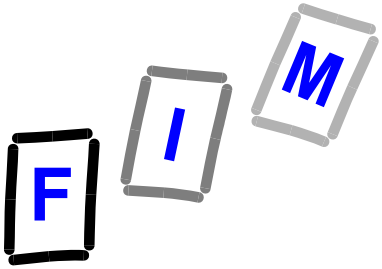


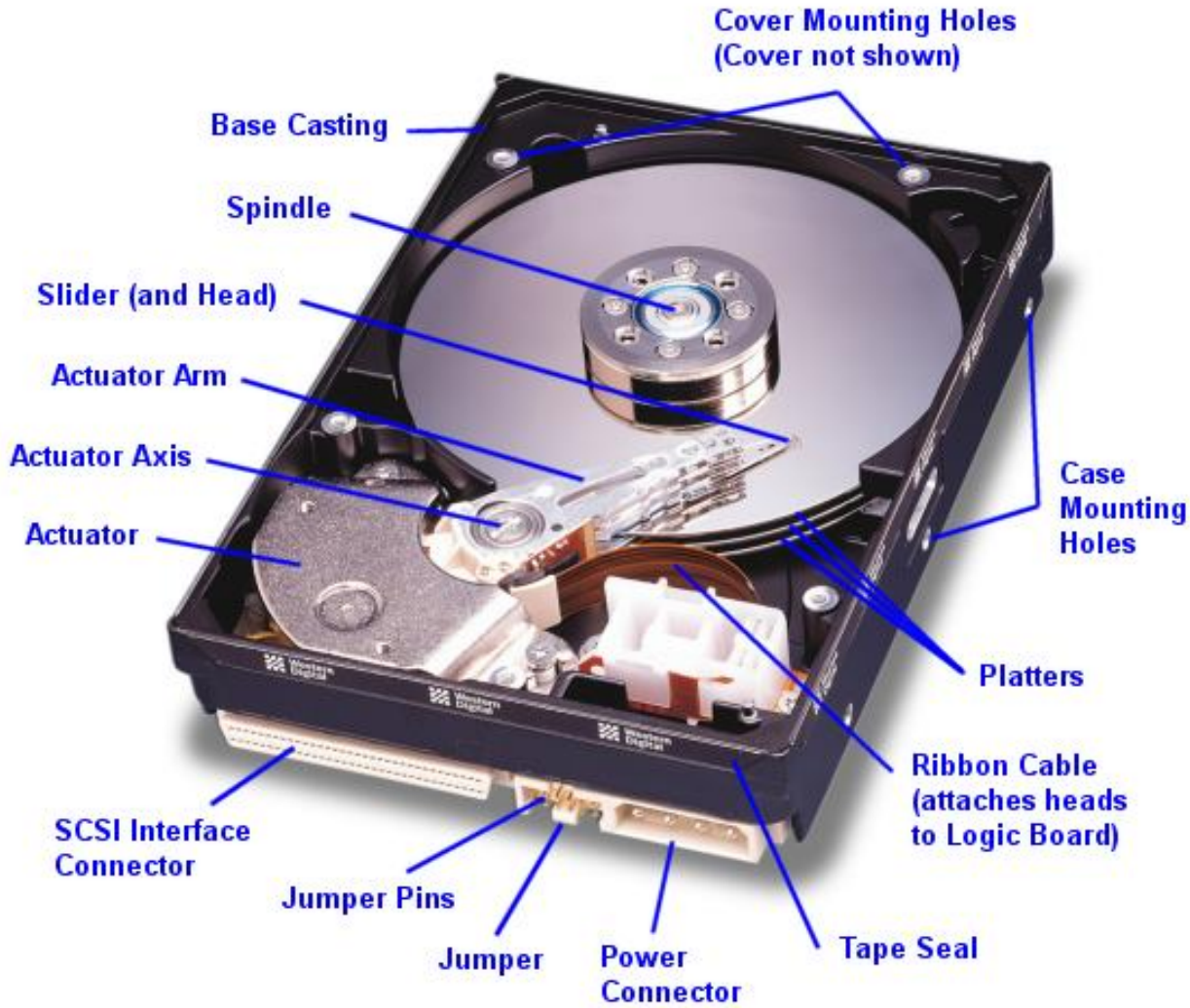
SS 2007

KV Betriebssysteme

Datenpersistenz in Betriebssystemen
Dateien und Dateisysteme

Andreas Putzinger , Michael Sonntag, Rudolf Hörmanseder

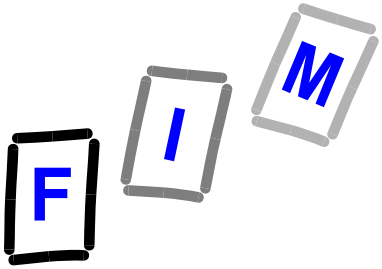
Beispiel: Festplatten-Hardware



(http://www.storagereview.com/guide2000/hdd/...)

Allgemeines zu Festplatten

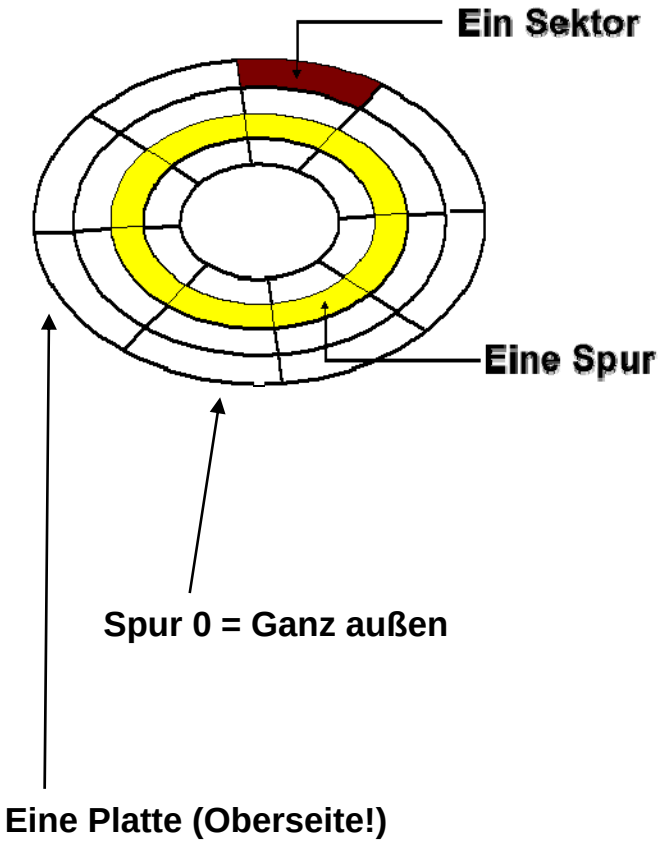
- Verschiedene Baugrößen
- Rotierende Scheiben = „Platten“
  - Aus Aluminium oder Legierung; event. Glas
  - Beschichtung: Eisenoxyd, Kobaltschicht, …
- "Kamm" mit Schreib-Leseköpfen
- Landing Zone / Auto Parking
  - Früher "manuell" nötig; erfolgt heute automatisch!
- Staubdicht, aber nicht luftdicht
- „Geometrie“
  - Anzahl Sektoren, Köpfe, Zylinder
- Reserve-Spuren
- Ansteuerung per (Serial) ATA / SCSI üblich
- SMART = Self-Monitoring Analysis and Reporting Technology



