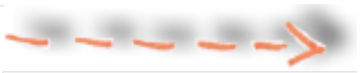
Der Linux-Kernel

Lernziele

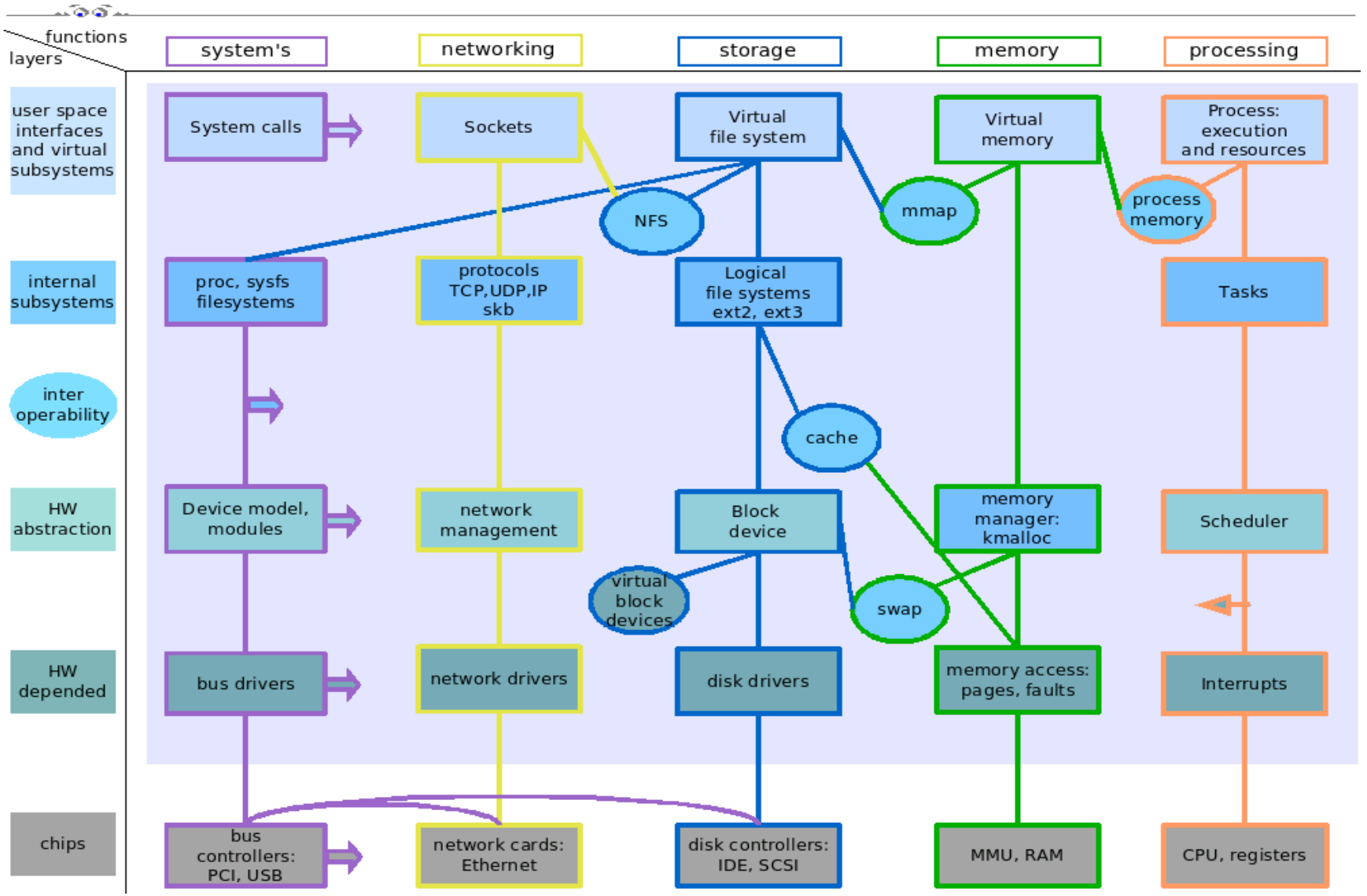


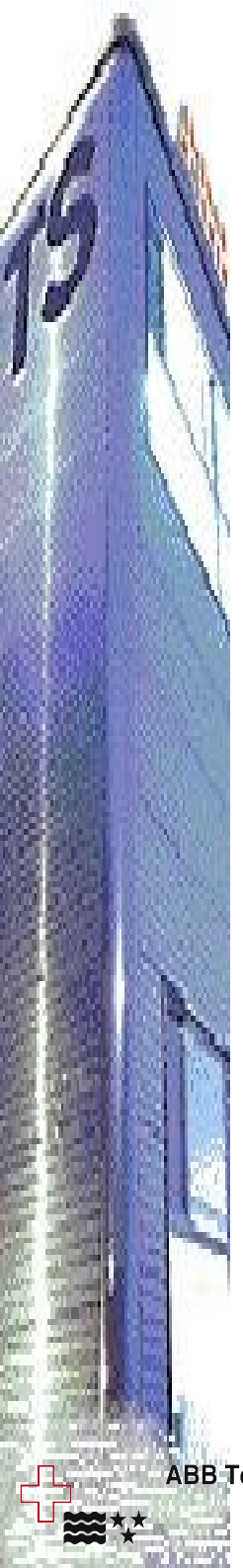
- Die verschiedenen Kerneltypen kennen
- Die Version des laufenden Kernels identifizieren können
- Module anzeigen, laden und entladen können
- Die Konfiguration des Modul-Lademechanismus beherrschen

Der Linux-Kernel

- Der Linux-Kernel ist der innerste Teil des Betriebssystems
- Er stellt elementare Funktionen zur Verfügung
  - Steuerung der Hardware
  - Netzwerk, Interkommunikation
  - Zugriff auf Massenspeicher
  - Speicherverwaltung
  - Prozessverwaltung

Der Linux-Kernel



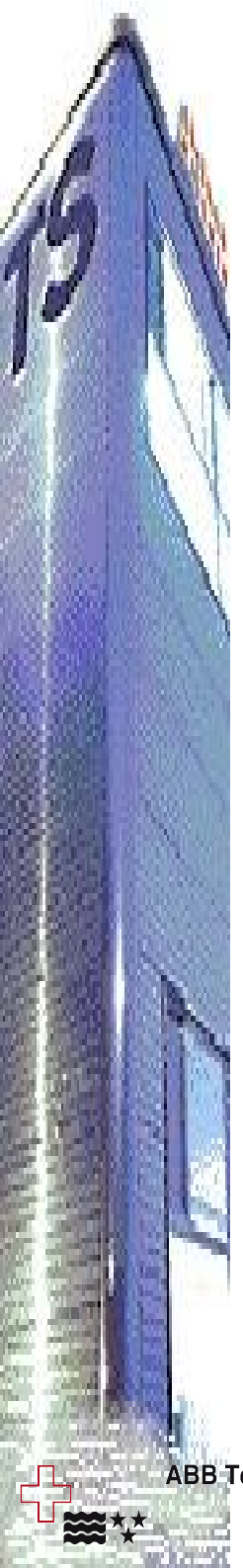


Monolithische und modulare Kernel

- Von seiner Struktur her ist der Linux-Kernel ein traditioneller, monolithischer Unix-artiger Betriebssystemkern
- Neue Funktionalität, erfordert damit immer eine Veränderung des Kernels
- Den monolithischen Kerneln steht der Micro-Kernel-Ansatz gegenüber
- Monolithische Kernel sind viel einfacher effizient zu implementieren

Monolithische und modulare Kernel

- Bei frühen Linux-Versionen (Kernel 1.x) mit monolithischen Kernel war der Austausch der Netzwerkkarte daher gleichbedeutend mit der Neuübersetzung des Kernels
- Je nach Hardware konnte das durchaus mehrere Stunden in Anspruch nehmen
- Die Folge davon sind Modulare Kernel (ab Kernel 2.x)



