

Echtzeit-BS vs. Standard-BS

Unterschiede zwischen Realzeit- und Standard-BS

Einsatz von Standardbetriebssystemen in technischen Bereichen

Vorteile:

- **Applikationsvielfalt**
- **Einarbeitungsaufwand**
- **gute Entwicklungswerkzeuge (bis auf spezielle RT-Werkzeuge)**
- **grafische Benutzeroberfläche mit gewohnter Benutzerführung**
- **lauffähig auf preiswerter Hardware**
- **Einbettung von Parametern aus dem technischen Prozess in Office-Produkte (DCOM/OLE)**



Echtzeit-BS vs. Standard-BS

Unterschiede zwischen Realzeit- und Standard-BS

Einsatz von Standardbetriebssystemen in technischen Bereichen

Nachteile:

- Die preiswerte Hardware ist nicht industrietauglich
- Die Lizenzkosten von Standardbetriebssystemen sind für den Einsatz in eingebetteten Systemen zu hoch (Lizenzgebühren für Echtzeitbetriebssysteme schwanken abhängig von der abgenommenen Anzahl zwischen 1\$ und 50\$)
- Einarbeitungsaufwand in ein Realzeitbetriebssystem fällt bei der Entwicklung an. In dem Fall, dass das Betriebssystem in einem eingebetteten System eingesetzt wird, ist für die Bedienung der Einarbeitungsaufwand irrelevant. Auch die Applikationsvielfalt spielt in diesem Falle keine Rolle



Echtzeit-BS vs. Standard-BS

Unterschiede zwischen Realzeit- und Standard-BS

Einsatz von Standardbetriebssystemen in technischen Bereichen

Nachteile:

- **Realzeitfähigkeit, oftmals nicht deterministisch:**
 - Paging
 - Swapping
 - Caching (Determinismus vs. Performance)
- **File I/O (Determinismus, Interruptlast, Flash/RAM-Filesysteme in Standard-BS oft nicht unterstützt)**
- **Headless und Diskless-Betrieb häufig nicht möglich**
- **Schedulingstrategien (unterschiedliche Ziele zwischen RT- und Standard-BS, kommt gleich)**
- **Abhängigkeit von Dienstprogrammen (Standard-BS erfordert oft viele Services oder Dämonen)**
- **Anforderungen an Ressourcen (Hauptspeicher, Verarbeitungsleistung, Hintergrundspeicher)**



Echtzeit-BS vs. Standard-BS

Standard-Betriebssystem versus Realzeit-Betriebssystem

Standard-Betriebssystem	Realzeit-Betriebssystem
Optimiert für Effizienz und Durchsatz.	Optimiert auf Determinismus.
Sammeln von Systemaufträgen (effizienter Plattenzugriff).	Aufträge werden direkt abgearbeitet.
Abschnitte werden längere Zeit gelockt.	Kurzes Locking.
Kein deterministisches Schedulingverfahren.	Deterministisches Scheduling.
Beim Scheduling wird jede Task berücksichtigt.	Solange hochpriorie Tasks lauffähig sind, bleiben andere Tasks unberücksichtigt.
Headless und Diskless-Betrieb nicht möglich.	OS ist für den eingebetteten Betrieb geeignet.
Paging und Swapping.	Paging und Swapping wird nicht unterstützt.



Echtzeit-BS vs. Standard-BS

Einsatz von Standard-BS in technischen Bereichen

Windows 95/98/XP/7

- Selten, z.B. Visualisierungsplattform (z.B. HP-Logicanalyzer) ohne Realzeitaufgaben
- Nachteilig bei Windows 95/98/XP
 - Ressourcenverbrauch (Rechenzeit, Speicherplatz ...),
 - mangelnde Realzeitfähigkeit
 - Instabilität

Windows NT, Windows Server 2000,2003,2008 R2

- Verwendung z.B. in Bankautomaten
- Für kleinere Systeme wegen hoher Ressourcenanforderungen ungeeignet
- Realzeiteigenschaften (Memory-Locking, RT-Scheduling), nicht deterministisch
- Latenzzeiten bis in den Millisekundenbereich (500 ms ???)
- Es gibt Erweiterungen, die NT realzeitfähig machen



Echtzeit-BS vs. Standard-BS

Einsatz von Standard-BS in technischen Bereichen

Embedded NT (ca. ab 1999)

- Abgespeckte Windows NT Version, benötigt *nur* noch 14-24 MByte
- Embedded NT ist nicht realzeitfähig
- Vorteil: unterstützt Win32s Interface
- Embedded NT läuft nur auf Intel-Prozessoren.