SMART

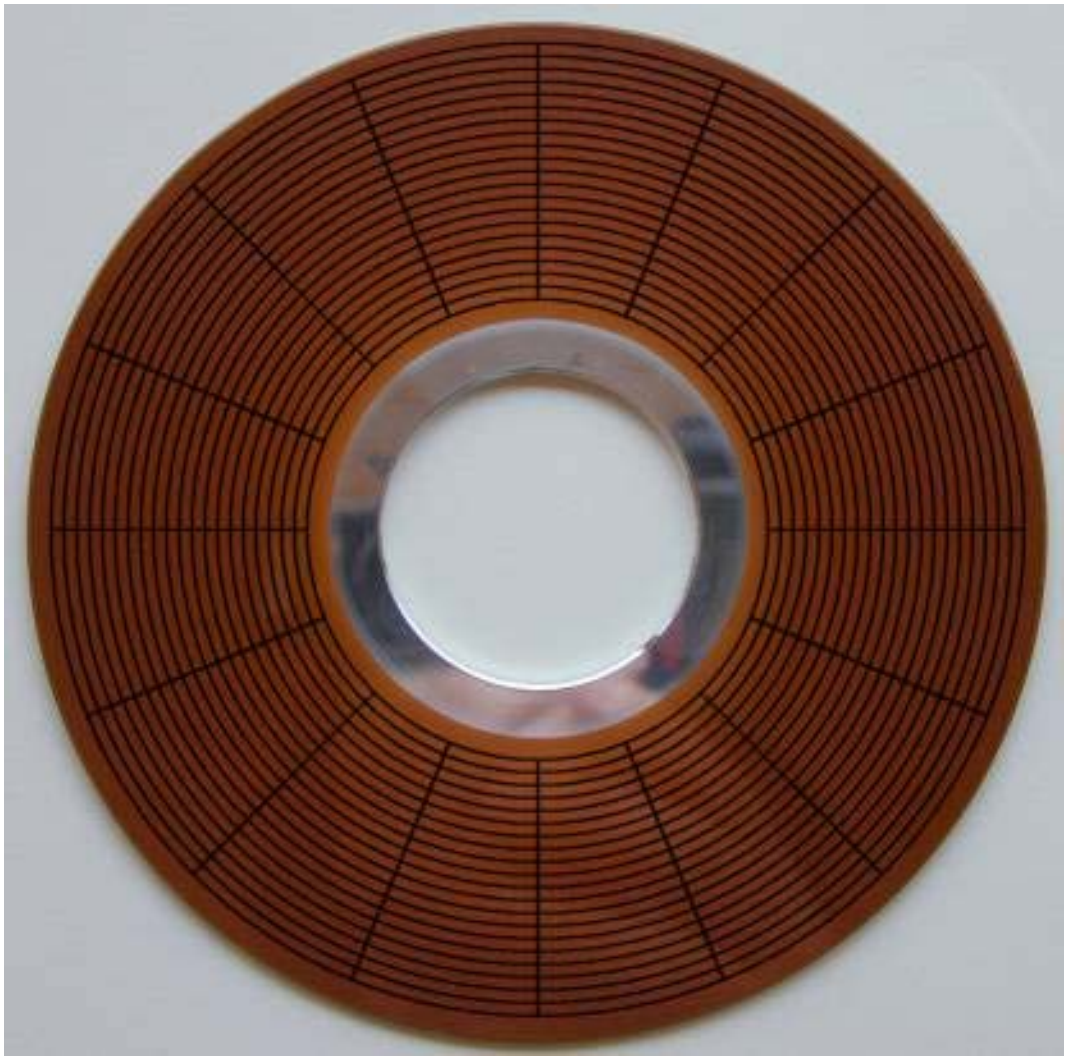
- **Fehler können in zwei Gruppen aufgeteilt werden**
  - **Nicht vorhersehbar: Hier lässt sich nichts machen!**
    - » **Beispiele: Chip defekt, Kondensator explodiert, …**
  - **Vorhersehbar: Bestimmte Parameter ändern sich langsam und kontinuierlich, bis sie irgendwann eine Schwelle über-/unterschreiten und ein Fehler auftritt**
- **Beispiele:**
  - **Motor bzw. Motor-Lager: Anlauf-/Bremszeit, Lager-Temperatur, Strombedarf des Motors etc.**
  - **Head-Crash: Höhe des Kopfes über der Oberfläche**
  - **Oberflächendefekte: Anzahl/Rate defekter Sektoren**
- **Festplatte erzeugt keine Hinweise! Es wird lediglich der Status gespeichert**
  - **Software erforderlich, die den Status regelmäßig ausliest!**

Wozu Spuren und Sektoren?



- **Das Formatieren erzeugt ein Dateisystem auf einem Speichermedium**
  - **Dieses muss einzelne "Teile" adressieren können!**
- **Eine Platte ist eingeteilt in tausende konzentrische Kreise = Spuren (tracks)**
- **Unterteilung der Spuren in Sektoren zu je 512 Byte**
- **Sektor = Kleinste adressierbare Einheit auf Festplatte (="Teile")**

Spuren und Sektoren



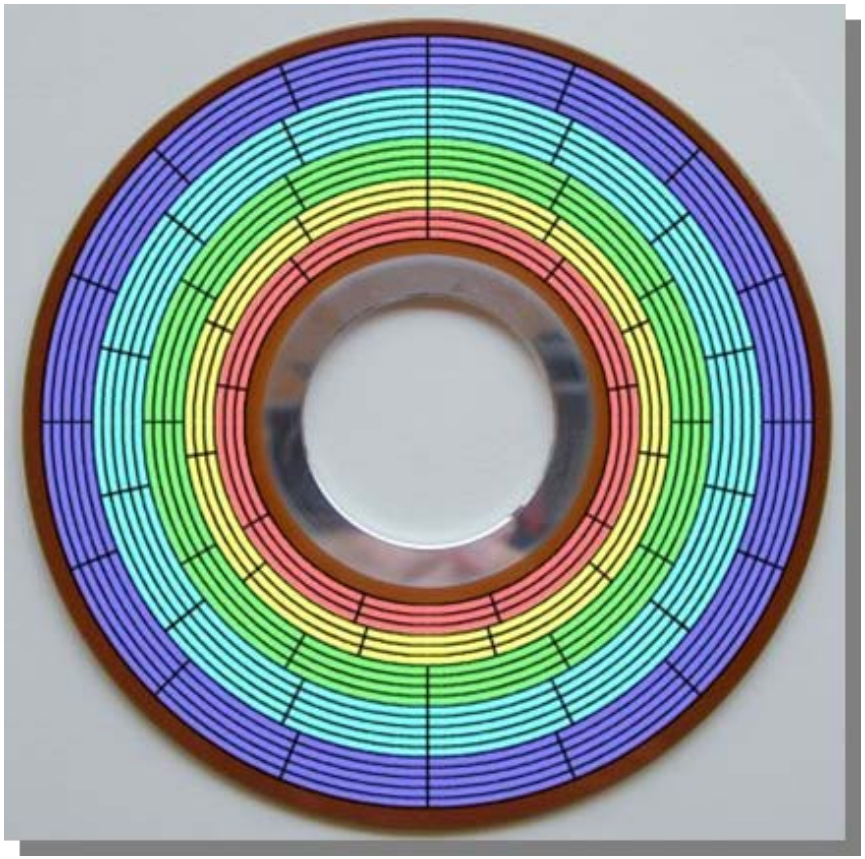
- **5,25" Diskette**
  - **2 Seiten**
  - **Je 40 Spuren**
  - **Je 9 Sektoren**
- **Datenmenge:**
  - **2*40*9*512**
  - **368640 Bytes**
    - **=360 kBytes**