Monolithische und modulare Kernel

- Module sind Kernel-Teile, die dynamisch (also zur Laufzeit) hinzugeladen und auch wieder entfernt werden können
- Der Zugriff auf modulare Treiber erfolgt nur marginal langsamer als auf statisch eingebundene
- Module sind Erweiterungen des Kernels
- Jedes verwendete Modul muss genau zum Kernel passen

Seinen Kernel kennen

- Linux Systeme haben die unterschied-lichsten Kernel
- Welcher Kernel läuft?
- Dies kann man herausfinden mit dem Kommando
  - `uname -rv`
  - Die Optionen zeigen den Kernel Release (-r) und die Kernel Version (-v)

Seinen Kernel kennen

- `uname -rv`

  ```
  2.4.20-4GB #1 Wed Dec 3 10:42:32 UTC 2003
  ```

  - Es handelt sich hier um einen 2.4er Kernel mit Release Status 20
  - »4GB« ist eine Kompiler Option
  - »#1« bezeichnet den Kompilerlauf
  - und das Datum des Kompilates
- Der Kernel Release bestimmt die Charakteristik des Kernels

Seinen Kernel kennen

- `uname -rv`

  ```
  2.6.13-15.13-smp #1 SMP Tue Nov 28 13:43:50
  ```

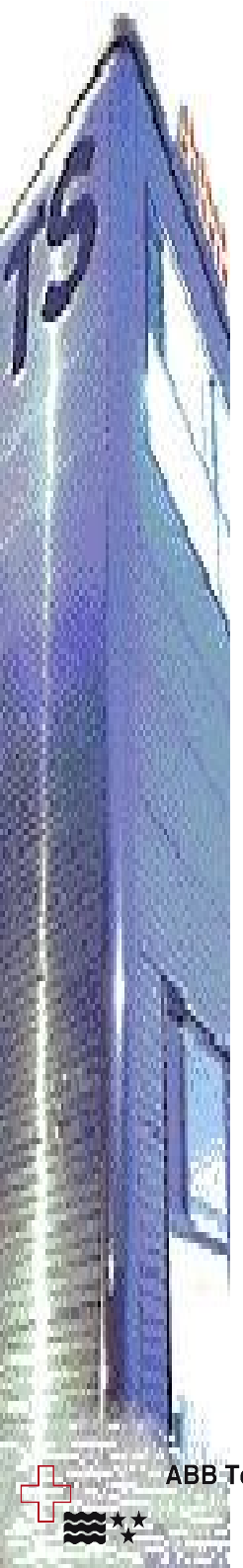
  - Es handelt sich hier um einen 2.6er Kernel mit Release Status 13
  - Der Subrelease 15.13 deutet auf einen Distributions-Kernel hin
  - »smp« ist die Kompiler-Option für Symetric-Multi-Prozessor Architektur

Seinen Kernel kennen

- Die Versionsnummer wird wie folgt interpretiert
  - Die erste Nummer ist für sehr grosse Umstellungen des Kernels vorgesehen
    - Bei 2.0 unter anderem der Modul-Mechanismus und die Mehrprozessor Unterstützung
  - Die zweite Nummer bedeutet immer noch grosse Umstellungen
    - Austausch von Modulen und Unterstützung für neue Hardware

Seinen Kernel kennen

- Die zweite Nummer hat noch eine zusätzliche Bedeutung:
  - Ist sie ungerade, handelt es sich um einen Kernel aus dem Entwickler-Zweig
    - sie sind allesamt nicht für den Produktiv-Einsatz geeignet
  - Bei gerader zweiter Nummer handelt es sich um eine stabile Kernel-Version
- Die dritte Nummer beschränkt sich auf Bugfixes

