Ablauf

Inhalt

Theorie

- Motivation
- Eine Frage der Lizenz
- Programmentwicklung
- Linux aus Anwendersicht
- Betriebssystem-Theorie

Infos:

- Quade/Kunst: Linux-Treiber entwickeln. 2. Auflage, dpunkt-Verlag, 2006, S. 7-55.

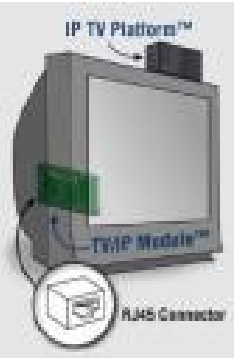
Motivation

Linux Feature-Übersicht

- Merkmale
  - hoch portierbar
  - hohe Skalierbarkeit
  - geeignet für kleine, eingebettete Systeme
  - geeignet für Mainframes
  - Open Source
  - keine Lizenzgebühren
  - Fully Featured

Motivation

Linux Feature-Übersicht



- Merkmale (Fortsetzung)
  - große Zahl von Anwendungen
  - zunehmende Verbreitung (Marktanteil)
    - Server
    - Desktop
    - Embedded Systems

  - gute Dokumentation
  - guter Support – aber anders geartet (community versus company)

Linux Feature-Übersicht

- Nachteile:
  - Der native Linux-Kernel ist kein Hard-Realtime-Kernel.
  - Unternehmen tun sich zeitweilig schwer die Lizenzanforderungen zu erfüllen.
- Tipp: Use it!

Lizenzfrage

- Kernel steht unter der GNU Public License (GPL):
  - Modifikationen am Code, die nicht für den Eigenbedarf bestimmt sind, müssen veröffentlicht werden.
  - Lizenzbestimmungen müssen dem Gerät/Produkt beigelegt werden.

Lizenzfrage

- Torvalds: Gerätetreiber sind bereits „Modifikation des Kernel-Codes“.
- Binärtreiber werden (ungern und noch) toleriert.
- Treibercode muss Lizenzbedingung spezifizieren.
- Kernel-Interfaces differenzieren zwischen GPL und Non-GPL-Treibern:
  - Non-GPL-Treiber können nur auf ein Subset der Funktionen zurückgreifen.

Programmentwicklung unter Linux

Spezifische Werkzeuge/Programme sind nicht notwendig!

- Editor (vi, emacs, kate)
- Make
- (Cross-) Compiler
- (Cross-) Linker
- Versionsverwaltung
- IDE steht nur eingeschränkt zur Verfügung
- Sehr eingeschränktes Debugging

Editor

- vim = vi
  1. Leistungsfähiger Editor.
  2. Auf allen (Unix-) Plattformen vorhanden.
  3. Nur für den Profi: gewöhnungsbedürftige Bedienung.
  4. Keine grafische Oberfläche notwendig.
- Emacs
  1. Leistungsfähiger Editor.
  2. Keine grafische Oberfläche notwendig.
- Kate
  1. Einfach zu bedienender Editor.
  2. Nur unter X (daher für „Embedded Systems“ nur eingeschränkt verwendbar).

Debugging

"I'm afraid that I've seen too many people fix bugs by looking at debugger output, and that almost inevitably leads to fixing the symptoms rather than the underlying problem."

Linux Kernel Coding Style

Linus Torvalds:
„*This is a short document describing the preferred coding style for the linux kernel. Coding style is very personal, and I won't _force_ my views on anybody, but this is what goes for anything that I have to be able to maintain, and I'd prefer it for most other things too. Please at least consider the points made here.*

First off, I'd suggest printing out a copy of the GNU coding standards,“

`/usr/src/linux/Documentation/CodingStyle`

Linux Kernel Coding Style

„*First off, I'd suggest printing out a copy of the GNU coding standards*,“

„*and* **NOT** *read it. Burn them, it's a great symbolic gesture*.“

Linux Kernel Coding Style

- Im Kernel wird der Kernighan & Ritchie Stil verwendet.
- Variablen werden klein geschrieben.
- Im Variablennamen verbirgt sich keine Typinformation.
- Öffnende Block-Klammern werden am Ende des vorausgehenden Statements gesetzt.
- Schließende Block-Klammern werden bündig zur aktuellen Einrückungstiefe gesetzt.
- Es wird grundsätzlich ein Tabulator von 8 Zeichen verwendet.
- Gotos sind erlaubt, wenn sie zu kürzerem, effizienten Code führen.

Kernelprogrammierung

- Innerhalb des Kernels stehen nur eingeschränkt Bibliotheksfunktionen zur Verfügung.
- Innerhalb des Kernels darf kein Floating-Point verwendet werden.
- Kernelcode steht nur ein eingeschränkter Stack zur Verfügung (4-8kByte).
- Kernelcode ist performance-optimiert programmiert (Stichwort `goto`).
- Innerhalb des Kernels gibt es keinen Speicherschutz.

Linux im Überblick

Aus Anwendersicht

Betriebssystem

Unter einem Betriebssystem versteht man alle Programme (Plural), die

- ➔ die Ausführung von Benutzerprogrammen und
- ➔ die Verteilung von Betriebsmitteln (z.B. Interrupts, Speicher, Prozessorzeit)

steuern und überwachen.

Systemmerkmale aus Anwendersicht

- Multitasking:
  1. Kommando „ps“ zeigt alle im System gestarteten Prozesse
- Multiuser:
  1. Mehrere User nutzen parallel das System. Jeder User ist durch seine „User Identifikation“ (UID) gekennzeichnet.
  2. Der „Superuser“ (root) ist mit besonderen Rechten ausgestattet.
  3. Um Superuser-Rechte zu bekommen, reicht das Kommando „su“ aus.

Systemmerkmale aus Anwendersicht

- Alle wichtigen Ereignisse werden im System durch den „syslogd“ protokolliert.

```
1. tail -f /var/log/messages
```

- Systeminformationen und Systemzustände sind im Proc-Filesystem abrufbar:

```
1. cat /proc/cpuinfo
2. cat /proc/interrupts
```

Anwendersicht

Systemmerkmale aus Anwendersicht

- Der Zugriff auf Peripherie ist auf den Dateizugriff abgebildet.
- 3 „Dateien“ sind für jeden Rechenprozess direkt zugreifbar:
  1. STDIN
  2. STDOUT
  3. STDERR
- Ausgaben von Rechenprozessen können mit Eingaben anderer Rechenprozesse verknüpft werden (Pipen).

Anwendersicht

Shell

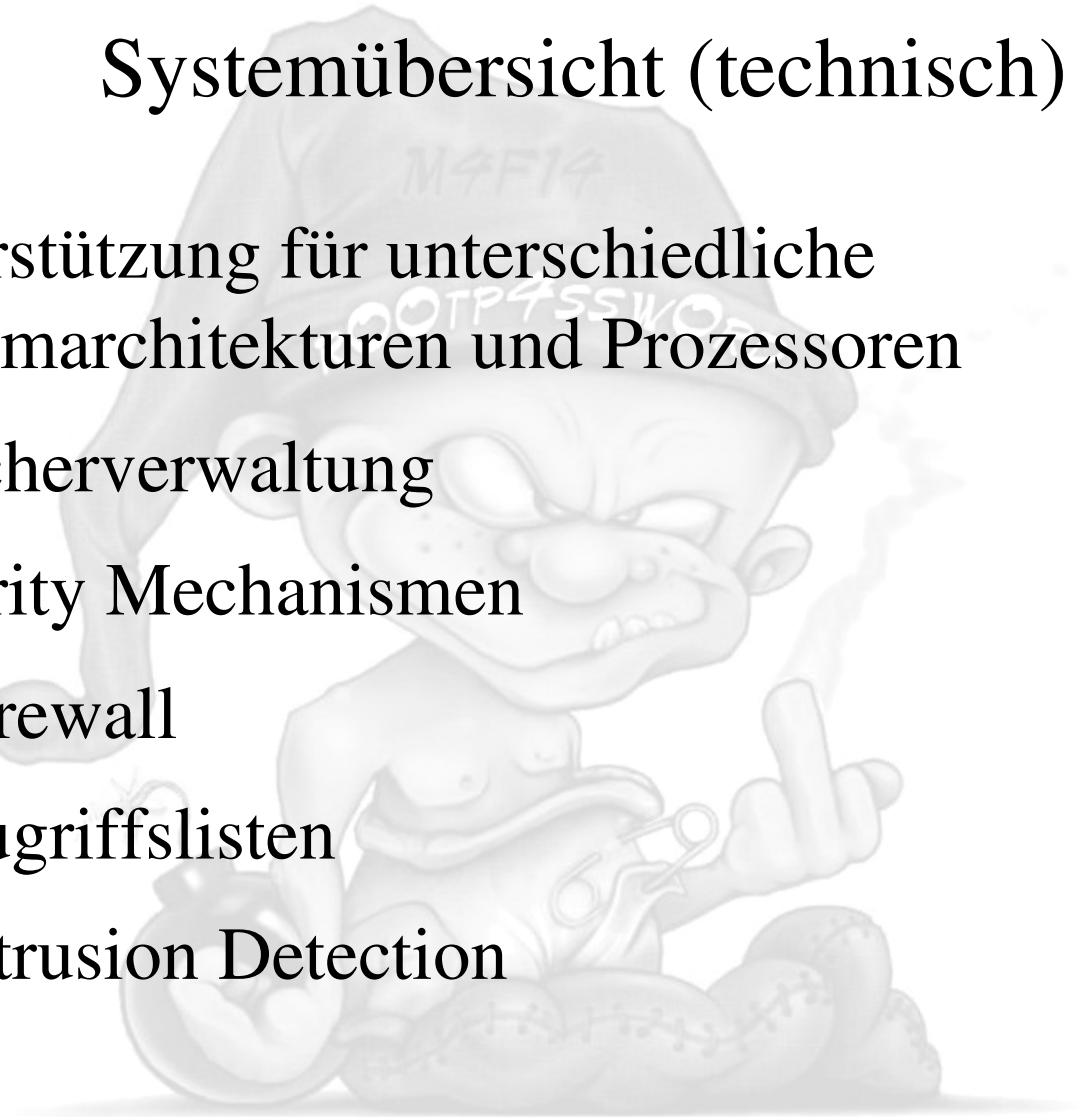
- Kommandointerpreter:
  1. ls = list files
  2. mkdir = make directory
  3. rm = delete file
  4. cd = change directory
  5. cat <filename> = Inhalt der Daten <filename> ausgeben
  6. > Ausgabe umlenken
  7. < Eingabe umlenken

