



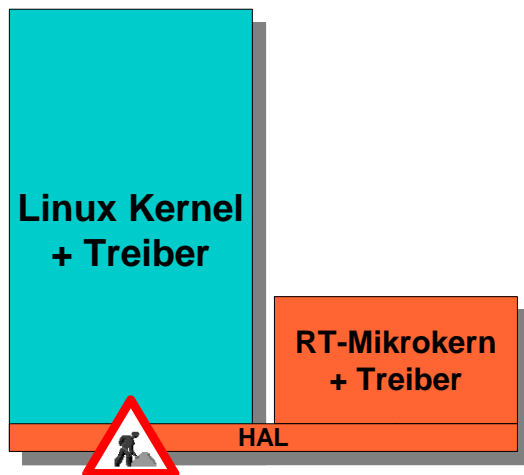
Echtzeit-Linux durch RTAI

Universität Hannover
ISE – Fachgebiet Echtzeitsysteme

Inhalt

- **Einführung**
- **RTAI/fusion: Konzept und Realisierung**
- **2-Seiten-Tutorial**
- **Werkzeuge**
- **Bewertungen und Ausblick**

Harte Echtzeit für Linux



- RTLinux
- RTAI Classic, RTAI/fusion



- RT-Preemption-Patch
- LibeRTOS
- (RTAI/fusion)

Weitere Dual-Kernel-Ansätze durch Mikrokernel:

- Linux auf L4-Mikrokernel, L4Linux (TU Dresden)
- Linux auf PikeOS (Sysgo, kommerziell)
- Linux auf Jaluna OSware (Jaluna, kommerziell)
- Linux auf Integrity (Green Hills, kommerziell)
- und wohl noch einige andere...

Probleme:

- In der Regel müssen alle Treiber echtzeitfähig für den Mikrokernel entwickelt werden
- Teilweise sehr unkomfortable, harte Trennung zwischen Echtzeit- und Nicht-Echtzeitanwendungen

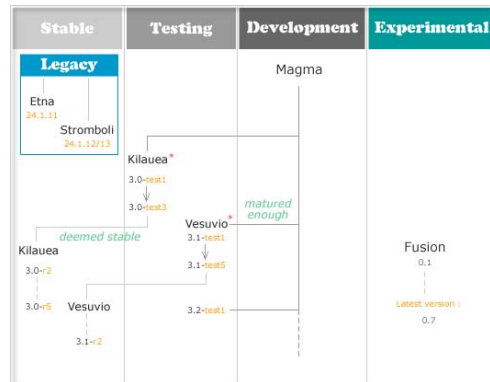
Die RTAI-Geschichte

- In Teilen Abspaltung früher RTLinux-Version (1999)
- Weltweite Entwicklergruppe um Paolo Mantegazza, DIAPM (Department of Aerospace Engineering, Politecnico di Milano)
- **Userspace Real-Time Support** (1999)
- Populäre 24.1.x-Serie für 2.4er-Kernel (2000-2003)
- Parallel: Permanenter Streit um RTLinux-Patent und Copyright
- Umstellung auf Adeos-Kernel-Patch von Philippe Gerum (2003)
- **RTAI 3.1 für 2.6er-Kernel** (2004)
- **Neuer Entwicklungszweig „Fusion“** des zweiten Maintainers Philippe Gerum, Openwide (2004)

...to be continued!

Von Vulkanen und Versionen

- **Vulkannamen bezeichnen Entwicklungszweige der Classic-Serie**
- **Aktuelle Namen:**
 - Vesuvio = 3.1 (stabil)
 - Vulcano = 3.2 (im Test)
- **Neuer Entwicklungszweig Fusion steht „abseits“**
 - diesmal kein Vulkan
 - eigene Versionsnummer, aktuell 0.7 (29.03.2005)
- **Namen gelten ebenfalls im CVS**
- **Über Sinn und Zweck ließe sich vortrefflich streiten...**



* Snapshot: brand new symbolic name

Inhalt

- Einführung
- ▶ **RTAI/fusion: Konzept und Realisierung**
- 2-Seiten-Tutorial
- Werkzeuge
- Bewertungen und Ausblick

