

# Betriebssysteme 2

WS 2017/18

Prof. Dr.-Ing. Hans-Georg Eßer  
Fachhochschule Südwestfalen

## Foliensatz C:

- Partitionierung
- Dateisysteme
- Zugriffsrechte

v1.0, 2016/11/11

# Partitionierung

# Datenträger (1)

Unterteilung des gesamten Speichers in einzeln adressierbare Bereiche

- physikalische Adressierung:
- Anzahl der Köpfe (Schreib-/Leseköpfe) (z. B. 20)
- Tracks, Zylinder (= Track auf jeder Pl.) (z. B. 65536 Zyl.)
- Sektoren (z. B. 63 pro Track)
- Block: 1 Sektor oder Sektorgruppe (Größe z. B. 512 Byte)
- ergibt:  $63 \times 20 \times 65536 = 82.575.360$  Sektoren,  
 $82.575.360 \times 512 \text{ Byte} = 42.278.584.320 \text{ Byte} \approx 39 \text{ GB}$

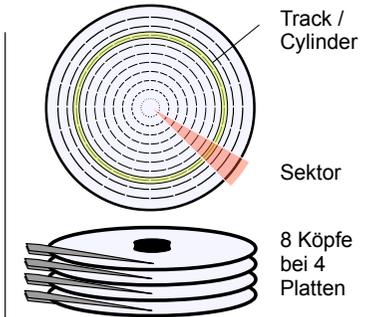


Bild: [https://commons.wikimedia.org/wiki/File:Cylinder\\_Head\\_Sector.svg](https://commons.wikimedia.org/wiki/File:Cylinder_Head_Sector.svg) (public domain)

# Datenträger (2)

- Platten-Controller setzt physikalische Adressierung in logische Adressierung um
- logisch: Zylinder oder Sektoren, durchnummeriert
- z. B. mit Zylindern:

```
# fdisk -l /dev/sda
```

```
Platte /dev/sda: 300.0 GByte, 300090728448 Byte
255 Köpfe, 63 Sektoren/Spuren, 36483 Zylinder
Einheiten = Zylinder von 16065 x 512 = 8225280 Bytes
```

Gerät	boot.	Anfang	Ende	Blöcke	Id	System
/dev/sda1	*	1	12562	100897146+	7	HPFS/NTFS
/dev/sda2		12563	35566	184779630	f	W95 Erw. (LBA)
/dev/sda3		35569	35836	2152710	c	W95 FAT32 (LBA)
/dev/sda4		35837	36483	5197027+	1c	Verst. W95 FAT32 (LBA)
/dev/sda5		12563	12691	1036161	82	Linux Swap / Solaris
/dev/sda6		12692	35566	183743406	83	Linux

## Partitionen (1)

- Festplatten werden in Partitionen unterteilt
  - traditionell („MBR-Modell“): vier Partitionen
    - Anfang, Ende, Größe: in Partitionstabelle im MBR (Master Boot Record)
    - Bezeichnung: **primäre Partitionen**
  - falls mehr nötig: eine der vier Partitionen zur **erweiterten** Partition machen
  - darin: **logische Partitionen**
  - moderne Variante: → **GPT**

## Partitionen (3)

- Festplatten
  - sda, sdb, sdc, ...: SCSI und moderne SATA
  - hda, hdb, hdc, ...: klassische IDE
- Partitionen
  - 1, 2, 3, 4: primäre Partitionen
  - 5, 6, 7, ...: logische Partitionen (dann muss mind. eine der primären Part. eine erweiterte sein)
- Zugriff über Gerätedateien:
  - sda3 → /dev/sda3

## Partitionen (2)

- Windows vergibt für jede (Windows)-Partition einen Laufwerksbuchstaben (C:, D: etc.)
  - unabhängig von Status primär/logisch
  - Reihenfolge kann wechseln
- Linux verwendet Bezeichnungen, die sich aus
  - Typ der Platte (IDE, SCSI)
  - Gerätenummer
  - Partitionsnummerzusammensetzen (sda1 = SCSI disk a, part. 1)

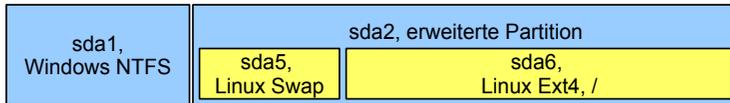
## Partitionen (4)

- Gerätedateien erzeugen moderne Linux-Versionen dynamisch:

```
esser@dissdevel:~$ ls -l /dev/sd*
brw-rw---- 1 root disk 8, 0  2. Jun 17:15 /dev/sda
brw-rw---- 1 root disk 8, 1  2. Jun 17:15 /dev/sda1
brw-rw---- 1 root disk 8, 2  2. Jun 17:15 /dev/sda2
brw-rw---- 1 root disk 8, 5  2. Jun 17:15 /dev/sda5
```
- in alten Linux-Versionen: große Mengen an passenden Gerätedateien statisch erzeugt

## Partitionen (5)

### Typische Partitionierung



- sda1: 1. primäre Partition: Windows, NTFS („Laufwerk C.“)
- sda2: erweiterte Partition, enthält logische
- sda5: 1. logische Partition: Linux, Swap
- sda6: 2. logische Partition: Linux, Ext4

## Partitionen (7)

### Partitionieren unter Linux

- fdisk: Standard-Tool
- cfdisk: „grafisches“ Tool



- sfdisk: für Skript-gesteuertes Partitionieren

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-9

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

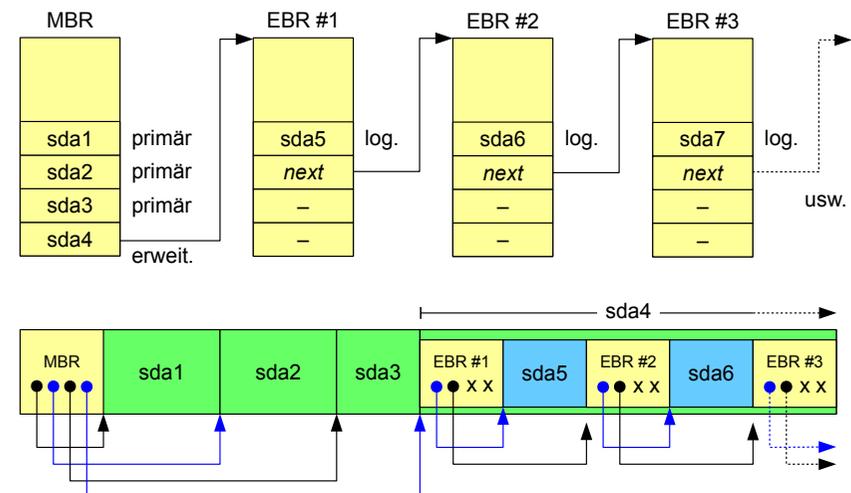
Folie C-11

## Partitionen (6)

### Arbeiten mit Gerätedateien

- head /dev/sda1  
gibt Anfang der Partition sda1 aus
- dd if=/dev/sda1 of=/tmp/image.dat  
erzeugt 1:1-Kopie der Partition sda1 in Datei, if=input file, of=output file
- fdisk /dev/sda  
bearbeitet Partitionstabelle der Festplatte sda
- mkfs.ext3 /dev/sda7  
formatiert Partition sda7 mit Ext3-Dateisystem

## Datenstrukturen im MBR & EBR



12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-10

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-12

## fdisk (1)

### Partitionsliste anzeigen

```
server:~# fdisk -l
```

```
Disk /dev/sda: 10.7 GB, 10694426624 bytes
255 heads, 63 sectors/track, 1300 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000ce798
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	1	1241	9965568	83	Linux
/dev/sda2		1241	1301	475137	5	Extended
/dev/sda5		1241	1301	475136	82	Linux swap / Solaris

### Platte partitionieren

```
server:~# fdisk /dev/sda
```

```
Command (m for help): _
```

## fdisk (3)

### Neue primäre Partition erzeugen

```
Command (m for help): n
```

```
Command action
```

```
  e  extended
```

```
  p  primary partition (1-4)
```

```
p
```

```
Partition number (1-4): 2
```

```
First cylinder (1241-1300, default 1241):
```

```
Using default value 1241
```

```
Last cylinder, +cylinders or +size{K,M,G} (1241-1300, default 1300):
```

```
Using default value 1300
```

```
Command (m for help): p
```

```
Disk /dev/sda: 10.7 GB, 10694426624 bytes
255 heads, 63 sectors/track, 1300 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000ce798
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	1	1241	9965568	83	Linux
/dev/sda2		1241	1300	475658	83	Linux

## fdisk (2): Kommandoübersicht

- **p** – zeigt die Partitionstabelle (wie in `fdisk -l /dev/sda`).
- **n** – legt eine neue Partition an; fragt Partitionstyp, Nummer der Partition und Größe ab.
- **t** – Ändert den Typ einer Partition. Nach dem Aufruf des Kommandos erhalten Sie mit dem Kommando `L` eine Übersicht über die `fdisk` bekannten Partitionstypen.
- **d** – Löscht eine Partition.
- **w** – schreibt die von Ihnen überarbeitete Partitionstabelle. Danach beendet sich `fdisk`.
- **q** – Programm beendet sich, ohne die Partitionstabelle zu ändern.
- **m** – Menü, in dem alle Befehle aufgeführt sind, nur in Englisch und noch ein paar mehr.

