Einführung in Real-Time Linux (Preempt-RT)

Stefan Assmann
Red Hat
2010

Agenda

- Was bedeutet Real-Time
- Wir bauen einen Real-Time-Kernel
- Kernel Preemption
- Linux Scheduling Richtlinien
- Prioritäten in Linux
- Speichermanagement
- Real-Time Hello World Programm
- Real-Time OS Tests
- Herausforderungen / Zusammenfassung
- Fragen ?

Was bedeutet Real-Time

Real-Fast ?

Was bedeutet Real-Time

- DIN 44300

  Echtzeitbetrieb ist ein Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig derart betriebsbereit sind, daß die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind.

- Ergebnis muss innerhalb einer vorgegebenen Zeit verfügbar sein.
- Ergebnis muss korrekt sein.

Was bedeutet Real-Time

- Aus Sicht eines Real-Time OS
  - Hohe Geschwindigkeit (nice to have)
  - Kurze Reaktionszeit (nice to have)
  - **Deterministisches Verhalten**

- Real-Time OS ist nicht die eierlegende Wollmilchsau!

Performance vs. Determinismus

Wir bauen einen RT-Kernel

```
# wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.33.tar.bz2
# wget http://www.kernel.org/pub/linux/kernel/projects/rt/patch-2.6.33-rt4.bz2

# tar xjf linux-2.6.33.tar.bz2
# bunzip2 patch-2.6.33-rt4.bz2

# cd linux-2.6.33
# patch -p1 < ../patch-2.6.33-rt4

# make menuconfig
# make ; make modules_install ; make install
```

Processor type and features

- **Preemption Mode (Complete Preemption (Real-Time))**
- **High Resolution Timer Support**

Power management and ACPI options

- **Power Management support**

Kernel hacking

Preemption

- Complete Preemption CONFIG_PREEMPT_RT
  - Unterbrechbare Critical Sections
    - Spinlock -> RTMutex
    - Alte Spinlocks noch verfügbar als raw_spinlock_t
  - Unterbrechbare Interrupt Handler
    - Interrupt Handler als Threads
  - Latenz Tweaks
    - Blockieren “langsamer” MMX/SSE Befehle
  - Priority Inheritance für Spinlocks und Semaphoren

Preemption

- Preemption = Unterbrechbarkeit

Timer Interrupt

```
static int __devinit igb_probe([...])
{
[...]
pci_set_drvdata(pdev, netdev);
adapter = netdev_priv(netdev);
adapter->netdev = netdev;
adapter->pdev = pdev;
hw = &adapter->hw;
hw->back = adapter;
[...]
}
```


Preemption

- Preemption = Unterbrechbarkeit

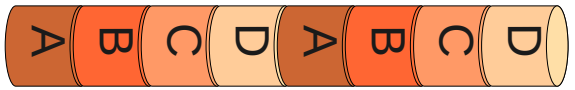
Timer Interrupt

```
static int __devinit igb_probe([...])
{
[...]
pci_set_drvdata(pdev, netdev);
adapter = netdev_priv(netdev);
adapter->netdev = netdev;
adapter->pdev = pdev;




hw = &adapter->hw;
hw->back = adapter;
[...]
}
```

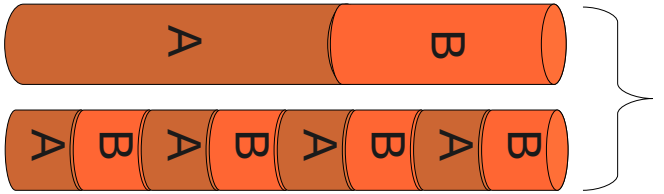
```
smp_apic_timer_interrupt([...])
{
[...]
}
```