RTAI/fusion im Überblick

- Vollständig neue Code-Basis
- Nutzt Adeos-gepatchten 2.6er-Kernel
- Kernel-unabhängige Hardware-Abstraktionsschicht (wie Classic-RTAI)
- Xenomai: Generischer Scheduler ohne User-API
- Nutzer-APIs als austauschbare „Skins“
- Unterstützung für Kernel- und Userspace-Echtzeit
- RTAI-Skin als Brücke zur Classic-Serie
- Plattformen: x86, PPC

Anwendung

RTOS-Skin

Xenomai

HAL

Adeos

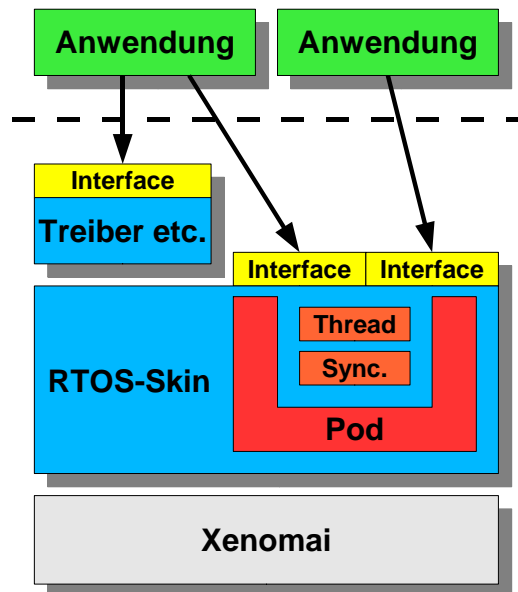
Der Xenomai-Nanokernel

- **Abstraktionen**
 - **Threads**
 - **Generische Synchronisationsobjekte**
 - Timer / Systemuhr
 - Echtzeitspeicher (Heaps)
 - Interrupts
 - Pipes (RT-NRT-Kommunikation)
 - **Features**
 - Scheduling: prioritätsbasiert (mit Prioritätsvererbung), Round-Robin
 - periodischer System-Timer oder One-Shot-Modus
 - **Userspace-Threads** („Shadow“-Threads)
 - **Userspace-Interfaces** (Syscall-Erweiterungen)
 - Thread-Signals, FPU, SMP, Software-Watchdog, ...
- => Plattform für beliebige RTOS-APIs**

The Xenomai logo is displayed in a rectangular box. The word "Xenomai" is written in a stylized, serif font with a slight shadow effect, giving it a three-dimensional appearance.

Skins, Pods, Interfaces?

- Skin implementiert API eines bestimmten RTOS
- Skin registriert Pod als Organisationsstruktur gegenüber Xenomai
- Skin oder Erweiterungen bieten API
 - für andere Kernelmodule per Symbolexport
 - für Userspace-Programme per Interfaces
- Verfügbare Skins
 - POSIX
 - pSOS+
 - RTAI
 - VRTX
 - uITRON
 - VxWorks



Am Fachgebiet Echtzeitsysteme entsteht z.Z. ein auf Security ausgerichteter Xenomai-Skin. Ziele:

- Praktische Anwendung eines spezielle für Echtzeitsysteme entworfenen Sicherheitsmodells
- Sichere Ausführung von potentiell vertrauensunwürdigen Programmen mit Echtzeitprivilegien
- Weitgehend POSIX-konforme API
- Integration von Echtzeit-Hardware-Treibern in die Erzwingung des Sicherheitsmodells