## fdisk (4)

### Neue erweiterte Partition erzeugen

```
Command (m for help): n
```

```
Command action
```

```
  e  extended
```

```
  p  primary partition (1-4)
```

```
e
```

```
Partition number (1-4): 2
```

```
First cylinder (1241-1300, default 1241):
```

```
Using default value 1241
```

```
Last cylinder, +cylinders or +size{K,M,G} (1241-1300, default 1300):
```

```
Using default value 1300
```

```
Command (m for help): p
```

```
Disk /dev/sda: 10.7 GB, 10694426624 bytes
255 heads, 63 sectors/track, 1300 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000ce798
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	1	1241	9965568	83	Linux
/dev/sda2		1241	1300	475658	5	Extended

## fdisk (5)

### Neue logische Partition erzeugen

```
Command (m for help): n
Command action
  l   logical (5 or over)
  p   primary partition (1-4)
1
First cylinder (1241-1300, default 1241):
Using default value 1241
Last cylinder, +cylinders or +size{K,M,G} (1241-1300, default 1300): 1260

Command (m for help): p

Disk /dev/sda: 10.7 GB, 10694426624 bytes
255 heads, 63 sectors/track, 1300 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000ce798

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1 *          1         1241     9965568   83  Linux
/dev/sda2            1241       1300     475658    5  Extended
/dev/sda5            1241       1260     154326+   83  Linux
```

Auswahl geändert, weil es  
jetzt eine erweiterte  
Partition gibt! **!**

## GPT (1)

- MBR-Partitionsschema wird abgelöst durch **GPT** (GUID Partition Table) (GUID: Globally Unique Identifier)
- Teil von **UEFI** (BIOS-Nachfolger)
- nur ein Partitionstyp
- beliebig viele Einträge
- kann sehr große Platten adressieren (Sektornummern in 64 Bit kodiert)

## fdisk (6)

```
Command (m for help): t
Partition number (1-5): 1
Hex code (type L to list codes): L
```

### Partitionstypen

0	Empty	24	NEC DOS	81	Minix / old Lin	bf	Solaris
1	FAT12	39	Plan 9	82	Linux swap / So	c1	DRDOS/sec (FAT-
2	XENIX root	3c	PartitionMagic	83	Linux	c4	DRDOS/sec (FAT-
3	XENIX usr	40	Venix 80286	84	OS/2 hidden C:	c6	DRDOS/sec (FAT-
4	FAT16 <32M	41	PPC PreP Boot	85	Linux extended	c7	Syrinx
5	Extended	42	SFS	86	NTFS volume set	da	Non-FS data
6	FAT16	4d	QNX4.x	87	NTFS volume set	db	CP/M / CTOS / .
7	HPFS/NTFS	4e	QNX4.x 2nd part	88	Linux plaintext	de	Dell Utility
8	AIX	4f	QNX4.x 3rd part	8e	Linux LVM	df	BootIt
9	AIX bootable	50	OnTrack DM	93	Amoeba	e1	DOS access
a	OS/2 Boot Manag	51	OnTrack DM6 Aux	94	Amoeba BBT	e3	DOS R/O
b	W95 FAT32	52	CP/M	9f	BSD/OS	e4	SpeedStor
c	W95 FAT32 (LBA)	53	OnTrack DM6 Aux	a0	IBM Thinkpad hi	eb	BeOS fs
e	W95 FAT16 (LBA)	54	OnTrackDM6	a5	FreeBSD	ee	GPT
f	W95 Ext'd (LBA)	55	EZ-Drive	a6	OpenBSD	ef	EFI (FAT-12/16/
10	OPUS	56	Golden Bow	a7	NeXTSTEP	f0	Linux/PA-RISC b
11	Hidden FAT12	5c	Priam Edisk	a8	Darwin UFS	f1	SpeedStor
12	Compaq diagnost	61	SpeedStor	a9	NetBSD	f2	SpeedStor
14	Hidden FAT16 <3	63	GNU HURD or Sys	ab	Darwin boot	f2	DOS secondary
16	Hidden FAT16	64	Novell Netware	af	HFS / HFS+	fb	VMware VMFS
17	Hidden HPFS/NTF	65	Novell Netware	b7	BSDI fs	fc	VMware VMKCORE
18	AST SmartSleep	70	DiskSecure Mult	b8	BSDI swap	fd	Linux raid auto
1b	Hidden W95 FAT3	75	PC/IX	bb	Boot Wizard hid	fe	LANstep
1c	Hidden W95 FAT3	80	Old Minix	be	Solaris boot	ff	BBT
1e	Hidden W95 FAT1						

## GPT (2)

- Linux: fdisk / gdisk
  - Ältere Linux-Versionen: fdisk unterstützt nur MBR-Schema; Alternativ-Tool gdisk
  - Neuere Linux-Versionen: fdisk beherrscht GPT
  - Partitionsnamen: sda1, sda2 usw.
- Windows
  - seit Vista Support für GPT
  - ab Windows 8: Standard
- OS X setzt seit dem Intel-Umstieg auf GPT

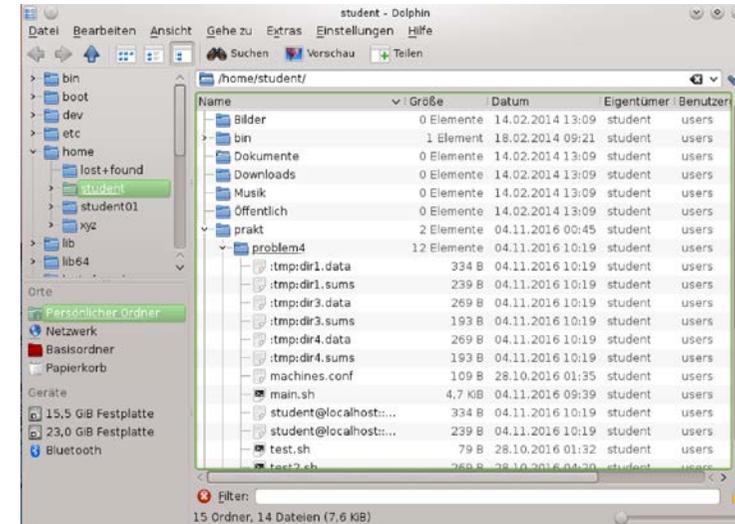
## GPT (3)

```
linux:~/# fdisk -l /dev/sdb
```

```
Disk /dev/sdb: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: gpt
```

#	Start	End	Size	Type	Name
1	2048	12584959	6G	Linux filesystem	
2	12584960	23070719	5G	Linux filesystem	
3	23070720	33556479	5G	Linux filesystem	
4	33556480	35653631	1G	Linux filesystem	
5	35653632	37750783	1G	Linux filesystem	
6	37750784	41943006	2G	Linux swap	

## Ohne Dateien ...



12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-21

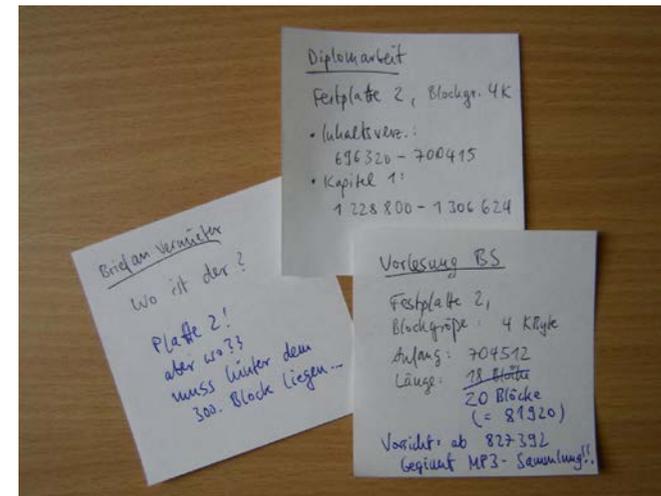
12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-23

## Dateisysteme

## ... ist die Arbeit schwer!



12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-22

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-24

## Aufgaben eines Dateisystems (1)

Abstraktion des Speicherplatzes auf (meist nicht-flüchtigen) Datenträgern:

- Festplatten
- Disketten
- CD, DVD, ...
- sonstige (wieder-) beschreibbare Medien (USB-Stick, ZIP-Disketten, Compact Flash, ...)
- Bandlaufwerke (?)
- Es gibt sehr kleine (178 KByte) und sehr große (200 GByte) Datenträger

## Aufgaben eines Dateisystems (3)

Zweites Konzept: **Verzeichnis**

- Dateisammlung strukturieren
- Verzeichnisse enthalten Dateien und ggf. Unterverzeichnisse

Verwaltung des freien Speichers auf dem Datenträger

- zusammenhängende Verwaltung (contiguous)
- nicht-zusammenhängend (non-contiguous) → Fragmentierung

## Aufgaben eines Dateisystems (2)

Zentrales Konzept: **Datei**

- Datei erzeugen
- Datei auf Datenträger finden (Inhaltsverzeichnis)
- Datei öffnen / schließen
- Datei sequentiell lesen (vom Anfang bis zum Ende)
- Datei im Direktzugriff lesen (seek & read)
- Datei (um-) benennen

## Aufgaben eines Dateisystems (4)

Schutz vor **Datenverlust**

- redundantes Speichern wichtiger Verwaltungs-informationen
- **Journaling**
  - Datei-„Transaktionen“
  - System legt vor Beginn einer Transaktionen einen Journaleintrag an
  - Bei Systemausfall können (beim nächsten Systemstart) partielle Transaktionen rückgängig gemacht werden
- (parallelen) Zugriff mehrerer Anwender ermöglichen

## Aufgaben eines Dateisystems (5)

- **Zugriffsschutz** (z. B.: lesen, schreiben, anhängen, ausführen, finden, Rechte ändern) für Dateien
  - auf Basis von Benutzern / Gruppen (Unix / Linux)
  - auf Basis von Benutzern / Access Control Lists (ACLs) (Windows)
- **Zugriffs-Scheduler:** In welcher Reihenfolge Zugriffe ausführen (Optimierung der Plattenzugriffe)
- **Caching:** Aus Performance-Gründen gelesene Daten in Cache (Hauptspeicher) vorrätig halten und Änderungen verzögert zurück schreiben

## Klassische Dateisysteme: CP/M (1)

### CP/M-Dateisystem

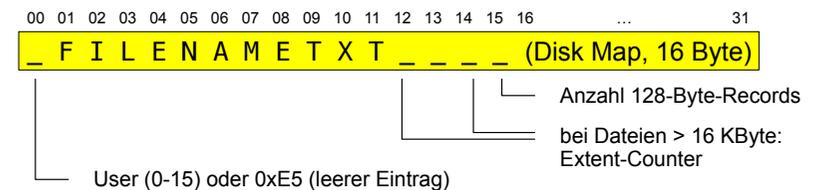
- Flache Struktur (keine Verzeichnisse)
- Dateien: 8+3-Konvention (Name + Extension)
  - Programme: \*.COM
  - Textdateien: \*.TXT, etc.
- Etwas Struktur durch „User“ (16 mögliche Dateibesitzer)
- Blockgröße: 1 KByte

## Abstraktionsebenen

- Klassisch: eine Ebene (Dateisystemtreiber)
  - MS-DOS, CP/M, altes Mac OS
  - Systemfunktionen für Zugriff auf Medien, die mit dem Dateisystem des BS formatiert sind
- Modern: zwei Ebenen
  - virtuelles Dateisystem
  - Systemfunktionen zum Zugriff auf beliebige Medien
  - Treiber für spezielle Dateisysteme
    - Linux: ext3, ext4, reiserfs, iso9660, vfat, ntfs, udf, ...
    - Windows: NTFS, FAT (-12, -16, -32), exFAT

## Klassische Dateisysteme: CP/M (2)

- Am Anfang eines Datenträgers: Inhaltsverzeichnis (für 64 Dateien)
- Für jeden Eintrag 32 Byte reservieren



- obere Bits der Erweiterungen enthalten drei Attribute (read-only, hidden, archived)
- Dateigröße >16 KByte → mehrere Einträge (Extents)

## Klassische Dateisysteme: CP/M (3)

- Alle Betriebssystem-Funktionen zum Schreiben / Lesen etc. müssen Aufbau des Dateisystems kennen
- Umstieg auf neuere Version des Dateisystems (mit veränderten Verwaltungsstrukturen)

→ Fallunterscheidungen in Treiberfunktionen:

```
void create_file (char* name) {
    switch (fs_version) {
        "1.4": create_file_type1 (name);
        "2.2": create_file_type2 (name);
        ....
    }
}
```

Mehr Informationen zu CP/M:  
<http://www.seasip.info/Cpm/>

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-33

## Klassische Dateisysteme: FAT (2)

- Keine Liste aller benutzten Blöcke (wie CP/M)
- Zusätzlich zu Verzeichniseinträgen File Allocation Table (FAT)
- Verzeichniseintrag enthält „First Block“-Feld, das als Index in die FAT dient
- Jeder FAT-Eintrag
  - entspricht einem Block auf dem Datenträger;
  - enthält die Nummer des nächsten FAT-Eintrags (verkettete Liste) oder -1 (letzter Block)

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-35

## Klassische Dateisysteme: FAT (1)

Jeder Verzeichniseintrag ist 32 Byte groß:

```
00 01 02 03 04 05 06 07 08 09 10 11 12 ... 21 22 23 24 25 26 27 28 29 30 31
F I L E N A M E T X T _ _ _ _ _ _ _ _ _ _
```

- 0-10 Dateiname (8 Byte) und Dateierweiterung (3 Byte)
- 11 Datei-Attribute:
  - Bit 0: read-only (**R**), Bit 1: hidden(**H**), Bit 2: system file (**S**),
  - Bit 3: volume label, Bit 4: Verzeichnis, Bit 5: archiviert (**A**),
  - Bits 6-7: ungenutzt – attrib +RHS c:\io.sys
- 12-21 reserviert
- 22-23 Zeit: Stunde (5 Bit), Minute (6 Bit), Doppel-Sekunde (5 Bit)
- 24-25 Datum: Jahr seit 1980 (7 Bit), Monat (4 Bit), Tag (5 Bit)
- 26-27 Erster Block (0 = leere Datei)
- 28-31 Dateigröße in Byte (32 Bit: theoretisch bis zu 4 GByte)

## Klassische Dateisysteme: FAT (3)

Kleines Experiment mit FAT:

```
$ mkdosfs -F 12 /tmp/fatfs.img
$ mount -o loop -t msdos /tmp/fatfs.img /dos
$ cp /tmp/readme.txt /dos/
$ sync
$ cp /dos/readme.txt /dos/kopie.txt
$ sync
$ cp /dos/readme.txt /dos/geloescht.txt
$ sync
$ rm /dos/geloescht.txt
$ sync
$ umount /dos
$ hexdump /tmp/fatfs.img
```

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-34

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-36

## Klassische Dateisysteme: FAT (4)

```

00000000 eb 3c 90 6d 6b 64 6f 73 66 73 00 00 02 01 01 00 |ë<.mkdosfs.....|
                03-0A: OEM Identifier
00000010 02 e0 00 40 0b f0 09 00 12 00 02 00 00 00 00 00 |.à.ê.ð.....|
00000020 00 00 00 00 00 00 29 c6 52 68 45 20 20 20 20 20 |.....)ÆRhE|
00000030 20 20 20 20 20 20 46 41 54 31 32 20 20 20 0e 1f | FAT12 ..|
                36-3D: FAT-Typ
00000040 be 5b 7c ac 22 c0 74 0b 56 b4 0e bb 07 00 cd 10 |ÿ[|-"Ät.VŽ.»..í.|
00000050 5e eb f0 32 e4 cd 16 cd 19 eb fe 54 68 69 73 20 |^èð2äï.í.ëþThis|
00000060 69 73 20 6e 6f 74 20 61 20 62 6f 6f 74 61 62 6c |is not a bootabl|
00000070 65 20 64 69 73 6b 2e 20 20 50 6c 65 61 73 65 20 |e disk. Please|
00000080 69 6e 73 65 72 74 20 61 20 62 6f 6f 74 61 62 6c |insert a bootabl|
00000090 65 20 66 6c 6f 70 70 79 20 61 6e 64 0d 0a 70 72 |e floppy and..pr|
000000a0 65 73 73 20 61 6e 79 20 6b 65 79 20 74 6f 20 74 |ess any key to t|
000000b0 72 79 20 61 67 61 69 6e 20 2e 2e 2e 20 0d 0a 00 |ry again ... ..|
000000c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
000001f0 00 00 00 00 00 00 00 00 00 00 00 00 55 aa |.....Uª|
/* Anfang der FAT im 2. Cluster (ab 512d = 0200x) */
00000200 f0 ff ff ff ff ff 00 00 00 00 00 00 00 00 00 00 |ðÿÿÿÿ.....|
00000210 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00001400 f0 ff ff ff ff ff 00 00 00 00 00 00 00 00 00 00 |ðÿÿÿÿ.....|
00001410 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|

```

## Probleme klass. Dateisysteme

- Systemfunktionen (öffnen, lesen, schreiben etc.) auf ein spezielles Dateisystem zugeschnitten
- Support für fremde / neue Dateisysteme schwierig; oft ganz unmöglich
- Software (von Drittanbietern) für Zugriff auf fremde Dateisysteme schlecht integriert
- Lösung: **Virtuelles Dateisystem**

## Klassische Dateisysteme: FAT (5)

```

00002600 52 45 41 44 4d 45 20 20 54 58 54 20 00 00 00 00 |README TXT ....|
00002610 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....o.y5..²...|
00002620 4b 4f 50 49 45 20 20 20 54 58 54 20 00 00 00 00 |KÖPIE TXT ....|
00002630 00 00 00 00 00 00 83 83 79 35 03 00 b2 00 00 00 |.....y5..²...|
00002640 e5 45 4c 4f 45 53 43 48 54 58 54 20 00 00 00 00 |ËELOESCHTXT ...|
00002650 00 00 00 00 00 00 91 83 79 35 04 00 b2 00 00 00 |.....y5..²...|
00002660 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00004200 48 61 6c 6c 6f 2c 20 69 63 68 20 62 69 6e 20 6e |Hallo, ich bin n|
00004210 75 72 20 65 69 6e 65 20 6b 6c 65 69 6e 65 0a 54 |ur eine kleine.T|
00004220 65 73 74 64 61 74 65 69 20 66 75 65 72 20 64 69 |estdatei fuer di|
00004230 65 20 56 6f 72 6c 65 73 75 6e 67 0a 42 65 74 72 |e Vorlesung.Betr|
00004240 69 65 62 73 73 79 73 74 65 6d 65 2e 20 49 63 68 |iebssysteme. Ich|
00004250 20 6c 69 65 67 65 20 61 75 66 0a 65 69 6e 65 72 |liege auf.einer|
00004260 20 28 76 69 72 74 75 65 6c 6c 65 6e 29 20 44 4f |(virtuellen) D0|
00004270 53 2d 44 69 73 6b 65 74 74 65 2c 0a 61 6c 73 6f |S-Diskette,also|
00004280 20 61 75 66 20 65 69 6e 65 6d 20 46 41 54 2d 31 |auf einem FAT-1|
00004290 32 2d 44 61 74 65 69 73 79 73 74 65 6d 2e 0a 0a |2-Dateisystem...|
000042a0 56 69 65 6c 20 53 70 61 73 73 20 6e 6f 63 68 21 |Viel Spass noch!|
000042b0 0a 0a 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
000042c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00004400 48 61 6c 6c 6f 2c 20 69 63 68 20 62 69 6e 20 6e |Hallo, ich bin n|
00004410 75 72 20 65 69 6e 65 20 6b 6c 65 69 6e 65 0a 54 |ur eine kleine.T|

```

## Virtuelles Dateisystem – VFS (1)

- Zwei Schichten einführen
- VFS-Treiber stellt High-Level-Dateioperationen bereit (create, delete, rename, open, close, read, write, seek, link, ...)
- Kommunikation aller Programme (und auch des BS selbst) nur mit dem VFS-Treiber
- VFS-Treiber leitet Anfragen an Spezialtreiber für die Dateisysteme weiter
- Spezialtreiber beherrschen einzelne FS

## VFS (2)

### VFS-Treiber

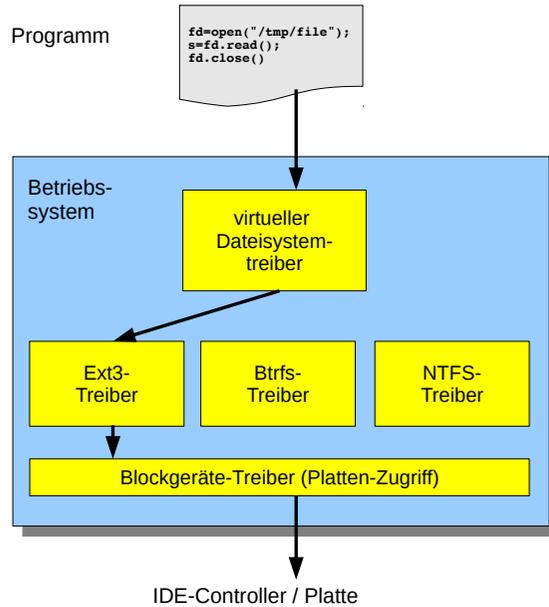
- kennt abstrakte Dateieigenschaften
- weiß, welchen speziellen Dateisystem-Treiber er braucht

### Dateisystem-Treiber

- kennt das jeweilige Dateisystem-Format (weiß nichts von HW)

### Blockgeräte-Treiber

- kann mit der Hardware sprechen



## VFS (4): Standard-Funktionen

- |                           |   |
|---------------------------|---|
| <code>seek ();</code>     | in Datei an bestimmte Stelle springen   |
| <code>get_attr ();</code> | Datei-Eigenschaften abfragen (was auch immer das BS hier für Eigenschaften zulässt) |
| <code>set_attr ();</code> | Datei-Eigenschaften setzen  |
| <code>rename ();</code>   | Dateinamen ändern   |

## VFS (3): Standard-Funktionen

In jedem Betriebssystem unterstützt das VFS mindestens

- |                         |                                   |
|-------------------------|-----------------------------------|
| <code>create ();</code> | Datei erzeugen                    |
| <code>delete ();</code> | Datei löschen                     |
| <code>open ();</code>   | Datei öffnen                      |
| <code>close ();</code>  | offene Datei schließen            |
| <code>read ();</code>   | aus Datei lesen                   |
| <code>write ();</code>  | in Datei schreiben                |
| <code>append ();</code> | ans Ende der Datei etwas anhängen |

## Formatieren (1)

- Einfaches Anlegen einer neuen Partition macht diese noch nicht benutzbar
- Partition muss man vor erster Nutzung **formatieren** (= mit einem **Dateisystem** versehen)
- Kommando allgemein: `mkfs` (make filesystem)
  - `mkfs -t TYP /dev/GERÄT`
  - ruft spezialisiertes Tool `mkfs.TYP` (z. B. `mkfs.ext3`) auf