Bild: 20 Spuren, 16 Sektoren

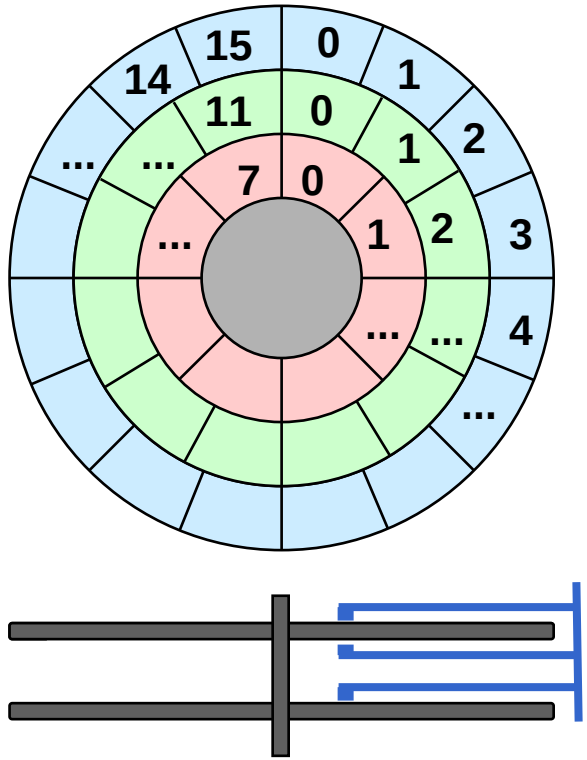
Quelle: http://www.storagereview.com/guide2000/ref/hdd/geom/tracks.html

ZBR
Zoned Bit Recording

Zonen mit verschiedener Anzahl von Sektoren pro Spur

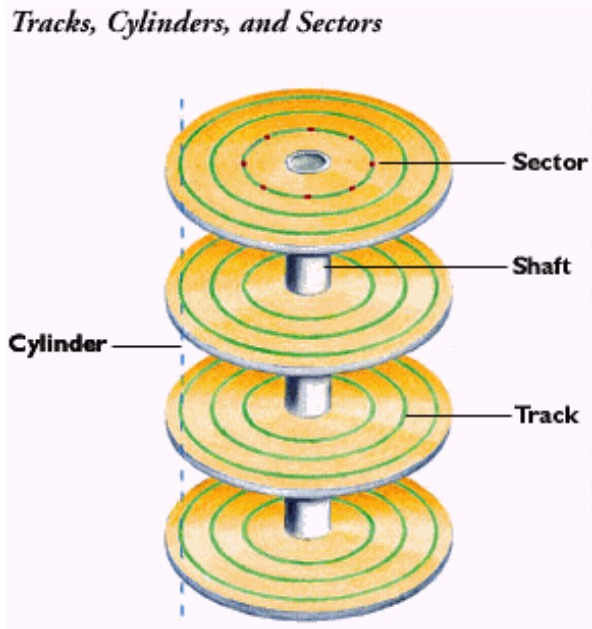


(http://www.storagereview.com/guide2000/hdd/...)



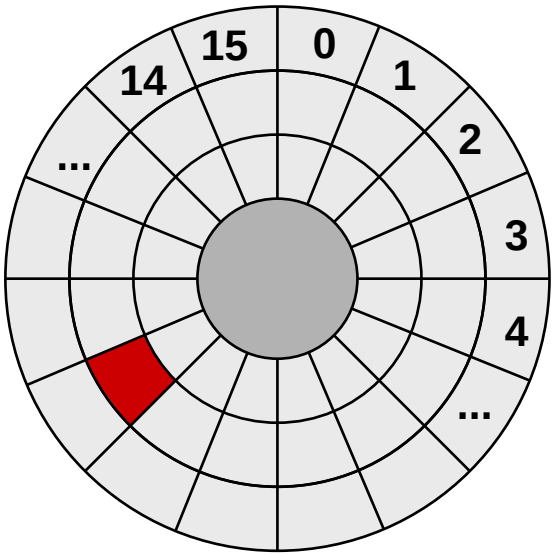
Zylinder

- **Alle identischen Spuren einer Festplatte**
- **Auf Daten im Zylinder kann ohne Bewegung des Schreib-Lesekopfes gleichzeitig zugegriffen werden**
- **Bsp: Zylinder bei 4 Platten umfasst wie viele Spuren?**



Disk-Struktur

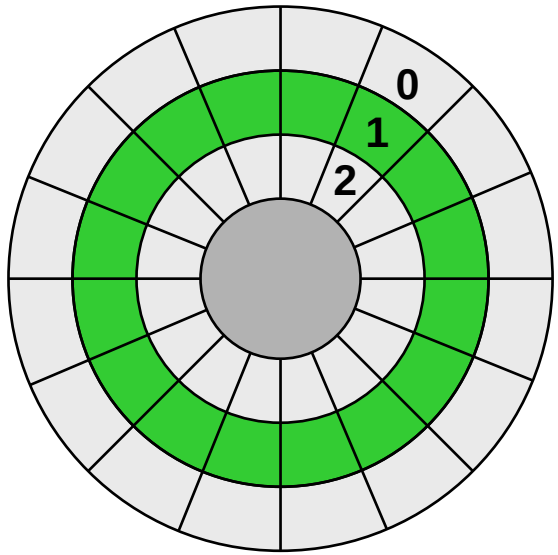
Sektor



sec_per_track
(16)

sec
[0 .. sec_per_track-1]

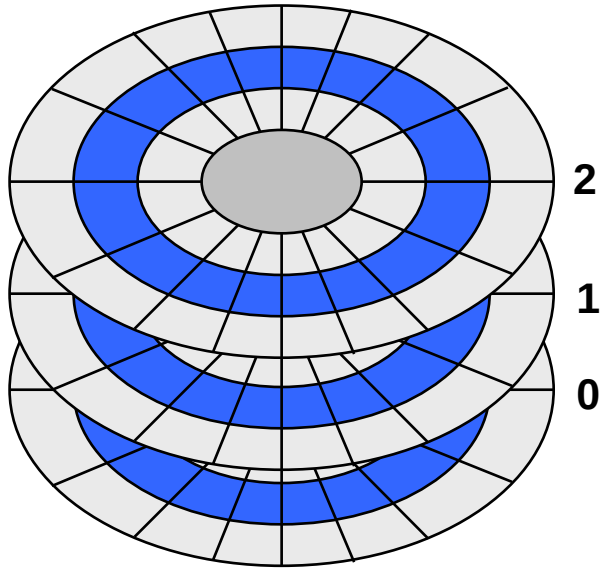
Spur



nr_cyl
(3)

cyl
[0 .. nr_cyl-1]

Zylinder



tracks_per_cyl
= Kopfanzahl
(3)

head
[0 .. tracks_per_cyl-1]

Wozu Cluster?

- Zusammenfassung mehrerer Sektoren zu 1 Cluster
- Cluster = kleinste Einheit, die vom Betriebssystem angesprochen werden kann
- Verwaltungstechnische Erfindung, damit Betriebssysteme mit unterschiedlich großen Festplatten umgehen können
- Fragmentierung?
- Vor- und Nachteile von Clustergrößen?