Kontakt: Jan Kiszka <kiszka@rts.uni-hannover.de>

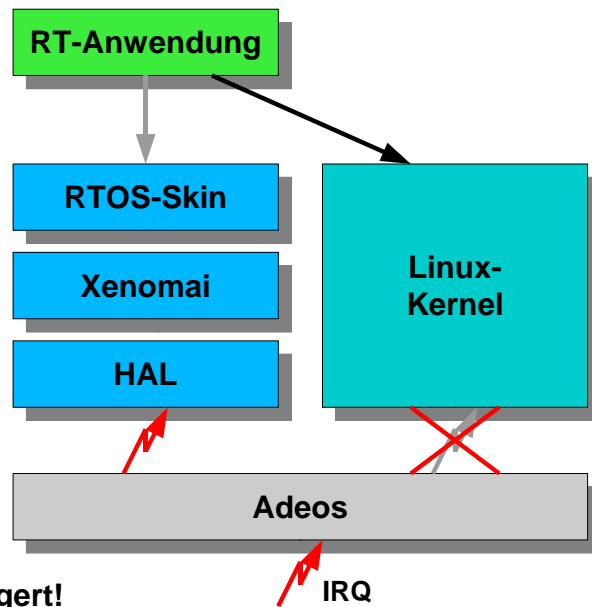
RTAI Native Skin

- Ziel: **Ersatz der RTAI-3.x-API**
- Kernel- und Userspace
- Monolithische API:
nur ein Modul, Programmierung nur gegen den Skin
- Reguläres Schema für Funktionsnamen:
rt_<Klasse>_<Funktion>
z.B. rt_timer_start, rt_timer_stop
- Invertiertes Prioritätsschema:
mit Zahlenwert aufsteigende Prioritäten (1..99)
- Zeitrepräsentation:
durchgängig in Nanosekunden
- weitere Details:
`fusion/skins/rtai/doc/refactoring.txt`
=> RTAI-Classic-Anwendung müssen umgeschrieben werden



Fusion: „Pervasive Real-Time“

- Userspace-Echtzeit-Threads können Standard-Linux aufrufen
- Interrupts für Linux-Kernel bleiben gesperrt
- Kernel erhält Priorität des RT-Threads
- Sehr vereinfacht: **Ähnlich LXRT, nur „echtzeitfähiger“**
- Aber: **Determinismus kann zerstört werden!**
- Niedriger priorisierte Threads werden u.U. undeterministisch verzögert!



Wer mehr Information sucht:

<http://www.rta.org/modules.php?name=Content&pa=showpage&pid=1>

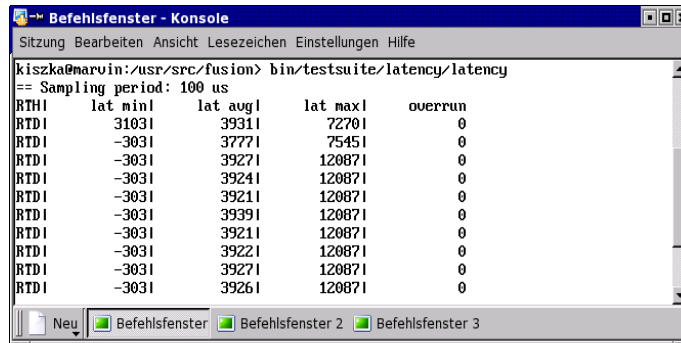
Darüber hinaus existieren konkrete Pläne der Entwickler, Fusion mit dem RT-Preemption-Patch zu kombinieren. Die Hoffnung: mehr Kernel-Funktionen können deterministisch im Secondary Mode ausgeführt werden.

Inhalt

- Einführung
- RTAI/fusion: Konzept und Realisierung
- ▶ **2-Seiten-Tutorial**
- Werkzeuge
- Bewertungen und Ausblick

Echtzeit-Linux in zwei Slides (1)

- Kernel (z.B. 2.6.10) und Fusion (ab 0.7) herunterladen, entpacken
- Kernel mit Adeos patchen
- Kernel-Konfiguration des laufenden Systems übernehmen, ggf. anpassen (z.B. kein Power-Management)
- Kernel kompilieren, installieren, System neu starten
- Fusion konfigurieren (Pfade setzen), kompilieren, installieren
- Erster Test:



```
Sitzung Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
kiszka@marvin:/usr/src/fusion> bin/testsuite/latency/latency
== Sampling period: 100 us
RTH|  lat min|  lat avg|  lat max|  overrun
RTD|  3103|  3931|  7270|  0
RTD|  -303|  3777|  7545|  0
RTD|  -303|  3927|  12087|  0
RTD|  -303|  3924|  12087|  0
RTD|  -303|  3921|  12087|  0
RTD|  -303|  3939|  12087|  0
RTD|  -303|  3921|  12087|  0
RTD|  -303|  3922|  12087|  0
RTD|  -303|  3927|  12087|  0
RTD|  -303|  3926|  12087|  0
```