Windows CE/Pocket PC -> Windows Powered, Windows Embedded Compact 7

- Windows CE API: „Kleines“-Subset der PC Funktionalität → kein „1:1“ Sourcecode
- Läuft auch auf Nicht-Intel-Prozessoren (z.B. ARM).
- CE <6.0 : Benötigt 400KByte bis 2.5 Mbyte Hauptspeicher
- Im Gegensatz zu den eigentlichen Windows Varianten und auch Embedded NT kann Windows CE direkt aus dem ROM gestartet werden
- 7: Unterstützt MultiCore und BS bis zu 3 GByte phys. RAM



Echtzeit-BS vs. Standard-BS

Einsatz von Standard-BS in technischen Bereichen

Linux

- verschiedenste Prozessoren (mit und ohne MMU)
- Skalierbar
- lauffähig auf Systemen mit 256 KByte ROM und 1 MByte RAM.
- keine Lizenzkosten
- (noch)-nicht realzeitfähig, immerhin einige Realzeiterweiterungen (Memory-Locking, POSIX-Scheduling) PREEMPTION_PATCH, LOW_LATENCY_PATCH
- MontaVista plant, Linux-Kernel echtzeitfähig zu machen („MobiLinux“, seit ca. April 2005 verfügbar):
- WindRiver (VxWorks) setzt massiv auf eigene Embedded Linux Distribution („Device Software Optimization“)
- Echtzeit-Varianten RTAI, XENOMAI und RTLinux
- Linux für CPUs ohne MMU (Microcontroller): μ CLinux



Echtzeit-BS vs. Standard-BS

Realzeiteigenschaften von Standard-Betriebssystemen

In Windows NT und Linux findet man die folgenden Realzeiteigenschaften

- Memory-Locking
- POSIX-Scheduling
- Disablen von Caches (Determinismus, aber erheblicher Performanceverlust!!!)

In Linux zusätzlich:

- POSIX-Threads

„In Linux findet man Wartezeiten (Zeitpunkt vom Eintritt des Interrupts bis zum Start der Bearbeitung, Latency-Times) von 600 usec und Periodische Tasks werden schon mal um 20 msec auf einem PC mit Pentium Prozessor verzögert...“

„Die Wartezeit unter RT-Linux beträgt weniger als 15 usec und periodische Tasks werden nicht mehr als 35 usec verzögert.“ [Yodaiken, *The RTLinux Manifesto*, 1999].



Echtzeit-BS vs. Standard-BS

Standard-Betriebssysteme in Realzeitanwendungen

Realzeitanwendungen mit unmodifiziertem Standard-BS

Zeitanforderungen im Sekundenbereich (bis einige 10 ms)

- auf Applikationsebene

Zeitanforderungen im Millisekunden- oder Mikrosekundenbereich

- Bearbeitung von Echtzeitaufgaben im Betriebssystemkern, z.B. innerhalb eines Gerätetreibers

Zeitanforderungen im Mikrosekundenbereich

- ISR (Latenzzeit: Interruptsperrung durch Kernel- oder Treiberfunktionen)
 - keine langwierigen Berechnungen
 - meist kein bzw. nur erschwerter Floating Point Zugriff möglich
 - keine komplizierte Programmstrukturen möglich
- Vorteil: Standardbetriebssystem!!!
- Nachteil: Lösung der Echtzeitaufgabe verteilt sich auf unterschiedliche Komponenten und wird auf das Betriebssystem angepasst
- Beispiel aus dem Automatisierungsbereich: TwinCAT



Echtzeit-BS vs. Standard-BS

Realzeiterweiterungen von Standard-Betriebssystemen

Um die Skalierbarkeit und Flexibilität eines Standard-Betriebssystems auch im Realzeitbereich zu bekommen, gibt es die folgenden beiden Möglichkeiten:

- **Erweiterung des Standard-BS um Eigenschaften für harte Echtzeit**
 - Extrem aufwändig
 - Eingriffe an vielen Stellen: Synchronisationspunkte für Datenstrukturen, Scheduling, Interruptverarbeitung, Gerätetreiber, critical sections, memory management,...
 - Zusätzlich: verminderte Effizienz
- **Parallelbetrieb des Standardbetriebssystems mit dem Realzeitsystem.**
 - Echtzeitkern, unter dem ein Standard-BS als Task läuft (gibt's für NT, Linux,...)
 - Wenig aufwändig
 - Standard-BS-Komponenten (z.B. TCP/IP-Stack, ...) nicht von Echtzeittask nutzbar (Verlust des Determinismus)
 - Hardwarevirtualisation (z.B. Realtime Systems) → EchtzeitBS läuft direkt auf der Hardware mit realem Hardwarezugriff aber hat nicht gesamten Speicher zur Verfügung → StandardBS läuft in virtueller Maschine mit gekapselter virtueller Hardware



