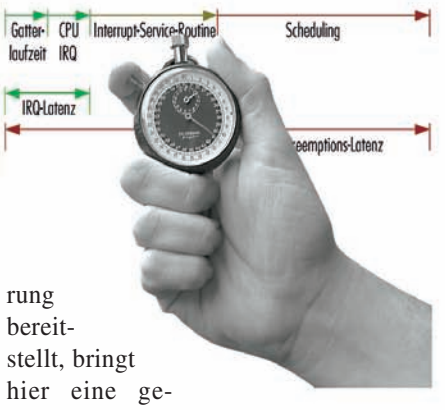
Computer & AUTOMATION

Echtzeit inklusive

Carsten Emde, Robert Schwebel

Mit Aufnahme von RT-Preempt-Patches in den Mainline-Kernel ist Linux echtzeitfähig geworden. Werden damit die anderen Echtzeit-Erweiterungen für Linux obsolet?



(Grafik: Computer & AUTOMATION, Bilder: OSADL, Trumpf)

Zwei Open-Source-Methoden konkurrieren, um Linux mit Echtzeit-Eigenschaften auszustatten: das Dual-Kernel-Konzept und die seit Sommer 2006 im Mainline-Kernel von Linux enthaltenen RT-Erweiterungen basierend auf den RT-Preempt-Patches.

Bei Dual-Kernel-Konzepten kontrolliert ein separater Echtzeit-Kernel wie RTAI/ Xenomai (Real-Time Application Interface) den eigentlichen Linux-Kernel und übernimmt die Interrupt-Verarbeitung und das echtzeitfähige Scheduling. Das sorgt für das Echtzeit-Verhalten und das Betriebssystem kann weitgehend unverändert betrieben werden. Gleichzeitig ist die nicht-echtzeitgemäße Verarbeitung normaler Linux-Applikationen möglich.

Bei diesem Konzept werden die echtzeitpflichtigen Applikationen mit einem Framework programmiert. Die Achillesferse dieses Konzepts ist dessen exklusive Funktionsbibliothek, denn die damit entwickelten Softwarekomponenten lassen sich nicht einfach auf andere Systeme portieren. Xenomai, das einen API-Plug-in-Layer für die Programmierung bereitstellt, bringt hier eine gewisse Entlastung. Darüber hinaus bietet das Xenomai-API weitere, Posix-konforme Funktionen (Posix: Portable Operating System Interface for Unix).

Echtzeit-API für Linux

Bei der zweiten Variante wurden die Schwachstellen von Linux in Bezug auf seine Echtzeit-Eigenschaften von Ingo

Molnar, Thomas Gleixner und anderen Entwicklern der RT-Preempt-Patches analysiert und eliminiert. Diese RT-Preempt-Patches enthalten Elemente wie Interrupt-Threading, High-Resolution-Timer, Priority-Inheritance und Realtime-Mutexe sowie eine vollständige Kernel-Preemption und Diagnose-Tools. Preemption bezeichnet die Fähigkeit eines hoch priorisierten Prozesses, einen niedriger priorisierten Prozess zu unterbrechen – speziell dann, wenn dieser Prozess gerade einen Kernel-, Speicher, I/O- oder andere System-Funktionen ausführt. Bis einschließlich Kernelversion 2.4 war Mainline-Linux nicht preemptiv und damit für Echtzeit-Anwendungen praktisch ungeeignet – zumindest ohne Dual-Kernel-Erweiterung.

Im Juli 2006 wurde auf dem Kernel-Summit in Ottawa entschieden, wesentliche Bestandteile dieser RT-Preempt-Patches mit der Version 2.6.18 des Linux-Kernels zu übernehmen. Damit ist Linux offiziell auf dem Weg, ein Echtzeit-Betriebssystem (RTOS) zu werden. Bis Ende 2007 sollen die restlichen RT-Komponenten, unter anderem spezielle Timer sowie Diagnose- und Optimierungshilfen im Mainline-Kernel, verfügbar sein.