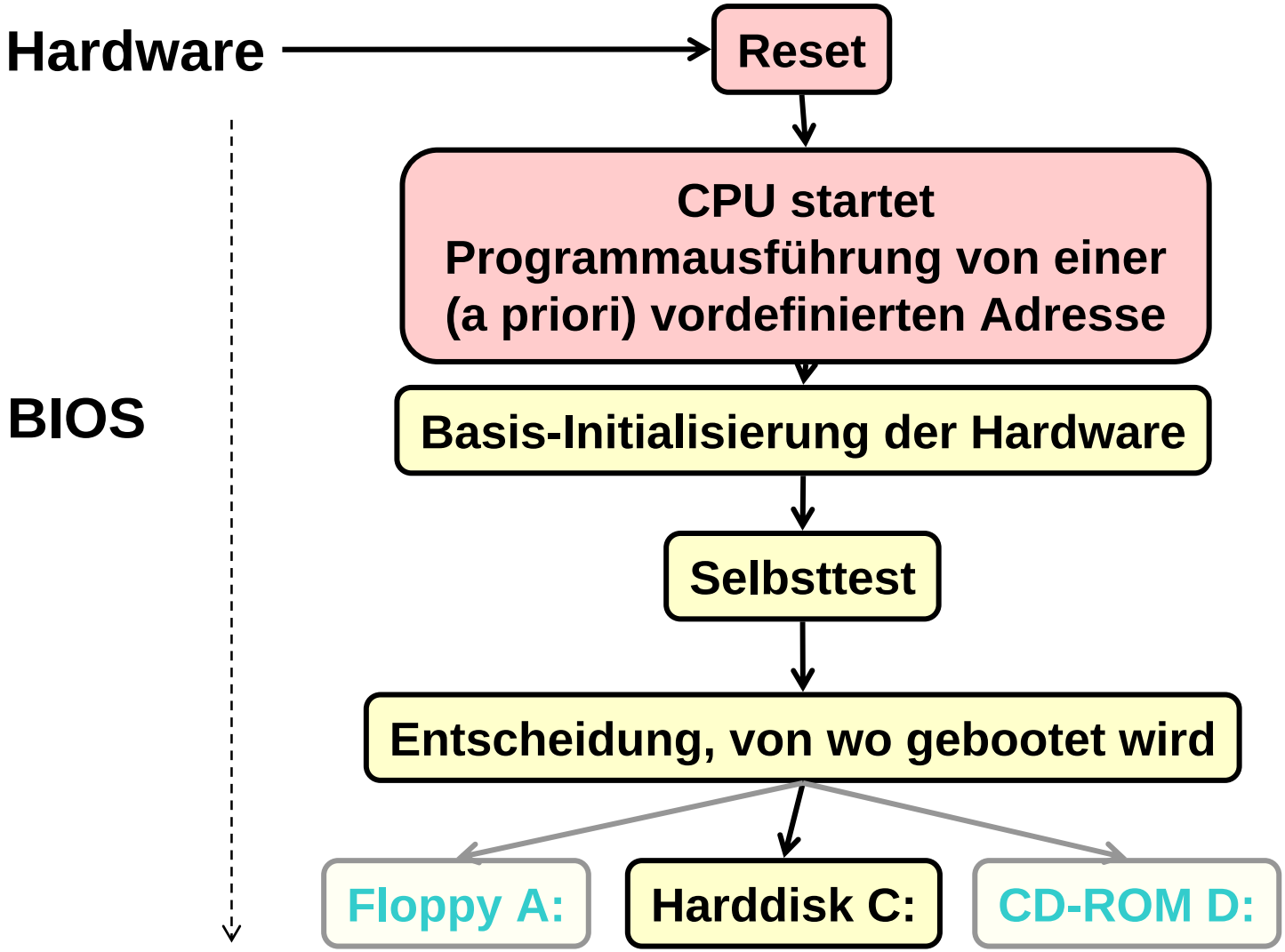
Clustergrößen?

	Maximal 2^{16} Cluster	Maximal 2^{28} Cluster	
Partitionsgröße	**FAT16**	**FAT32**	**NTFS**
< 16 MB	2 KB	-	512 Byte
< 32 MB	512 Byte	-	512 Byte
< 64 MB	1 KB	512 Byte	512 Byte
< 128 MB	2 KB	1 KB	512 Byte
< 256 MB	4 KB	2 KB	512 Byte
< 512 MB	8 KB	4 KB	512 Byte
< 1 GB	16 KB	4 KB	1 KB
< 2 GB	32 KB	4 KB	2 KB
< 4 GB	64 KB	4 KB	4 KB
< 8 GB	-	4 KB	4 KB
< 16 GB	-	8 KB	4 KB
< 32 GB	-	16 KB	4 KB
< 2 TB	-	-	4 KB

Disk-Partition und OS-BOOT

- **BIOS**
  - **„Basic Input / Output System“**
    Stellt u. a. Informationen über die Disks bereit
- **MBR**
  - **Master Boot Record**
    Enthält Partitions-Information über die Disk sowie ein kleines Codestück.

OS BOOT (1)


Hardware
Reset
CPU startet Programmausführung von einer (a priori) vordefinierten Adresse
BIOS
Basis-Initialisierung der Hardware
Selbsttest
Entscheidung, von wo gebootet wird
Floppy A:
Harddisk C:
CD-ROM D:

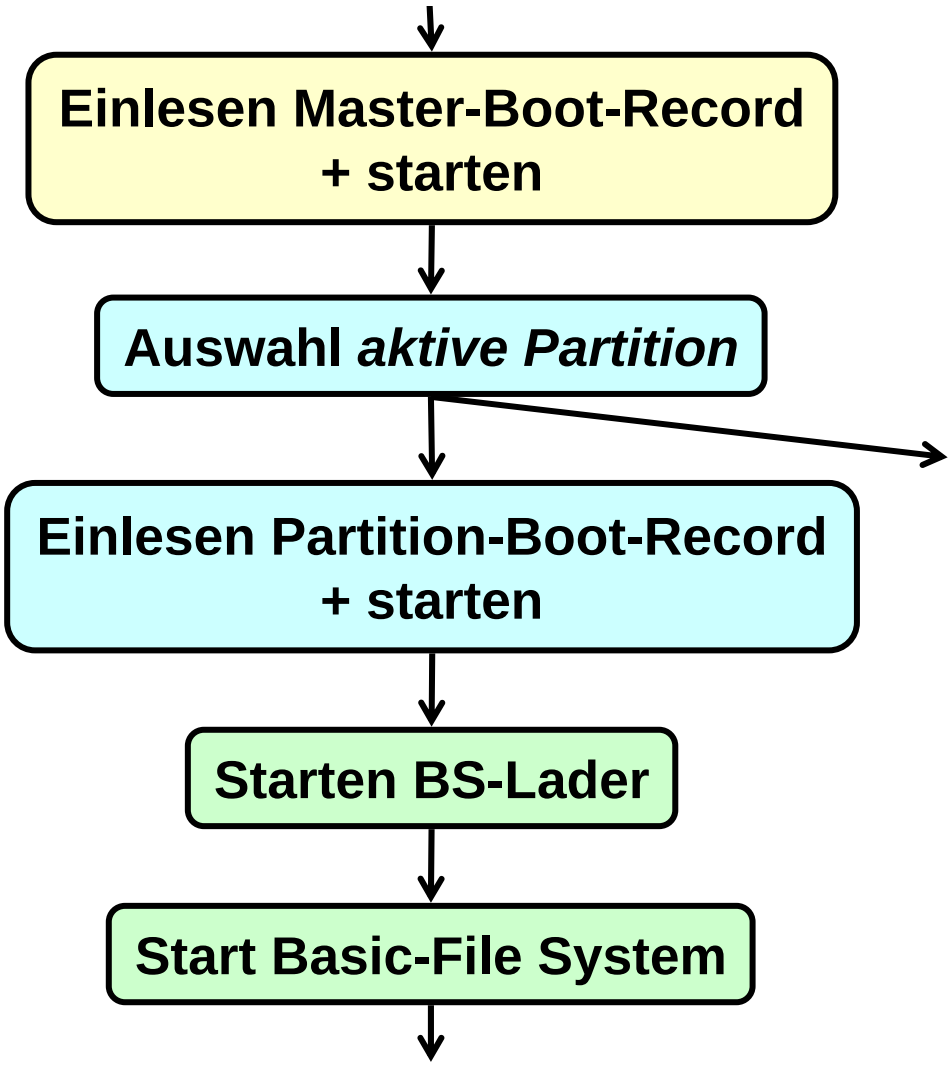
OS
BOOT (2)