```
root@dissdevel:/# ls /sbin/mkfs*
/sbin/mkfs      /sbin/mkfs.ext2  /sbin/mkfs.ext4dev /sbin/mkfs.ntfs
/sbin/mkfs.bfs  /sbin/mkfs.ext3  /sbin/mkfs.minix   /sbin/mkfs.vfat
/sbin/mkfs.cramfs /sbin/mkfs.ext4  /sbin/mkfs.msdos
```

## Formatieren (2)

```
root@dissdevel:/# mkfs -t ext3 /dev/sda8
mke2fs 1.41.12 (17-May-2010)
Dateisystem-Label=
OS-Typ: Linux
Blockgröße=1024 (log=0)
Fragmentgröße=1024 (log=0)
Stride=0 Blöcke, Stripebreite=0 Blöcke
2560 Inodes, 10240 Blöcke
512 Blöcke (5.00%) reserviert für den Superuser
Erster Datenblock=1
Maximale Dateisystem-Blöcke=10485760
2 Blockgruppen
8192 Blöcke pro Gruppe, 8192 Fragmente pro Gruppe
1280 Inodes pro Gruppe
Superblock-Sicherungskopien gespeichert in den Blöcken:
  8193

Schreibe Inode-Tabellen: erledigt
Erstelle Journal (1024 Blöcke): erledigt
Schreibe Superblöcke und Dateisystem-Accountinginformationen: erledigt

Das Dateisystem wird automatisch nach jeweils 30 Einhäng-Vorgängen bzw.
alle 180 Tage überprüft, je nachdem, was zuerst eintritt. Dies kann durch
tune2fs -c oder -i geändert werden.
```

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-45

## Formatieren (3)

- Auch **Swap-Partition** (Bereich, der für das Auslagern von Speicherseiten verwendet wird; → **Paging**) muss formatiert werden
- Tool heißt **mkswap**:

```
root@dissdevel:/# mkswap /dev/sda5
Setting up swap space version 1, size = 475132 KiB
no label, UUID=5c43f2b7-8801-4fde-94a2-f154ffbabb42
```
- Swap-Bereich darf auch Datei sein  
→ hilfreich, wenn keine Swap-Partition angelegt werden kann

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-46

## Mounten (1)

- Linux bindet beim Systemstart nicht automatisch alle Dateisysteme (meist: Partitionen) ein, sondern tut dies nur für eine Auswahl, die durch Einträge in einer Konfigurationsdatei festgelegt wird. Ausnahme: Root-Dateisystem /, ohne das kein Systemstart möglich ist.
- Den Einbindevorgang nennt Linux (wie alle Unix-Systeme) **mounten**, die umgekehrte Operation, bei der das System nicht länger auf einen Datenträger zugreift, heißt **unmounten**.
- Die dafür zuständigen Kommandos heißen **mount** und **umount** (nicht **unmount**!).
- Automatisches Mounten über Einträge in `/etc/fstab`

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-47

## Mounten (2)

- Das Mounten stellt eine Verknüpfung zwischen einem Datenträger und einem Verzeichnis her, unter dem dann die Inhalte des Datenträgers erreichbar sind
- Diese Verzeichnisse (**Mount-Points**) sind das Gegenstück zu Windows-Laufwerksbuchstaben
- Linux- (Unix-) Ansatz ist flexibler

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-48

## Mounten (3)

### Datenträger unter Windows und Linux

Linux-Partition: nicht sichtbar	[Root-Dateisystem /dev/sda6 auf /] /home /usr /etc /var ...
C: [Win] C:\Windows C:\Windows\System C:\Users C:\Users\Esser C:\Users\Esser\Documents	[/dev/sda1 auf /mnt/win1] /mnt/win1/Windows /mnt/win1/Windows/System /mnt/win1/Users /mnt/win1/Users/Esser /mnt/win1/Users/Esser/Documents
D: [Restore] D:\Restore.Tmp	[/dev/sda2 auf /mnt/win2] /mnt/win2/Restore.Tmp
E: [OfficeDVD] E:\Files	[Office-DVD auf /media/OfficeDVD] /media/OfficeDVD/Files

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-49

## Mounten (5)

### • Was braucht man fürs Mounten?

- Gerätedatei des Datenträgers (Partition o. ä.)
- Mount-Point (Verzeichnis, muss schon existieren)
- evtl. Typ des Dateisystems
- evtl. Optionen fürs Mounten

mount

-t *TYP* -o *OPTIONS*  
/dev/*PARTITION* /*MOUNTPOINT*

mount -t ext3 -o ro /dev/sda7 /mnt

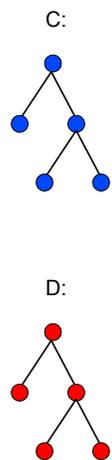
12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

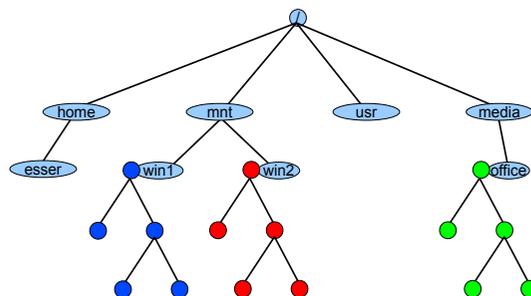
Folie C-51

## Mounten (4)

### Windows



### Linux



12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-50

## Mounten (6)

### • Dateisystemtyp (-t *TYP*), Auswahl

- ext4: 4<sup>th</sup> extended filesystem (Linux, aktuell)
- ext3: 3<sup>rd</sup> extended filesystem (Linux, älter)
- ext2: 2<sup>nd</sup> extended filesystem (Linux, veraltet)
- btrfs: B-tree filesystem
- ntfs: New Technology Filesystem (Windows)
- vfat: Virtual File Allocation Table (DOS, Windows)
- iso9660: CD-/DVD-Dateisystem
- udf: DVD-Dateisystem (z. B. Video-DVD)

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-52

## Mounten (7)

- Dateisystemtyp (`-t TYP`)
  - Liste tatsächlich noch viel länger
  - Welche Dateisysteme unterstützt das System?

```
essser@essser-lm18kde ~/~/tex $ uname -a ; grep -v nodev /proc/filesystems
| pr -3 -t
Linux essser-lm18kde 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:33:3
7 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux
ext3          ext4          fuseblk
ext2          vfat          btrfs
essser@essser-lm18kde ~/~/tex $ ls /lib/modules/$(uname -r)/kernel/fs
9p             cachefiles   fat           jfs           ocfs2         sysv
adfs           ceph         freevxfs     lockd         omfs          ubifs
affs           cifs         fscache      minix         overlayfs    udf
afs            coda         fuse          ncpfs         pstore        ufs
aufs           configfs     gfs2         nfs           qnx4          xfs
autofs4        cramfs       hfs          nfs_common    qnx6
befs           dlm          hfsplus      nfsd          quota
bfs            efs          hpfs         nilfs2        reiserfs
binfmt_misc.ko exofs        isofs        nls           romfs
btrfs          f2fs         jffs2        ntfs          squashfs
```

## Mounten (9)

- Swap-Partitionen werden nicht gemountet, sondern aktiviert (`swapon`) oder deaktiviert (`swapoff`)