Betriebssystem-Theorie

Systemübersicht (technisch)

- Monolithischer Betriebssystemkern.
- Prioritätengesteuertes Scheduling (mit Round-Robin oder FCFS).
- Untertützung für Tasks und Threads.
- Verschiedenste Dateisysteme:
  - ext2/ext3/ext4
  - vfat
  - jffs2 (journaled flash filesystem)
  - ...

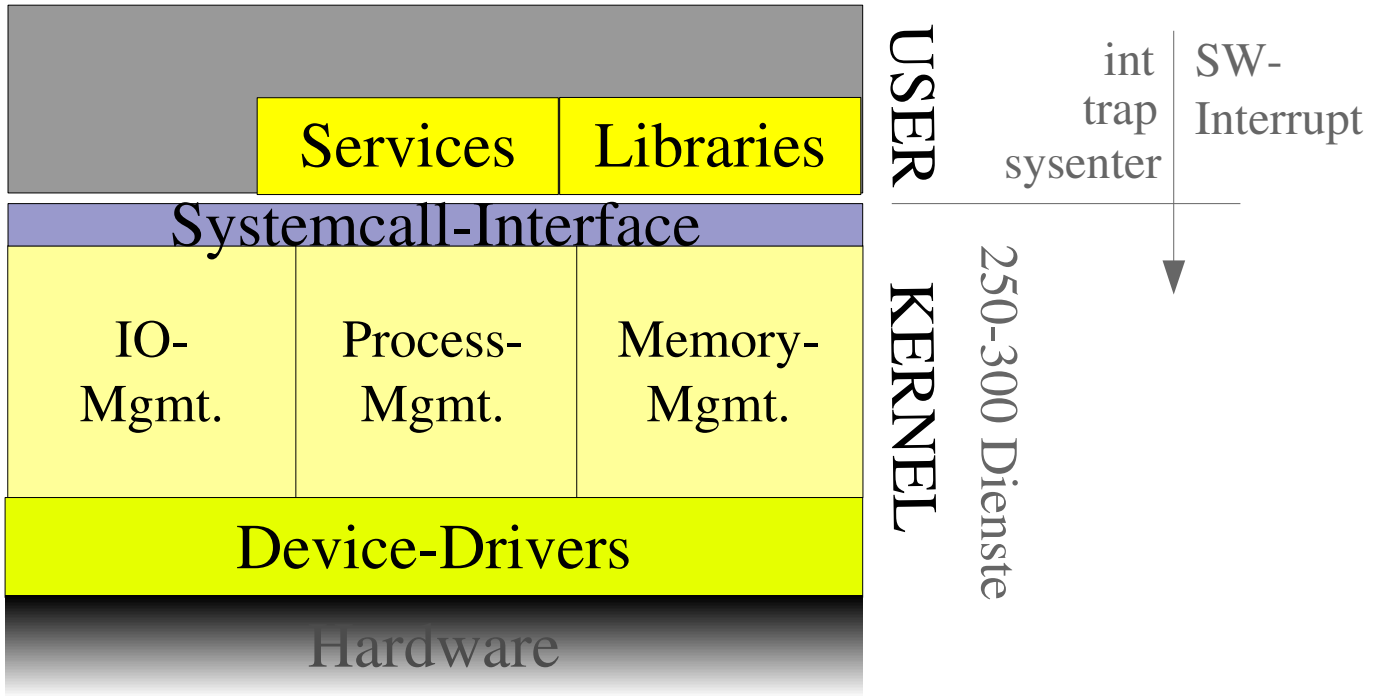
Systemübersicht (technisch)



- Unterstützung für unterschiedliche Systemarchitekturen und Prozessoren
- Speicherverwaltung
- Security Mechanismen
  - Firewall
  - Zugriffslisten
  - Intrusion Detection
  - ...
- Bekannte Programmierschnittstelle

Technik

Überblick über den Linux-Kernel



Systemcallinterface

- Dienstzugangsschnittstelle
- Unabhängig von Programmiersprachen
- Realisiert über Softwareinterrupt 0x80 (synchron zum Programmablauf, Assemblerbefehl `INT` oder `sysenter`)
- Argumentenübergabe über Register oder über den Stack
- Alle Systemcalls unter Linux sind im Headerfile ***<asm/unistd.h>*** aufgeführt

Systemcalls

```
#define __NR_exit          1
#define __NR_fork          2
#define __NR_read          3
#define __NR_write         4
#define __NR_open          5
#define __NR_close         6
#define __NR_waitpid       7
#define __NR_creat         8
#define __NR_link          9
#define __NR_unlink       10
#define __NR_execve       11
#define __NR_chdir        12
#define __NR_time         13
#define __NR_mknod        14
#define __NR_chmod        15
#define __NR_lchown       16
#define __NR_break        17
#define __NR_oldstat      18
#define __NR_lseek        19
#define __NR_getpid       20
#define __NR_mount        21
#define __NR_umount       22
#define __NR_setuid       23
#define __NR_getuid       24
#define __NR_stime        25
#define __NR_ptrace       26
#define __NR_alarm        27
#define __NR_oldfstat     28
#define __NR_pause        29
#define __NR_utime        30
#define __NR_stty         31
...
```

Systemcallbeispiel

```
.text
.globl write_hello_world
write_hello_world:
   movl $4,%eax          ; //code fuer write systemcall
   movl $1,%ebx          ; //file descriptor fd (1=stdout)
   movl $message,%ecx    ; //Adresse des Textes (buffer)
   movl $12,%edx         ; //Laenge des auszugebenden Textes
   int $0x80             ; //SW-Interrupt, Auftrag an das BS
   ret
.data
message:
   .ascii "Hello World\n"
```