BIOS

MBR

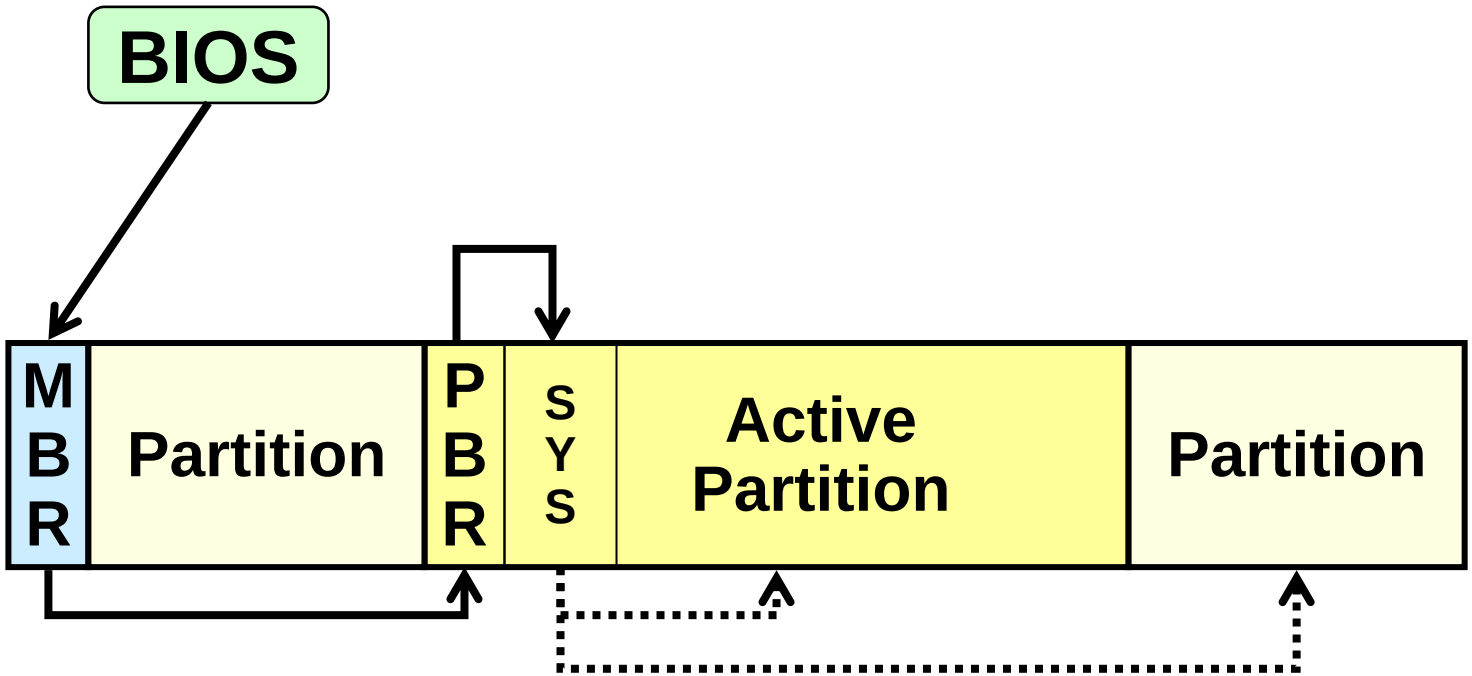
PBR

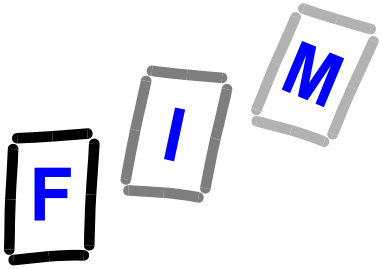
OS



Disk-Management
Logisch != Physisch

- **Boot-Sequenz**



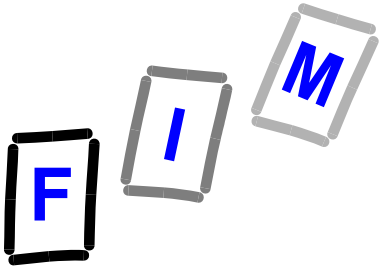


Was ist ein Dateisystem?

- **Permanente Speicherung von Daten**
- **Schnittstelle zwischen Betriebssystem und Laufwerken**
  - **Abstrahiert von den physischen Gegebenheiten des Laufwerks**
  - **Ergebnis: Linearer permanenter Speicher mit Adressen von 0 bis <SIZE> (=Datei)**
- **Organisation der Informationen**
  - **Dateien**
  - **Verzeichnisse**

Welche Dateisysteme gibt es?

- FAT (File Allocation Table), 16 Bit DOS-System, FAT16
- FAT32, Windows 95B (OSR2)
- HPFS (High Performance File System), OS/2, 32-Bit-Dateisystem
- NTFS, Windows NT, 32-Bit-Dateisystem
- NetWare, eigenes 32-Bit-Dateisystem von Novell
- ISO 9660 für CD-ROM und ISO 13346 für DVD
- UDF (Universal Disk Format) ist für Speichermedien mit einer großen Kapazität gedacht, wie z.B. DVD-RAM
  - ISO 13346; Nachfolger von ISO 9660
- ReiserFS, ext, ext2, ext3, XFS, JFS, GPFS (Linux)
- etc.



Betriebssysteme und Dateisysteme

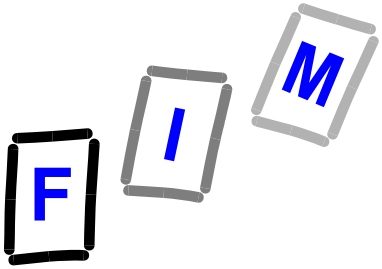
Betriebssystem	Dateisystem(e)
DOS	FAT16
Windows 95	FAT16
Windows 95 OSR2	FAT16, FAT32
Windows 98, ME	FAT16, FAT32
Windows NT 4	FAT16, NTFS4
Windows 2000	FAT16, FAT32, NTFS4
Windows XP	FAT16, FAT32, NTFS4, NTFS5
Windows Vista	NTFS *(Ursprünglich geplant: WinFS)*
OS/2	FAT16, HPFS
Novell NetWare	eigenes Dateisystem
Linux	ReiserFS, ext, ext2, ext3, XFS, JFS, GPFS, FAT16, FAT32, NTFS4, NTFS5, …

Betriebssysteme und Größen?

Betriebssystem / Dateisystem	Größe
DOS-Versionen vor 3.0	bis 16 MB
DOS-Version 3.0 und 3.32	bis 32 MB
DOS 4.0	bis 128 MB
DOS 5.0	bis 528 MB

- DOS 5.0 und das BIOS, welches für IDE-Laufwerke zuständig ist, akzeptierten nur 1024 Zylinder und Festplatten bis zu 528 MB. Dieses Limit wurde durch den EIDE-Standard gebrochen.

