Eingebettete Betriebssysteme

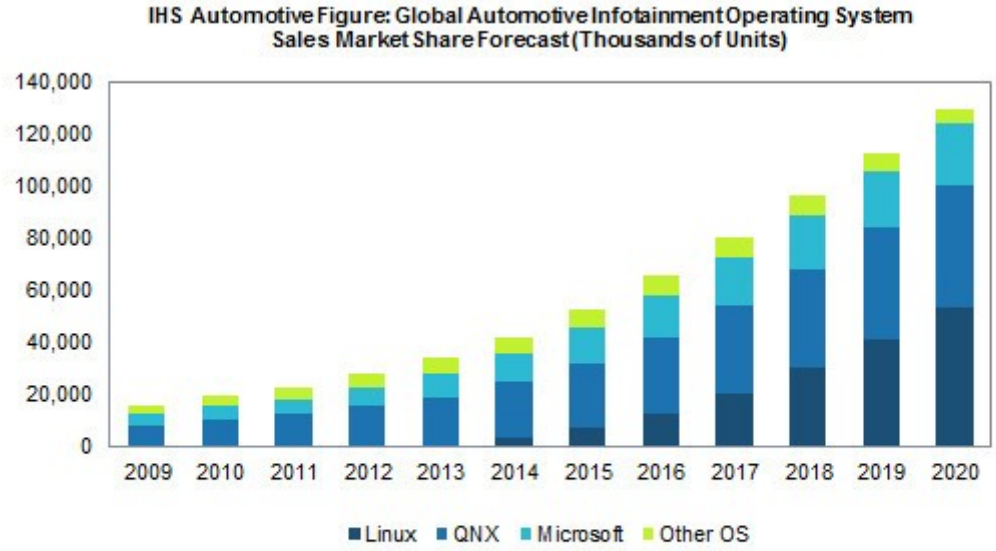
Bachelorstudiengang
Technische Informatik
Hochschule Pforzheim

Dipl.-Ing.(FH) Marc Jüttner

Linux

- 1991 von Linux Torvalds veröffentlicht
- Ursprünglich nur für IBM-PCs geplant
- Unix-ähnliches Betriebssystem als Alternative zu Minix
- Frei verfügbar unter GPLv2
- Multiuser und Multitasking
- Weltweite Entwicklung