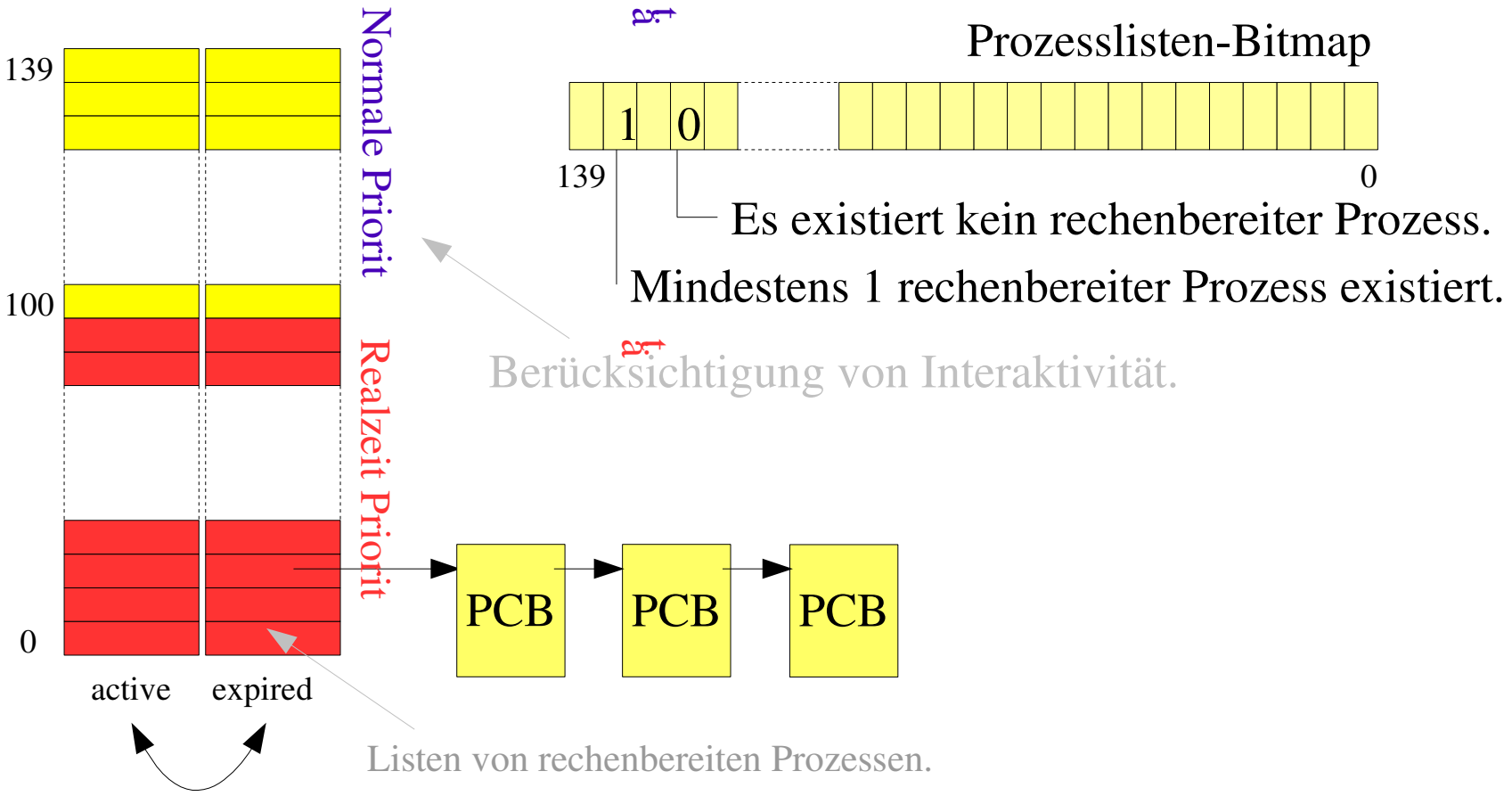
Prozess-Management

- Aufgabe
  - Verteilung der Ressource CPU (Scheduling)
- Schedulingverfahren:
  - Prioritätengesteuertes Scheduling mit überlagertem Round-Robin oder FCFS.

Technik

Scheduling

Prioritätengesteuertes Scheduling
Innerhalb einer Prioritätsebene: Zeitscheiben

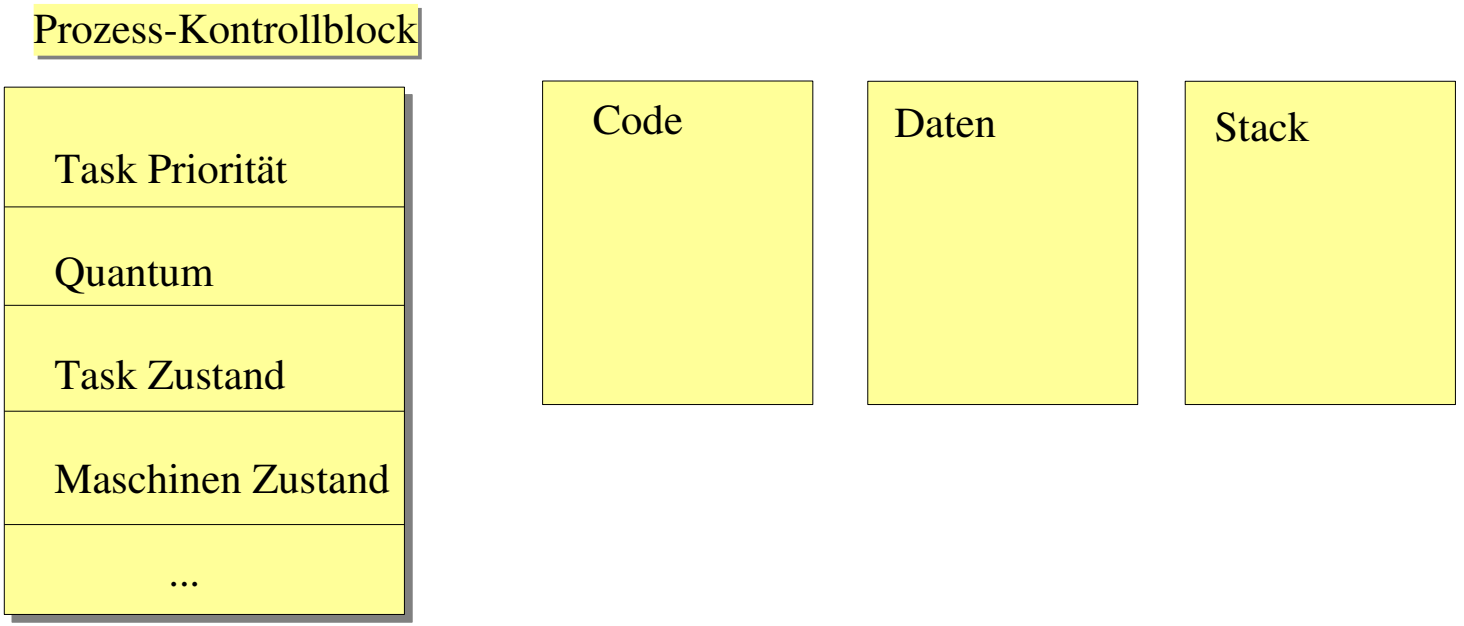


Multikern-Scheduling

- Jeder Kern hat seine eigene Prozess-Listen (eigener Scheduler).
- Migrationskosten werden abhängig von der Architektur (HT, SMP, NUMA) berechnet.
- Architektur wird beim Booten evaluiert (Stichwort Scheduling Domains).
- Zeitpunkte für die Prozessmigration:
  - Exit, fork, clone.
  - Periodisch (kthread).

Technik

Prozessmanagement
Tasks, Threads, Scheduler und Context Switch



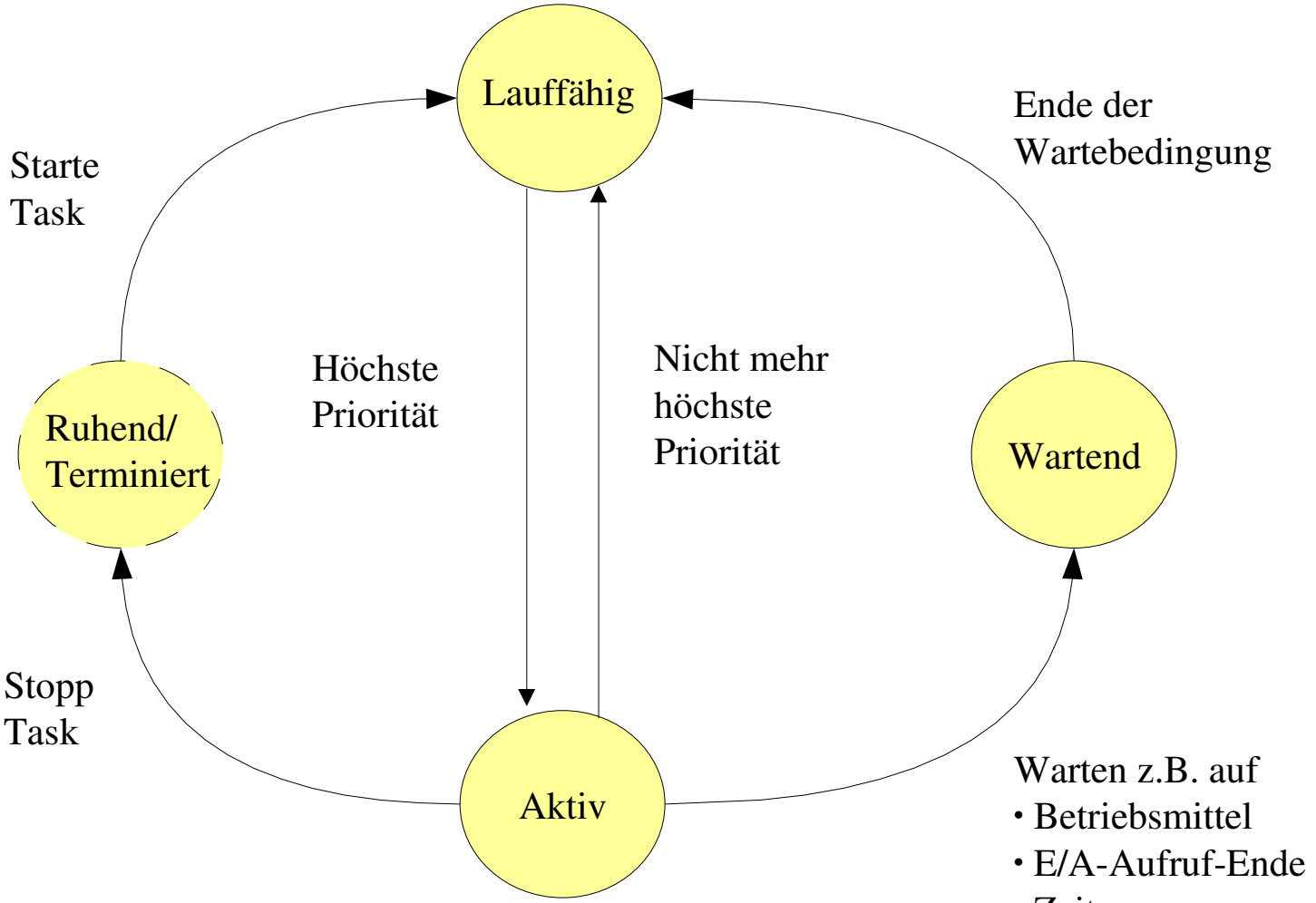
Task = TCB + Code + Daten + Stack
Thread = TCB + Stack (Code + Daten geteilt)