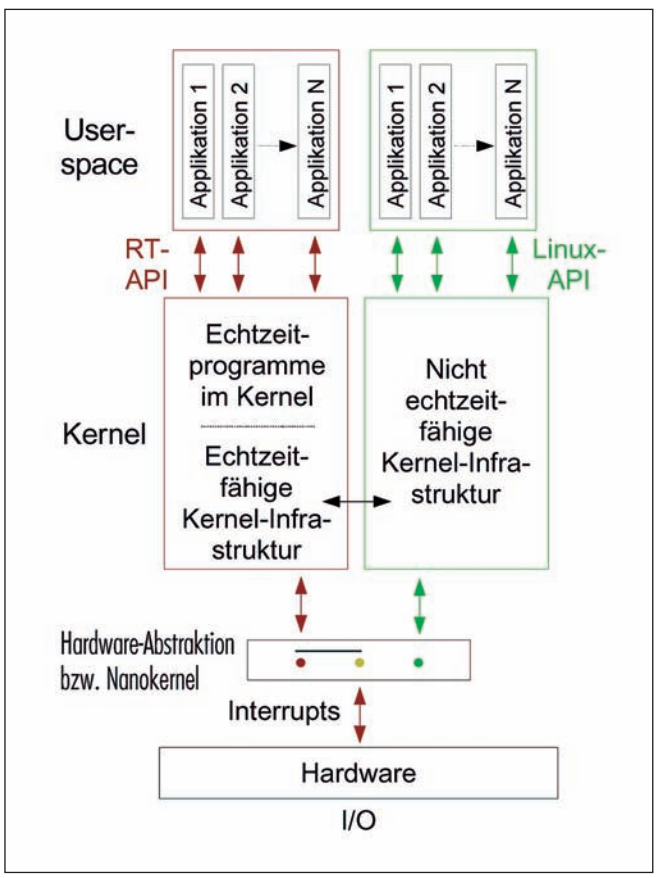
Die Programmierung und Nutzung der im Mainline-Kernel integrierten Echtzeit-Fähigkeit geschieht grundsätzlich anders als beim Dual-Kernel-Konzept: Für die Interprozess-Kommunikation, die Programmierung des Zeitverhaltens und anderer Mechanismen werden vorzugsweise in Posix standardisierte Funktionen verwendet. Deswegen lässt sich die damit erstellte Software leichter auf andere Echtzeit-Systeme portieren.

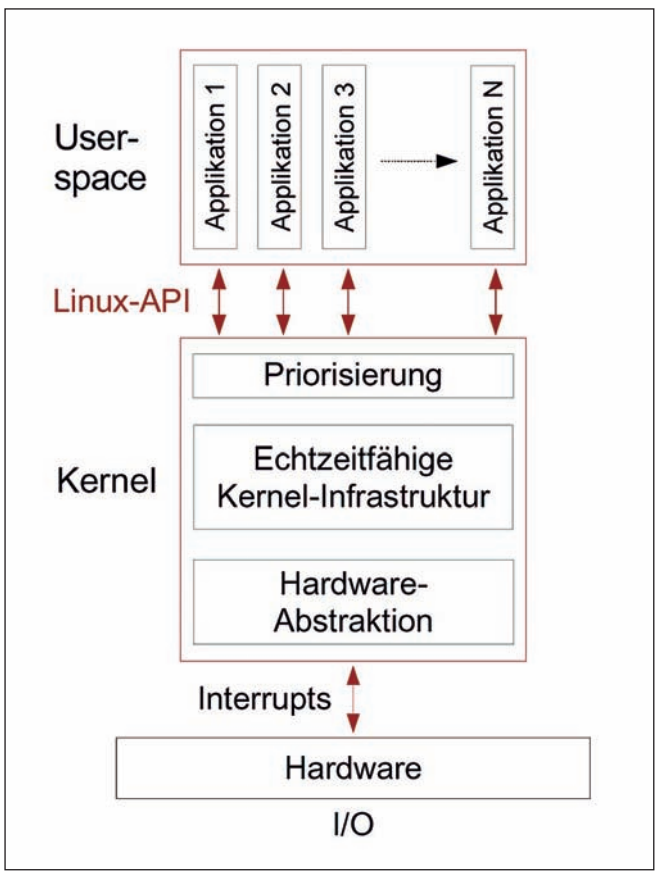
In Bezug auf die Treiber-Auswahl bestehen wohl die größten Unterschiede zwischen den beiden Konzepten. RTAI/Xenomai unterstützt ein breites Spektrum von Protokollen und Kontrollern, für die es unter dem Linux-Kernel mit RT-Preempt-Patches noch keine Treiber gibt. Dagegen punktet der Ansatz mit RT-Preempt-Patches aufgrund der Integration in den Mainline-Kernel von Linux bei der Langzeitstabilität und -weiterentwicklung.