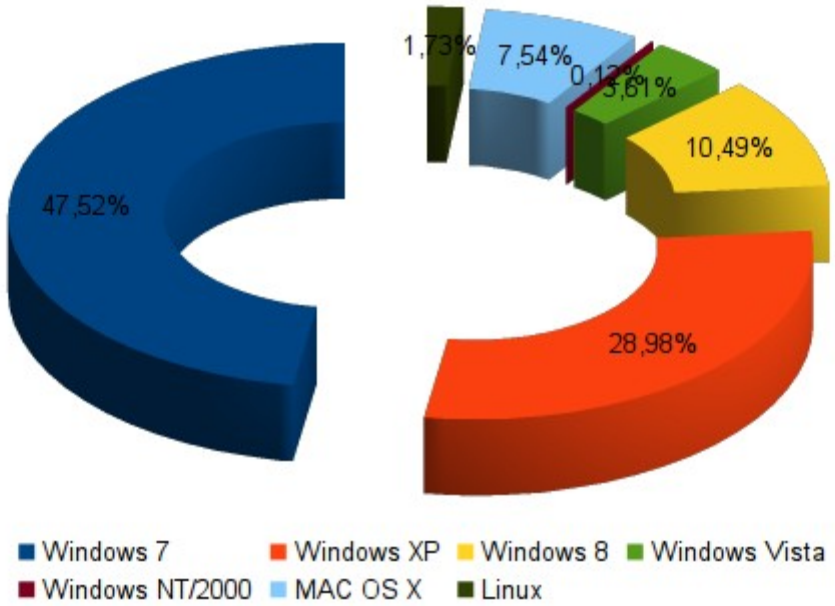
Marktanteil Automotive DRI



Source: IHS Automotive November 2013

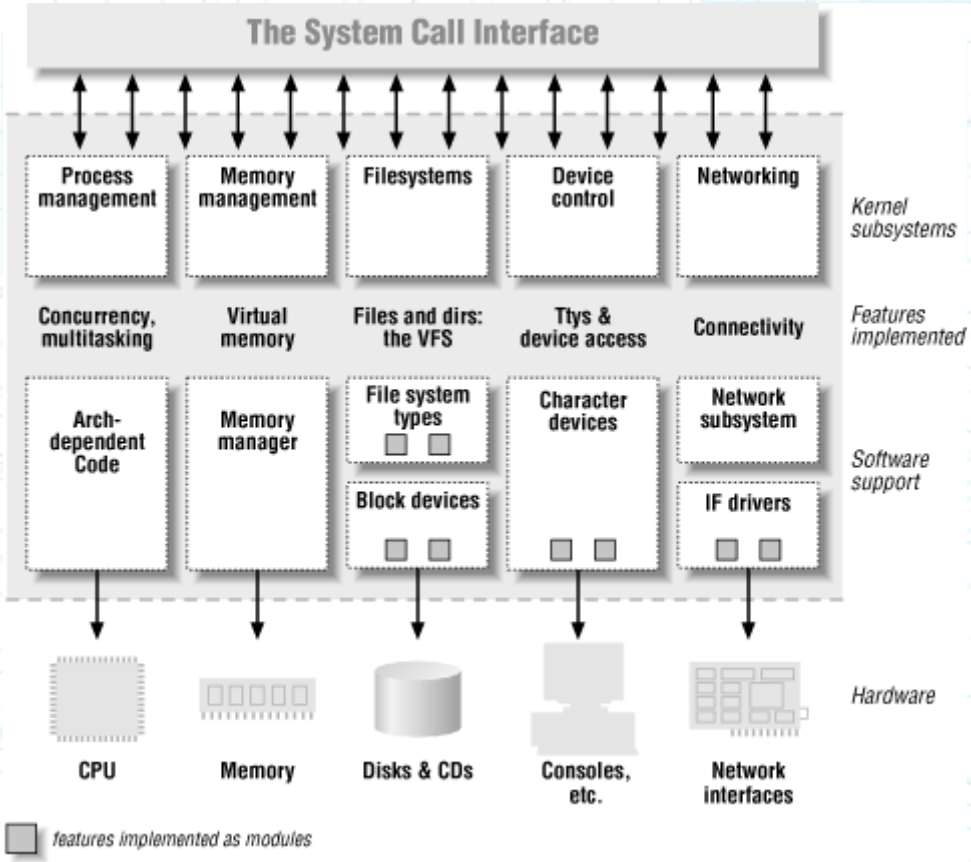
Quelle: bitbloke.de

Linux-PC-Marktanteil



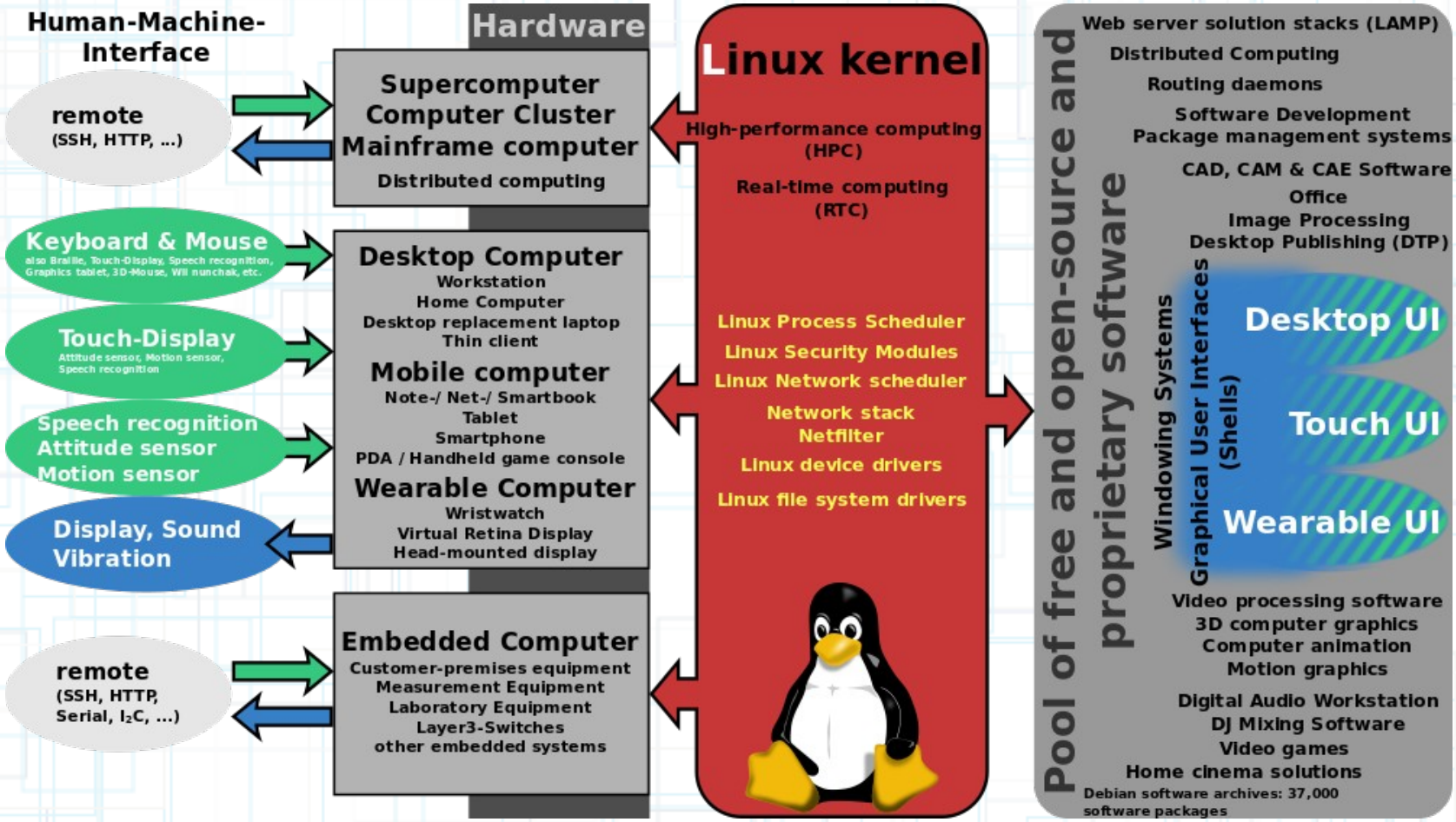
Datenquelle: netmarketshare.com

Linux-Aufbau



Quelle: Linux Device Drivers 3rd Edition

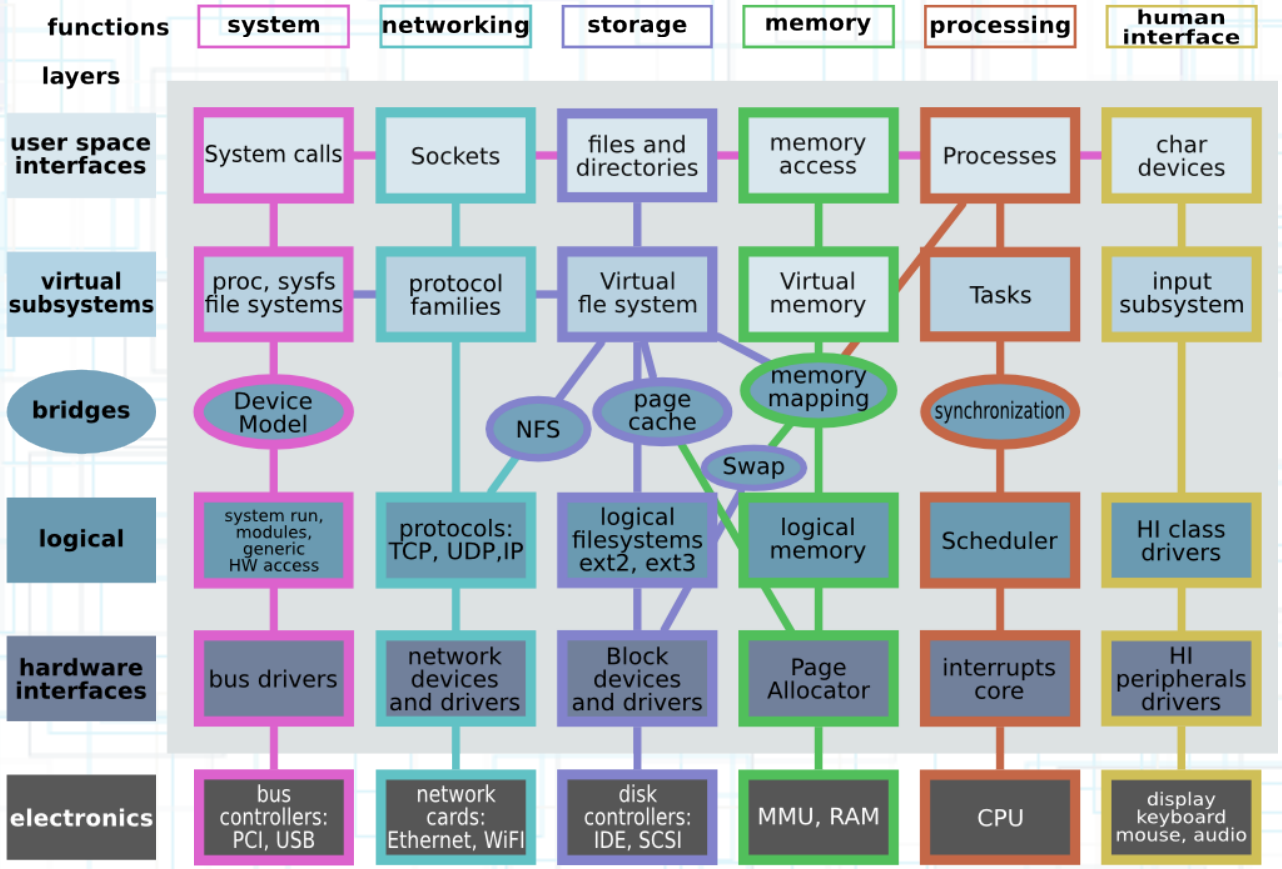
Linux



Quelle: Wikipedia

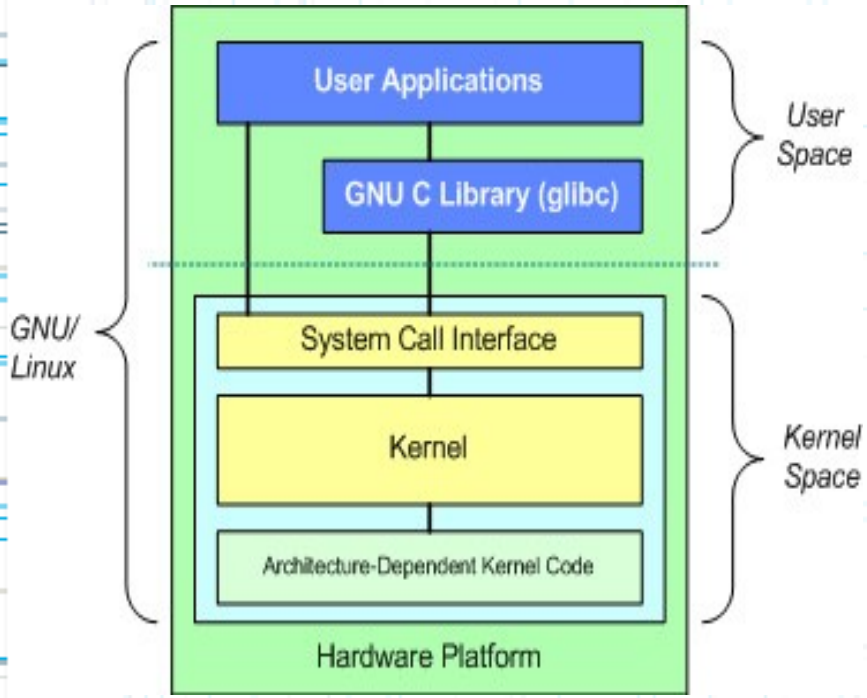
Linux

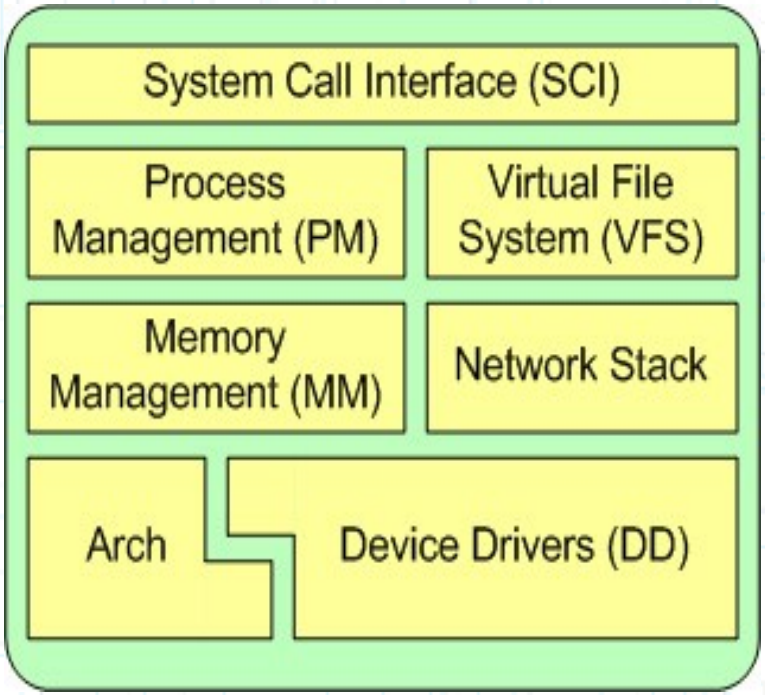
Linux kernel diagram



Quelle: http://makelinux.net/kernel/diagram

Linux-Struktur

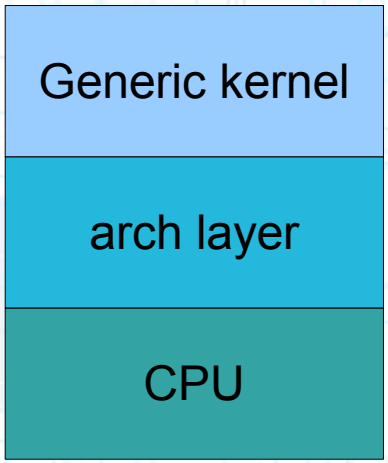




Quelle: M. Tim Jones, Architecture of the Linux kernel, http://www.ibm.com/developerworks/linux/library/l-linux-kernel/