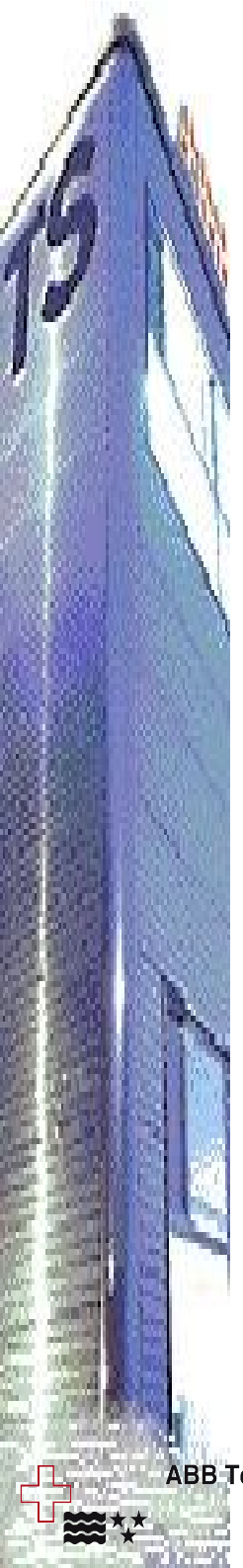


Seinen Kernel kennen

Version	Datum	Anmerkung	Dateien [2]	LoC [3]	Aktuell
0.01	17. September 1991	Initial Public Release	90	8.413	-
0.02	5. Oktober 1991		—	-	-
1.0.0	13. März 1994		563	170.581	-
1.1.0	6. April 1994	Entwicklungsversion	561	170.320	-
1.2.0	7. März 1995		909	294.623	-
1.3.0	12. Juni 1995	Entwicklungsversion	992	323.581	-
2.0.0	9. Juni 1996		2.015	716.119	2.0.40
2.1.0	30. September 1996	Entwicklungsversion	1.727	735.736	-
2.2.0	26. Januar 1999		4599	1.676.182	2.2.26
2.3.0	11. Mai 1999	Entwicklungsversion	4.721	1.763.358	-
2.4.0	4. Januar 2001		8.187	3.158.560	2.4.36
2.5.0	23. November 2001	Entwicklungsversion	9.893	3.833.603	-
2.6.0	18. Dezember 2003		15.007	5.475.685	2.6.24

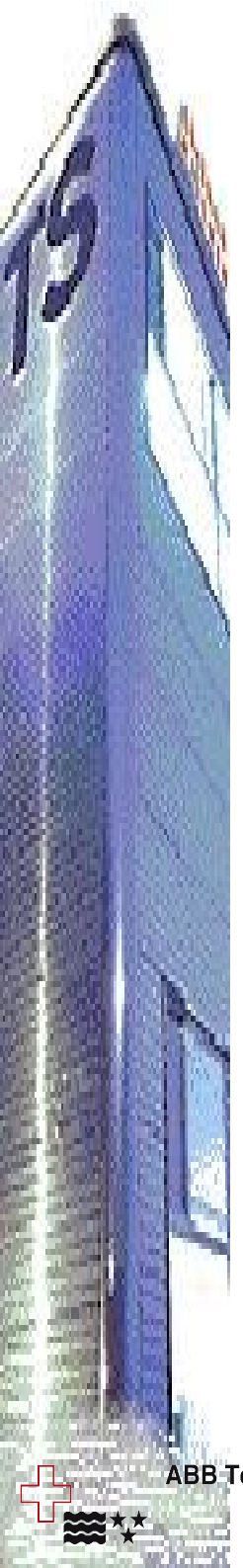
[2] Gezählt mit "find . -type f | wc -l"
[3] Lines of Code, gezählt mit "find . -name *.[hcS] | xargs cat | wc -l"

Fragen?



Übungen

- [12.1] Welche Version hat Ihr aktueller Kernel und wann wurde er übersetzt?
- Ist es ein Multiprozessor Kern?
- Ist er für 32-Bit oder 64-Bit Architektur?



Übungen

- [12.2] Was bedeuten die anderen Felder in der Ausgabe von »uname -a«
- Wofür sind sie gut?