RT-Systeme – Markt im Umbruch

Bislang werden in erster Linie proprietäre Echtzeit-Betriebssysteme wie OS-9, QNX und VxWorks eingesetzt. Allerdings steigen die funktionalen Anforderungen an diese Betriebssysteme,

Echtzeit-System mit Dual-Kernel-Design: Ein Hardware-Abstractionlayer bei RTAI beziehungsweise der ADEOS-Nanokernel bei Xenomai sind für die Verarbeitung sämtlicher Interrupts zuständig.

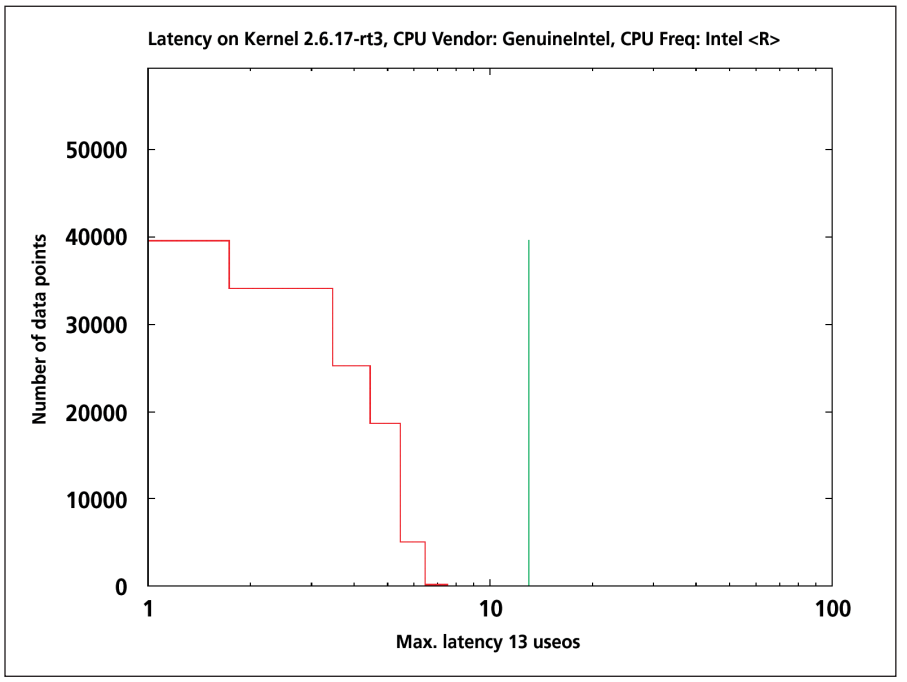




Single-Kernel-Design: Mit RT-Preempt-Patches ist Mainline-Linux ab Version 2.6.18 ein Echtzeit-System.

speziell im Anlagen- und Maschinenbau: Unterstützung von CD und DVD, ISDN-Einwahl, Internet-Anschluss, USB-Interface, WLAN und sogar Bluetooth sind immer häufiger geforderte Systemeigenschaften. Zudem müssen die Systeme eine kaum noch zu übersehende Vielfalt an Hersteller-spezifischen Schnittstellenkarten und Controllern unterstützen. Diese Vielfalt können proprietäre Echtzeit-Betriebssysteme immer weniger erfüllen. Daher liegt es für Anlagen- und Maschinenbauer nahe, auf ein Open-Source-Betriebssystem wie Linux umzusteigen. Hier sorgt die weltweite Entwicklergemeinde für die notwendige Hardware- und Software-Unterstützung. Als einziger Nachteil galt bislang die fehlende Echtzeit-Fähigkeit.

Die Performancegrenze eines echtzeitfähigen Linux liegt zurzeit zwischen 10 und 100 µs Latenzzeit (siehe *Kasten* „Das kleine Einmaleins“) – entsprechend leis- tungsfähige Prozessoren vorausgesetzt. Verlangt die Applikation eine geringere Latenz, ist eine dedizierte Hardware-Lösung auf Basis eines DSPs (Digitaler Signal-Prozessor) meist kostengünstiger. Liegen die Anforderungen darüber, lässt sich eine Linux-Echtzeit-Lösung mit einem Standard-Prozessor einsetzen. Hier unterstützen die RT-Preempt-Patches von Linux inzwischen alle gängigen i386-, PowerPC- und ARM-Prozessoren. Für die Entscheidung, welches Echtzeit-Konzept – RTAI/Xenomai oder Linux-Kernel mit RT-Preempt-Erweiterungen – am besten geeignet ist, spielen Faktoren eine Rolle wie die Art der benötigten Treiber, Zeitpunkt der Inbetriebnahme und Verfügbarkeit entsprechender Board-Support-Packages.