Linux Scheduling Richtlinien

- Nicht-Real-Time Scheduling Richtlinie
  - SCHED_NORMAL

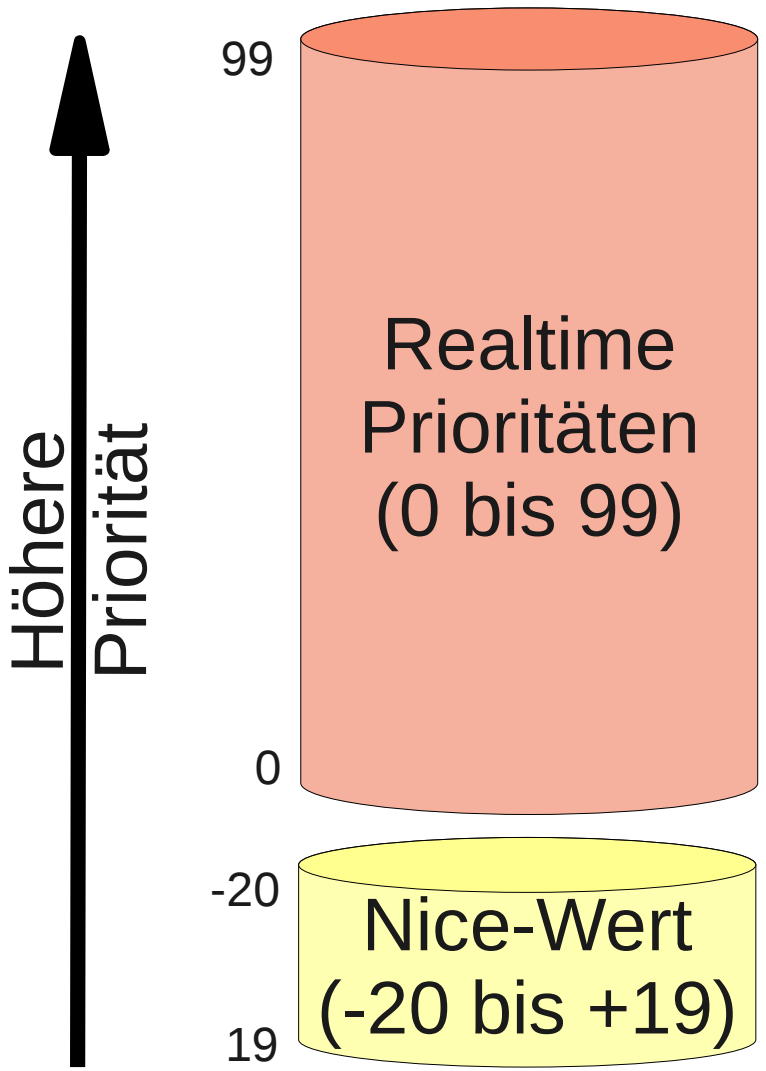


- Real-Time Scheduling Richtlinien
  - SCHED_FIFO