```
root@dissdevel:/# swapon -v /dev/sda5
swapon on /dev/sda5
swapon: /dev/sda5: found swap signature: version 1, page-size 4,
same byte order
swapon: /dev/sda5: pagesize=4096, swappiness=486539264,
devsize=486539264

root@dissdevel:/# swapoff -v /dev/sda5
swapoff on /dev/sda5
```

- (ohne Option `-v` keine Ausgabe)
- Swap darf auch eine Datei sein

## Mounten (8)

- Mount-Optionen (`-o OPTIONEN`); Auswahl:
  - `ro`: read-only (nur lesen)
  - `rw`: read-write (lesen und schreiben; Standard)
  - `async`, `sync`: alle Zugriffe asynchron bzw. synchron (sofort schreiben, kein Puffer) ausführen
  - `noatime`: Zugriffe auf Dateien nicht in Metadaten speichern (u. a. für Flash-Datenträger sinnvoll)
  - `nodiratime`: wie `noatime`, für Verzeichnisse
  - `noexec`: Programme sind nicht ausführbar
  - `remount`: bereits gemountetes FS nochmal mounten
  - `loop`: Dateisystem-Image mounten

## Mounten (10)

- Übersicht über aktive Swap-Bereiche

```
root@dissdevel:/# cat /proc/swaps
Filename      Type      Size      Used      Priority
/dev/sda5     partition 475128    0         -1
/tmp/swapfile file       10232     0         -2
```

## Unmounten (1)

- Dateisystem wieder aushängen (unmounten)
  - Kommando `umount`
  - Argument: Wahlweise Name der Gerätedatei (`/dev/...`) oder Mount-Point
- Beispiele:  
`umount /dev/sda6` (Gerätedatei)  
`umount /mnt/win1` (Mount-Point)

## Mounten mit `/etc/fstab` (1)

- Konfigurationsdatei `/etc/fstab` (filesystem table) legt fest, welche FS beim Systemstart eingebunden werden

- Aufbau einer Zeile der Datei:

```
# <fs>      <mount point>  <type>      <options>      <dump>  <pass>
```

- Beispiel:

```
proc        /proc          proc         defaults        0          0
/dev/sda1   /              ext3         errors=remount-ro 0          1
/dev/sda5   none           swap         sw              0          0
/dev/scd0   /media/cdrom0  udf,iso9660 user,noauto     0          0
```

## Unmounten (2)

- Kommando `umount` schlägt manchmal fehl:

```
root@dissdevel:/mnt/tmp# pwd
/mnt/tmp
root@dissdevel:/mnt/tmp# umount /mnt
umount: /mnt: device is busy.
(In some cases useful info about processes that use
the device is found by lsof(8) or fuser(1))
root@dissdevel:/mnt/tmp# cd /
root@dissdevel:/# umount /mnt/
root@dissdevel:/# _
```

- Es darf keine Datei im FS geöffnet sein
- Es darf keine Shell (oder ein anderes Programm) das aktuelle Arbeitsverzeichnis in diesem FS haben

## Mounten mit `/etc/fstab` (2)

- Einige Einträge haben im Optionenfeld die Option `noauto`
- Solche Einträge werden nicht automatisch gemountet, können aber einfacher von Hand gemountet werden

```
root@server:~# grep scd0 /etc/fstab
/dev/scd0 /media/cdrom udf,iso9660 user,noauto 0 0
root@server:~# mount /media/cdrom
```

- Zusatzoption `user` bedeutet: Mounten auch ohne Root-Rechte möglich

## Mounten mit /etc/fstab (3)

- Neben /etc/fstab gibt es noch eine Datei /etc/mtab (**mount table**)
- Diese enthält Informationen über gemountete Dateisysteme und wird automatisch (vom System) erstellt und aktualisiert

```
root@dissdevel:/# cat /etc/mtab
/dev/sda1 / ext3 rw,errors=remount-ro 0 0
tmpfs /lib/init/rw tmpfs rw,nosuid,mode=0755 0 0
proc /proc proc rw,noexec,nosuid,nodev 0 0
sysfs /sys sysfs rw,noexec,nosuid,nodev 0 0
udev /dev tmpfs rw,mode=0755 0 0
tmpfs /dev/shm tmpfs rw,nosuid,nodev 0 0
devpts /dev/pts devpts rw,noexec,nosuid,gid=5,mode=620 0 0
fusectl /sys/fs/fuse/connections fusectl rw 0 0
Daten /media/sf_Daten vboxsf gid=1001,rw 0 0
```

## Filesystem Check (2)

- Automatische Überprüfung beim Systemstart:

```
Activating swap...done.
Checking root file system...fsck from util-linux-ng 2.17.2
/dev/sda1 contains a file system with errors, check forced.
/dev/sda1: 187822/623392 files (0.8% non-contiguous), 1128272/2491392 blocks
done.
Loading kernel modules...done.
Cleaning up ifupdown...
Setting up networking...
Activating lvm and md swap...done.
Checking file systems...fsck from util-linux-ng 2.17.2
done.
Mounting local filesystems...done.
```

- Welche Dateisysteme überprüft werden, legt letzte Spalte in /etc/fstab fest: 1 = prüfen

proc	/proc	proc	defaults	0	0
/dev/sda1	/	ext3	errors=remount-ro	0	1
/dev/sda5	none	swap	sw	0	0
/dev/scd0	/media/cdrom0	udf,iso9660	user,noauto	0	0

## Filesystem Check (1)

- Dateisysteme werden i. d. R. beim Systemstart auf Konsistenz überprüft (filesystem check)
- Auf Wunsch auch manuelle Überprüfung möglich
- Dateisystem darf dabei nicht gemountet sein
- Generisches Tool: `fsck` (filesystem check)

```
root@dissdevel:/# fsck /dev/sda1
fsck from util-linux-ng 2.17.2
e2fsck 1.41.12 (17-May-2010)
/dev/sda1 ist eingehängt.
```

WARNUNG!!! Die Benutzung von e2fsck auf einem eingehängten Dateisystem führt zu SCHWERWIEGENDEN SCHÄDEN im Dateisystem.

Wirklich fortfahren (j/n)?

## Filesystem Check (3)

- Statt `fsck` besser direkt das für das Dateisystem passende Tool (`fsck.TYP`) aufrufen  
→ dann sind auch individuelle Optionen möglich
- Beispiel `fsck.ext3`, Optionen:
  - `-f` : force, auch als „clean“ erkanntes FS prüfen
  - `-p` : versuche, Fehler automatisch zu beheben
  - `-y` : alle Fragen, die `fsck.ext3` stellt, automatisch mit „y“ (yes) beantworten
  - `-c` : Programm `badblocks` aufrufen (findet defekte Blöcke und trägt diese in Bad Blocks List ein)

## Filesystem Check (4)

- Beispiel `fsck` auf Ext3-Dateisystem

```
root@dissdevel:/# fsck /dev/sda8
fsck from util-linux-ng 2.17.2
e2fsck 1.41.12 (17-May-2010)
/dev/sda8: sauber, 11/65536 Dateien, 12644/262144 Blöcke
```

(jetzt mit `-f` erzwingen)

```
root@dissdevel:/# fsck -f /dev/sda8
fsck from util-linux-ng 2.17.2
e2fsck 1.41.12 (17-May-2010)
Durchgang 1: Prüfe Inodes, Blocks, und Größen
Durchgang 2: Prüfe Verzeichnis Struktur
Durchgang 3: Prüfe Verzeichnis Verknüpfungen
Durchgang 4: Überprüfe die Referenzzähler
Durchgang 5: Überprüfe Gruppe Zusammenfassung
/dev/sda8: 11/65536 Dateien (0.0% nicht zusammenhängend), 12644/262144
Blöcke
```

## Alternativnamen mkfs, fsck

- Die FS-spezifischen `mkfs`- und `fsck`-Tools sind meist noch unter anderen (kürzeren) Namen erreichbar:
  - `mkfs.ext3` = `mke2fs`    `fsck.ext3` = `e2fsck`
  - `mkfs.ext4` = `mke2fs`    `fsck.ext4` = `e2fsck`
  - `mkfs.vfat` = `mkdosfs`    `fsck.vfat` = `dosfsck`
  - `mkfs.msdos` = `mkdosfs`    `fsck.msdos` = `dosfsck`
- Aber: dann bei `mk*fs` aufpassen, welches das Standard-FS ist (`mke2fs`: Ext2, also nicht sinnvoll...)
- `mkfs` ohne `-t`: auch Ext2
- `vfat` und `msdos` sind identische FS

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-65

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-67

## Filesystem Check (5)

- Beispiel `fsck.ext3` – mit Fehlern

```
root@dissdevel:/home/esser# fsck.ext3 -f /dev/sda8
e2fsck 1.41.12 (17-May-2010)
Durchgang 1: Prüfe Inodes, Blocks, und Größen
Durchgang 2: Prüfe Verzeichnis Struktur
Eintrag »..« in ??? (41972) hat gelöscht/unbenutzt Inode 19152.  Bereinige<j>? ja
Eintrag »..« in ??? (42004) hat gelöscht/unbenutzt Inode 19167.  Bereinige<j>? ja
Eintrag »..« in ??? (42006) hat gelöscht/unbenutzt Inode 19167.  Bereinige<j>? ja
Durchgang 3: Prüfe Verzeichnis Verknüpfungen
Durchgang 4: Überprüfe die Referenzzähler
Durchgang 5: Überprüfe Gruppe Zusammenfassung

Die Anzahl freier Inodes ist falsch (59759, gezählt=58271).
Repariere<j>? ja

/dev/sda8: ***** DATEISYSTEM WURDE VERÄNDERT *****
/dev/sda8: 7265/65536 Dateien (0.0% nicht zusammenhängend), 44392/262144 Blöcke
```

## FS-Informationen, du/df (1)

- Speicherplatz-Verbrauch
  - `df` (**d**isk **f**ree) zeigt freien Platz auf einem Datenträger (oder auf allen) an
  - `du` (**d**isk **u**sage) zeigt verwendeten Platz in einem Verzeichnis an
  - für beide Tools: mit Optionen die Ausgabe anpassen

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-66

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-68

## FS-Informationen, du/df (2)

- df

```
root@dissdevel:~/tmp# df
Dateisystem 1K-Blöcke Benutzt Verfügbar Ben% Eingehängt auf
/dev/sda5 9809032 6446020 2864736 70% /
/dev/sda1 118022124 87966344 30055780 75% /mnt/win
tmpfs 517240 0 517240 0% /lib/init/rw
udev 512884 176 512708 1% /dev
tmpfs 517240 0 517240 0% /dev/shm
```

-h = „human-readable“

