

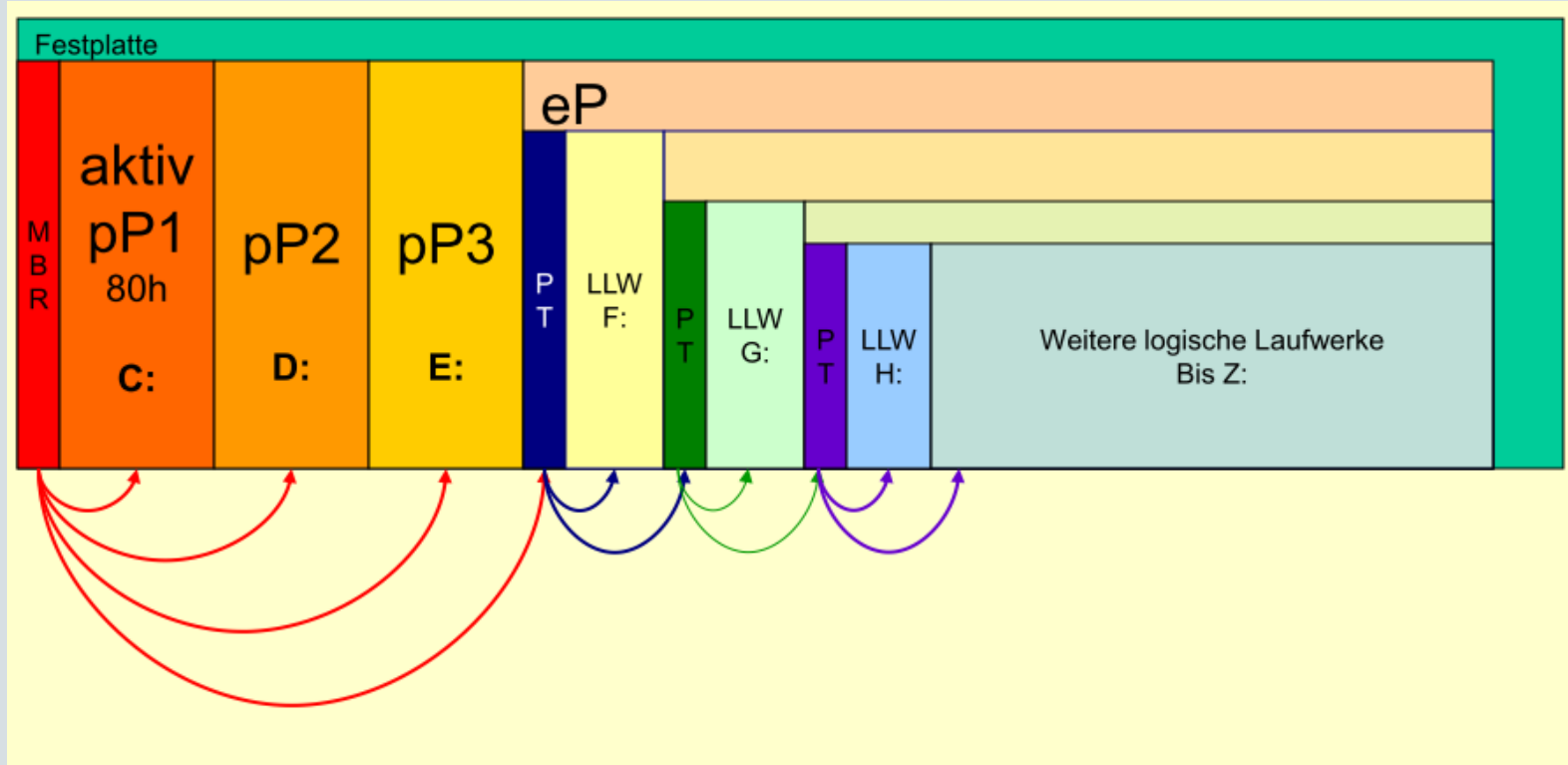
# MBR, GPT, Partitionen, Dateiformate

---

Einführung

Dozent: Dr.-Ing. Reiner Kupferschmidt

# Aufbau MBR



Jedes LLW hat eine Partitionstabelle mit 2 Einträgen:

- das eigentliche LLW
- nächste PT für nächstes LLW

MBR: Masterbootrecord mit Partitionstabelle der Festplatte  
pP#: primäre Partition  
eP: erweiterte Partition  
PT: Partitionstabelle je LLW, Bootsektor  
LLW: Logisches Laufwerk

# Berechnung des Speichervolumens

Dezimal	Binär	Hexadez.	Dezimal	Binär	Hexadez.
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F
8	1000	8	16	1 0000	10

Berechnung der Speicherkapazität:  
C (Zylinder) x H (Köpfe) x S (Sektoren)  
x 512 Byte = Speichervolumen  
(ODER Anzahl der Blöcke x 512 Byte)

## Beachte!

1 kiByte = 1 024 Byte (realer Speicherplatz)  
1 kByte = 1 000 Byte (Markt)

# Sektoren und Cluster

- Die Einteilung der Festplatte in Zylinder (Spuren), Köpfe (Seiten), Sektoren ist physisch
- Das Zusammenfassen von Sektoren zu **Clustern** ist logisch und erforderlich wegen der begrenzten Adressbits.
- Es sind Clustergrößen von einem Sektor (512 Byte) bis 64 Sektoren (32 kByte) möglich
- Das verwendete Dateisystem/Betriebssystem bestimmt die Clustergröße