  - SCHED_RR



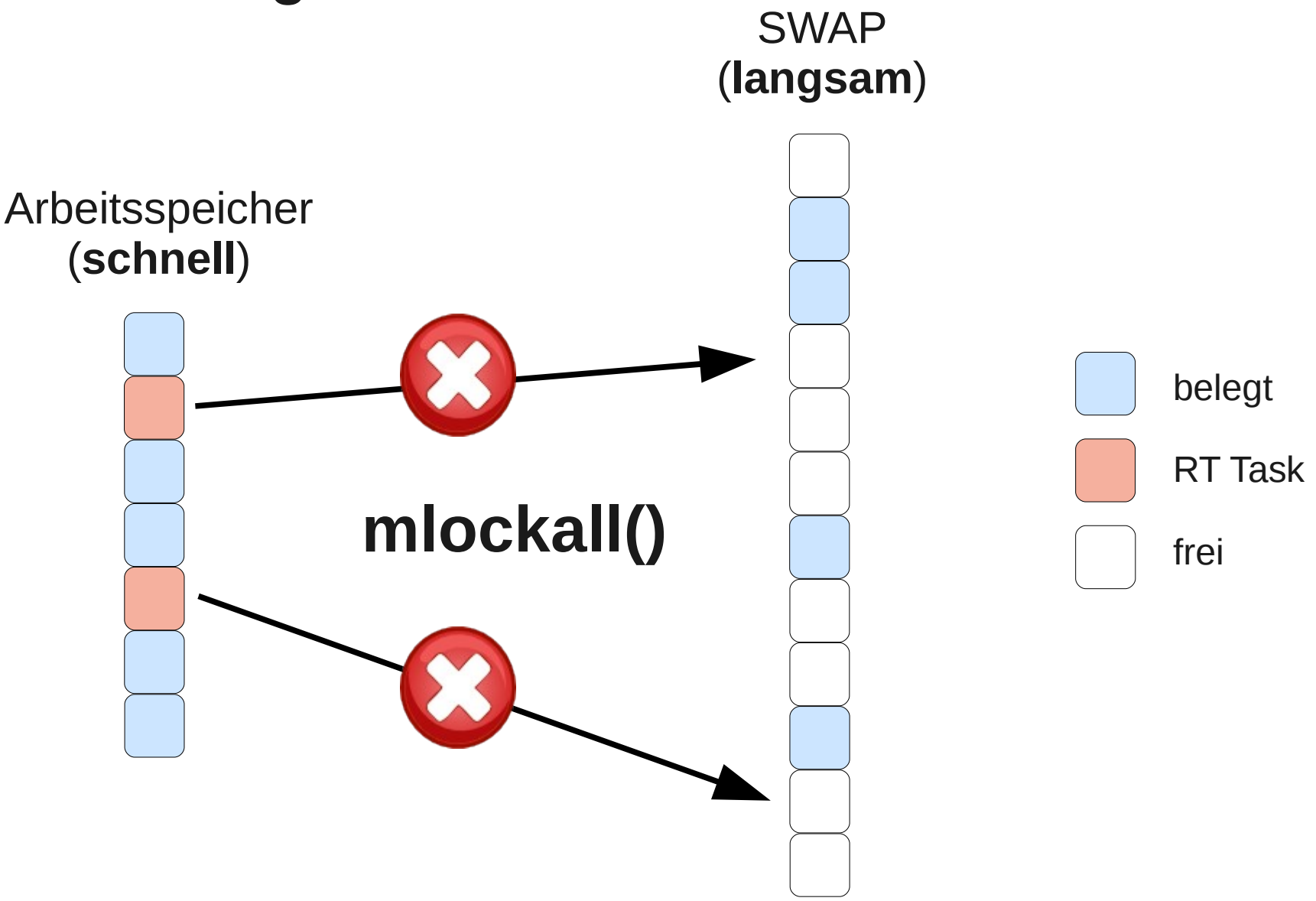
Die Scheduler Richtlinie kann mit den Systemcalls sched_setscheduler() und sched_getscheduler() gesetzt und ausgelesen werden.