Original- und Distributions-Kernel

- Original Kernel findet man unter

  http://www.kernel.org

- Patches und Module müssen immer zum aktuellen Kernel passen
- Zum Kompilieren von Kernelmodulen müssen die Header Dateien des aktuellen Kernels installiert sein
- Distributionen verwenden oft angepasste Kernel

Kernel Module

- Die Kernel-Module befinden sich unter */lib/modules/<Kernel-Version>*
- `ls /lib/modules/`uname -r``
- Eine Liste aller verfügbaren Module können mit *modprobe* ausgegeben werden
-
```
modprobe -l
/lib/modules/2.4.20-4GB/kernel/abi/cxenix/abi-cxenix.o
/lib/modules/2.4.20-4GB/kernel/abi/ibcs/abi-ibcs.o
/lib/modules/2.4.20-4GB/kernel/abi/isc/abi-isc.o
...
```

Kernel Module

- Welche Module gerade im Kernel geladen sind, verrät das Programm *lsmod*

```
# lsmod
Module                  Size  Used by
vfat                   10476  0 (unused)
fat                    31640  0 [vfat]
usbserial              19580  0 (autoclean) (unused)
3c59x                  27024  1
usb-uhci               23664  0 (unused)
usbcore                63116  1 [usbserial usb-uhci]
ext3                   85928  1 (autoclean)
jbd                    50896  1 (autoclean) [ext3]
```

Kernel Module

- Zur Laufzeit des Kernels können Module jederzeit nachgeladen werden
- Mit dem Programm *insmod* kann ein Kernelmodul geladen werden
- `# insmod cramfs`
  `Using /lib/modules/2.4.20-4GB/kernel/fs/cramfs/cramfs.o`
- Umgekehrt kann man mit *rmmod* ein Kernelmodul wieder entladen
- `# rmmod cramfs`

Module manuell laden und entladen

- Bei *modprobe* handelt es sich um einen Aufsatz auf *insmod*
- *modprobe* kümmert sich um Abhängigkeiten und ist konfigurierbar
- `modprobe cramfs`
- `modprobe -r cramfs`
- *modprobe* kann ganze Klassen von Modulen probeweise laden

Module automatisch laden und entladen

- Der Kernel ist in der Lage, Module automatisch nachzuladen
- Dazu dient der kernelinterne Daemon *kmod*
  - Bei SUSE erledigt das der *hwscand*
- Automatisch geladene Module werden markiert
  - lsmod zeigt die Ergänzung »(autoclean)«
- Module werden nicht automatisch entfernt

Module automatisch laden und entladen

- Um unbenutzte Module automatisch entfernen zu lassen, verwendet man
  - `# modprobe -r`

    oder
  - `# rmmod -a`
- Nur automatisch geladene Module werden entfernt (erst beim 2. Aufruf)
- Manuell geladene Module müssen auch manuell wieder entfernt werden

Modul-Abhängigkeiten

- Module können voneinander abhängen
  - `lsmod`

```
Module  Size  Used by
vfat   10476  0 (unused)
fat    31640  0 [vfat]
```

- Obwohl das Modul *fat* nicht benutzt ist, kann es nicht entfernt werden
  - `rmmod fat`

```
fat: Device or resource busy
```

- Man muss zuerst das Modul *vfat* entfernen

Modul-Abhängigkeiten

- Umgekehrt kann man ein Modul nicht laden, wenn es andere Module benötigt
  - `# insmod vfat`

```
Using /lib/modules/.../vfat.o
/lib/modules/.../vfat.o: unresolved symbol fat_scan
/lib/modules/.../vfat.o: unresolved symbol fat_dir_empty
```

- Die Modulabhängigkeiten werden vom Programm *depmod* aufgelöst
  - `/sbin/depmod -a`
- Wird meist in den Boot-Skripten aufgerufen

Modul-Abhängigkeiten

- *depmod* schreibt aufgrund der in den Modulen gefundenen Informationen (Symbole) die Abhängigkeiten in die Datei

  `/lib/modules/<Kernel-Version>/modules.dep`

  - `# grep vfat /lib/modules/2.4.20-4GB/modules.dep`
    `/lib/modules/2.4.20-4GB/kernel/fs/vfat/vfat.o: fat.o`