Echtzeit-BS vs. Standard-BS

RT-Linux

- Entwickelt am *Department of Computer Science* der Universität New Mexico
- Echtzeitkern, der Linux in einer eigenen Task ablaufen lässt (als *ldletask*, wenn keine Echtzeittask bereit)
- Versuch der Linux-Task, Interrupts zu sperren wird
 - abgefangen (Emulation der Interrupt-Hardware) und vermerkt
 - auftretende Interrupts im RT-Kontext beantwortet oder
 - zurückgestellt (*pending*), bis Linux Interrupts wieder freigibt
- Interrupt sperren durch Linux führt zu keiner Latenzzeit

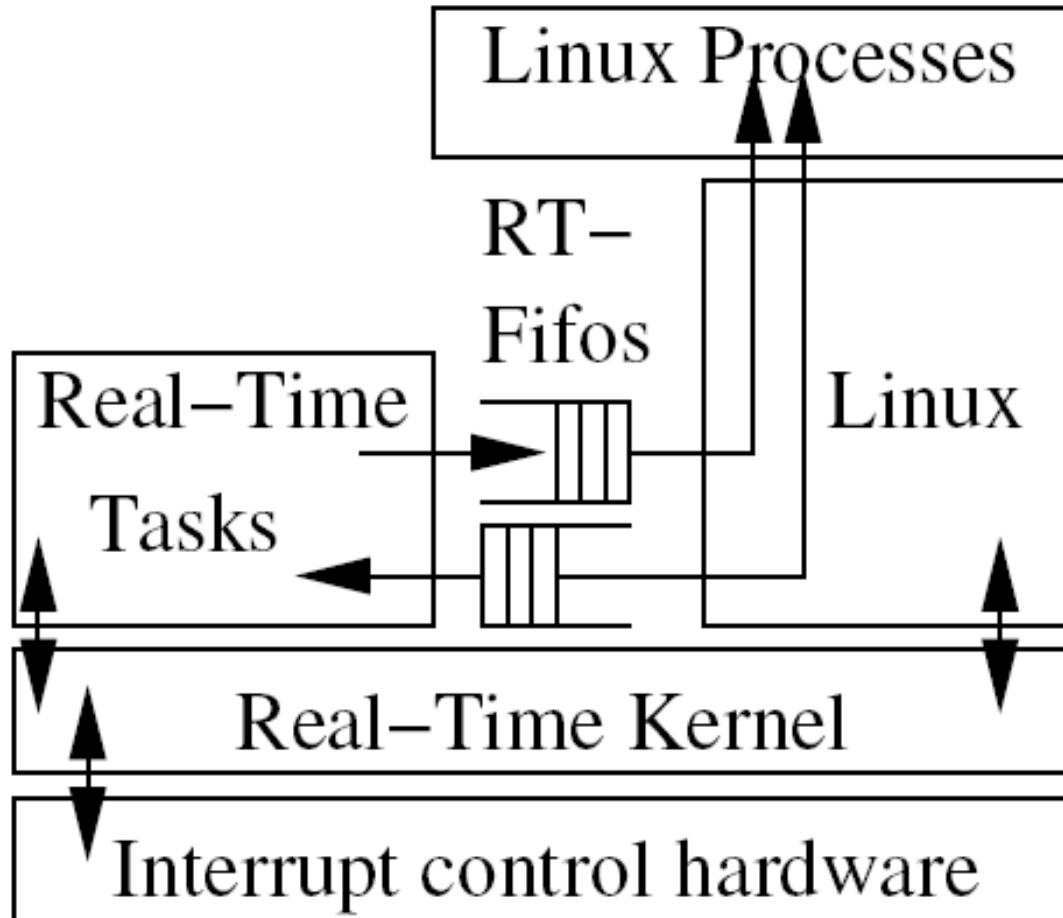
Standard- und Echtzeitbetriebssystem parallel betreibbar,
Informationsaustausch:

1. Real-Time-Fifos und
2. Shared-Memory



Echtzeit-BS vs. Standard-BS

RT-Linux



Echtzeit-BS vs. Standard-BS

RT-Linux

- **RTLinux mittlerweile kommerzialisiert, freie Version verfügbar, allerdings ohne Weiterentwicklung**
- **RTAI /XENOMAI sehr ähnlich, sehr aktive Entwicklercommunity**
- **Problem: RT-Entwicklungswerkzeuge fehlen (s. später)**
- **Geeignet für einfache Anwendungen**



Echtzeit-BS vs. Standard-BS

Trend: Mehrere BS auf einer CPU

z.B. VMWare, XEN,...

Auch möglich mit RT-BS:

- **VxWin, CeWin**
 - Windows + VxWorks/WinCE
 - Ähnlich RT-Linux: Non-RT-BS als Idle-Task des RT-BS
- **PikeOS**
 - „BS-Partitionen“
 - Mikro-Kernel
 - Verwaltung von Zeitquanten pro BS-Partition!!!