Prioritäten in Linux

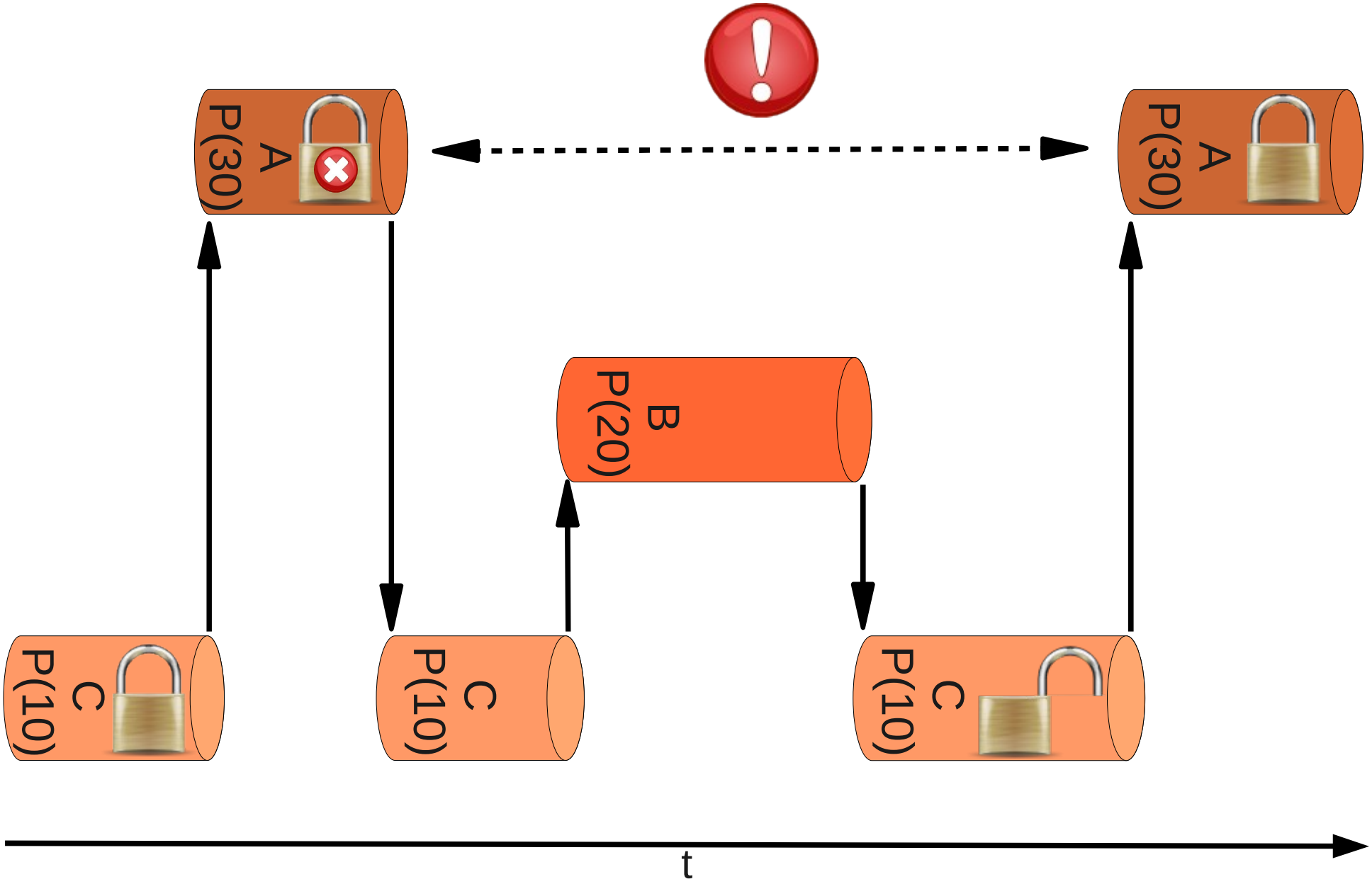



```
ps -eo pid,rtprio,cmd
  PID RTPRIO CMD
    3     99 [migration/0]
    4     49 [sirq-high/0]
    5     49 [sirq-timer/0]
    6     49 [sirq-net-tx/0]
    7     49 [sirq-net-rx/0]
   10     49 [sirq-tasklet/0]
   11     49 [sirq-sched/0]
   12     49 [sirq-hrtimer/0]
   14     99 [posixcputmr/0]
   16     99 [migration/1]
   17     99 [posixcputmr/1]
   [...]
  268     50 [irq/9-acpi]
  713     50 [irq/14-ata_piix]
  714     50 [irq/15-ata_piix]
  770     50 [irq/19-ath9k]
  804     50 [irq/12-i8042]
  805     50 [irq/1-i8042]
  821     50 [irq/8-rtc0]
  903     50 [irq/27-hda_inte]
 2629     50 [irq/16-i915@pci]
```