# Berechnung Clustergröße

→ Adressbit bei den Dateisystemen

→ FAT12 12 bit  $2^{12} = 4.096$

→ FAT16 16 bit  $2^{16} = 65.536$

→ FAT32 32 bit  $2^{32} = 4.294.967.296$

→ NTFS 64 bit  $2^{64} = 18.446.744.073.709.551.616$

$\text{Clustergröße}_{\min} = \text{Partitionsgröße} / 2^{\text{Anzahl der Adressbit}}$

$\text{Sektoren pro Cluster} = \text{Clustergröße} / 512 \text{ Byte}$

# Clustergrößen FAT 16

Partitionsgröße	Sektoren/Cluster	Clustergröße
Bis 32 Mbyte	1	512 byte
Bis 64 Mbyte	2	1 kbyte
Bis 128 Mbyte	4	2 kbyte
Bis 256 Mbyte	8	4 kbyte
Bis 512 Mbyte	16	8 kbyte
Bis 1024 Mbyte	32	16 kbyte
Bis 2048 Mbyte	64	32 kbyte
Bis 4096 Mbyte	128	64 kbyte

# Clustergrößen FAT 32

Partitionsgröße	Sektoren/Cluster	Clustergröße
512 Mbyte bis 1	1	512 byte
Bis 2 Gbyte	2	1 kbyte
Bis 4 Gbyte	4	2 kbyte
Bis 8 Gbyte	8	4 kbyte
Bis 16 Gbyte	16	8 kbyte
Bis 32 Gbyte	32	16 kbyte
> 32 Gbyte	64	32 kbyte

# Clustergrößen NTFS

Partitionsgröße	Sektoren/Cluster	Clustergröße
512 Mbyte bis 1	1	512 byte
Bis 1 Gbyte	2	1 kbyte
Bis 2 Gbyte	4	2 kbyte
Bis 4 Gbyte	8	4 kbyte
Bis 8 Gbyte	16	8 kbyte
Bis 16 Gbyte	32	16 kbyte
Bis 32 Gbyte	64	32 kbyte
> 32 Gbyte	128	64 kbyte



# Betriebssysteme und Dateisysteme

Betriebssystem	Unterstützte Dateisysteme
DOS	FAT16
Windows 3.x	FAT16
Windows95	VFAT (16bit und lange Dateinamen)
Windows95B	VFAT und FAT32
Windows98	VFAT, FAT32
Windows NT	FAT16, VFAT, NTFSv4
Windows 2000	FAT16, VFAT, NTFSv5
WindowsXP, Windows2003	FAT16, VFAT, NTFSv5
OS/2	FAT16, HPFS
Linux	FAT16, FAT32, NTFS, extfs2, extfs3, riserfs und andere

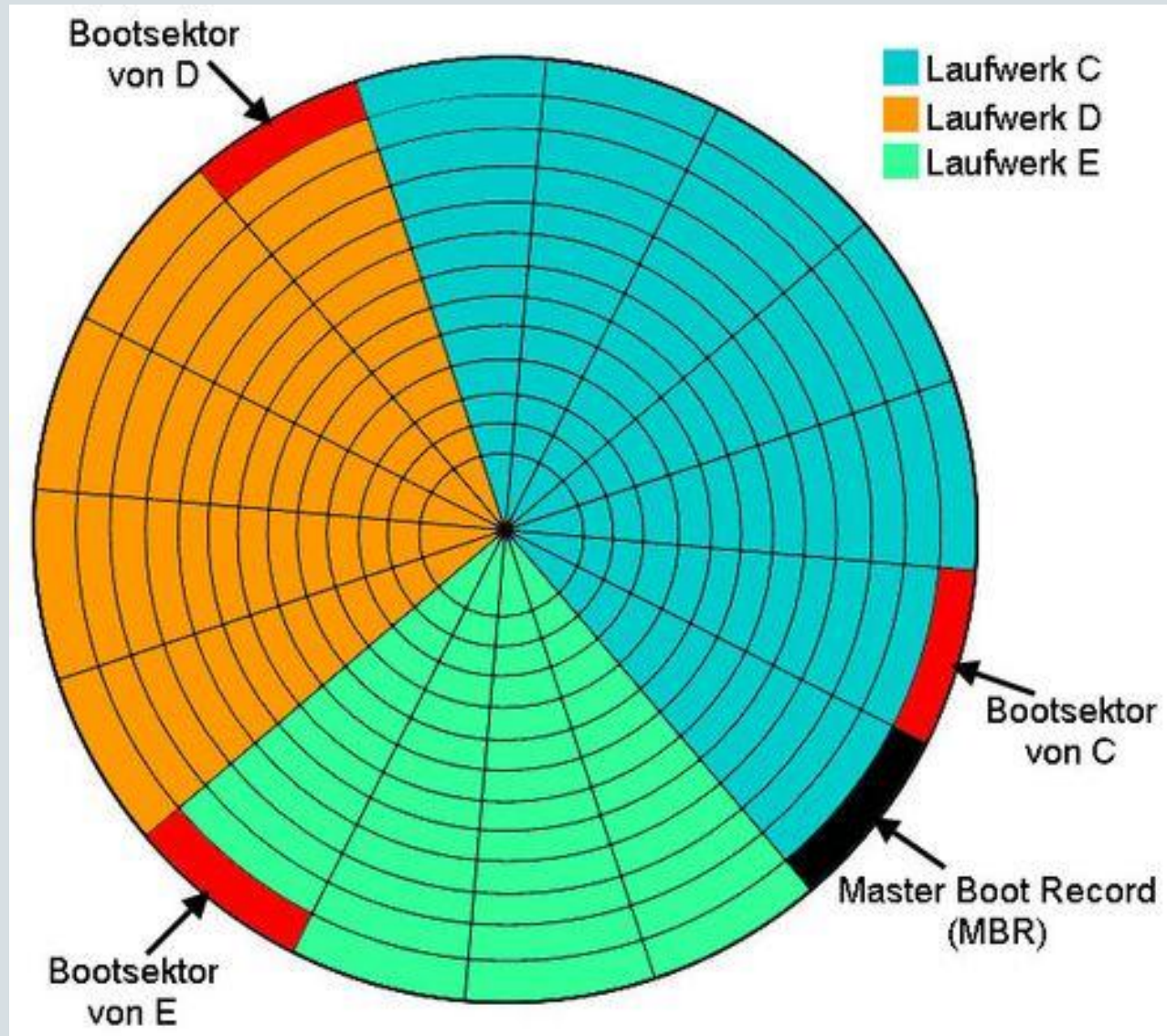
# Optimale Clustergröße?