Latenzmessung: Die grüne Linie kennzeichnet die „worst-case latency“.

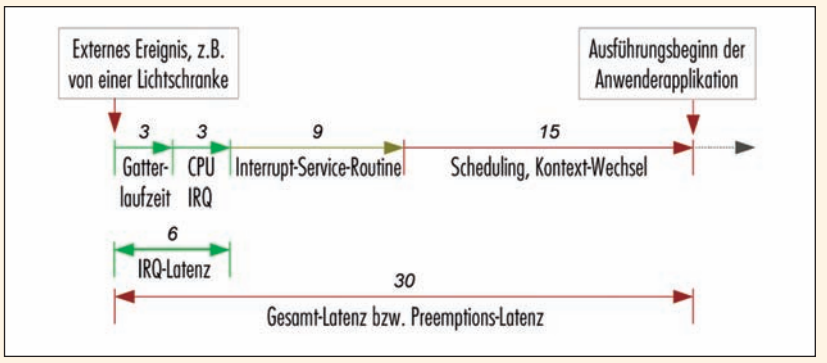
Echtzeit betrifft das Gesamtsystems

Echtzeit-Fähigkeit bezieht sich nicht auf die Reaktionsgeschwindigkeit eines Systems, sondern auf die Zuverlässigkeit der Reaktion. Entscheidend ist die Fähigkeit eines Gesamtsystems, auf externe Ereignisse innerhalb eines definierten Zeitfensters zu reagieren. Dazu ein Beispiel: Bei einer Sortieranlage muss eine Weiche auf den Impuls einer Lichtschranke und das Ergebnis einer Computer-Berechnung innerhalb einer bestimmten Zeit reagieren. Andernfalls kommt es zu einer Fehlfunktion. Innerhalb des vorgegebenen Zeitin-tervalls muss über Sensor und I/O-Karte ein Prozess auf der CPU angestoßen, die

Latenzzeiten

Das kleine Einmaleins

IRQ-Latenz: Intervall zwischen Spannungssprung und Start der Interrupt-Service-Routine (ISR).
IRQ-Latenz = Gatterlaufzeit + CPU-IRQ-Umschaltzeit
Scheduling-Latenz: Zeitintervall von Beendigung der ISR bis zum Start der Anwenderapplikation.
Preemptions-Latenz: Gesamtzeit vom Spannungssprung bis zur ersten Instruktion des aufgeweckten Prozesses.
Preemptions-Latenz = IRQ-Latenz + ISR-Verarbeitung + Scheduling-Latenz.



Typischer Ablauf einer Interrupt-Verarbeitung (Angaben in µs)

OSADL

Linux in der Automation

Die Linux-Erweiterung RT-Preempt sowie andere Aspekte des Einsatzes von Open-Source-Software sind für die Automatisierungsindustrie besonders relevant. Die Genossenschaft der Open Source Automation Development Lab (OSADL) setzt sich für die Entwicklung der für die Automatisierungsbranche benötigten spezifischen Erweiterungen von Linux ein. Das OSADL hat für die Echtzeitfähigkeit des Linux-Kernels wesentliche Softwareprojekte angestoßen und finanzierte die Anpassung der PowerPC- und ARM-Architekturen an die Realtime-Preempt-Patches.

Operation ausgeführt und ein Stellsignal an die Weiche ausgeben werden.

Das zeigt, dass nur ein Gesamtsystem im Hinblick auf seine Echtzeit-Fähigkeit (Reaktionszeit) beurteilt werden kann. Eine CPU-Karte, ein I/O-Adapter, ein Betriebssystem oder eine Applikation können für sich echtzeitfähig sein; für ein aus diesen Komponenten zusammengesetztes System gilt dies aber noch lange nicht. Daher sollten echtzeitpflichtige Systeme vor der Freigabe immer daraufhin untersucht werden, ob sie die geforderte Echtzeit-Fähigkeit erreichen.