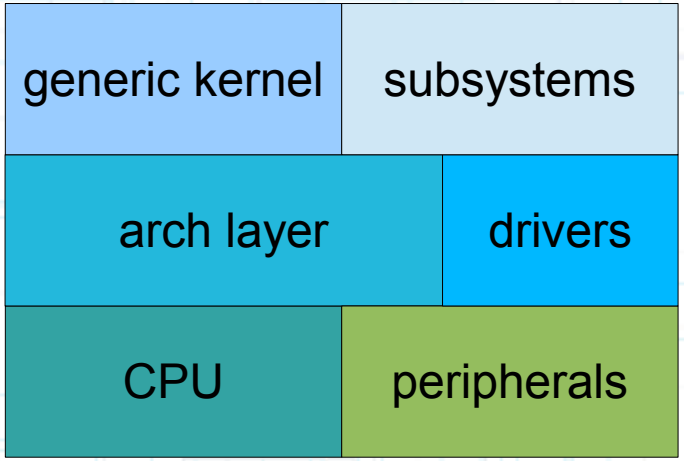
Linux-Kernelstruktur

- Aufteilung des Kernels in
  - Architektur
    - Interrupts
    - Dispatcher
  - Generischer Teil
    - Scheduler
    - Speicherverwaltung
    - Prozessverwaltung

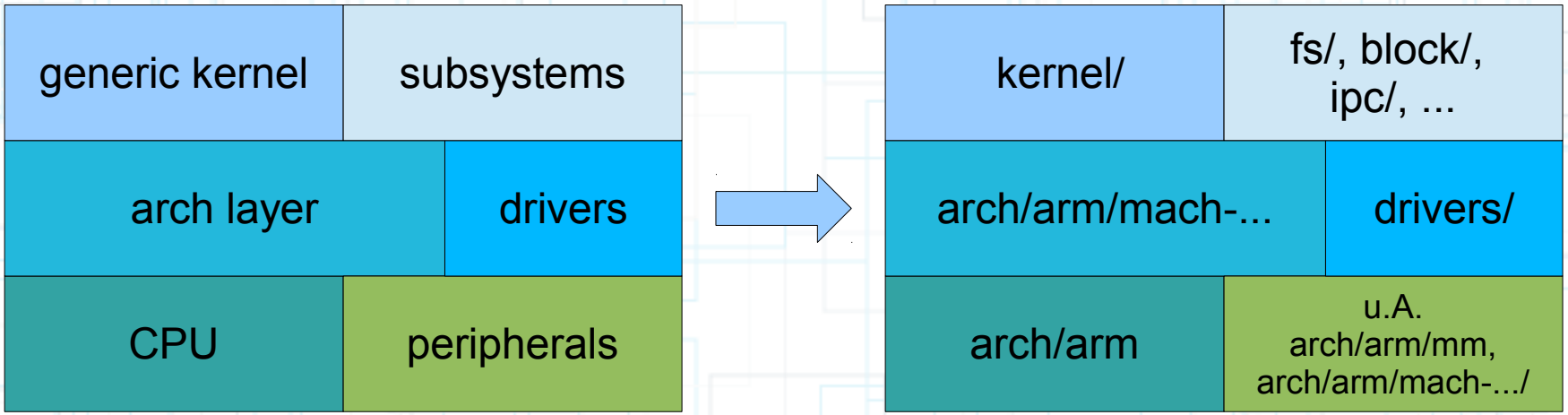


Linux-Kernelstruktur

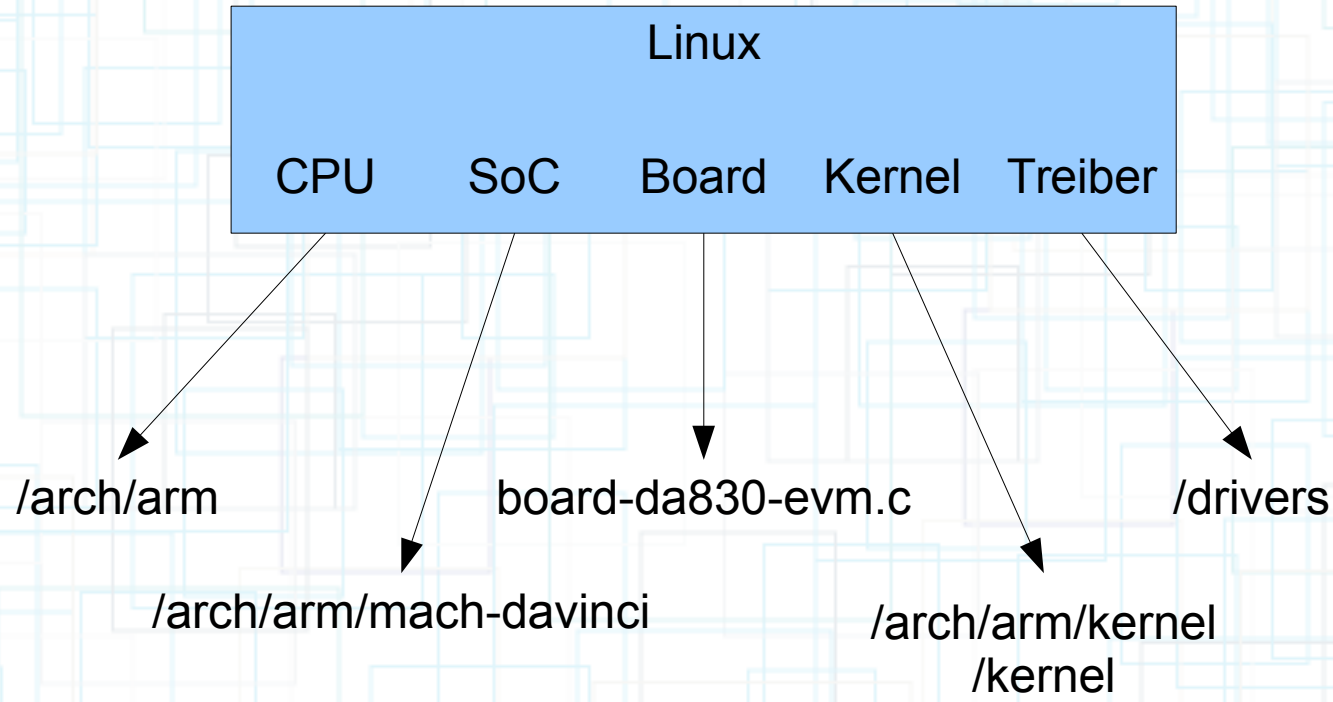
- Ein System besteht nicht nur aus der CPU
  - SoC
  - Boards (Gesamtsysteme)
- Unterscheidung!