PCB im Linux Kernel

```
struct task_struct {
    volatile long state; /* -1 unrunnable, 0 runnable, >0 stopped */
    ...
    struct exec_domain exec_domain; /* code segment */
    ...
    pid_t pid;
    pid_t pgrp;
    pid_t tty_old_pgrp;
    pid_t session;
    pid_t tgid;
    ...
    int swappable:1;
    uid_t uid, euid, suid, fsuid;
    gid_t gid, egid, sgid, fsgid;
    ...
    struct thread_struct thread; /* machine state */
    ...
};
```

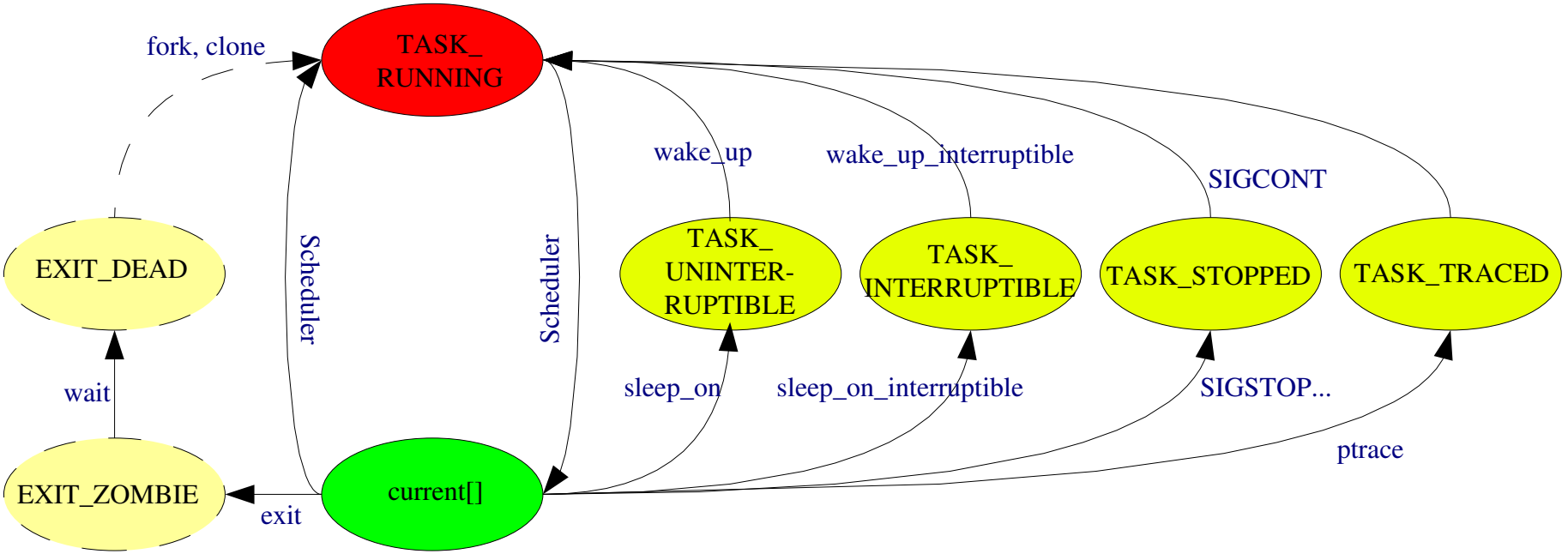
Technik

Taskzustände (Theorie)



Technik

Taskzustände



Memory-Management

- Aufgabe
  - Speicherschutz
  - Adressumsetzung
  - Virtuellen Speicher zur Verfügung stellen
  - Unterstützung von extended Memory (Highmem)
- User-Space: Speicherbereiche der Applikation
- Kernel-Space: Speicherbereiche des Kernels

Memory Management (I)

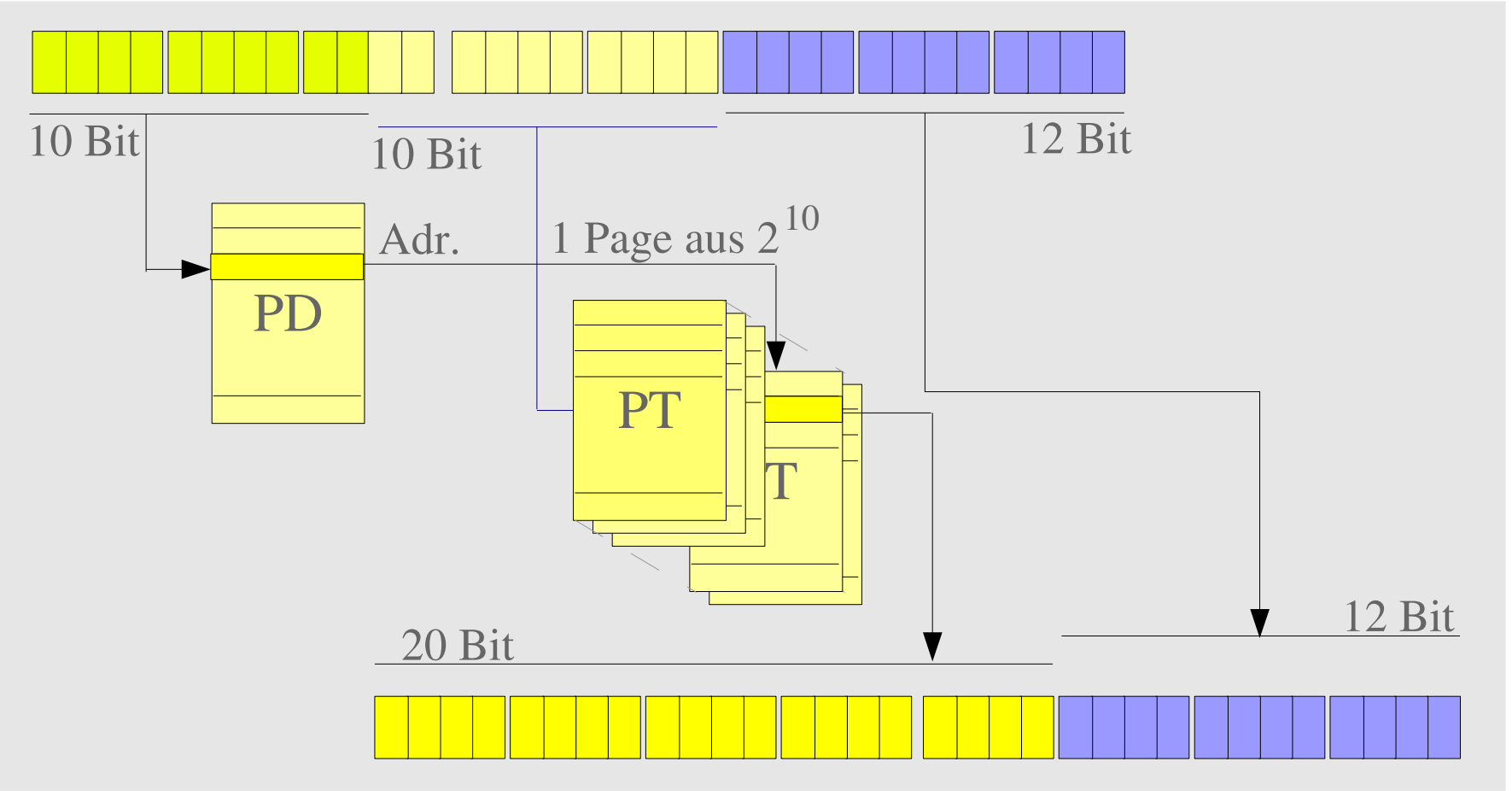
- Jede Task hat ihren eigenen Speicherbereich.
- Applikationen können nicht auf den Speicherbereich des Kernels zugreifen.
- Auch der Kernel kann nicht *einfach* auf den Speicherbereich einer Applikation zugreifen.