- Cluster ist die kleinste logische Speichereinheit
- Dateien, die kleiner als ein Cluster sind benötigen immer einen ganzen Cluster, dabei kann Speicherplatz ungenutzt bleiben
- Große Dateien haben nur im letzten Cluster etwas Verlust
  - Viele Große Dateien – große Cluster
  - Viele kleine Dateien – besser kleine Cluster

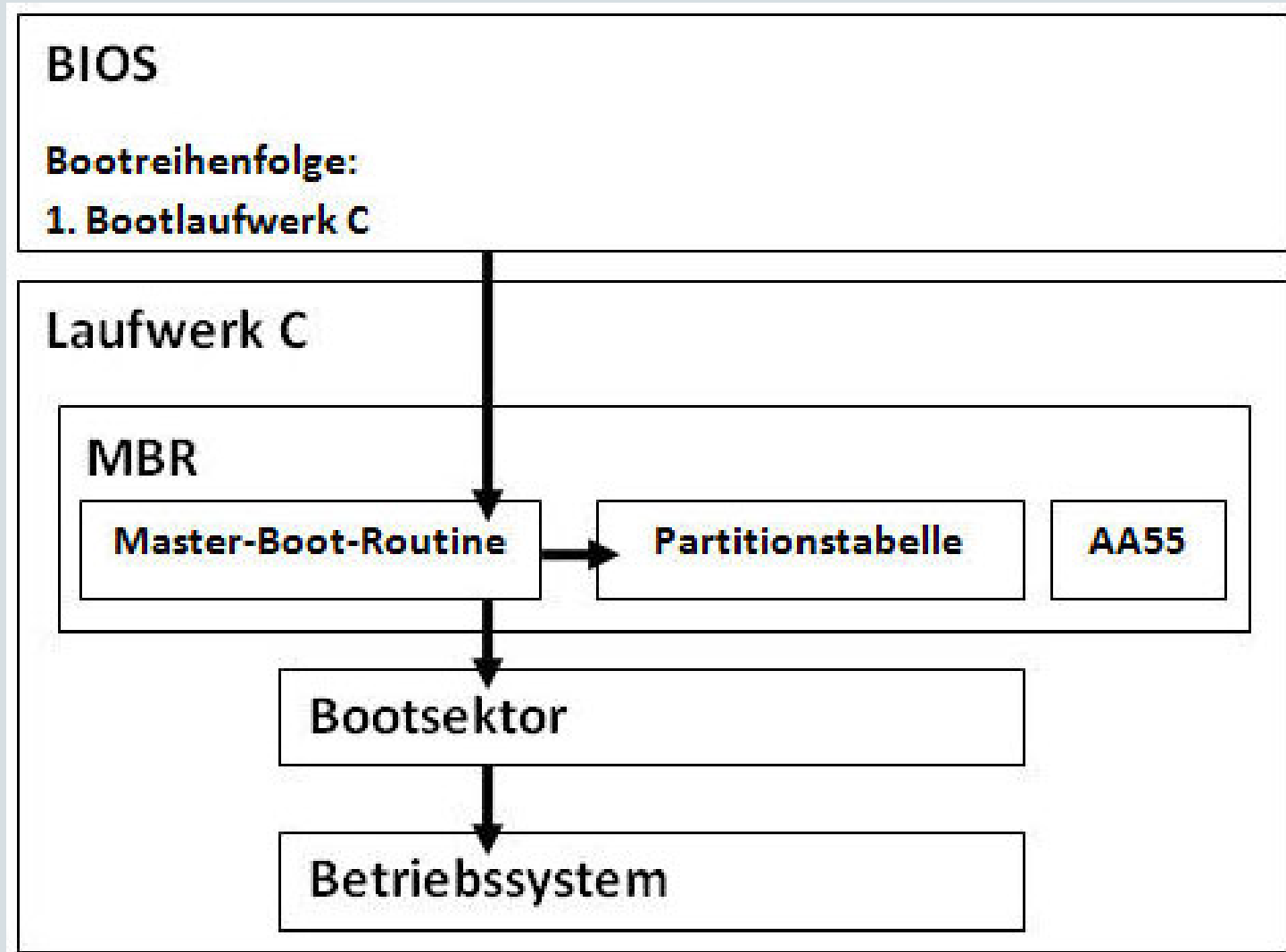
# Ablauf nach dem Einschalten

- BIOS wird angesprochen
- POST und danach BIOS in RAM unter FFFE0h laden
- IRQ 13h (Software-Interrupt) löst Bootvorgang aus:
  - Suche A: auf Seite 0, Spur 0, Sektor 1 einen MBR und lade diesen auf RAM 07C00h (nur DOS)
  - Oder (je nach BIOS-Einstellung Bootgerätefolge)  
Suche C: auf Kopf 0, Spur 0, Sektor 1 einen MBR und lade diesen auf RAM 07C00h (nur DOS)
- Führe MBR-Code aus. Es wird die Partitionstabelle gelesen.
- Erkenne aktive primäre Partition und suche dort im Sektor 1 nach Bootsektor der Partition
- Führe Bootsektorcode aus, lese FAT/MFT und lade Startdateien