Struktur und Quellcode



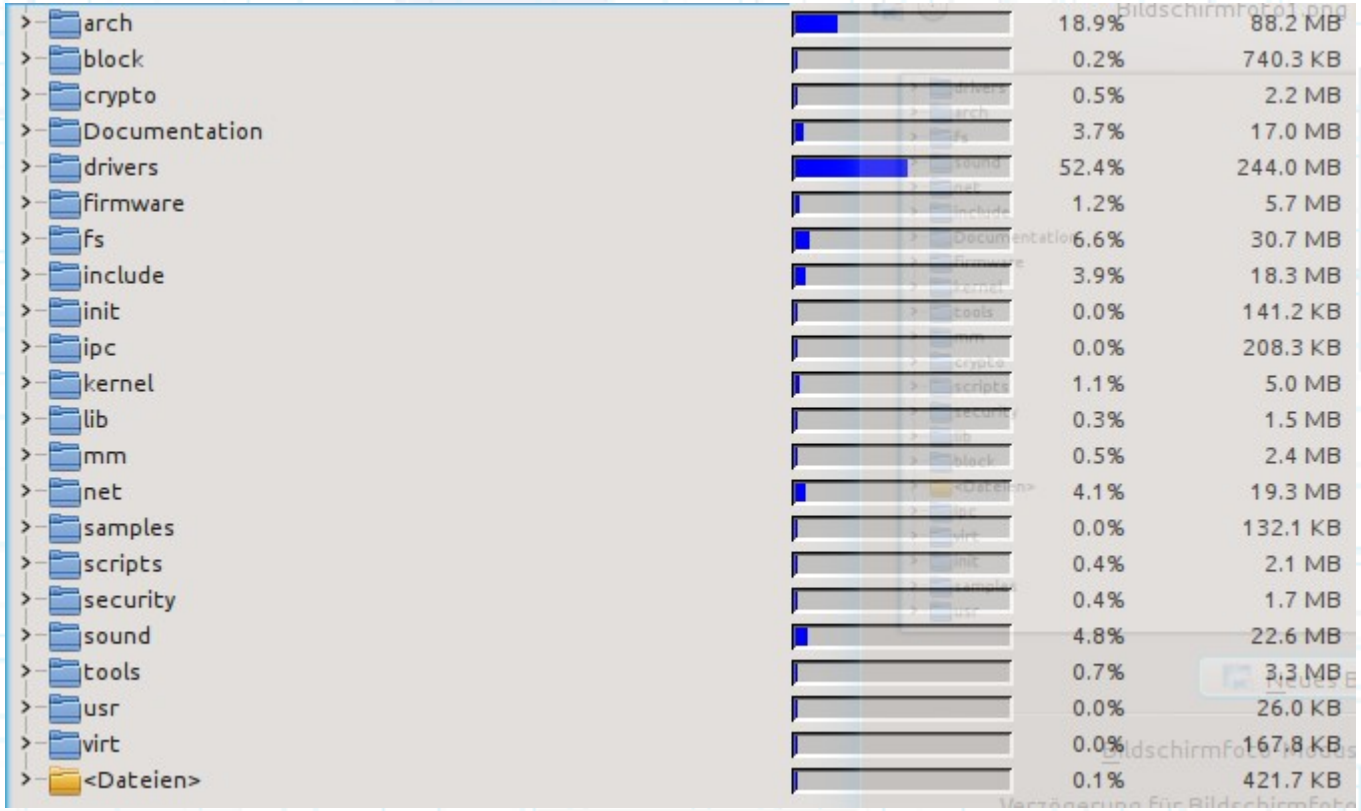
Boardkonzept



Quellenstruktur

- Linux ist im Quellcode verfügbar
  - Kann selbst modifiziert und übersetzt werden
- Die Kernelbestandteile sind strikt strukturiert
- Unterscheidung zwischen Quellestruktur und Rootdateisystem erforderlich

Linuxquellen



Verfügbare Treiber

- Unterverzeichnis „drivers“
- Alle Treiber, die Linux unterstützt
- Im Regelfall plattformunabhängig und nicht systemspezifisch

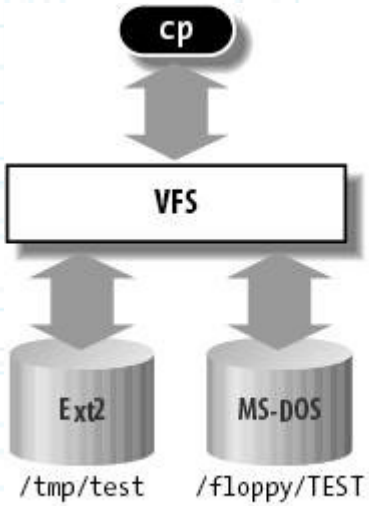
Everything is a file...

- Jedes Gerät wird wie eine Datei dargestellt
  - ... und behandelt
- Einheitliche Anwendungsschnittstelle
  - open(): Öffnen der Datei
  - close(): Schließen
  - read(): Daten lesen
  - write(): Daten schreiben

Virtuelles Dateisystem

- VFS, Virtual FileSystem
- Abstraktionsschicht oberhalb konkreter Dateien und Dateisysteme
- Versteckt Details der Dateizugriffe
- Einheitliche API
  - Auf jedes Dateisystem kann auf die gleiche Art zugegriffen werden

Beispiel



(a)

```
inf = open("/floppy/TEST", O_RDONLY, 0);
outf = open("/tmp/test",
       O_WRONLY|O_CREAT|O_TRUNC, 0600);
do {
    i = read(inf, buf, 4096);
    write(outf, buf, i);
} while (i);
close(outf);
close(inf);
```

(b)

Quelle: Understanding The Linux Kernel

Device files

- Geräte werden durch device files repräsentiert
- Je Gerät existiert ein device file
- Eindeutige Zuordnung zu einem Treiber
  - Major-/Minornummer
- Häufig in /dev
- Dynamische Dateien!

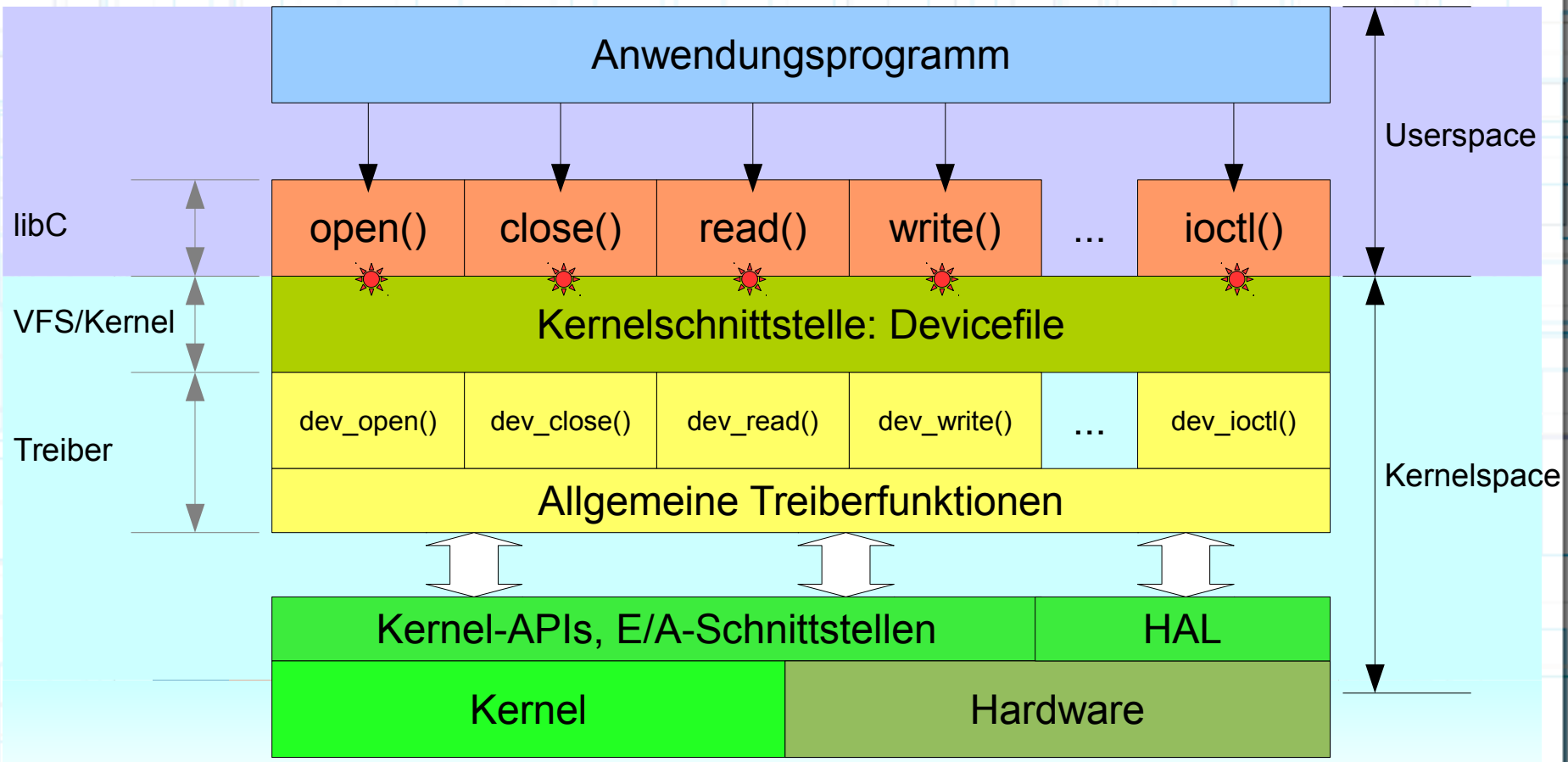
Device files

```
juemar@juemar-Compaq-610:~$ ls -l /dev/sda*
brw-rw---- 1 root disk 8, 0 Mär 28 11:26 /dev/sda
brw-rw---- 1 root disk 8, 1 Mär 28 11:26 /dev/sda1
brw-rw---- 1 root disk 8, 2 Mär 28 11:26 /dev/sda2
brw-rw---- 1 root disk 8, 3 Mär 28 11:26 /dev/sda3
juemar@juemar-Compaq-610:~$
```

```
juemar@juemar-Compaq-610:~$ ls -l /dev/ttyS1*
crw-rw---- 1 root dialout 4, 65 Mär 28 11:26 /dev/ttyS1
crw-rw---- 1 root dialout 4, 74 Mär 28 11:26 /dev/ttyS10
crw-rw---- 1 root dialout 4, 75 Mär 28 11:26 /dev/ttyS11
crw-rw---- 1 root dialout 4, 76 Mär 28 11:26 /dev/ttyS12
crw-rw---- 1 root dialout 4, 77 Mär 28 11:26 /dev/ttyS13
crw-rw---- 1 root dialout 4, 78 Mär 28 11:26 /dev/ttyS14
crw-rw---- 1 root dialout 4, 79 Mär 28 11:26 /dev/ttyS15
crw-rw---- 1 root dialout 4, 80 Mär 28 11:26 /dev/ttyS16
crw-rw---- 1 root dialout 4, 81 Mär 28 11:26 /dev/ttyS17
crw-rw---- 1 root dialout 4, 82 Mär 28 11:26 /dev/ttyS18
crw-rw---- 1 root dialout 4, 83 Mär 28 11:26 /dev/ttyS19
juemar@juemar-Compaq-610:~$
```

```
lrwxrwxrwx  1 root root           4 Mär 28 11:26 rtc -> rtc0
crw-------  1 root root      254,   0 Mär 28 11:26 rtc0
brw-rw----  1 root disk        8,   0 Mär 28 11:26 sda
brw-rw----  1 root disk        8,   1 Mär 28 11:26 sda1
brw-rw----  1 root disk        8,   2 Mär 28 11:26 sda2
brw-rw----  1 root disk        8,   3 Mär 28 11:26 sda3
crw-rw----  1 root disk       21,   0 Mär 28 11:26 sg0
crw-rw----+ 1 root cdrom      21,   1 Mär 28 11:26 sg1
lrwxrwxrwx  1 root root           8 Mär 28 11:26 shm -> /run/shm
crw-------  1 root root      10, 231 Mär 28 11:26 snapshot
drwxr-xr-x  3 root root         200 Mär 28 11:26 snd
brw-rw----+ 1 root cdrom      11,   0 Mär 28 11:26 sr0
lrwxrwxrwx  1 root root          15 Mär 28 11:26 stderr -> /proc/self/fd/2
lrwxrwxrwx  1 root root          15 Mär 28 11:26 stdin -> /proc/self/fd/0
lrwxrwxrwx  1 root root          15 Mär 28 11:26 stdout -> /proc/self/fd/1
crw-rw-rw-  1 root tty        5,   0 Mär 29 13:05 tty
```