Überblick über den Linux-Kernel

- Speicher wird in Pages eingeteilt:
  - 32 Bit-Systeme:
    - 2-stufige Speicherverwaltung (two-level paging)
    - Page Directory, Page Table, (Page)
    - 4096 Byte/Page
  - 64 Bit-Systeme:
    - 3-stufige Speicherverwaltung (three-level paging)
    - Page Directory, Page Middle Directory, Page Table, (Page)
    - 8192 Byte/Page
    - Adressraum: 43 Bit, 8 Terabyte

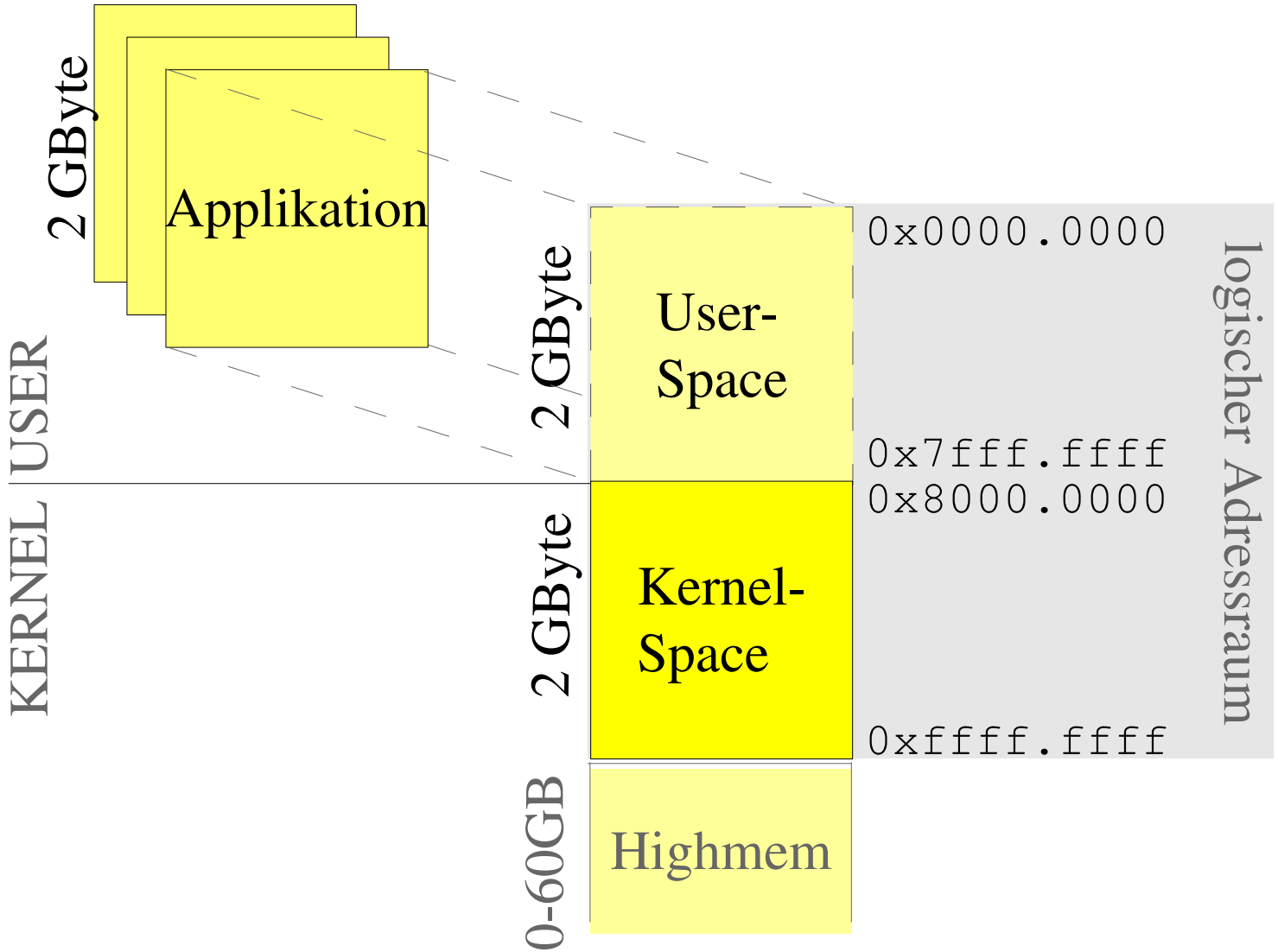
Technik

Überblick über den Linux-Kernel



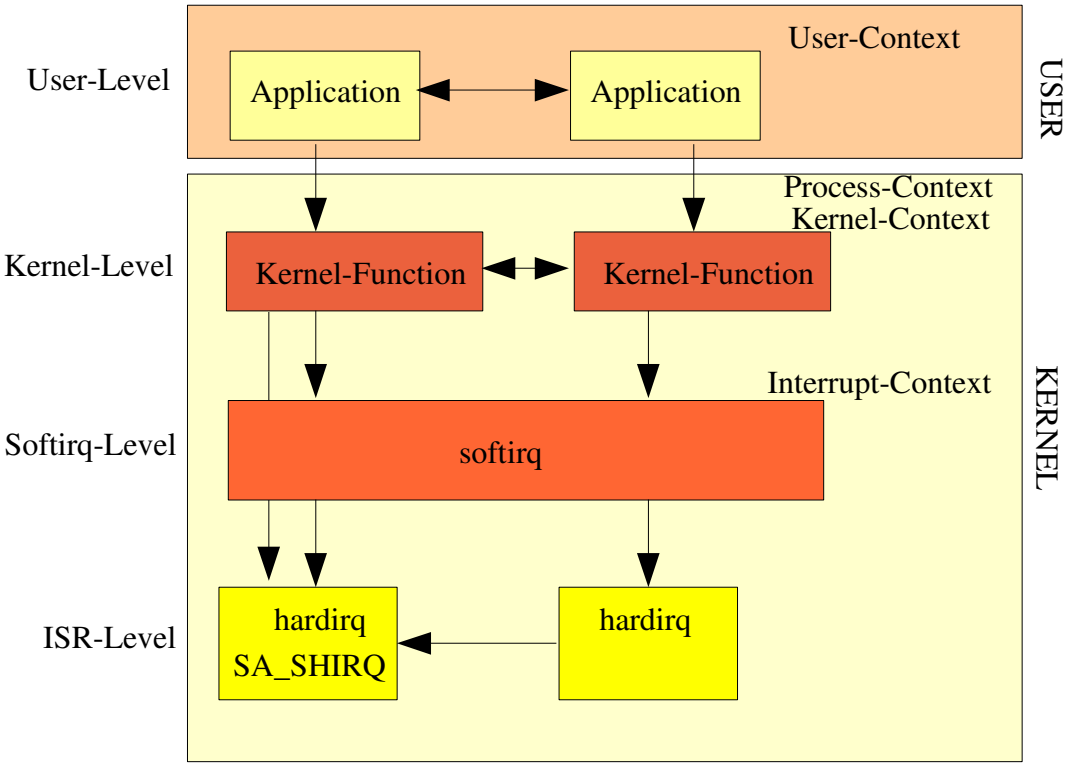
Technik

Überblick über den Linux-Kernel



Technik

Unterbrechungsmodell



Kernel-Preemption

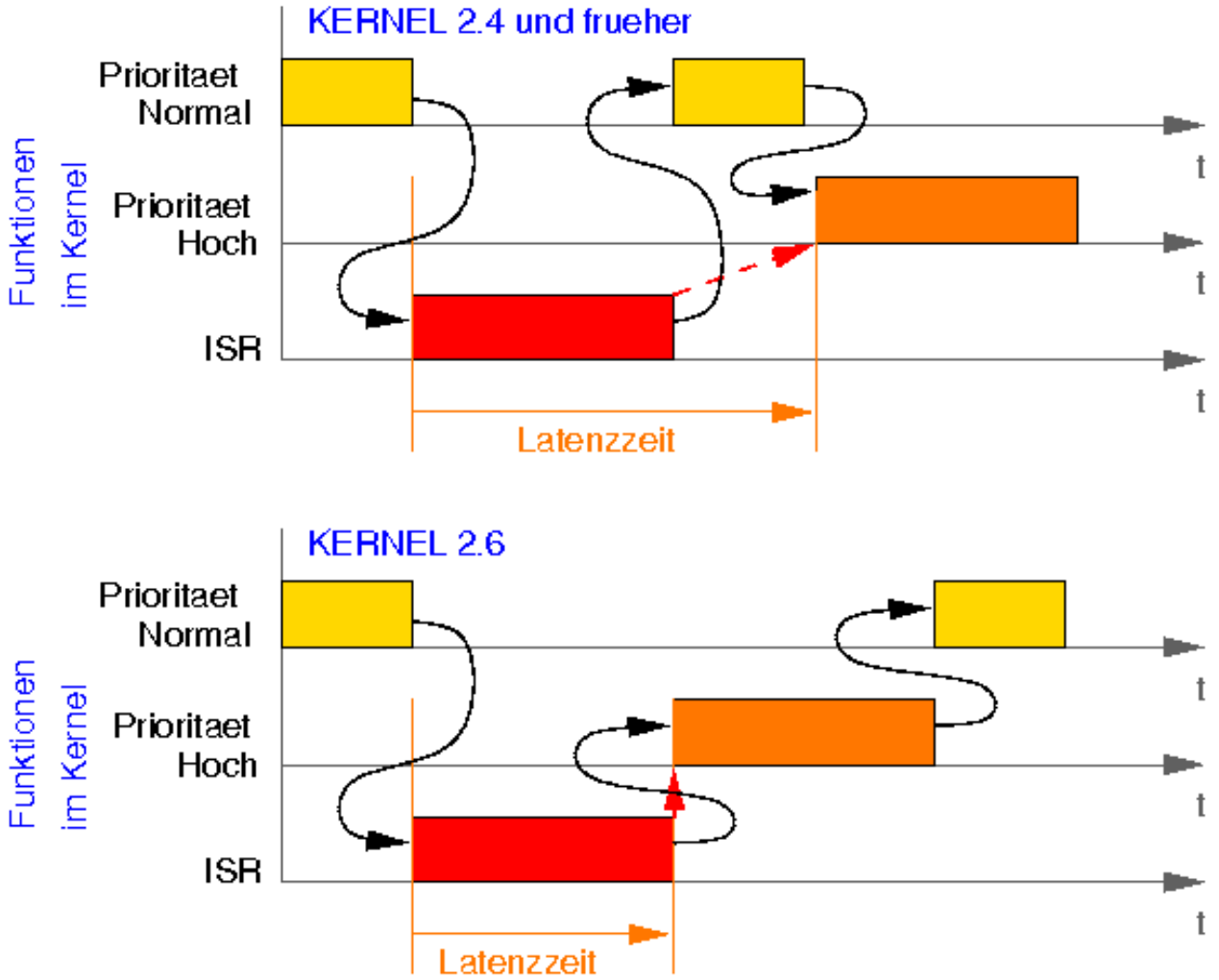
- Code, der im Kernel im Kontext eines Prozesses ausgeführt wird (Prozess-Kontext), wird unterbrochen, wenn
  - ein höherpriorer Rechenprozess lauffähig wird.
- Code, der damit unterbrechbar geworden ist:
  - Applikationsgetriggerte Treiberfunktionen (driver_open, driver_close, driver_read, driver_write)
  - Systemcalls (z.B. insmod, gettimeofday, ...)
  - Kernel-Threads (Kernel-Kontext)