```
root@dissdevel:~/tmp# df -h
Dateisystem Size Used Avail Use% Eingehängt auf
/dev/sda5 9,4G 6,2G 2,8G 70% /
/dev/sda1 113G 84G 29G 75% /mnt/win
tmpfs 506M 0 506M 0% /lib/init/rw
udev 501M 176K 501M 1% /dev
tmpfs 506M 0 506M 0% /dev/shm
```

```
root@dissdevel:~/tmp# df -h /
Dateisystem Size Used Avail Use% Eingehängt auf
/dev/sda1 9,4G 6,2G 2,8G 70% /
```

## FS-Informationen, du/df (4)

- du -s \* | sort -n

```
esser@dissdevel:~/Daten$ ls -ld *
Anstel Buecher Erlangen SWF FU-Hagen Heise HM LNM privat
Promotion
```

```
esser@dissdevel:~/Daten$ du -sm * | sort -n
```

```
1 privat
3 Heise
6 Anstel
6 Buecher
9 Erlangen
15 HM
60 FU-Hagen
61 LNM
147 SWF
1715 Promotion
```

sort -n = numerisch sortieren

## FS-Informationen, du/df (3)

- du

```
esser@dissdevel:~/Daten/SWF$ du
80 ./Briefe
7300 ./BS-Alt
60324 ./BS-Praxis
20184 ./BS-Theorie/Klausur
20 ./BS-Theorie/Uebung02-Loesungen/aufgabe-b
20 ./BS-Theorie/Uebung02-Loesungen/aufgabe-c
20 ./BS-Theorie/Uebung02-Loesungen/aufgabe-d
88 ./BS-Theorie/Uebung02-Loesungen
45492 ./BS-Theorie
440 ./IT-Infrastruktur
4780 ./Material und Downloads/IT-Infrastruktur_(REP)_510-r.15_sw
31920 ./Material und Downloads
2648 ./Seminar/bearbeitet
4196 ./Seminar
149812 .
```

```
esser@dissdevel:~/Daten/SWF$ du -s
149812 .
```

← -s = summary,

```
esser@dissdevel:~/Daten/SWF$ du -sm
147 .
```

← -m = megabytes

## debugfs, dumpe2fs, tune2fs (1)

- Arbeiten am Dateisystem (für Fortgeschrittene)
- Tools für die Familie der Ext-Dateisysteme (Ext2, Ext3, Ext4)
  - debugfs: Eingriffe in die „Interna“ des Dateisystems
  - dumpe2fs: Ausgabe aller wichtigen Metadaten des Dateisystems
  - tune2fs: „Tuning“ für Ext-Dateisysteme, Einstellen von Optionen

## debugfs, dumpe2fs, tune2fs (2)

```
root@dissevel:/tmp# dumpe2fs /dev/sda1
dumpe2fs 1.41.12 (17-May-2010)
Filesystem volume name: <none>
Last mounted on: <not available>
Filesystem UUID: 27a7303b-9479-47ea-8ae8-67f9b6206920
Filesystem magic number: 0xEF53
Filesystem revision #: 1 (dynamic)
Filesystem features: has_journal ext_attr resize_inode dir_index filetype needs_recovery
                    sparse_super large_file
                    signed_directory_hash
Filesystem flags: (none)
Default mount options: clean
Filesystem state: clean
Errors behavior: Continue
Filesystem OS type: Linux
Inode count: 623392
Block count: 2491392
Reserved block count: 124569
Free blocks: 1363120
Free inodes: 435570
First block: 0
Block size: 4096
Fragment size: 4096
Reserved GDT blocks: 608
Blocks per group: 32768
Fragments per group: 32768
Inodes per group: 8096
Inode blocks per group: 506
Filesystem created: Tue May 3 11:55:19 2011
Last mount time: Thu Jun 2 17:15:51 2011
Last write time: Thu Jun 2 17:15:41 2011
Mount count: 1
Maximum mount count: 20
Last checked: Thu Jun 2 17:15:41 2011
Check interval: 15552000 (6 months)
Next check after: Tue Nov 29 16:15:41 2011
...
```

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-73

## Journaling

- Moderne Dateisysteme (z. B. ext3, ext4, ReiserFS, BtrFS) verwenden **Journaling**
  - Vor jeder Änderung an den **Metadaten** einer Datei wird in einen Protokollbereich (das **Journal**) die geplante Änderung geschrieben
  - Ist Änderung erfolgreich abgeschlossen, wird Eintrag aus Journal wieder gelöscht
- Beschleunigt (nach Absturz) den FS-Check:
  - nur prüfen, welche Einträge im Journal stehen – diese wurden evtl. nicht erfolgreich durchgeführt
- Variante: nicht nur Metadaten, sondern auch Daten

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-75

## debugfs, dumpe2fs, tune2fs (3)

- `tune2fs`
  - Einstellen, was bei FS-Fehler passiert (continue, panic, remount-ro)
  - Intervall zwischen FS-Checks ändern
  - **Journal** ergänzen oder entfernen (→ Journaling, nächste Folie)
  - **Volume-Label** ändern
  - Größe des **reservierten Bereichs** ändern
    - dieser Teil des FS kann nur von root verwendet werden
    - für normale Nutzer erscheint das FS ggf. als voll

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-74

## Grundlagen von Unix-FS (1)

- Wichtige Konzepte in Linux-Dateisystemen:
  - I-Nodes
  - Dateien und Verzeichnisse
  - Datenblöcke

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-76

## Grundlagen von Unix-FS (2)

- I-Nodes:
  - Wenn eine neue Datei angelegt wird, sucht Linux zunächst einen freien **I-Node** (Index Node) – das ist ein Verwaltungseintrag auf der Partition
  - I-Node enthält Metadaten:
    - Dateigröße, Liste der verwendeten Blöcke
    - Besitzer und Standard-Gruppe
    - Zugriffsrechte, Timestamps ()
    - **nicht im I-Node: Dateiname und/oder Pfad (!)**
  - Danach zu I-Node Eintrag in Verzeichnis anlegen

## Grundlagen von Unix-FS (4)

- Dateien
  - Eine Datei besteht „klassisch“ aus
    - den eigentlichen Nutzdaten, die in Datenblöcken gespeichert sind,
    - einem Dateinamen (mit Pfadangabe)
    - Metadaten (Besitzer, Zugriffsrechte, Größe etc.)
  - Aus Linux-Sicht ist eine Datei zunächst die Sammlung der Datenblöcke + der I-Node (mit Metadaten und Blockliste)
  - Durch Eintragen in ein Verzeichnis (also Zuordnung: Dateiname → I-Node) wird die Datei im Dateisystem sichtbar

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-77

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-79

## Grundlagen von Unix-FS (3)

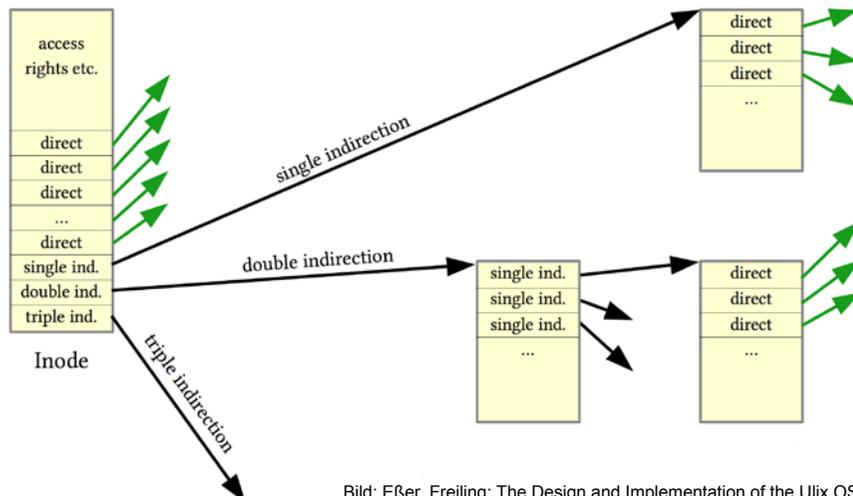


Bild: Eßer, Freiling: The Design and Implementation of the Ulix OS

## Grundlagen von Unix-FS (5)

- Verzeichnisse
  - ... sind in Linux-Dateisystemen spezielle Dateien, welche nur Zuordnungen Name → I-Node enthalten
  - entspricht der Unix-Philosophie „alles ist eine Datei“
  - Da Verzeichnis nur eine Datei ist, ist auch ein schnelles Verschieben eines kompletten Ordners mit Unterordnern schnell erledigt:  
`mv /home/esser/Videos /tmp/Videos`  
benötigt keine messbare Zeit (falls Verschieben innerhalb einer Partition!)

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-78

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-80

## Grundlagen von Unix-FS (6)

- Datenblöcke
  - Dateisystem verwaltet eine Liste freier / belegter Datenblöcke
  - Beim Löschen einer Datei werden alle verwendeten Datenblöcke als „frei“ gekennzeichnet (und bald wiederverwendet)

## Soft Links / Hard Links (2)

- Symbolische Links / Soft Links
  - erstellen mit `ln -s` (s = soft)
  - funktionieren auch Dateisystem-übergreifend (wenn anderes FS auch eingebunden ist)

```
esser@dissdevel:~$ ls -l /mnt/windows/config.sys
-rwxr-xr-x 1 root root 36  2. Jun 20:08 /mnt/windows/config.sys
esser@dissdevel:~$ ln -s /mnt/windows/config.sys config.sys
esser@dissdevel:~$ ls -l config.sys
lrwxrwxrwx 1 esser esser 31  2. Jun 20:08 config.sys -> /mnt/windows/config.sys
esser@dissdevel:~$ ln -s /mnt/windows/BROKEN broken.txt
esser@dissdevel:~$ ls -l broken.txt
lrwxrwxrwx 1 esser esser 27  2. Jun 20:09 broken.txt -> /mnt/windows/BROKEN
esser@dissdevel:~$ cat broken.txt
cat: broken.txt: Datei oder Verzeichnis nicht gefunden
esser@dissdevel:~$ file broken.txt
broken.txt: broken symbolic link to `~/mnt/windows/Windows/BROKEN'
```

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-81

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-83

## Soft Links / Hard Links (1)

- Grundidee hinter Links: Datei unter mehreren Namen (und ggf. an verschiedenen Orten) ansprechen
  - **symbolische Links (soft links)**: spezielle Dateien, die den Pfad (absolut oder relativ) zu einer anderen Datei speichern
    - können „broken“ sein, also auf etwas zeigen, das es nicht gibt (wie im Web: broken link)
  - **Hard Links**: Eintrag in einem Verzeichnis, der auf denselben I-Node zeigt

## Soft Links / Hard Links (3)

- Hard Links
  - erstellen mit `ln` (ohne Option)
  - Quelle und Ziel zeigen auf gleichen I-Node
    - darum nur innerhalb eines Dateisystems möglich