FAT16	bis 2 GB (16-Bit-Cluster)
FAT32	bis 2048 GB

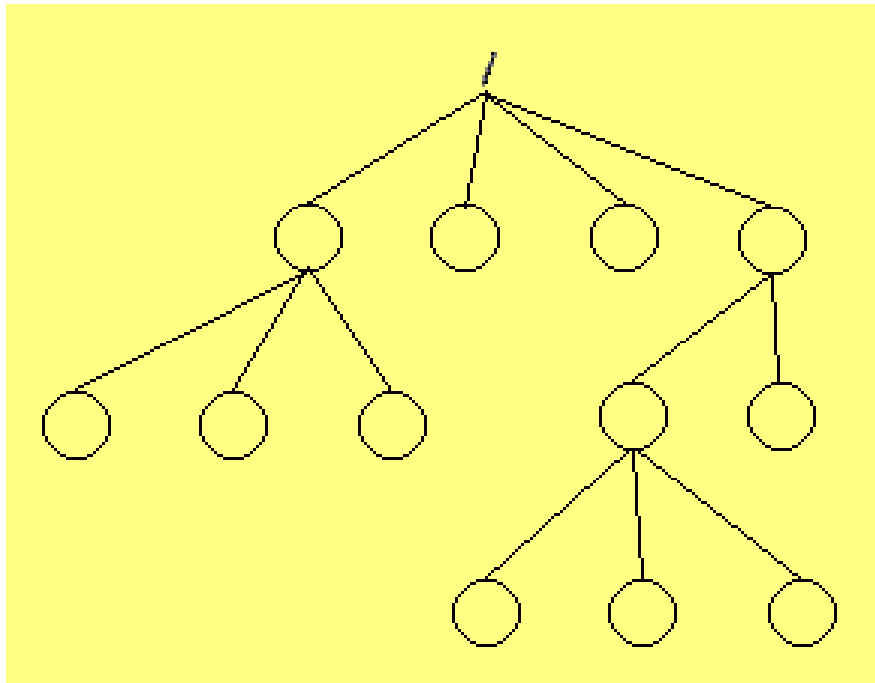


Was ist eine Datei?

- **Aus der Sicht des Betriebssystems ist eine Datei eine Folge von Bytes, deren Bedeutung dem Betriebssystem nicht bekannt ist. Diese wird von der Anwendung definiert.**
- **Zur eindeutigen Identifikation hat eine Datei einen Namen.**
- **Blockstruktur: Zugriff auf ganze Blöcke best. Größe**
- **Darüber hinaus kann jede Datei noch weitere Attribute haben: Typ, Größe, Zugriffsberechtigungen, Datum und Uhrzeit der Dateianlage, der letzten Änderung, des letzten Zugriffs, etc.**
- ***nix Philosphie: Möglichst viele Konzepte auf Dateikonzept abbilden (Virtuelles Dateisystem)**
  - **Alles ist eine Datei. Ist es keine, dann ist es ein Prozess.**

Was ist ein Verzeichnis?

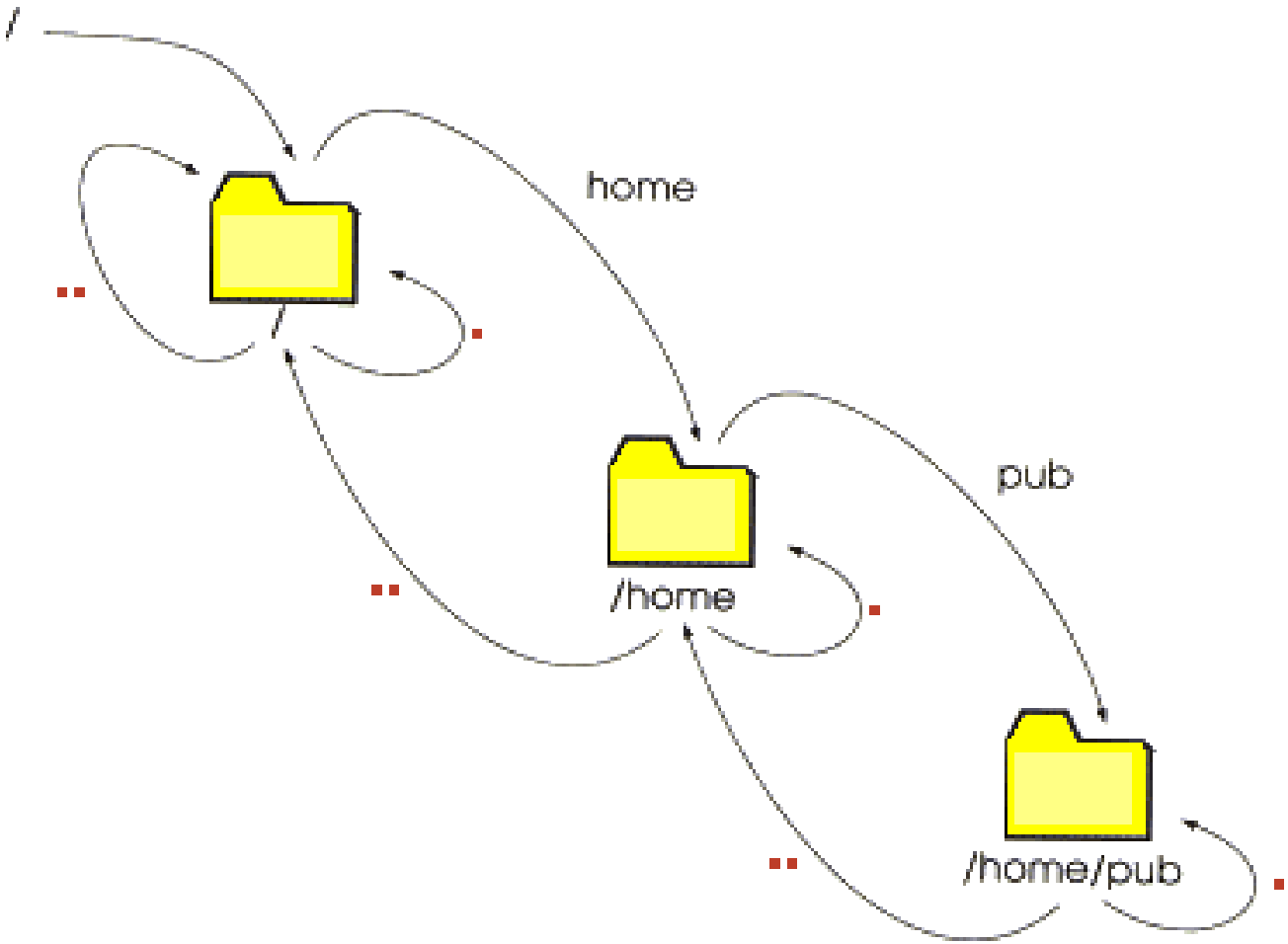
- Zur Strukturierung der Dateien
- Ein Verzeichnis kann Dateien und Unterverzeichnisse beinhalten
- Jedes Verzeichnis hat einen Namen
  - Es entsteht ein Baum
- Hierarchisches Dateisystem vs. Datenbank-basiertes FS



Datei- und Pfadnamen (1)

- Verzeichnisse und Dateinamen durch \ oder / trennen,
  - /Home/Hoe/Lva/Betriebssystem/2006/Lektion3/Test.doc
- Absolute Pfadangaben (Beginn beim Wurzelverzeichnis)
  - C:\Hoe\Lva\Betriebssystem\2006\Lektion3\Test.doc
  - /var/srv/htdocs/Portal/index.html
- Relative Pfadangaben (Beginn beim akt. Verzeichnis)
  - Test.doc, .\Test.doc, ..\Test.doc
  - ../../Test/./../Library.so
    - » Kanonisierung sinnvoll bzw. für Sicherheit oft erforderlich!
    - » Ergebnis: ../../Library.so
- Spezielle Verzeichnisse
  - . Aktuelles Verzeichnis
  - .. Übergeordnetes Verzeichnis

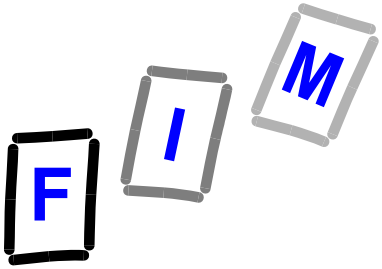
Datei- und Pfadnamen (2)



Dateisystem-Standardisierung bei Linux

http://www.pathname.com/fhs/

- Bei Linux existier(t)en versch. Varianten, das Dateisystem zu organisieren
  - Webserver-Daten unter "/srv/www/htdocs", "/var/www/html/, …
- Schwierig insb. für Programm-Installation!
- Daher: Standard-Layout
  - Beschreibt, wo Dateien liegen sollen
  - Aber nicht:
    - Ob es die Datei gibt
    - Das Format der Datei
- Ähnliches gibt es auch von Microsoft!