Kernel-Preemption

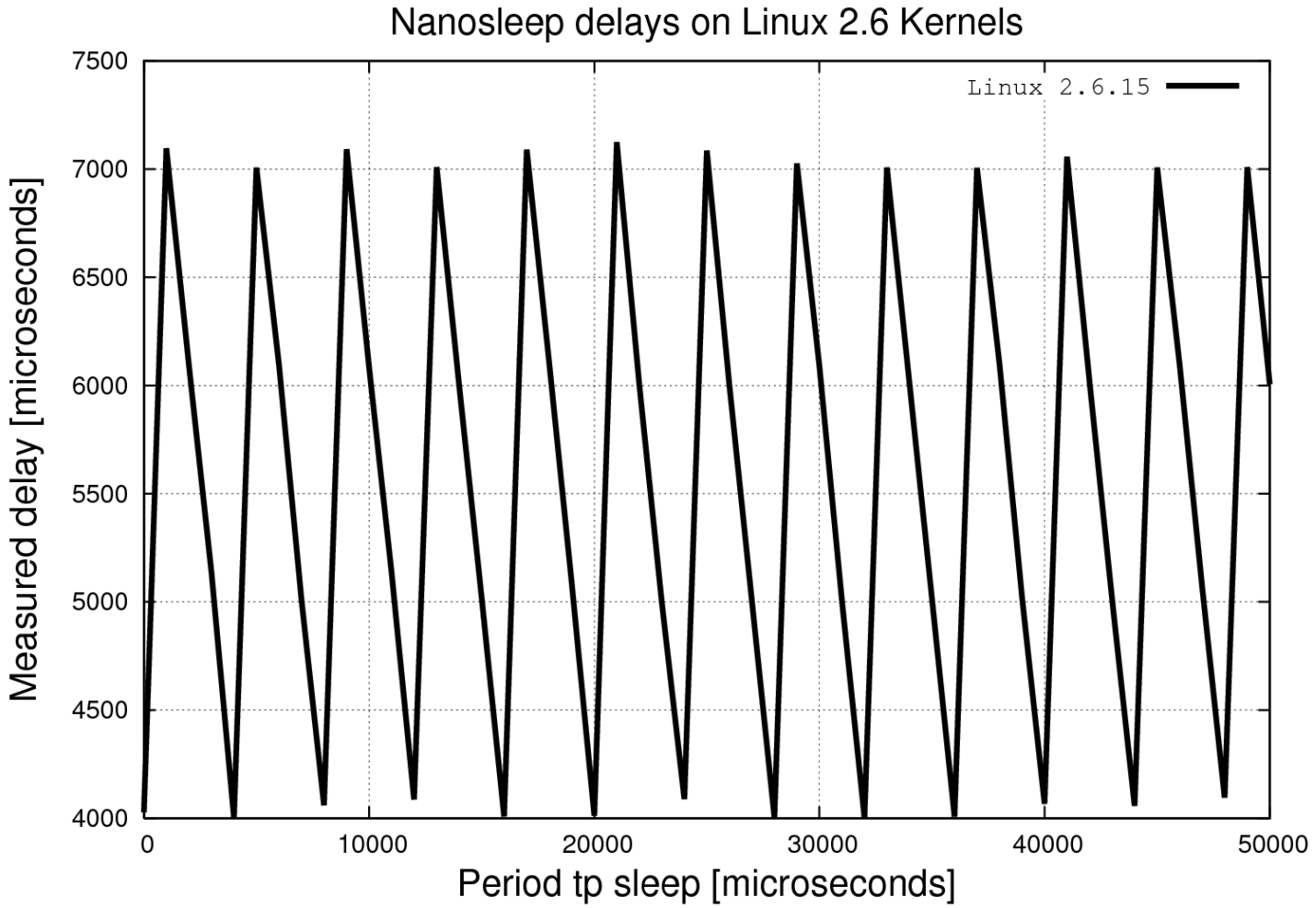
- Funktionen im Kernel- oder Prozesskontext waren schon immer durch Funktionen im Interruptkontext unterbrechbar:
  - Soft-IRQs (bottom-halves, Taskqueues, etc.)
  - Hardware-ISR

Technik

Kernel-Preemption



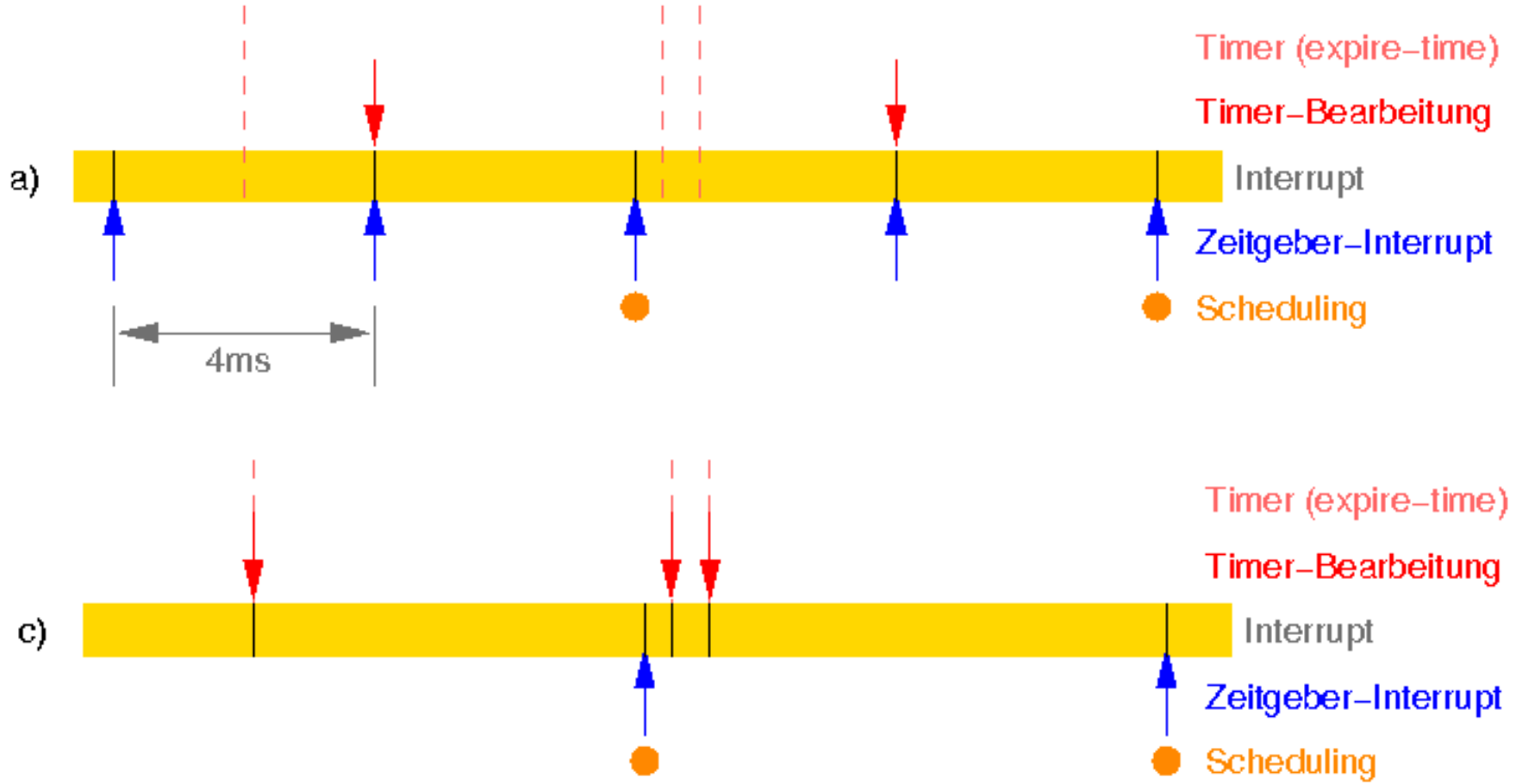
Time-Keeping (I)