Treiberkommunikation

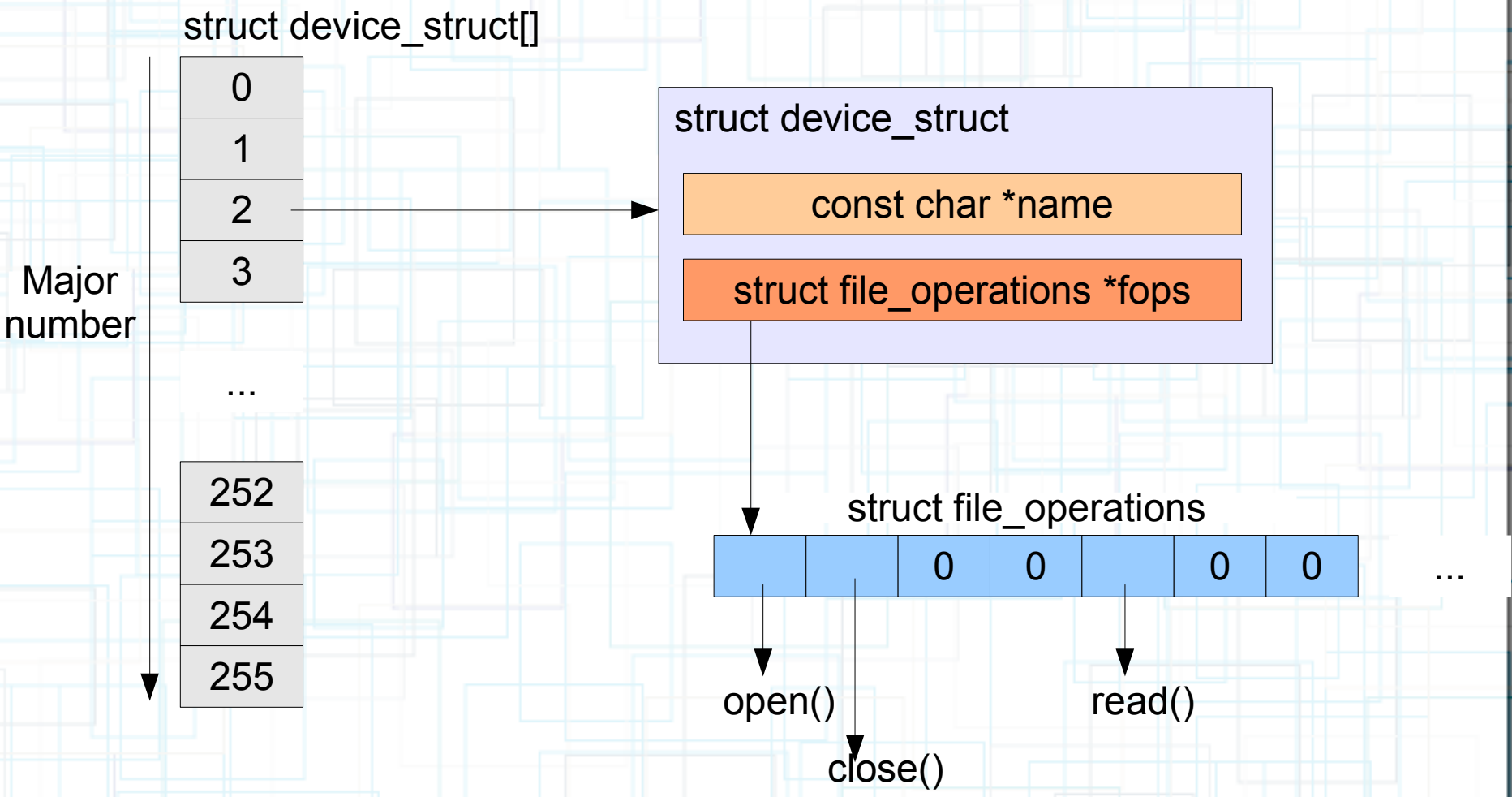


Treiberfunktionen

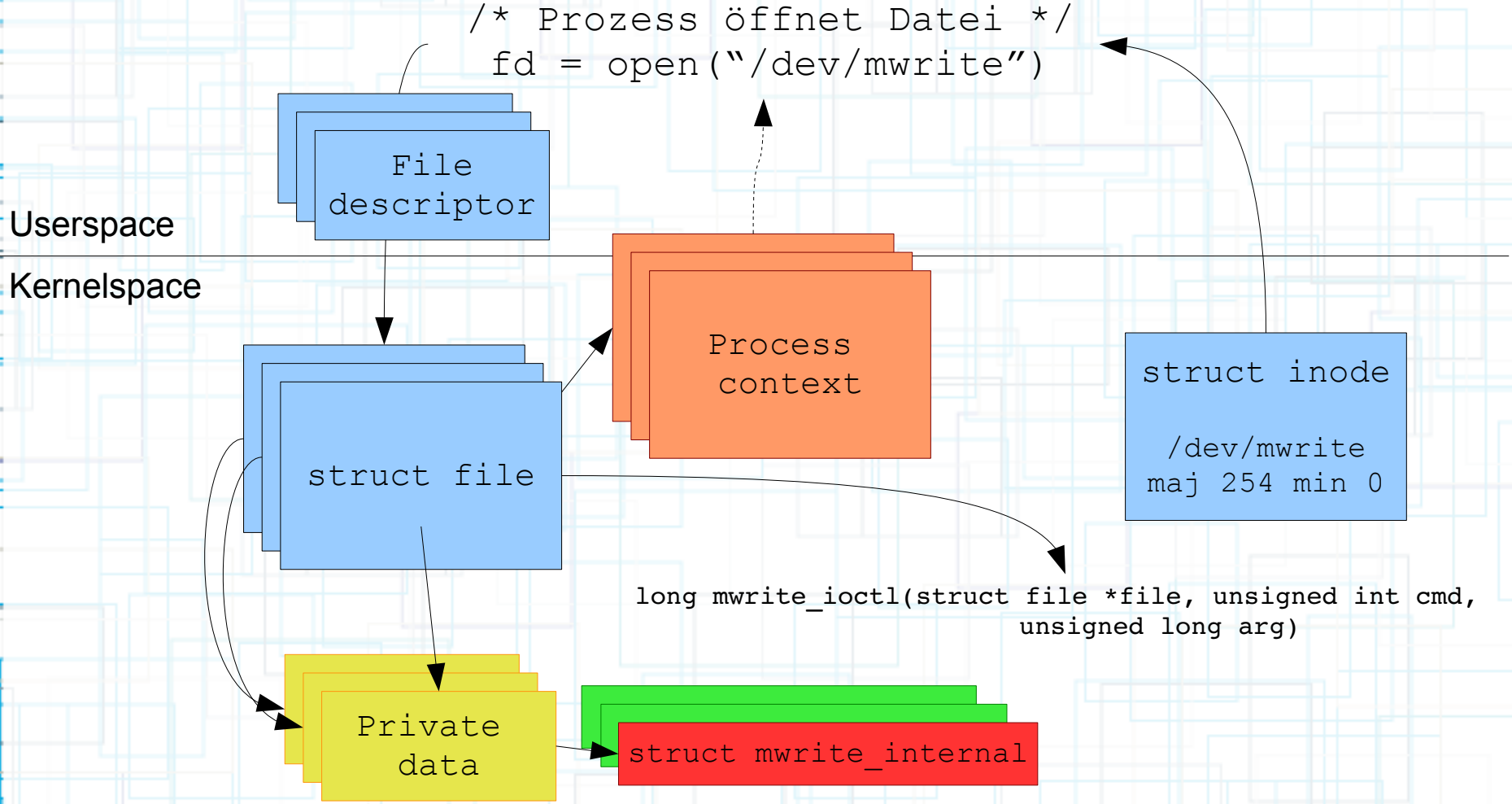
```
struct file_operations {
        struct module *owner;
        loff_t (*llseek) (struct file *, loff_t, int);
        ssize_t (*read) (struct file *, char *, size_t, loff_t *);
        ssize_t (*write) (struct file *, const char *, size_t, loff_t *);
        int (*readdir) (struct file *, void *, filldir_t);
        unsigned int (*poll) (struct file *, struct poll_table_struct *);
        int (*ioctl) (struct inode *, struct file *, unsigned int, unsigned long);
        int (*mmap) (struct file *, struct vm_area_struct *);
        int (*open) (struct inode *, struct file *);
        int (*flush) (struct file *);
        int (*release) (struct inode *, struct file *);
        int (*fsync) (struct file *, struct dentry *, int datasync);
        int (*fasync) (int, struct file *, int);
        int (*lock) (struct file *, int, struct file_lock *);
        ssize_t (*readv) (struct file *, const struct iovec *, unsigned long, loff_t *);
        ssize_t (*writev) (struct file *, const struct iovec *, unsigned long, loff_t *);
        ssize_t (*sendpage) (struct file *, struct page *, int, size_t, loff_t *, int);
        unsigned long (*get_unmapped_area)(struct file *, unsigned long, unsigned long, unsigned long, unsigned long);
#ifdef MAGIC_ROM_PTR
        int (*romptr) (struct file *, struct vm_area_struct *);
#endif /* MAGIC_ROM_PTR */
};
```