Typische Dateioperationen

- **Datei öffnen / anlegen, z.B.: new File**
- **Datei lesen, z.B.: FileInputStream.read**
- **Datei schreiben, z.B.: FileOutputStream.write**
- **Datei positionieren, z.B.: RandomAccessFile.seek**
- **Datei schließen, z.B.: FileInputStream.close**
- **Datei löschen, z.B.: File.delete**
- **Datei leeren (Truncate): RandomAccessFile.setLength(..)**

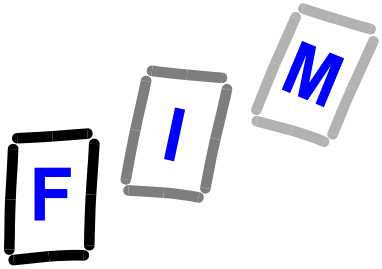
Links in Dateisystemen (1)

- ***Grundsatzüberlegung:*** **Eine Datei / Verzeichnis / Partition existiert physisch ein einziges Mal, soll aber mehrmals an verschiedenen Stellen in der logischen Sicht enthalten sein. Es wird nur eine Referenz („Link“) auf die physische Struktur bzw. auf die Beschreibungsstruktur der physischen Daten gespeichert.**
  - **Eine Dateneinheit unter mehreren Namen (Referenzen)**
- **Windows**

  - **Konzept der „Verknüpfungen“**
    - **Findet nicht auf Dateisystem-Ebene statt, sondern jede Windows-Verknüpfung ist eine separate Datei mit der Extension „LNK“, die vom Explorer beim Doppelklick interpretiert wird. Also kein richtiger Link.**
    - **Unabhängig vom Filesystem (bzw. abhängig von Applikation)!**
  - **NTFS beherrscht auch Links auf Dateisystem-Ebene, welche aber selten verwendet werden. Ganze Laufwerke könnten beispielsweise in ein Unterverzeichnis „gemappt“ werden.**

Links in Dateisystemen (2)

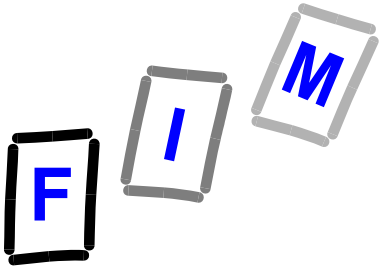
- **UNIX(e)**
  - **Es existiert nur ein globaler Verzeichnisbaum („/“ - root). Mittels dem Vorgang des „Mountens“ werden Partitionen und Laufwerke in diesen Verzeichnisbaum eingebunden.**
  - **Das „Temp“ Verzeichnis kann sich z. B. entweder in „/tmp“ oder auch in „/var/tmp“ befinden. Einige Distributionen definieren „/tmp“ beispielsweise als SymLink auf „/var/tmp“. Somit sind sie zu fast allen Programmen in dieser Hinsicht kompatibel.**
  - **Eine Datei gilt erst dann als gelöscht, wenn alle Links auf diese Datei gelöscht sind (Es wird ein Zähler mitgeführt)**
  - **Ähnlich wie „Verknüpfungen“ unter Windows sind unter UNIX die SymLinks vorhanden.**
    - **Unterschied zu Windows-Verknüpfungen: SymLink wird von Betriebssystem aufgelöst (anders als in Windows, dort muss die Verknüpfung vom Explorer aufgelöst werden)**
    - **Wird die eigentliche Datei gelöscht, ist sie weg, auch wenn SymLinks existieren. Die SymLinks bleiben bestehen und sind ungültig („Dangling Pointer“ Problem)**
  - **SymLink (soft) vs. (Hard)Link**



Journaling (1)

- **Eine „Datei“ ist eigentlich nur ein Abstraktionselement. Sie besteht einerseits aus Meta-Information (Dateiname, in welchem Verzeichnis, Größe, etc.) und den eigentlichen Daten, die eine logische Einheit darstellen, physisch aber meist in mehreren Blöcken (Cluster, etc.) auf der Festplatte verteilt liegen.**
- **D.h. etwa, dass das Erstellen einer Datei aus mehreren Schritten besteht (Anlegen der Metadaten im Directory-Teil, Reservieren der Cluster, etc.). In vielen Filesystemen wird dies in unabhängigen sequentiellen Operationen durchgeführt → Problem der Inkonsistenz, vgl. Datenbanken! („ScanDisk“ zum Beheben solcher Probleme [oft mit Datenverlust verbunden!] wird wahrscheinlich jeden schon einmal die eine oder andere halbe Stunde genervt haben!)**
- ***Lösung*: Eine Operation auf das Dateisystem wird in einer „Transaktion“ ausgeführt. D.h., ein Zugriff wird vollständig durchgeführt oder gar nicht.**

→ Vorteil: Keine Inkonsistenzen, schnelles Wiederherstellen eines konsistenten Zustands beim Neustart des Systems nach vorhergehendem Absturz.



Journaling (2)

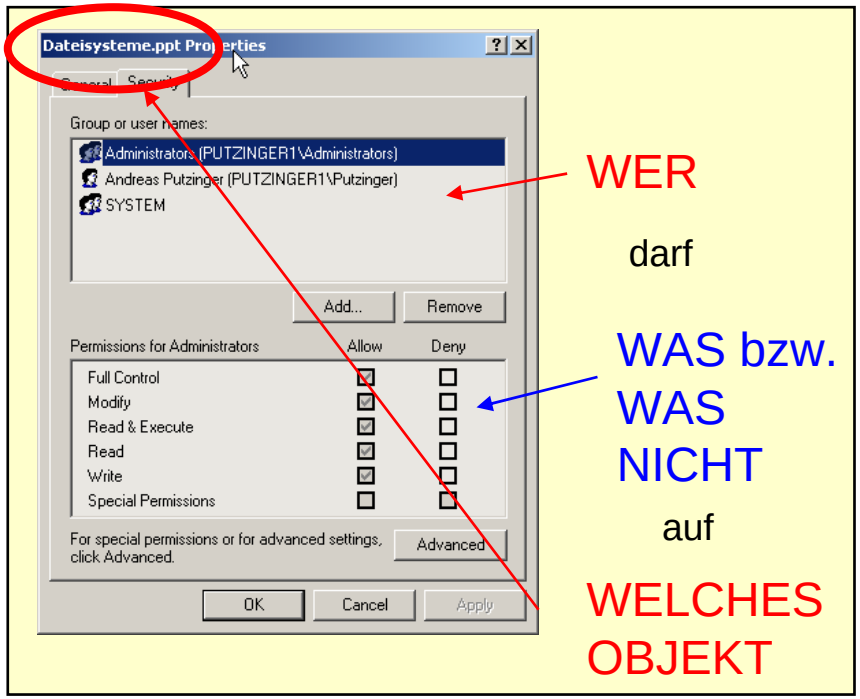
- **Windows**
  - **FAT bis FAT32 unterstützen kein Journaling**
  - **NTFS (alle Versionen) unterstützen Journaling**
    - » **WinFS sowieso, da es auf Datenbank basieren würde!**
- **UNIX(e)**
  - **ext, ext2 haben kein Journaling**
  - **Bei ext3 wird eine Journaling Schicht über ext2 gelegt, die das Journaling durchführt**
    - » **Großer Vorteil der Kompatibilität zu ext2!**
  - **JFS, ReiserFS und XFS unterstützen Journaling schon seit ihrem Beginn**