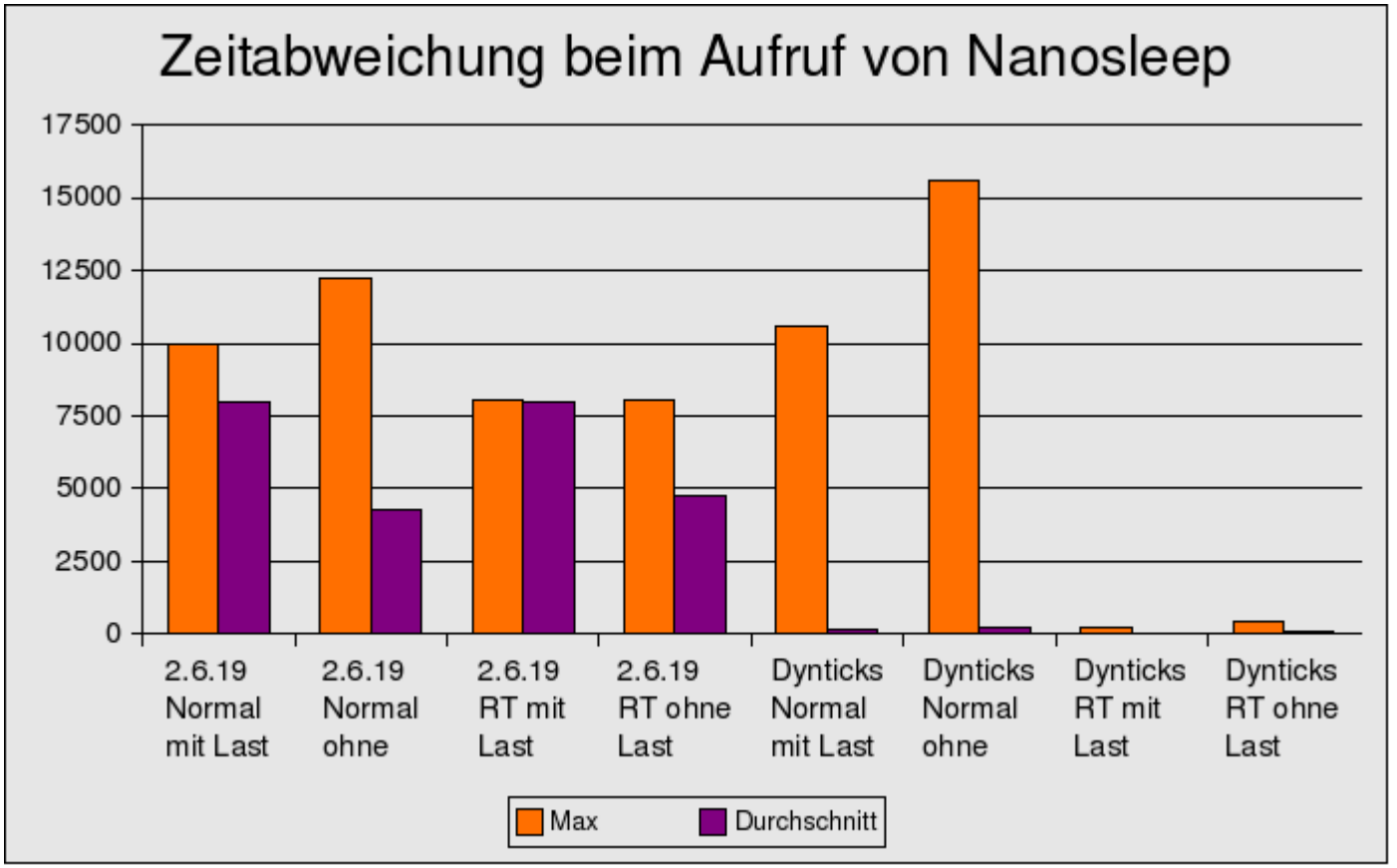
Technik

Time-Keeping

- Tickless-System



Testreihe



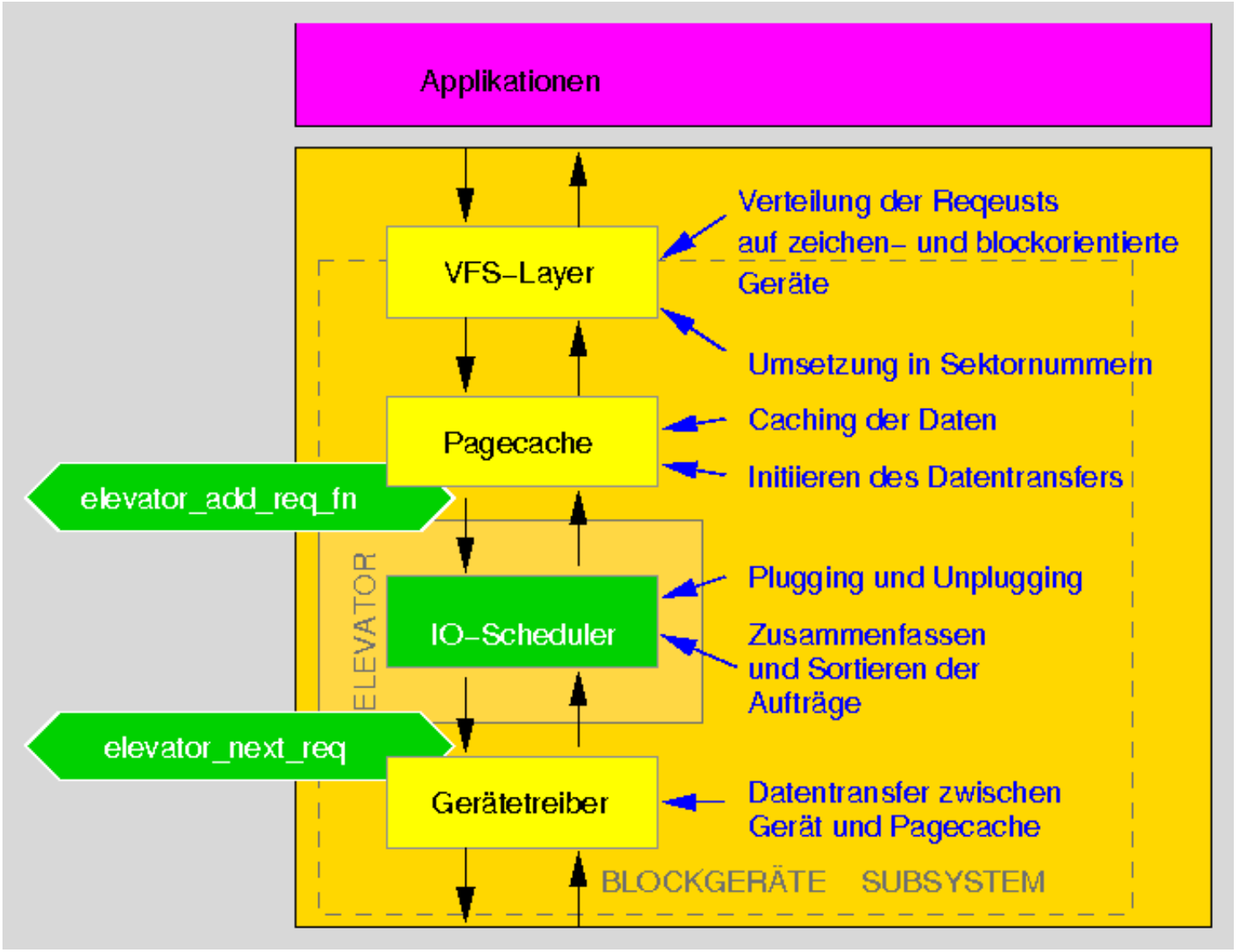
Testrechner: Pentium M, 1,4 Ghz, 512 MB Ram (zufällige Messung)

IO-Subsystem

- Das IO-Subsystem ermöglicht den einheitlichen Zugriff auf Peripherie.
- Implementiert Filesysteme.
- Ermöglicht die systemkonforme Einbindung von Hardware.