Aus: include/fs.h

Funktion im Kernel

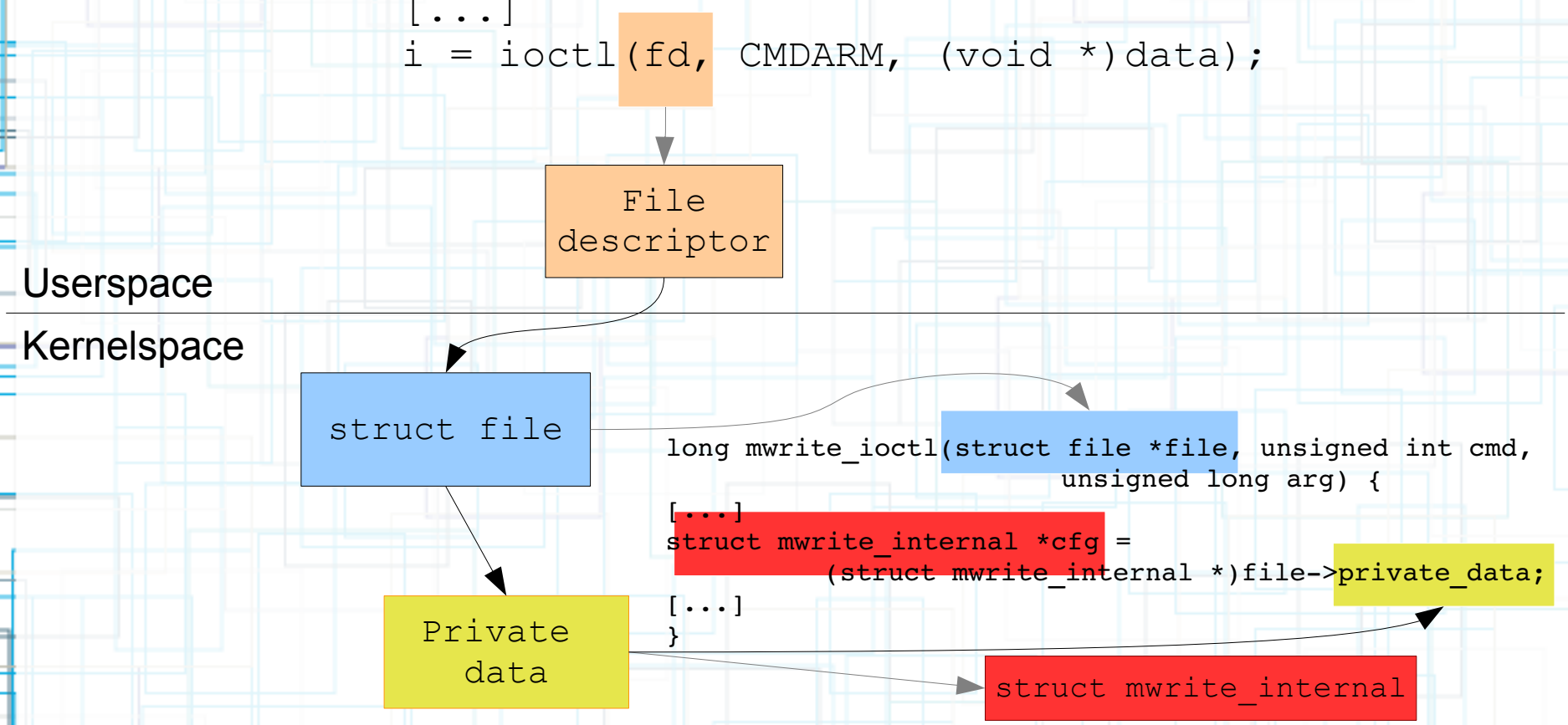


Funktionsübersicht



Funktionsweise schematisch

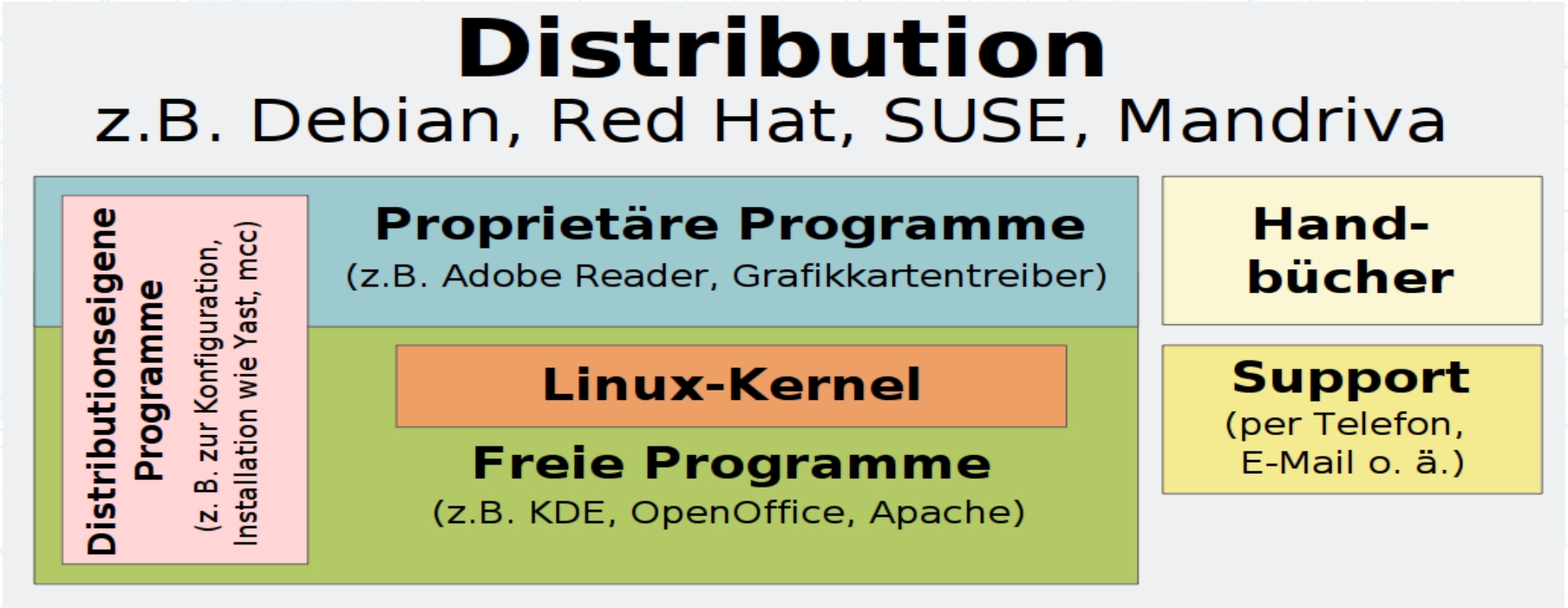
```
/* Prozess öffnet Datei */
fd = open("/dev/mwrite")
[...]
i = ioctl(fd, CMDARM, (void *)data);
```



Linux-System

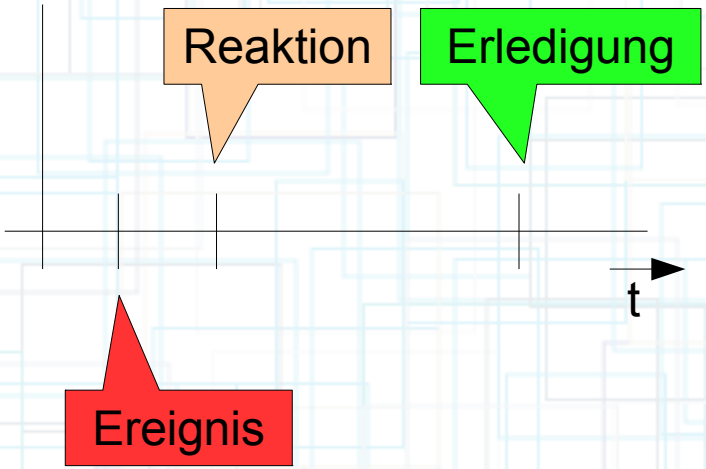
- „Linux“ bezeichnet nur den Kernel
- Rest der Systemsoftware meist aus GNU-Projekten
- Distributoren stellen kompatible Pakete zusammen
  - Ziel: Benutzbarkeit durch Endanwender
  - Distributionen
  - Versionsabhängigkeiten berücksichtigt

Distributionsstruktur



Echtzeitforderung

- Kurz gesagt...
  - Antwortzeit
    - *Responsivity*
  - Ausführungszeit
    - *Deadline*



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.