Berechtigungen (1)
FAT und NTFS

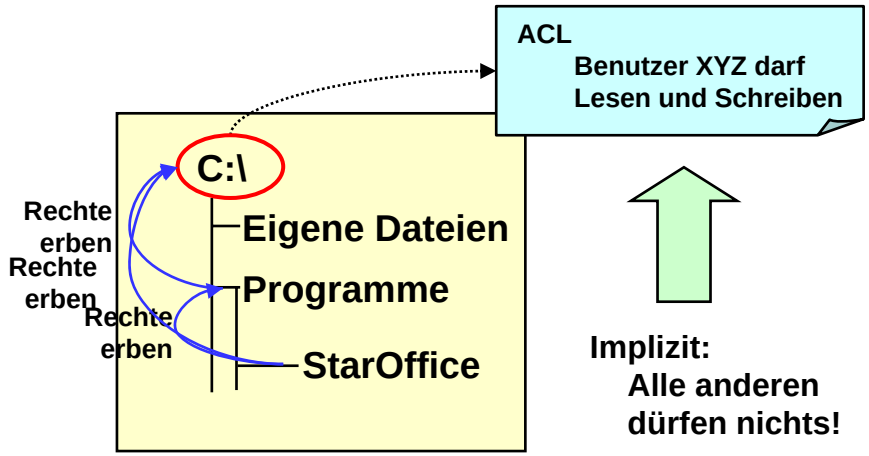
- **Unter DOS wird das Standard-FAT Dateisystem (siehe Skript VL Betriebssysteme) verwendet, welches keinerlei Berechtigungs-vergabe für einzelne Dateien oder Verzeichnisse ermöglicht**
  - **Es gibt nur „globale“ Schalter**
    - **„Schreibgeschützt“, „Archiv“, „Versteckt“, „System“)**
  - **Die von jedem Benutzer geändert werden können!**
- **Seit WinNT wird NTFS (New Technology File System) eingesetzt:**
  - **Feingranulare Berechtigungsvergabe durch Access-Control-Lists (ACL)**
  - **Zu jeder Datei werden Information gespeichert:**
    - **Welcher Benutzer bzw. welche Gruppe welches Recht**
    - **auf die Datei besitzt (grant)**
    - **explizit verweigert wird (deny)**
  - **Rechtevererbung (Ordner auf enthaltene Objekte bzw. Unterordner) spielt hierbei eine wichtige Rolle**

Berechtigungen (2)
NTFS contd.

NTFS Rechte-Dialog



NTFS Rechte-Vererbung

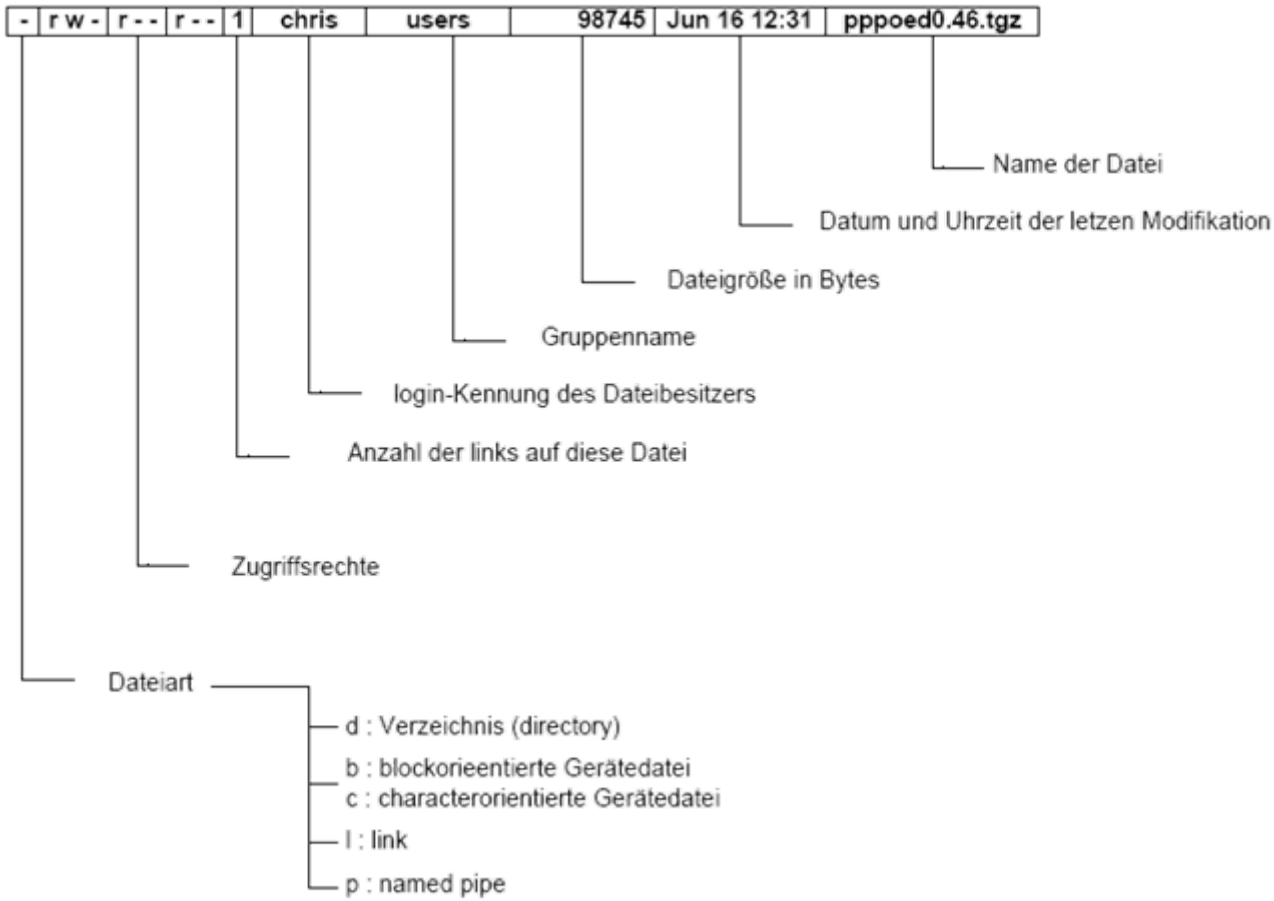


Berechtigungen (3) ext

- **Unix bzw. Linux bietet traditionellerweise ein sehr einfaches Rechtekonzept:**
  - **Es gibt Benutzer und Gruppen**
  - **Jeder Benutzer ist einer Primärgruppe und beliebig vielen Sekundärgruppen zugeordnet**
  - **Es gibt einen speziellen Benutzer („root“), der alle Rechte auf alle (normalen) Dateien hat bzw. sich diese verschaffen kann**
  - **Jede Datei hat einen Besitzer und eine „besitzende Gruppe“**
  - **Es gibt lediglich 3 Rechte: "read", "write" und "execute"**
  - **Eine Kombination dieser 3 Rechte kann getrennt für 3 Personengruppen definiert werden: für Besitzer, für besitzende Gruppe und für alle anderen**
  - **Weiters existieren einige Spezialbits (Ausführen als Besitzer, als besitzende Gruppe, etc.)**

Berechtigungen (4) ext contd.

Befehl: **`ls -la`**



Berechtigungen (5)
ACL für Linux

- **ACLs gibt es inzwischen auch für Linux (bas. auf POSIX 1003.1e)**
  - **Ext2, Ext3, XFS, JFS, ReiserFS**
- **Bekannte Berechtigungen (rwx) können einer Datei zusätzlich für beliebige weitere Benutzer oder Gruppen zugeordnet werden**
  - **Kommandos: getfacl, setfacl**
- **Beispiel:**
  - **"getfacl index.html"**
  - **# file: index.html**
    # owner: root
    # group: apache
    user::rw-
    user:sonntag:rwx
    group::r--
    other::---

Achtung: Dateisystem muss meist erst auf Benutzung von ACLs umgestellt werden (/etc/fstab !)