# Bereiche der Festplatte



# Masterboot-Routine



# MBR – Master Boot Record

- ➡ MBR enthält
  - ➡ Ausführbaren Code (liest PT)  
(ab 000h bis 1BDh: 446 Byte gesamt)
  - ➡ Partitionstabelle 64 Byte (ab 1BEh bis 1FDh –  
4x16Byte, 1FEh=55, 1FFh=AA)
- ➡ Partitionstabelle immer 64 Byte lang (x86)  
je 16 Byte pro Partition, also max. 4 Partitionen möglich
- ➡ Davon max. 4 primäre Partitionen (haben Bootsektor)  
oder  
3 primäre Partitionen und max. 1 erweiterte Partition
- ➡ (kann viele logische Laufwerke enthalten – freie Buchstaben,  
normalerweise nicht bootfähig)
- ➡ Es darf nur eine primäre Partition aktiv sein
- ➡ MBR reparieren mit fdisk des jeweiligen Betriebssystems

# MBR (Sektor 1: 00h bis 1FEh)

## ***Einträge Master Boot Record***

Adresse	Inhalt	Größe
+00h	Master Boot-Routine (beim MBR, denn Windows 98 schreibt nur 139 Byte Code, 80 Byte sind für Fehlermeldungstext und 227 Byte bleiben frei)	446 Byte
+1BEh	1 Eintrag der Partitionstabelle	16 Byte
+1CEh	1 Eintrag der Partitionstabelle	16 Byte
+1DEh	1 Eintrag der Partitionstabelle	16 Byte
+1EEh	1 Eintrag der Partitionstabelle	16 Byte
+1FEh	Erkennungscode des MBR <b>55AAh</b>	2 Byte



# Partitionstabelleninhalt je Partition (CHS)

- 8 bit Boot-Indikator (80h=aktiv)
  - 8 bit System-ID (Dateisystem)
- Max. Kapazität der Partition  
7,875 Gbyte (BIOS-CHS)*

- 8 bit erster Kopf
- 6 bit erster Sektor
- 10 bit erster Zylinder

- 8 bit letzter Kopf
- 6 bit letzter Sektor
- 10 bit letzter Zylinder

*Max. Kapazität der Partition  
theoretisch 2048 GByte*

- 32 bit relativer Sektor
- 32 bit Sektorenzahl der Partition



# Partitionstabelle

## **Einträge Partitionstabelle (16 Byte)**

Adresse	Inhalt	Größe
+00h	Status der Partition: 00h=inaktiv 80h=Boot-Partition (aktiv)	1Byte
+01h	Schreib-/Lesekopf, mit dem die Partition beginnt	1Byte
+02h	Sektor und Zylinder, mit dem die Partition beginnt	1Word
+04h	Partitionstyp: 00h=Eintrag nicht belgt 01h=Primäre DOS-Partiton mit 12-Bit-FAT 04h=Primäre DOS-Partiton mit 16-Bit-FAT 05h=Erweiterte Partiton etc.	1Byte
+05h	Schreib-/Lesekopf, mit dem die Partiton endet	1Byte
+06h	Sektor und Zylinder, mit dem die Partiton endet	1Word
+08h	Entfernung des ersten Sektors der Partition (Boot-Sektor) vom Partitonssektor in Sektoren	1DWord
+0Ch	Anzahl Sektoren der Partition	1DWord