Dazu wird möglichst häufig das Zeitintervall – auch als Preemption-Latenz bezeichnet – zwischen dem Eintreffen des externen Signals und dem Programmstart der Applikation gemessen. Anzugeben ist natürlich nicht ein Einzelwert oder der Mittelwert der Latenzzeiten, sondern deren Verteilung und speziell der höchste Wert – die „worst-case latency“.

Stationen eines Interrupt

Bei dem Beispiel läuft das Anwendungsprogramm im sogenannten Userspace des Betriebssystems und setzt Echtzeit-Eigenschaften sowohl im Kernel als auch bei der Benachrichtigung des Ereignisses an den Userspace-Prozess voraus.

Unmittelbar nach dem Spannungssprung am Eingang des I/O-Controllers durchläuft der Impuls die verschiedenen Gatter des Controllers und löst schließlich einen CPU-Interrupt aus. Die Unterbrechung bewirkt, dass die CPU zum nächstmöglichen Zeitpunkt auf die Adresse des Interrupt-Dispatcher im Kernel verzweigt, der die Kontrolle über die CPU an die Interrupt-Service-Routine des Treibers weitergibt.
Diese Operationen benötigen bei einfachen CPU-Architekturen nur wenige CPU-Takte. Bei Architekturen mit massiver Pa-rallelverarbeitung, können aber durchaus mehrere Mikrosekunden vergehen, bevor die Programmausführung in der Interrupt-Service-Routine fortgesetzt wird. Das Intervall zwischen Interrupt (Spannungssprung) und Start der Interrupt-Service-Routine (ISR) wird *IRQ-Latenz* genannt und hängt ausschließlich von der Hardware ab.

Nach Bearbeitung der ISR geht die Kontrolle an den RT-Kernel zurück. Er entscheidet, welcher Prozess aufgrund des Interrupt „aufgeweckt“ werden muss. Dieser Prozess erhält dann unter Berücksichtigung seiner Priorität eine Zeitscheibe zur Programmausführung. Die Dauer dieses Vorgangs wird *Scheduling-Latenz* genannt. Die Gesamtzeit vom Spannungssprung am Eingang des I/O-Controllers bis zur ersten Instruktion des aufgeweckten Prozesses entspricht der *Preemption-Latenz*. Sie stellt die wichtigste charakteristische Kenngröße eines Echtzeit-Systems dar.

Ungewollte Sabotage

Ein System kann seine Echtzeit-Fähigkeit verlieren, wenn Komponenten hinzugefügt werden, welche die Reaktionsgeschwindigkeit herabsetzen, beispielsweise ein Treiber, der die Interrupt-Verarbeitung länger als nötig blockiert. Ein spezielles Problem stellt das Phänomen der Priority-Inversion dar: System-Ressourcen wie Operationen bei der Speicherverwaltung können nicht gleichzeitig von verschiedenen Prozessen ausgeführt werden und sind deswegen über sogenannte Mutexe verriegelt (MUTual EXclusion: gegenseitiger Ausschluss). Belegt nun ein niedrig priorisierter Prozess eine solche Ressource, die ein höher priorisierter Prozess benötigt, kann dieser Prozess nicht mehr weiterarbeiten. Damit ist dessen Priorität invertiert. Eine Möglichkeit, dieses Problem zu umge- hen, besteht darin, die Zugriffe auf diese Ressource immer mit maximaler Priorität auszuführen. Bei einem anderen Verfahren „erbt“ die Ressource die Priorität des aufrufenden Prozesses. Dieses ist im Linux-Kernel ab Version 2.6.18 als PI-Mutexe (PI: priority inheritance) implementiert. *sk*

Carsten Emde

ist Inhaber der Firma Computer Experts und Geschäftsführer des Open Source Automation Development Lab (OSADL).

Robert Schwebel

ist Inhaber der Firma Pengutronix und Gründungsmitglied des OSADL.

Nähere Informationen:
www.rtai.org
http://rt.wiki.kernel.org/
http://lwn.net/Articles/191782/
www.osadl.org

Kontaktinformation:

Open Source Automation
Development Lab (OSADL) eG
Homagstr. 3-5
72296 Schopfloch
Tel +49(7443)13-3073
Fax +49(7443)13-8-3073
http://www.osadl.org
info@osadl.org