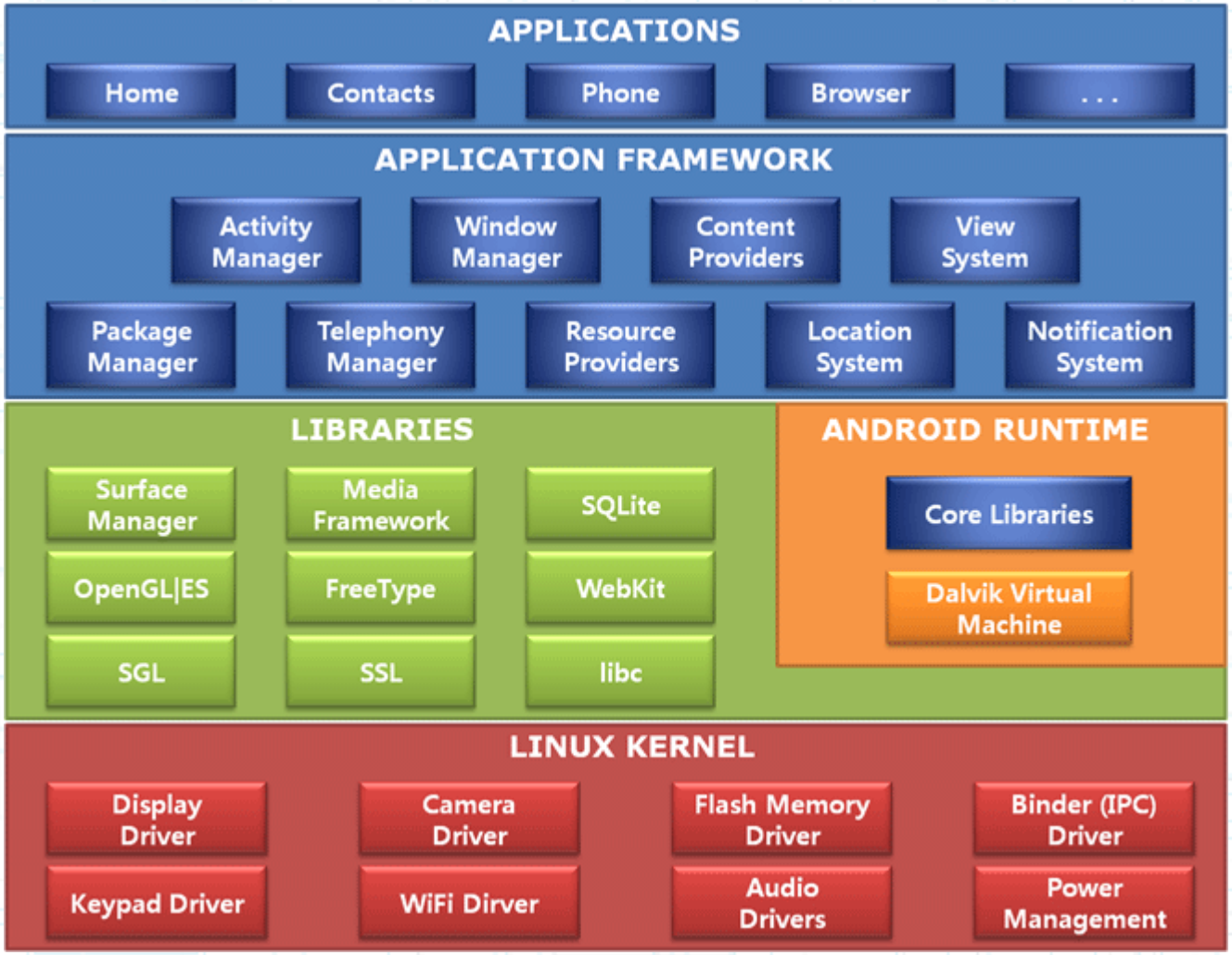
Realtime-Linux

- „Real“ ist nicht gleich „fast“!
- Erweiterung des Standard-Linux
  - Entfernen globaler Lockingmechanismen
  - Unterbrechbarkeit von Critical Sections
- Abwägung erforderlich: Performance vs. Determinismus
- Hauptunterschied liegt im Systemdesign!

Android

- Kein „Betriebssystem“ im eigentlichen Sinne
  - Zusätzliche Frameworkkomponenten
- Eher ein Gesamtsystemansatz
- Basierend auf Linux
- Sehr konkrete Hardwareanforderungen

Android-Struktur



Quelle:
blog.cubrid.org

Libraries

- Surface Manager verwaltet Puffer für die Bildausgabe von Apps
- Media framework zur Mediensteuerung und Codecverwaltung
- Database engine zur Datenspeicherverwaltung
- openGL für 2D-/3D-Rendering

Android Runtime

- Dalvik Virtual Machine
  - Höhere Effizienz bei begrenzten Ressourcen durch vorkompilierte Dateien
  - Parallelinstanzen möglich
    - Sicherheit, Isolierung
  - Speichermanagement und Threading
- Core Libraries
  - Bereitstellen von Java-Kernfunktionen

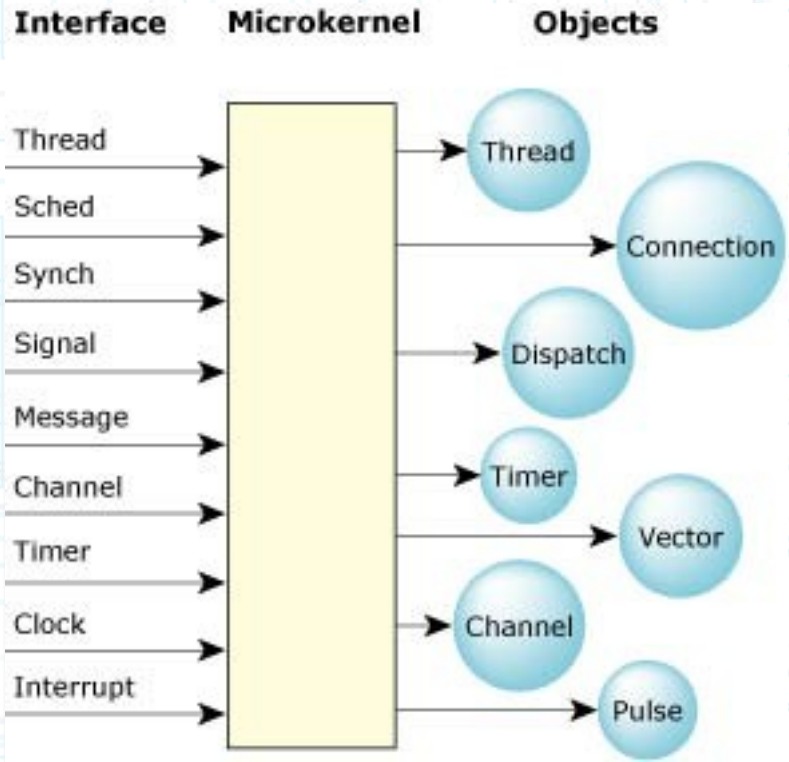
Application Framework

- Activity Manager steuert App-Lebenszyklus
- Content Providers zum Teilen von Daten zwischen Apps
- Telephony Manager
- Location Manager
  - GPS oder Funkortung (Netz oder Wifi)
- Resource Manager