```
esser@dissdevel:~$ touch datei.txt
esser@dissdevel:~$ cp datei.txt kopie.txt
esser@dissdevel:~$ ln datei.txt link.txt

esser@dissdevel:~$ ls -il *.txt
12589 -rw-r--r--  2 esser esser  0  2. Jun 20:16 datei.txt
12590 -rw-r--r--  1 esser esser  0  2. Jun 20:16 kopie.txt
12589 -rw-r--r--  2 esser esser  0  2. Jun 20:16 link.txt

esser@dissdevel:~$ ln /mnt/windows/config.sys config.sys
ln: Erzeuge harte Verknüpfung „config.sys“ => „/mnt/windows/config.sys“:
Ungültiger Link über Gerätegrenzen hinweg
```

*-i: I-Nodes anzeigen*  
*rot: link count*

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-82

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-84

## Soft Links / Hard Links (4)

- Hard Links / Links / Löschen
  - Jeder Eintrag in einem Verzeichnis ist ein Link
  - Das Anlegen eines Hard Links bedeutet also nur: Für die Datei (für den I-Node!) existieren jetzt zwei Einträge in einem (oder mehreren) Verzeichnissen
  - Linux kennt intern keine „Löschen“-Operation, sondern nur eine „Unlink“-Operation
    - sie entfernt den ausgewählten Link, also die Zuordnung Dateiname → I-Node
    - und zählt den Link Count um 1 runter
    - Wenn Link Count 0 erreicht wird, wird I-Node freigegeben

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-85

## Loop-Dateisysteme (1)

Themen:

- Loop-Geräte
- Zugriff auf Dateisystem-Images
- Verschlüsselung mit cryptsetup (LUKS)

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-87

## Loop-Dateisysteme

## Loop-Geräte

- Loop-Geräte ordnen Dateien virtuellen Geräten zu
- manuelle Einrichtung mit `losetup`

```
linux:~# dd if=/dev/zero of=/tmp/disk bs=1M count=100
linux:~# losetup -f /tmp/disk
linux:~# losetup -l
NAME          SIZELIMIT OFFSET AUTOCLEAR RO BACK-FILE
/dev/loop0    0          0          0      0 /tmp/disk
linux:~# mkfs.ext3 /dev/loop0
```

- automatische Einrichtung bei `mount -o loop`

```
linux:~# mount -o loop,ro /tmp/student.iso /mnt/
linux:~# losetup -l
NAME          SIZELIMIT OFFSET AUTOCLEAR RO BACK-FILE
/dev/loop0    0          0          0      0 /tmp/disk
/dev/loop1    0          0          1      0 /tmp/student.iso
```

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-86

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-88

## Verschlüsselung mit LUKS (1)

```
linux:~# dd if=/dev/urandom of=/tmp/crypt.file bs=1M count=100
linux:~# losetup -f /tmp/crypt.file
linux:~# losetup -l
NAME          SIZELIMIT OFFSET AUTOCLEAR RO BACK-FILE
/dev/loop0    0         0         0 0 /tmp/disk
/dev/loop1    0         0         1 0 /tmp/student.iso
/dev/loop2    0         0         0 0 /tmp/crypt.file
linux:~# cryptsetup -c aes-xts-plain64 -s 512 -h sha512 -y \
  luksFormat /dev/loop2
(Passwort festlegen)
linux:~# cryptsetup luksOpen /dev/loop2 privat
(Passwort eingeben)
linux:~# mkfs.ext4 /dev/mapper/privat
linux:~# mkdir /home/privat; mount /dev/mapper/privat /home/privat
linux:~# df -h /home/privat
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/privat 91M  1.6M  83M   2% /home/privat
```

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-89

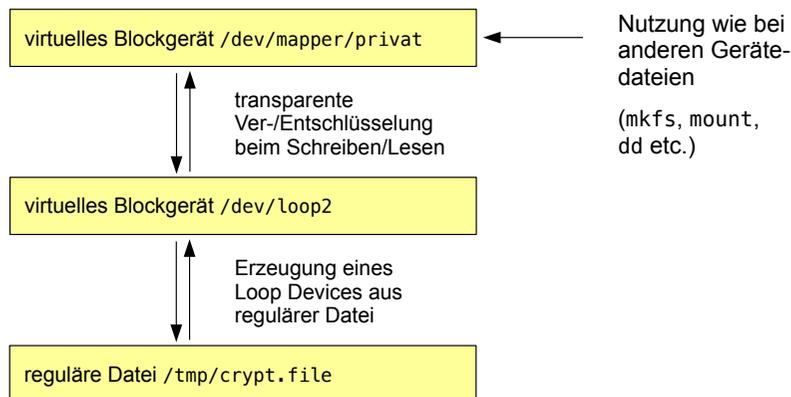
## Zugriffsrechte (Unix/Linux)

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-91

## Verschlüsselung mit LUKS (2)



## Dateien, Benutzer, Gruppen

- Jede Datei
  - ... gehört einem Benutzer (Besitzer, **user**)
  - ... und zu einer Gruppe (**group**)
- Benutzer können Mitglieder in verschiedenen Gruppen sein
- Zugriffsrechte entscheiden, ob eine Datei gelesen (**read**), geschrieben (**write**) oder ausgeführt (**execute**) werden darf

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-90

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-92

## Gruppen (1)

- In welchen Gruppen bin ich Mitglied?  
`$ groups`  
swf cdrom floppy audio dip video plugdev netdev  
powerdev scanner
- Mitgliedschaft durch Einträge in `/etc/group` geregelt:  
`$ grep swf /etc/group`  
cdrom:x:24:swf  
floppy:x:25:swf  
audio:x:29:swf  
...  
swf:x:1002:swf
- Gruppenmitgliedschaft bearbeiten:  
manuell oder (besser!) mit `gpasswd`

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-93

## Gruppen (3)

- Jeder Benutzer ist in einer Standardgruppe Mitglied. Welche ist das?  
`$ id`  
uid=1002(swf) gid=1002(swf) Gruppen=1002(swf),  
24(cdrom),25(floppy),29(audio),30(dip),...
- Zwei Standards für Standardgruppe
  - Debian-System: Jeder Benutzer hat seine eigene Standardgruppe (User: swf, Group: swf)
  - andere Systeme: Standardgruppe `users` für alle „normalen“ Benutzer
- Im Namen der Standardgruppe handeln Sie, bis Sie mit `newgrp` die Gruppe ändern.

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-95

## Gruppen (2)

- Gruppe mit `gpasswd -a user group` (add) ergänzen:  
`# gpasswd -a swf neugr`  
Benutzer swf wird zur Gruppe neugr hinzugefügt.  
`# groups swf`  
swf cdrom floppy audio dip video plugdev netdev  
powerdev scanner neugr
- Entfernen einer Gruppenmitgliedschaft:  
`gpasswd -d user group` (delete)  
`# gpasswd -d swf neugr`  
Benutzer swf wird aus der Gruppe neugr entfernt.

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-94

## Gruppen (4)

- Neue Gruppen kann der Administrator mit `groupadd` erzeugen, um Kooperation von Teams zu erleichtern
  - z. B. mit Dateien, die für alle Gruppenmitglieder (und nur diese) les- und schreibbar sind
- Beispielszenario folgt ...

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-96

## Beispielszenario (1)

- Gruppe `profs`: Mitglieder `prof1`, `prof2`
- Gruppe `studis`: Mitglieder `anna`, `tom`, `fritz` und (!) `prof1`, `prof2`
- Ziele:
  - `profs`-Mitglieder können Daten untereinander austauschen und teilweise auch Studenten zur Verfügung stellen
  - `studis`-Mitglieder können Daten untereinander austauschen und auf die von Profs zur Verfügung gestellten Skripte, Aufgaben etc. zugreifen

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-97

## Beispielszenario (3)

- Umsetzung: später
- Nachteil: keine vernünftige Zugriffsbeschränkung für `/srv/profs/public` möglich  
→ ACLs

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-99

## Beispielszenario (2)

- Verzeichnisstruktur

```
/srv/profs/  
/srv/profs/intern/           ; Austausch der Profs untereinander  
/srv/profs/intern/klausuren/  
/srv/profs/public/         ; Lesezugriff für Studenten möglich  
/srv/profs/public/skripte/  
/srv/studis/  
/srv/studis/mitschriften/  
/srv/studis/pruefungsprot/
```

- Gruppenzugehörigkeiten und Zugriffsrechte

- `/srv/profs/intern`: gehört Gruppe `profs`; lesen und schreiben für `profs` erlaubt, kein Zugriff für `studis`
- `/srv/profs/public`: gehört Gruppe `profs`; schreiben für `profs` erlaubt, lesen für alle (auch Nicht-`Studis`)
- `/srv/studis`: gehört Gruppe `studis`; lesen und schreiben für Gruppenmitglieder erlaubt

## Unix-Dateiattribute (1)

- Neben Besitzer und Gruppe gibt es noch die sonstigen Systembenutzer (`o`, `others`)
- ergibt 9 Zugriffsrechte; Notation bei `ls`:

```
-rwxrwxrwx  
Besitzer | Gruppe | sonstige
```

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-98

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-100

## Unix-Dateiattribute (2)

- `chown` (change owner) und `chgrp` (change group) ändern Besitzer und Gruppe einer Datei
- `chmod` (change mode) ändert Zugriffsrechte
- Beispiele:  