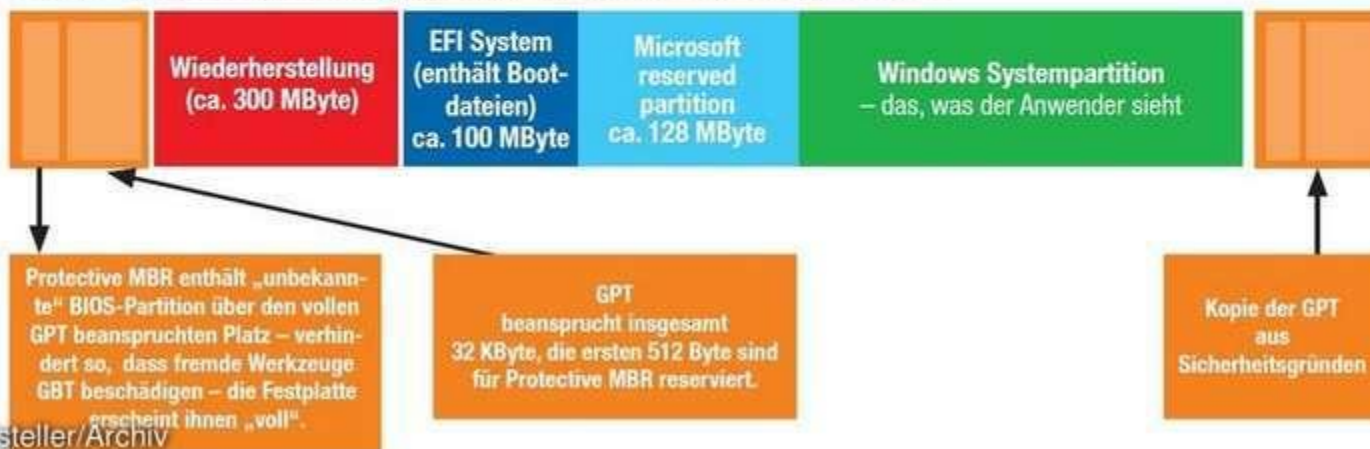


### Das Partitionsschema der Festplatte bei Rechnern mit BIOS und MBR

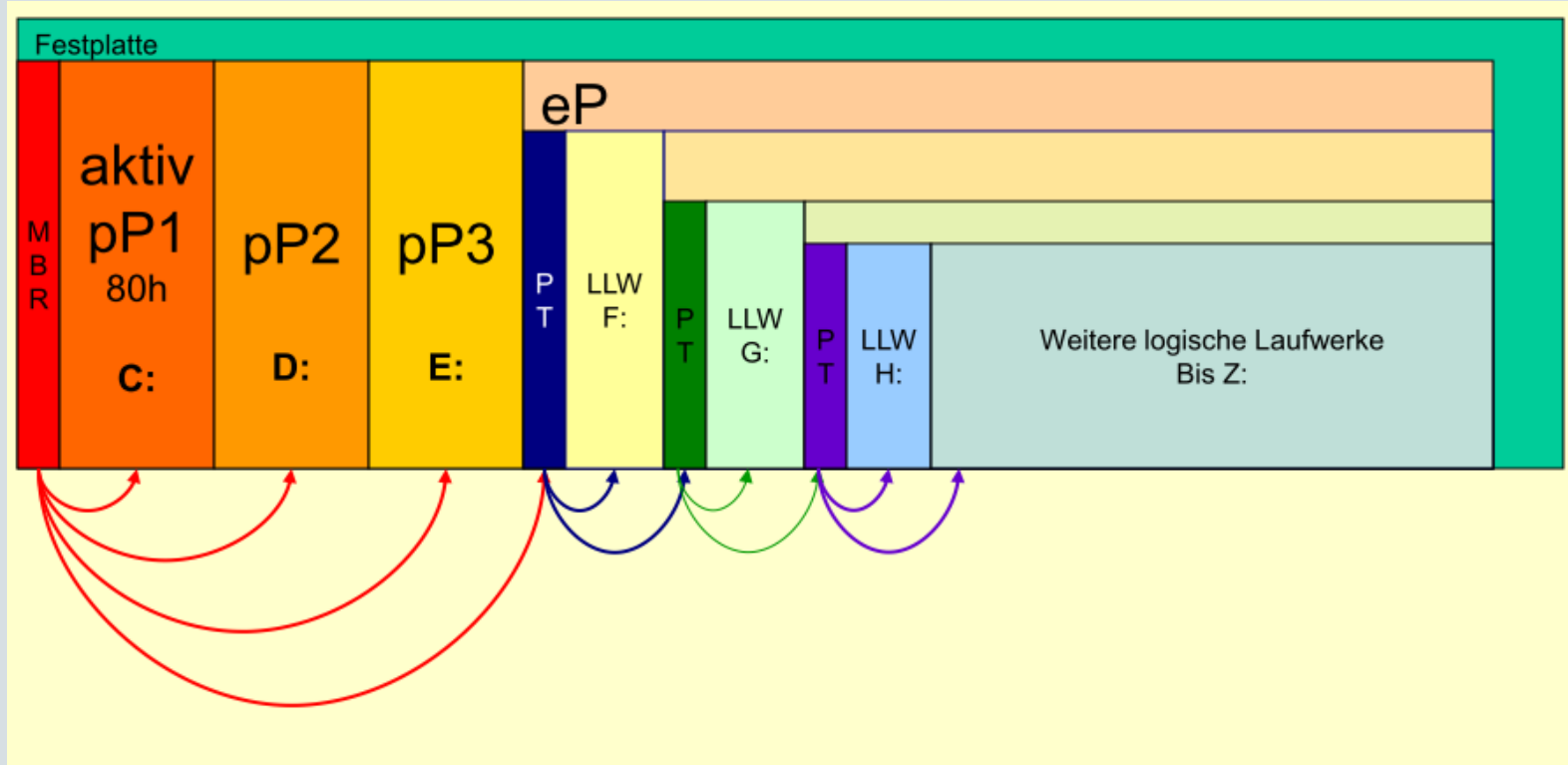


Der Klassiker: MBR formatierte Medien haben nur vier primäre Partitionen, weniger als zwei Terabyte können angesprochen werden – entsprechend einfach sieht das Partitionslayout aus.