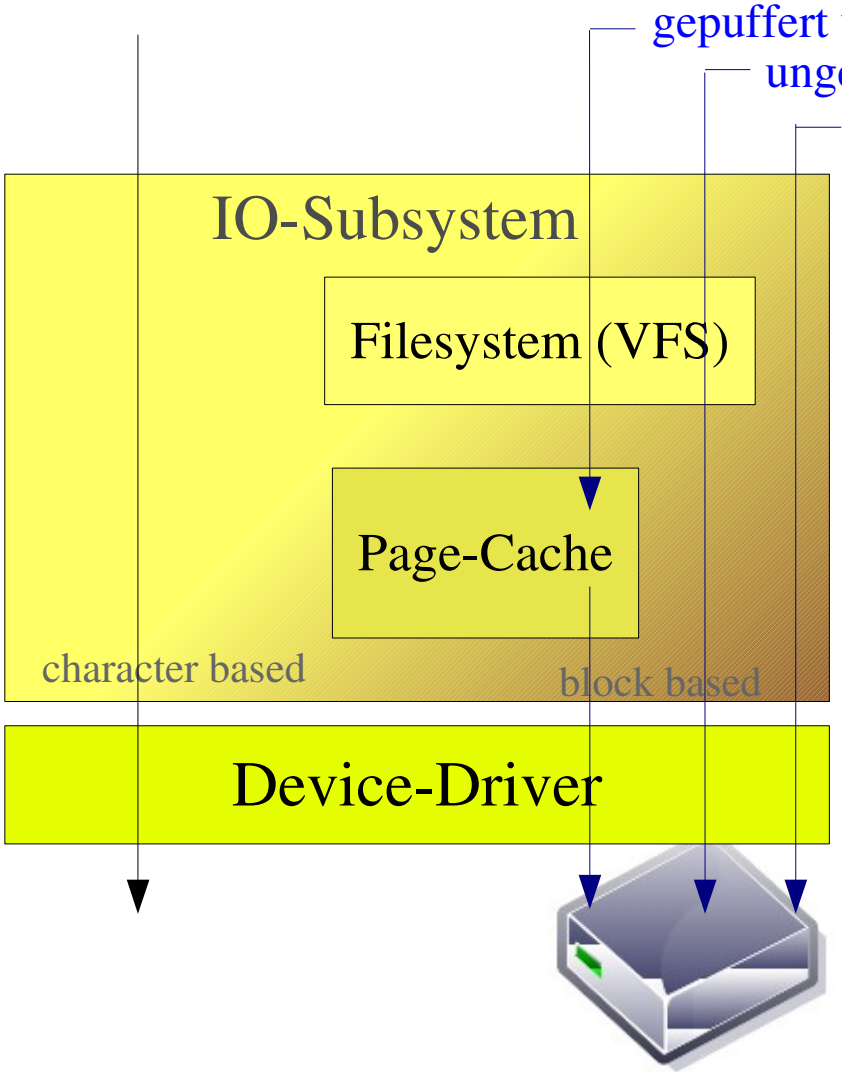
Technik

Architektur des Blockgeräte-Subsystem



Technik

Nutzungsmöglichkeiten des IO-Subsystems



Zusätzlich:
- blockierender Zugriff
- nicht blockierender Zugriff
- asynchroner Zugriff

Gerätetreiber

- Unix: Für den Applikations-Programmierer ist es kein Unterschied, ob er auf Dateien oder auf Geräte zugreift.
- Gerätetreiber führen den eigentlichen (Hardware-)Zugriff auf die Geräte durch.
- Es gibt Treiber für reale Geräte und für virtuelle Geräte (z.B. `/dev/null`).
- Treiber können als
  - Module oder als
  - Build-In-Treiber implementiert werden.

Gerätearten

- Betriebssystemintern werden mehrere Arten von Geräten unterschieden:
  - Character-Devices
  - Block-Devices
  - Network-Devices
  - USB-Devices
  - ...

Character-Devices

- Geräte, die die Ein- und/oder Ausgaben zeichenweise durchführen (z.B. Keyboard).
- Gelesene Zeichen können nicht ein zweites Mal gelesen werden.
- Zuordnung zwischen Gerätedatei und Treiber über Major- und Minornumber.

Gerätetreiber

Übersicht

Block-Devices

- Geräte, die die Daten in Blöcken organisieren und übertragen (z.B. Festplatte).
- Der „wahlfreie“ Zugriff auf die Daten ist möglich (seeken).
- Daten werden blockweise gelesen und geschrieben.
- Zuordnung zwischen Gerätedatei und Treiber über Major- und Minornumber.

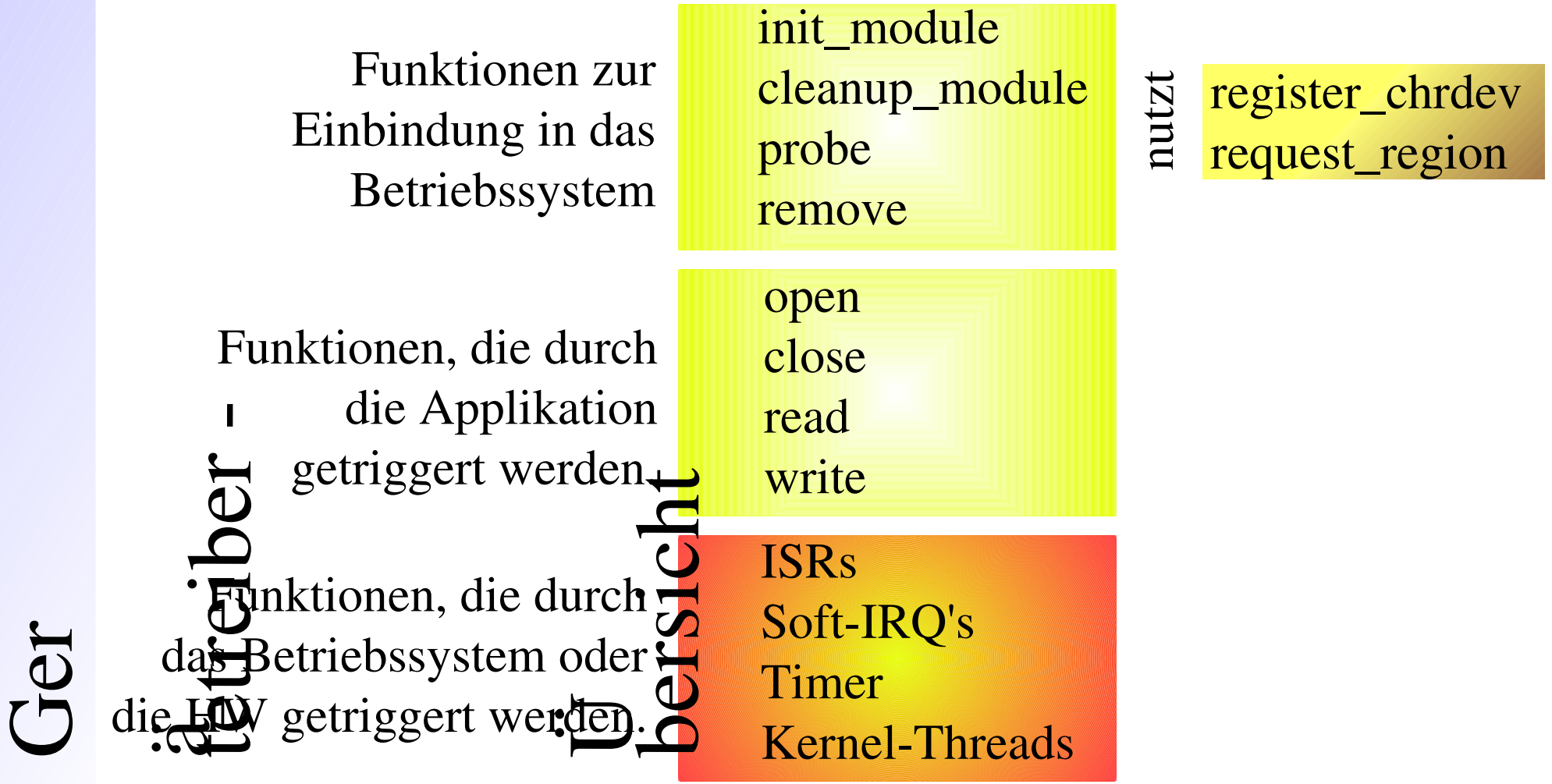
Gerätetreiber

Übersicht

Treiber-Identifikation

- Treiber müssen an der Applikationsschnittsstelle „sichtbar“ gemacht werden:
  - Character-Devices: Eintrag im Filesystem (Gerätedatei)
  - Block-Devices: Eintrag im Filesystem (Gerätedatei)
  - Network-Devices: Name (Parameter beim `ifconfig`)

Funktionen eines Gerätetreibers



Zusammenfassung

- Linux besitzt einen monolithischen Betriebssystemkern.
- Linux hat ein modernes Prozess-, Memory- und IO-Management.
- Linux entwickelt sich sehr schnell weiter.
- Besondere Entwicklungsschwerpunkte zur Zeit:
  - Virtualisierung
  - Echtzeitverhalten
  - Sicherheit