```
chown swf /tmp/log.txt
chgrp www-data /var/www/srv1
chmod o+r /tmp/log.txt
chmod o-rwx,ug+rw /tmp/log.txt
chmod u=rw,g=r,o= /tmp/log.txt
```
- Abkürzung `a` (all) für `ogu` (`chmod a=rw ...`)

## Unix-Dateiattribute (4)

- `chmod` mit numerischen Rechten nutzen
  - `rw- r-- --- = 640 (4+2+0, 4+0+0, 0+0+0)`
  - `chmod u=rw,g=r,o= /tmp/log.txt`  
`chmod 640 /tmp/log.txt`
- bei der numerischen Angabe kein „Geben“ und „Nehmen“ von Rechten möglich (wie mit `chmod u+x ...`, `chmod o-rwx ...`)

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-101

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-103

## Unix-Dateiattribute (3)

- numerische Rechte:
  - Leserecht: 4 ( $2^2$ )
  - Schreibrecht: 2 ( $2^1$ )
  - Ausführrecht: 1 ( $2^0$ )
  - aufaddieren, z. B.: `rw` = Lesen/Schreiben:  $4+2=6$
- für Benutzer, Gruppe und Sonstige: **nnn**
  - z. B. **640**:
    - Benutzer: 6 = lesen + schreiben (nicht ausführen)
    - Gruppe: 4 = lesen (nicht schreiben, nicht ausführen)
    - Sonstige: 0 = nichts

## Unix-Dateiattribute (5)

- Beim Erzeugen einer Datei werden Standardrechte gesetzt – welche das sind, bestimmt die **UMASK (user file creation mask)**

```
$ umask          Standard:
0022 ← Gruppe: nicht schreiben,
$ umask a=rw     Sonstige: nicht schreiben
$ umask
0111
$ touch Datei; ls -l Datei
-rw-rw-rw- 1 esser users 0 2008-12-04 20:48 Datei
$ umask u=rw,g=r,o=
$ umask
0137
$ touch Test; ls -l Test
-rw-r----- 1 esser users 0 2008-12-04 20:50 Test
```

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-102

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-104

## Unix-Dateiattribute (6)

- umask wird von 666 (rw-rw-rw-: Standardwert für Dateien) bitweise abgezogen, um konkrete Dateirechte zu berechnen;
- Ausführrecht wird beim Erzeugen einer Datei nie vergeben
- Linux unterstützt diese klassischen Unix-Dateiattribute und einige zusätzliche (→ Folien *Extra-Flags*, *Erweiterte Attribute*)

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-105

## Unix-Dateiattribute (7)

- Dateiattribute nur auf echten Unix-Dateisystemen nutzbar – auf Windows-Datenträgern nur stark eingeschränkt:

```
# mount | grep windows
/dev/sda3 on /windows/D type vfat (rw,gid=100,umask=0002)
# touch /windows/D/Testdatei
# ls -l /windows/D/Testdatei
-rwxrwxr-x 1 root users 0 2006-12-04 21:07 /windows/D/Testdatei
# chmod a-rwx /windows/D/Testdatei
# ls -l /windows/D/Testdatei
----- 1 root users 0 2006-12-04 21:07 /windows/D/Testdatei
# umount /windows/D; mount /windows/D; ls -l /windows/D/Testdatei
-r-xr-xr-x 1 root users 0 2006-12-04 21:07 /windows/D/Testdatei
```

- Windows kennt kein Ausführattribut – wohl aber ein Read-Only-Attribut

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-106

## Unix-Dateiattribute (8)

- Bedeutung der Attribute für Verzeichnisse:
  - read: Verzeichnisinhalt lesen (ls in einem Verzeichnis ausführen)
  - write: Verzeichnisinhalt ändern (z. B. neue Datei erzeugen, Datei umbenennen)
  - execute: Verzeichnis betreten, also zum aktuellen Arbeitsverzeichnis machen (cd)
  - Standardrechte, von denen die umask abgezogen wird, sind bei Verzeichnissen 777 (denn x = execute steht ja für „Verzeichnis betreten“)

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-107

## Beispielszenario (4)

- Zurück zum Beispielszenario

- Ersteinrichtung:

```
root# chown -R root /srv/profs /srv/studis
root# chgrp -R profs /srv/profs/intern
root# chmod ug=rwx,o= /srv/profs/intern
root# chgrp -R profs /srv/profs/public
root# chmod ug=rwx,o=rx /srv/profs/public
root# chgrp -R studis /srv/studis
root# chmod ug=rwx,o= /srv/studis
```

- neue Dateien erzeugen:

```
prof1$ newgrp profs # als „profs“-Mitglied arbeiten
prof1$ umask 0007 # Neue Dateien nicht für andere
prof1$ cd /srv/profs/intern
prof1$ touch pruefung.doc
prof1$ ls -l pruefung.doc
-rw-rw---- 1 prof1 profs ... pruefung.doc
```

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-108

## SUID, SGID (1)

- Problem: Es gibt Dateien, die Benutzer nur „unter kontrollierten Bedingungen“ ändern dürfen, z. B. die Passwortdatei `/etc/shadow`
  - Änderung an der Datei mit dem Tool `passwd`
  - Dafür sind Root-Rechte nötig
  - Normale Anwender haben keine Root-Rechte
- Zwei Lösungen
  - klassisch: SUID (siehe nächste Folie)
  - neuer: `sudo` (behandeln wir hier nicht)

## SUID, SGID (3)

- SUID- und SGID-Bits mit `chmod` setzen

```
# cp /usr/bin/passwd /tmp/mypasswd
# chmod u-s,g+s /tmp/mypasswd
# ls -l /usr/bin/passwd /tmp/mypasswd
-rwSr-xr-x 1 root root    ... /usr/bin/passwd
-rwxr-sr-x 1 root shadow  ... /tmp/mypasswd
```
- s-Bits erscheinen in der `ls`-Ausgabe immer an der Stelle, wo sonst das x steht
- Diese Bits sind bei Shell-Skripten wirkungslos (in einigen älteren Unix-Versionen funktionierte das auch mit Skripten)

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-109

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-111

## SUID, SGID (2)

- Ausführbare Dateien (nur Binaries) können ein SUID- (Set User ID) und/oder ein SGID-Bit (Set Group ID) haben
  - SUID: Programm läuft immer mit den Rechten des Dateibesitzers, meist `root`
  - SGID: Programm läuft immer mit den Gruppenrechten der Dateigruppe (seltener verwendet)
  - Beispiel: `passwd` muss Systemdateien ändern

```
$ ls -l /usr/bin/passwd /etc/shadow
-rwSr-xr-x 1 root root    ... /usr/bin/passwd
-rw-r----- 1 root shadow ... /etc/shadow
```

## Extra-Flags (1)

- Ext2-/Ext3-/Ext4-Extra-Flags (immutable, append-only etc.) mit `chattr` bearbeiten

### NAME

**chattr** - change file attributes on a Linux second extended file system

### SYNOPSIS

**chattr** [ -RV ] [ -v version ] [ mode ] files...

### DESCRIPTION

**chattr** changes the file attributes on a Linux second extended file system.

The format of a symbolic mode is +=[ASacDdijsTtu].

The operator '+' causes the selected attributes to be added to the existing attributes of the files; '-' causes them to be removed; and '=' causes them to be the only attributes that the files have.

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-110

12.10.2017

Betriebssysteme 2, WS 2017/18, Hans-Georg Eßer

Folie C-112

## Extra-Flags (2)

- Beispiel für `chattr`:

```
# cd /tmp; touch logdatei
# chattr +a logdatei
# echo Hallo >> logdatei      # >> = anhängen
# echo Welt >> logdatei
# cat logdatei
Hallo
Welt
# echo Ueberschreiben > logdatei
bash: logdatei: Die Operation ist nicht erlaubt
```

- Attribute anzeigen mit `lsattr`:

```
# lsattr -l logdatei
logdatei                Append_Only
```

## Erweiterte Attribute (2)

- bearbeiten mit `setfattr`, `getfattr`, `attr`:

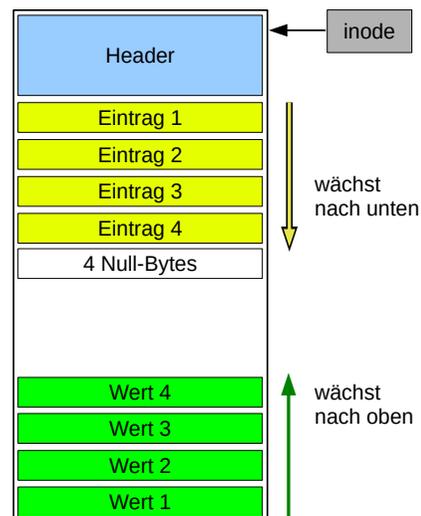
```
amd64:/home/esser # setfattr -n user.foo -v betriebssysteme test.txt
amd64:/home/esser # getfattr -d test.txt
# file: test.txt
user.foo="betriebssysteme"
```

```
amd64:/home/esser # attr -g user.foo test.txt
Attribute "user.foo" had a 15 byte value for test.txt:
betriebssysteme
```

- Software ist auf dem Debian-System nicht installiert → `apt-get install attr`
- Verwaltung von ACLs über das Paket `acl` → `apt-get install acl`
  - Tools: `getfacl`, `setfacl`

## Erweiterte Attribute (1)

- Erweiterte Attribute speichern beliebige Name-/Wert-Paare, u. a. ACLs
- Inode-Größe: 128 Byte
  - kein Platz für erweiterte Attribute
  - Vergrößerung auf 256 Byte nicht effizient
- Lösung: Separater Block für extended attributes



## Dreierlei Attribute

### Nicht verwechseln:

- Standard-Unix-Dateiattribute
  - UID, GID
  - Standardzugriffsrechte `rxw` für `user/group/others`
  - Zugriffszeiten, ...
- Extra-Flags
  - `immutable`, `compressed`, `secure deletion`, ...
- Extended Attributes
  - beliebige, frei definierbare Attribute (inkl. ACLs)