Echtzeit-Linux in zwei Slides (2)

- `gcc -o fusion-demo fusion-demo.c`
`-I <FusionInstDir>/include`
`-L <FusionInstDir>/lib -lrtai -lpthread`
- `./fusion-demo`

```
void demo(void *arg)
{
    rt_task_set_periodic(NULL,
        TM_NOW, 200000);

    while (1) {
        rt_task_wait_period();

        /* Echtzeitaufgaben */
    }
}
```

```
int main(int argc, char* argv[])
{
    signal(SIGINT, catch_signal);
    mlockall(MCL_CURRENT|MCL_FUTURE);
    rt_timer_start(TM_ONESHOT);
    rt_task_create(&demo_task,
        "kdemo",
        0, 99, 0);
    rt_task_start(&demo_task,
        &demo, NULL);

    pause();
    rt_task_delete(&demo_task);
    rt_timer_stop();
    return 0;
}
```

Zusätzliche Hinweise:

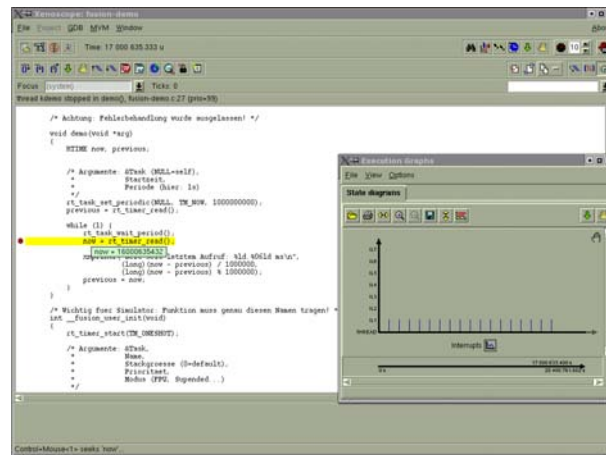
- `librtai.so.0` muss zur Ausführung von `rtaidemo` im `LD_LIBRARY_PATH` stehen oder nach `/usr/lib` etc. kopiert werden
- Beispielpaket `fusion-demo.tar.bz2` enthält Kernel-/Userspace-Demo + LXRT-Demo
- Fertige Makefiles dem Beispielpakets verlangen Aufrufparameter:
`make FUSIONDIR=<Fusion-Installationsordner>`
`make RTAIDIR=<RTAI-3.x-Installationsordner>`
- Weitere Beispiele (Code-Fragmente) in `skins/rtai/snippets`
- Userspace-Entwicklung in C++ sollte ebenfalls kein Problem darstellen...

Inhalt

- Einführung
- RTAI/fusion: Konzept und Realisierung
- 2-Seiten-Tutorial
- ▶ **Werkzeuge**
- Bewertungen und Ausblick

Minute Virtual Machine

- Ereignisgesteuerte Simulations-Engine
 - Ausführung von
 - Xenomai
 - Kernel-Skins
 - Kernel-Anwendungen
 - Grafisches Frontend Xenoscope
 - Eingebauter Debugger
 - Ausführungsgraph
 - Kernel-Objektinspektor
 - API-Tracer (in Vorbereitung)
- ...und weitere Funktionen – einfach einmal ausprobieren



Externe Abhängigkeiten:

- gcc-2.93.3 (tar-Archiv ist während der Konfiguration anzugeben)
- gdb 5.x (im Vorfeld kompilieren)
- tcl/tk8.3-dev oder neuer
- tix4.1 oder neuer
- libelf0-dev
- libpng2-dev

Zur Kompilation:

