently provides two communication mechanisms between real-time tasks and Linux, shared memory and RT fifos. RT fifos provide a standard UNIX device interface between Linux processes and real-time tasks which allows Linux processes to exchange data with real-time tasks, and shared memory allows areas of memory to be shared between real-time tasks and Linux processes.

One of the fundamental design philosophies of real-time Linux is to let the Linux operating system do as much as is practicable. Typical examples include system and device initialization, and blocking dynamic resource allocation. Any thread of execution that can be blocked when there are no available resources cannot have real time constraints.

Real-time Linux relies on the Linux loadable module mechanism to install components of the RT system, which keeps the RT system extensible and modular. Loading an RT module is not a real time operation, and so it can be done by Linux. The primary function of the RT kernel is to provide direct access to the raw hardware for real time tasks so that they can execute have minimal latency and maximal processing when required.

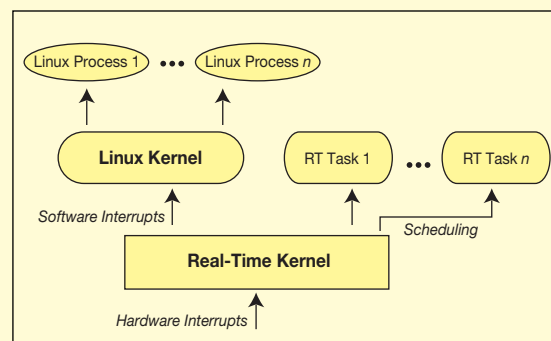


Figure 1. The RealTime Linux Interrupt Abstraction Layer

MARKET-TRENDS

James Norton is a co-founder and CEO of The Zentropic Computing Company. A self-proclaimed failure as an engineer, he comes to Linux from a thirty year career in commercial and military flight simulation. His last job in the world of the "reality challenged" was as director of program management for Hughes Training. He would most like to be remembered for his world renowned book "Bureaucracy in human organizations, why it occurs and how to avoid it". But he hasn't written it yet. He is also co-

founder of Advanced Simulation Technology, a successful privately held US company that produces DSP communication products for the simulation and training industry.

Clark Roundy is Vice President of Marketing and Sales at Alta Technology Corporation. He is a 10-year veteran of the embedded real-time systems industry. In addition to his passion for being on the leading edge of technology, he enjoys reading such books as "Bureaucracy in human organizations, why it occurs and how to avoid it".