### Das Partitionsschema der Festplatte bei Rechnern mit UEFI und GPT



# Aufbau MBR



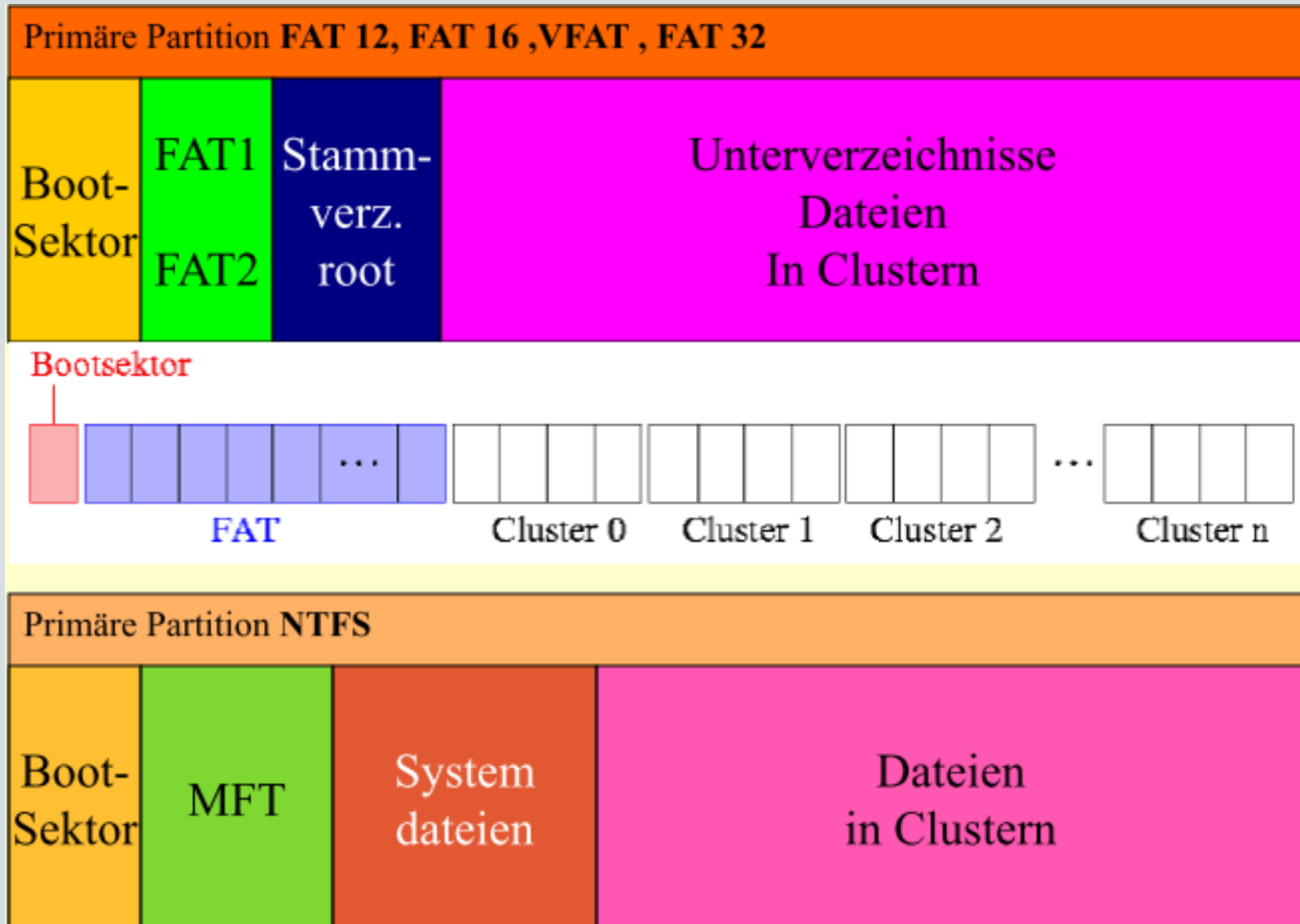
Jedes LLW hat eine Partitionstabelle mit 2 Einträgen:

- das eigentliche LLW
- nächste PT für nächstes LLW

MBR: Masterbootrecord mit Partitionstabelle der Festplatte  
 pP#: primäre Partition  
 eP: erweiterte Partition  
 PT: Partitionstabelle je LLW, Bootsektor  
 LLW: Logisches Laufwerk

# Dateisysteme –

## Die Verwaltung der Partition zur Datenspeicherung

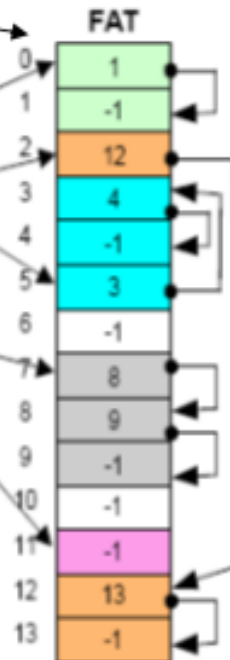


## Partitionstabelle

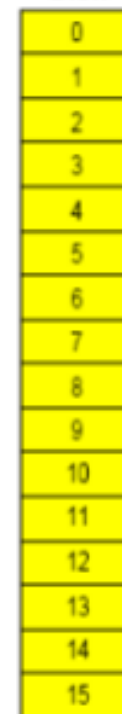


## Verzeichnis

Dateiname	Attribute	Cluster
test.txt	R/W	5
Beispiel.doc	Read-Only	0
Programm.exe	R/X	2
Vorlesung.doc	R/W	11
Klausur.doc	Hidden	7



## Disk



# GPT – GUID (Globally Unique Identifier) Partition Table

- Ein Standard zur Formatierung von Partitionstabellen
- Bestandteil von UEFI
- GPT-Partitionen lassen sich (mit Einschränkungen) auch unabhängig von UEFI nutzen
- OS und Datenträger müssen GPT unterstützen
- Wird MBR ablösen

# GPT - Vorteile

- Adressierung mit 64 bit – maximale Größe einer Partition liegt bei 18 ExaByte ( $18 \times 10^{18}$  Byte)
- Kein Limit für die Anzahl primärer Partitionen (128)
- Absicherung durch CRC32-Prüfsummen
- Eindeutige Identifikation von Partitionen und Datenträgern
- Backup-Header
- Abwärtskompatibilität
  