- Diese Datei wird von *modprobe* ausgewertet

  - `# modprobe vfat`

Konfiguration von Modulen

- An Module können die Parameter nicht als Kernel-Parameter mitgegeben werden
- Parameter müssen beim Laden des Moduls mitgegeben werden
  - `# insmod 3c501 io=0x280 irq=5`
- Diese Parameter kann man in die Datei `/etc/modules.conf`

  schreiben, wo sie auch von *modprobe* gelesen werden

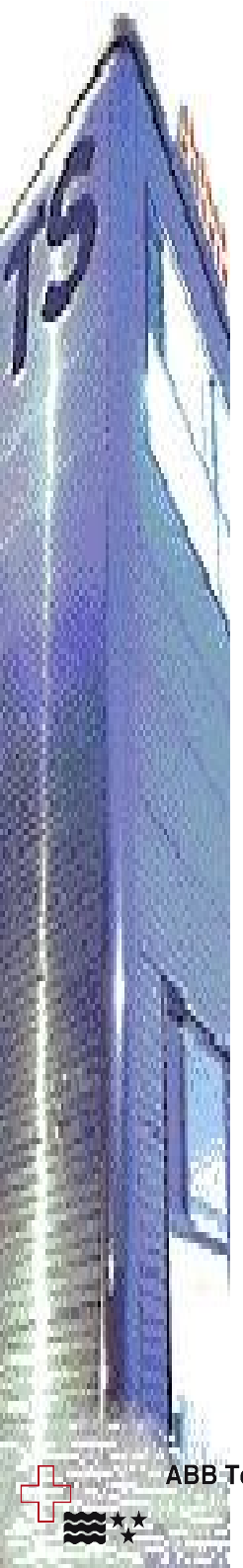
Konfiguration von Modulen

- Welche Parameter ein Modul akzeptiert, verrät die Ausgabe von *modinfo*
  - `# modinfo 3c501`

```
filename: /lib/modules/.../kernel/drivers/net/3c501.o
description: <none>
author: <none>
license: "GPL"
parm: io int, description "EtherLink I/O base address"
parm: irq int, description "EtherLink IRQ number"
```

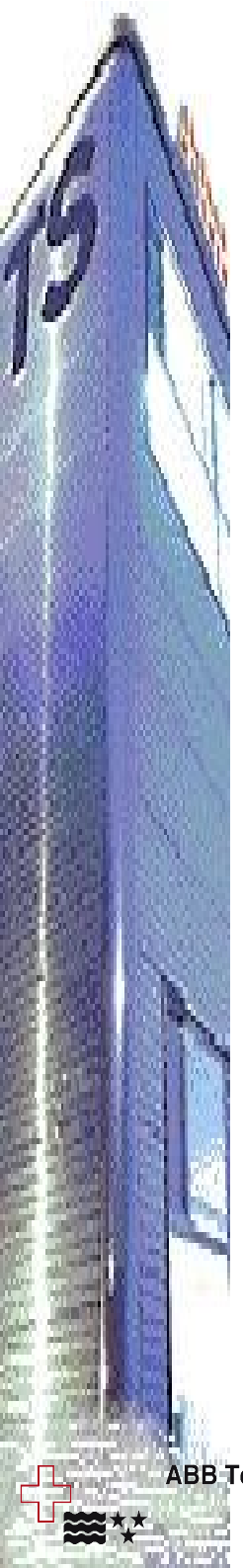
- Die Dokumentation der Module befindet sich unter */usr/src/linux/Documentation*

Fragen?



Übungen

- [12.3] Welche Module in Ihrem aktuellen Systemkern unterstützen Parameter und welche Parameter sind das?



Übungen

- [12.4] Warum ist die Angabe der Lizenz eines Moduls wichtig für den Linux-Kernel?

Modul-Aliase

- Neben den Modul-Parametern hat die Datei *modules.conf* noch eine weitere wichtige Bedeutung
- Modul-Aliase stellen eine Beziehung her zwischen abstraktem Interfacenamen und der Hardware der Peripheriegeräte
  - `alias eth0 3c501`
- So lässt sich die Netzwerkkarte mit dem Alias ansprechen
  - `ifconfig eth0 192.168.0.1`

Übungen

- [12.5] Welche Modul-Aliase sind in Ihrer /etc/modules.conf-Datei definiert?
- Wie korrespondieren diese mit der tatsächlich vorhandenen Hardware?
- Tipp: Konsultieren Sie »lspci -v«