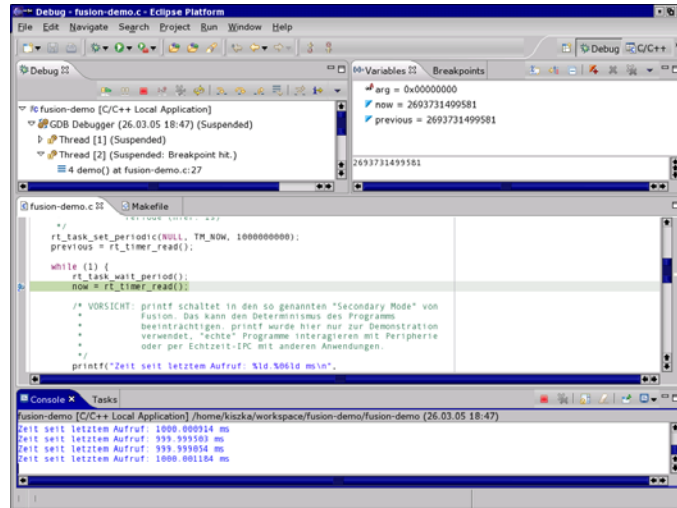
- Die MVM und virtuelle Usermode-Skins in Fusion anschalten
- Zur Anwendungskompilation Handbuch sim/doc/mvm-manual.txt befragen – oder das fusion-demo Beispieldpaket zu diesem Vortrag verwenden

Bei der Ausführung:

- Aufruf mit „xenoscope <Anwendung>
- Pfad zum gdb-5.x-Binary in den Einstellungen eintragen (erst dann Debugger starten!)

Real Debugging

- ...und Userspace-Debugging?
- Ganz knapp: Es geht einfach.



```
rt_task_set_periodic(NULL, TM_NOW, 1000000000);
previous = rt_timer_read();

while (1) {
    rt_task_wait_period();
    now = rt_timer_read();

    /* VORSICHT: printf schaltet in den so genannten "Secondary Mode" von
    * fusion. Das kann den Determinismus des Programms
    * beeinträchtigen. printf wurde hier nur zur Demonstration
    * verwendet. "echte" Programme interagieren mit Peripherie
    * oder per Echtzeit-LPC mit anderen Anwendungen.
    */
    printf("Zeit seit letztem Aufruf: %ld %6.1d ms\n",
```

Inhalt

- Einführung
- RTAI/fusion: Konzept und Realisierung
- 2-Seiten-Tutorial
- Werkzeuge
- ▶ **Bewertungen und Ausblick**

Bewertungen

Fusion ist die Zukunft von RTAI:

- Klares Design, saubere Realisierung
- Gute API-Dokumentation (gilt inzwischen auch für RTAI 3.x)
- Verbesserungswürdig (noch frühes Stadium):
Dokumentation zum Einstieg und zur Übersicht
- Hohe Stabilität, aber noch nicht „produktreif“ (greifbar!)

Allgemeine Vorteile:

- Echtzeitfähige Libraries werden mitgeliefert
- **Starke, wachsende Open-Source-Community**
- Sogwirkung auf andere FOSS-Projekte
- Hohe Rechtssicherheit
 - Keine Patentabhängigkeit, kein gemeinsamer Code mit RTLinux
 - **Userspace erleichtert Closed-Source-Anwendungen**

Ausblick

- **Fusion-Roadmap**
 - ARM-Support [0.8, ca. Mitte Juni]
 - Linux-Trace-Tool-Kit (LTT) [0.8]
 - Treiber-Framework (RTDM) [0.9, ca. August]
 - Userspace-POSIX [0.9]
 - Verbesserungen an Skalierbarkeit und Erweiterbarkeit des Schedulers [1.0, Oktober-Dezember]
- **RTAI-Classic-Roadmap**
 - Vollständige Umstellung auf LXRT-Scheduler [3.2, April?]
 - Langfristig: Plattform für Alt-Programme, insbesondere auf 2.4
 - Bevorzugte Umgebung für „Funktionsvielfalt“ in der API ;-)

Paolo: *„Meanwhile Linux will likely become real time [...] and we all go to pension :-).“*

Philippe: *„I don't think so actually.“*



www.rtai.org

kiszka@rts.uni-hannover.de