QNX

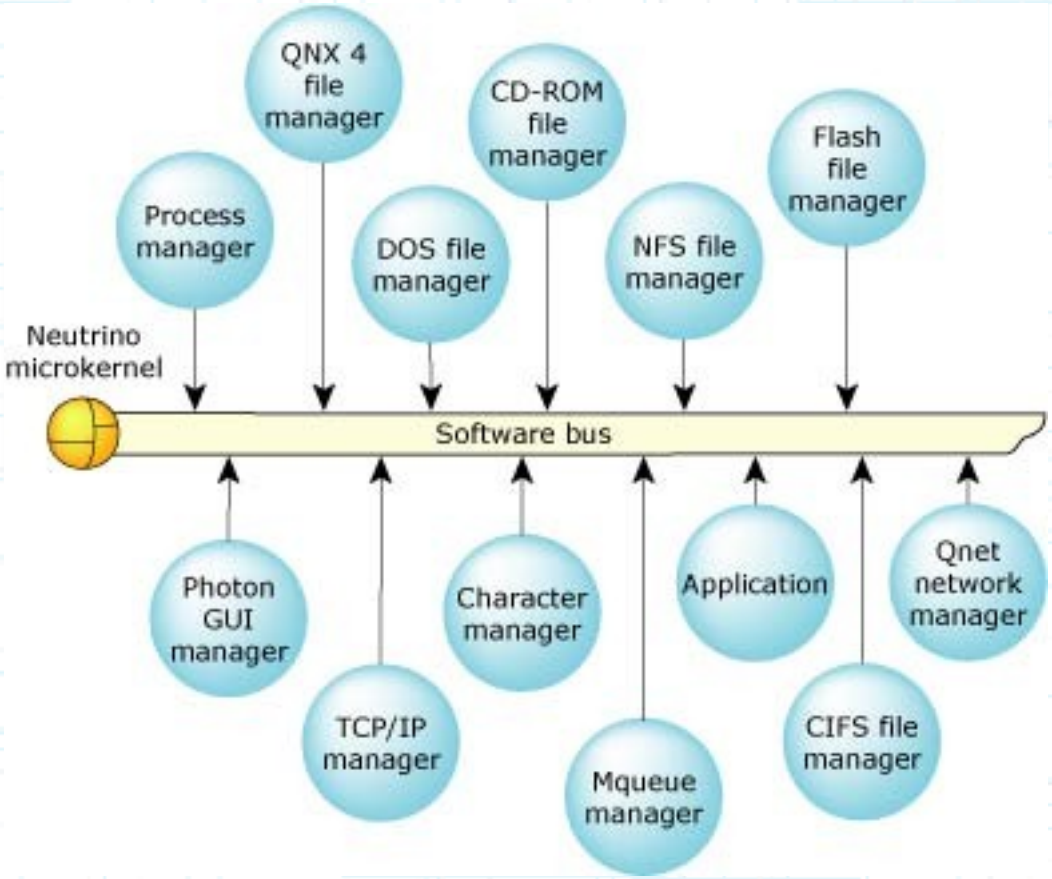
- Echtzeitfähiges Betriebssystem
- Mikrokernelansatz
- Usermode/Kernelmode
- Thread-/Taskprogrammierung über pthread-API (POSIX Threads)

QNX-Struktur



Quelle: qnx.com

QNX-Struktur



Quelle: qnx.com

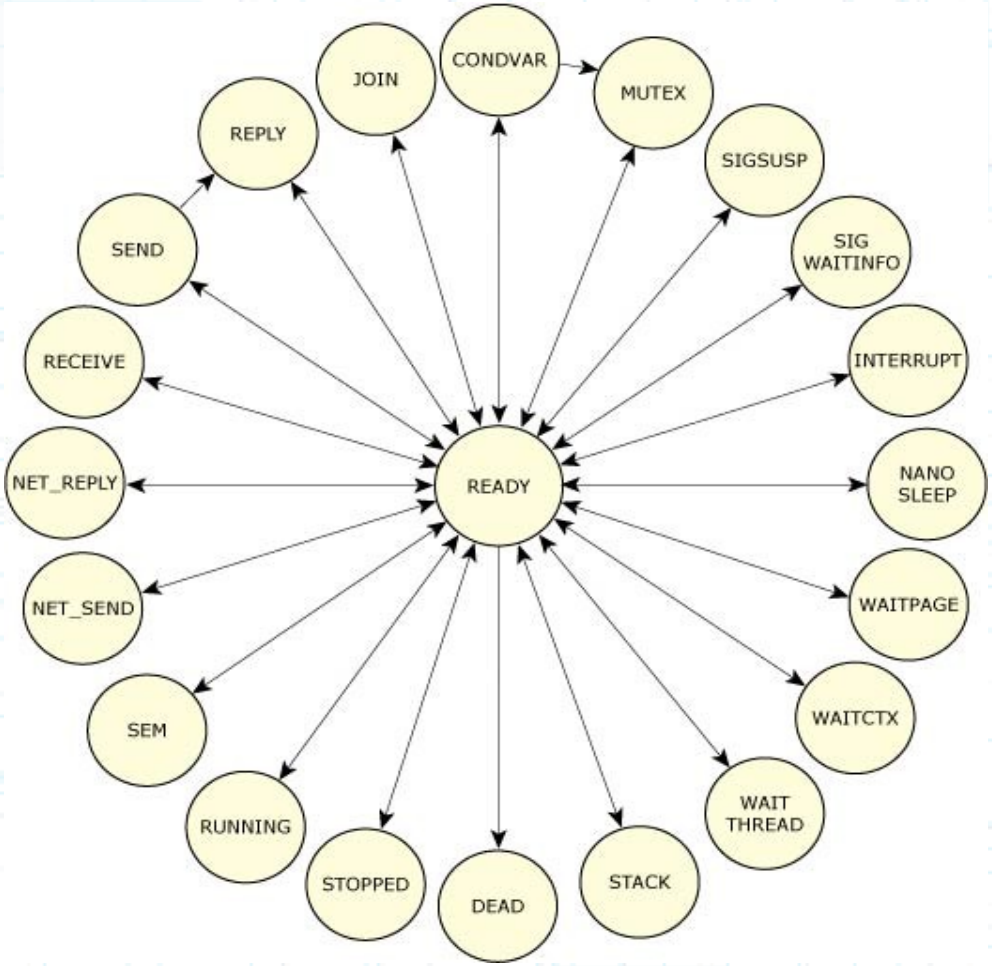
QNX: Kernelfunktionen

- Thread services
- Signaling services
- Message-passing
- Synchronization
- Scheduling und Prozessverwaltung
- Timer services

Spezialitäten

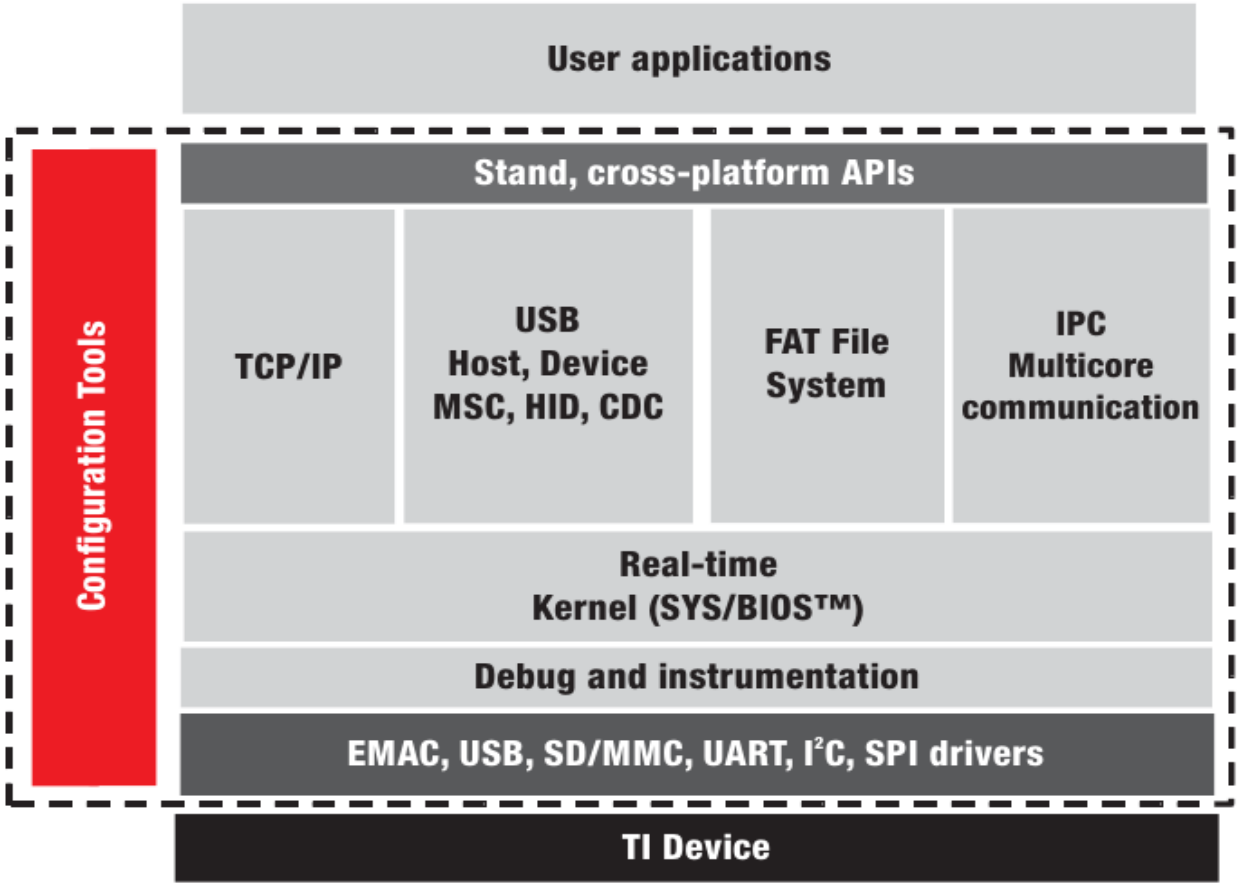
- Gerätetreiber sind normale Prozesse
  - Usermodeprozesse
  - Entwicklung und Debugging wie normale Anwendungen
- Transparente Netzwerkdienste
  - IPC und Message-Passing machen einzelne Systeme oder Systemteile transparent zugreifbar

Threadzustände



Quelle: qnx.com

SYS/BIOS



Quelle: www.ti.com