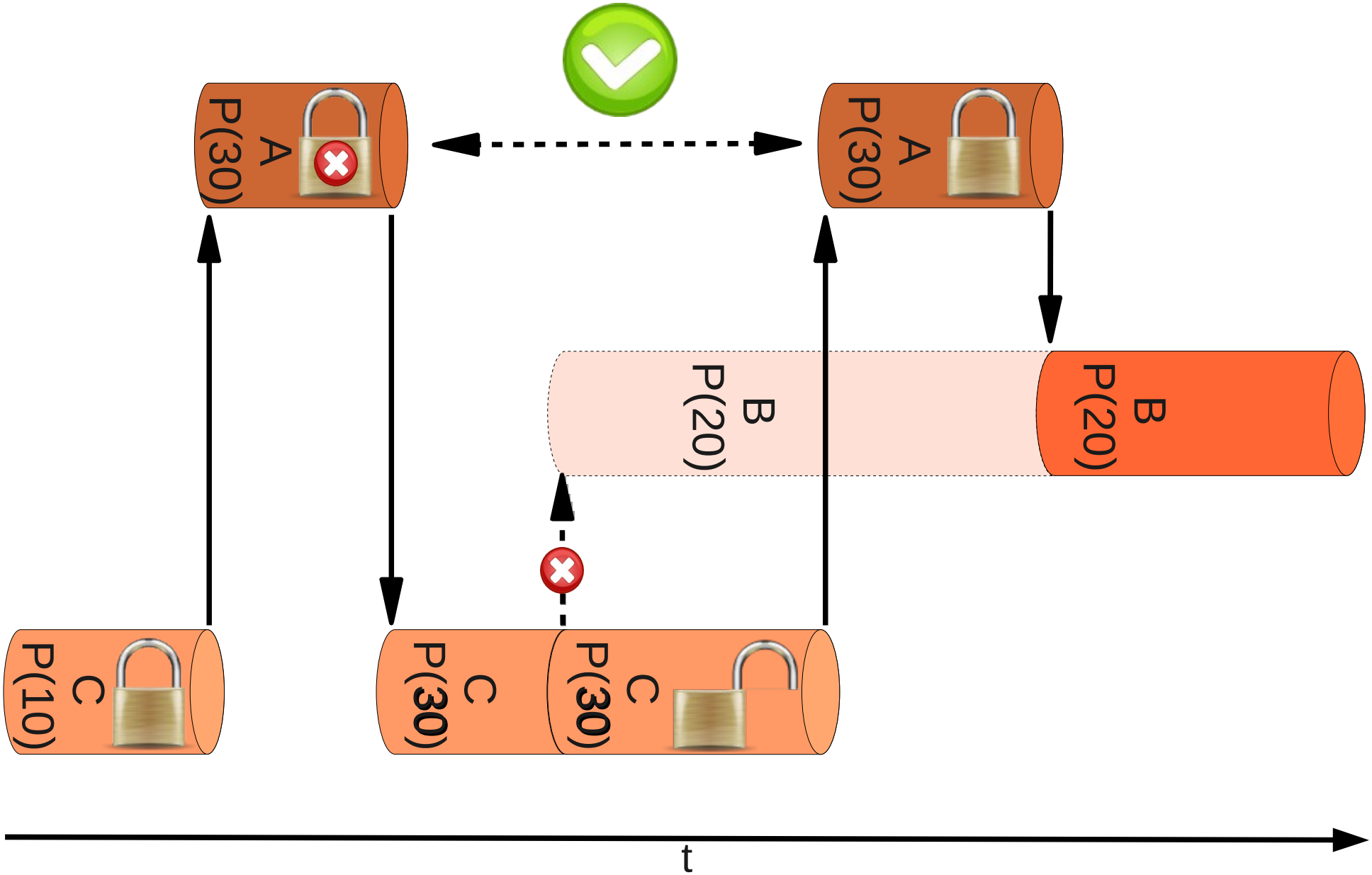

Speichermanagement



Priority Inversion



Priority Inheritance



Real-Time Hello World Programm

```
#include <[...]>
#define MY_PRIORITY (30)

int main(int argc, char* argv[])
{
	struct sched_param param;

	param.sched_priority = MY_PRIORITY;
	sched_setscheduler(0, SCHED_FIFO, &param)

	mlockall(MCL_CURRENT | MCL_FUTURE)

	while(1) {
		clock_nanosleep([...])

		/* insert RT stuff here */
		printf("Hello World with RT priority %d\n", MY_PRIORITY);
	}
}
```


Real-Time OS Tests

git://git.kernel.org/pub/scm/linux/kernel/git/tglx/rt-tests.git

- cyclictest --smp -p96
  - ein Thread pro CPU, Priorität 96

http://people.redhat.com/mingo/cfs-scheduler/tools/hackbench.c

- hackbench 100
  - 100*40 Threads

Weitere Tools

- http://freshmeat.net/projects/util-linux/
  - Chrt
- http://rt.wiki.kernel.org/index.php/Cpuset_management_utility
  - cpuset
- http://rt.wiki.kernel.org/index.php/LTP(Realtime_Test_Tree)
  - LTP Realtime Tests
- http://rt.wiki.kernel.org/index.php/Ftrace
  - ftrace

Herausfordrungen

- Nicht jeder Rechner ist für Real-Time geeignet
  - SMIs (hwlatdetect)
  - Interrupt Handling
  - Power Management
  - Skalierbarkeit
- Tuning erforderlich

Zusammenfassung

- Real-Time != Real-Fast
- Mächtiges Werkzeug, erst nachdenken
- RT Programme unterscheiden sich nicht wesentlich von nicht-RT Programmen
- Versteckte Fallen (Herausforderungen)
- Ausprobieren, ausprobieren, ausprobieren :-)

Don't Panic!

Weitere Informationen

- http://rt.wiki.kernel.org/index.php/Main_Page
- http://www.digitalhermit.com/linux/Kernel-Build-HOWTO.html
- http://ols.fedoraproject.org/OLS/Reprints-2007/rostedt-Reprint.pdf
- http://ols.fedoraproject.org/OLS/Reprints-2008/mckenney-reprint.pdf
- http://www.linuxjournal.com/article/5833
- http://lwn.net/Articles/146861/

- Real-Time Mailingliste: linux-rt-users@vger.kernel.org

Fragen ?

Feedback:
sassmann@redhat.com