- Erhöhter Schutz gegen Datenverlust bei HW-Defekten
- Einsatz bei portablen Speichermedien



# GPT - Schema

- **Protective Master Boot Record:** An erster Stelle steht der bereits erwähnte Protective-MBR, der für die Abwärtskompatibilität des Partitionierungsstils sorgt.
- **Primäre GUID-Partitionstabelle:** GPT-Header und Partitionseinträge
- **Partitionen:** Auf den Header und die Partitionseinträge folgen die jeweiligen Einheiten des aufgeteilten Speicherplatzes, also die verschiedenen Partitionen.
- **Sekundäre GUID-Partitionstabelle:** Backup von GPT-Header und Partitionseinträgen in gespiegelter Reihenfolge

# GPT – Schema 2

LBA 0	<b>Protective Master Boot Record</b>			
LBA 1	<b>Primärer GPT-Header</b>			
LBA 2	Partitionseintrag 1	Partitionseintrag 2	Partitionseintrag 3	Partitionseintrag 4
LBA 3 bis 33	Partitionseinträge 5 - 128			
LBA 34	Partition 1			
	Partition 2			
	weitere Partitionen			
LBA -34	Partitionseintrag 1	Partitionseintrag 2	Partitionseintrag 3	Partitionseintrag 4
LBA -33 bis -2	Partitionseinträge 5 - 128			
LBA -1	<b>Sekundärer GPT-Header</b>			

➔ **LBA-Blöcke** (Logical Block Addressing) => Sektor des Datenträgers **512 Byte**

# GPT – Header – LBA 1 (zweiter Sektor)

- Folgt auf protective MBR – Backup im letzten Sektor (LBA -1) – durch Prüfsumme geschützt - Position im Header gespeichert (92 Byte – Rest wird mit 0 aufgefüllt)

GUID Partition Table Header / GPT-Header		
Startbyte	Byte	Inhalt
0	8	Signatur („EFI PART“)
8	4	Revisionsnummer (Auskunft über GPT-Version)
12	4	Größe des Headers in Byte (Standardwert: 92)
16	4	CRC32-Prüfsumme des Headers
20	4	reservierter Bereich (muss den Wert „0“ haben)
24	8	Position des Headers (gegenwärtiger LBA-Block; LBA 1)
32	8	Position des Backup-Headers (LBA -1)
40	8	Angabe des ersten, für GTP-Partitionen nutzbaren LBA-Blocks
48	8	Angabe des letzten, für GTP-Partitionen nutzbaren LBA-Blocks
56	16	GUID zur eindeutigen Identifikation des Datenträgers
72	8	Angabe des Start-LBA-Blocks der Partitionseinträge
80	4	Anzahl der Partitionseinträge (Partitionen)
84	4	Größe eines einzelnen Partitionseintrags (Standard: 128 Byte)
88	4	CRC32-Prüfsumme der Partitionseinträge
92	420+	reservierter Bereich, der mit Nullen aufgefüllt wird (420 Byte bei der Standard-Sektorgröße von 512 Byte, bei größeren Sektoren entsprechend größer)

# GPT - Partitionseintrag

- Partitionseinträge folgen dem Header – 128 Byte pro Eintrag (4 Einträge pro log Block (512 Byte)), GUID-Partition-Table-Standard Blöcke 2 bis 33 – 128 Partitionen
- Bei Bedarf lässt sich die Zahl an freigegebenen Sektoren beliebig erhöhen

GPT-Partitionseintrag		
Startbyte	Byte	Inhalt
0	16	Partitionstyp-GUID (eindeutige ID, die den Partitionszweck beschreibt)
16	16	Partitions-GUID (eindeutige ID der Partition)
32	8	Angabe des Startblocks (LBA) der Partition
40	8	Angabe des Endblocks (LBA) der Partition
48	8	Attribute (z. B. „Systempartition“, „nur lesen“ oder „versteckt“)
56	72	Name der Partition (36 UTF-16LE-Zeichen)
128		

# Dateisysteme

- Zugriff über Pfadnamen
  - Namen aller Verzeichnisse auf dem Pfad von der Wurzel bis zur Datei, durch spezielles Trennzeichen separiert
  - Unix/Linux: `/home/ernie/oscar.jpg`
  - Windows: `C:\home\ernie\oscar.jpg`
- typische Operationen auf Verzeichnissen:
  - **Erzeugen** (mkdir)
  - **Löschen** (rmdir), i.a. nur von leeren Verzeichnissen möglich
  - **Öffnen** (opendir) und **Schließen** (closedir)
  - **Lesen** eines **Verzeichniseintrags** (readdir)
  - Schreiben eines neuen Verzeichniseintrags erfolgt implizit beim Erzeugen einer neuen Datei
  - Löschen eines Verzeichniseintrags erfolgt i.a. implizit beim Löschen einer Datei bzw. eines Unterverzeichnisses

# Stammverzeichnis FAT16

8	3	1	10	2	2	2	4
Name	Extension	Attribut	Ungenutzt	Zeit	Datum	Cluster	Dateigröße

- Im Stammverzeichnis max. 512 Einträge
- Jeder Eintrag 32 Byte lang
- Stammverzeichnis = 16 kByte
- Dateinamen 8.3 Zeichen
- At. = Attribute-Flags: Lesen (r), Archiv (a), Hidden(h), System (s)

- Attribute Flags
- Bit 6/7: res.;
- bit 5: Archive Flag (a);
- bit 4: Directory Flag;
- bit 3: Volume Label Flag;
- bit 2: System Flag (s);
- bit 1: Hidden Flag (h);
- bit 0: Read Only Flag (r)

# Verzeichnisaufbau FAT16

- Verkettung der Cluster wird in der FAT festgehalten:
  - Enthält Eintrag für jedes Cluster auf der Festplatte
  - Für jedes Cluster einer Datei ist die Nummer des nachfolgenden Clusters als 32-bit Zahl eingetragen
    - Die Nummer des Startblockes kann dem Verzeichnis entnommen werden
    - Dateiende wird durch den Eintrag -1 (EOF) markiert
- Freie Cluster werden durch Eintrag 0 markiert

Auszug aus einer FAT																							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	...	
-	-	0	8	5	6	20	0	9	15	11	17	0	0	0	16	18	4	-1	0	-1	0	...	
Plattenblöcke (Cluster) für Datei A																							
10			11			17			4			5			6			20					
Plattenblöcke (Cluster) für Datei B																							
3			8			9			15			16			18								

# Verzeichnisaufbau FAT32 - V(olume) FAT


8 Byte	3 Byte	1 Byte	10	2	2	2	4
Name	Ext.	Attribute	ungenutzt	Zeit	Datum	Cluster	Dateigröße

- ➔ Ermöglicht lange Dateinamen
- ➔ Durch Nutzung der freien Bit aus dem Attribute-Byte (Bit 6 und 7) und
- ➔ Für lange Dateinamen bis 255 Byte/Datei werden mehrere Dateieinträge umgesetzt
- ➔ Teilweise kompatibel zu FAT16



# Verzeichnisaufbau FAT32 - ab Windows 98

8 Byte	3 Byte	1 Byte	10 Byte	2 Byte	2 Byte	2 Byte	4 Byte
Name	Ext.	Attribute	Extra genutzt von FAT32	Zeit	Datum	Cluster	Dateigröße



1	1	2	2	2	2
F	S	T	d <sub>1</sub>	d <sub>2</sub>	Cluster obere 16 bit
Formatkennzeichnung	Sekundendauer beim Schreiben	Erstellungszeit	Erstellungsdatum	Datum letzter Zugriff	

# Aufbau FAT32-Verzeichnis

- Dateisystem für Windows 98
- einige Unterschiede zum Linux-Dateisystem EXT2:
  - **keine Benutzeridentifikation** für Dateien und Verzeichnisse!
  - Partitionen werden durch Laufwerke repräsentiert, die durch Buchstaben dargestellt werden, z. B.: A: (Floppy), C: (Platte), D: (DVD)
  - jedem Windows-Programm ist ein **aktuelles Laufwerk** und ein **aktuelles Verzeichnis** aus Dateibaum zugeordnet
  - es gibt keine Inodes: die Speicherung aller Attribute einer Datei erfolgt im Verzeichnis
  - es gibt keine Hard Links
  - die kleinste adressierbare Einheit heißt **Cluster** und ist ein Block mit einer Zweierpotenz von 1 bis 128 Sektoren (bei Formatierung wählbar)
  - die Blockadressierung erfolgt über eine Tabelle (**FAT** = File Allocation Table), in der die Verkettung der Cluster aller Dateien gespeichert ist

# FAT32 Dateisystem 2

## → Attribute einer FAT32-Datei:

### → Name

- im MS-DOS Modus: 8 Zeichen Name + 3 Zeichen Erweiterung (z. B. „**AUTOEXEC.BAT**“)
- im Windows 98 Modus: 255 Zeichen inklusive Sonderzeichen (z. B. „**Eigene Dateien**“)

### → Dateilänge

### → Typ: Verzeichnis, versteckte Datei (hidden), Systemdatei (system),

### → zu archivierende Datei (archive)

### → nur zwei Zugriffsrechte: „nur lesbar“ und „schreib- und lesbar“

### → Ortsinformation: Nummer des ersten Clusters einer Datei

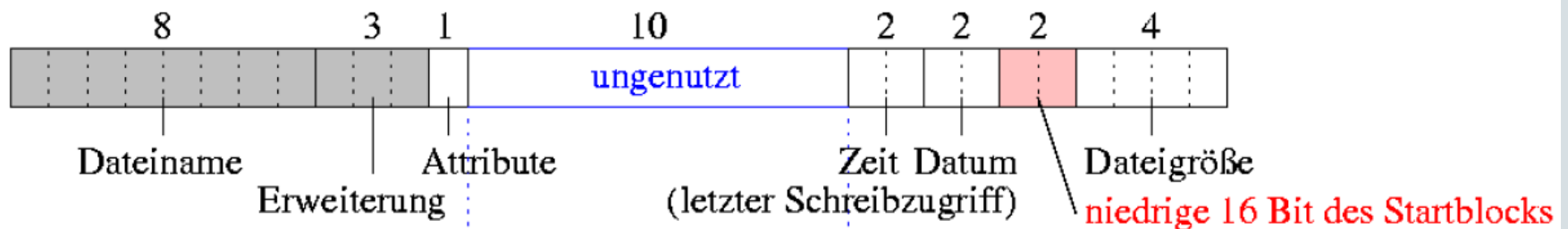
### → Zeitstempel:

- zunächst nur Datum und Uhrzeit des letzten Schreibzugriffs
- bei Windows 98 zusätzlich Datum und Uhrzeit der Erstellung, sowie
- Datum des letzten Lesezugriffs

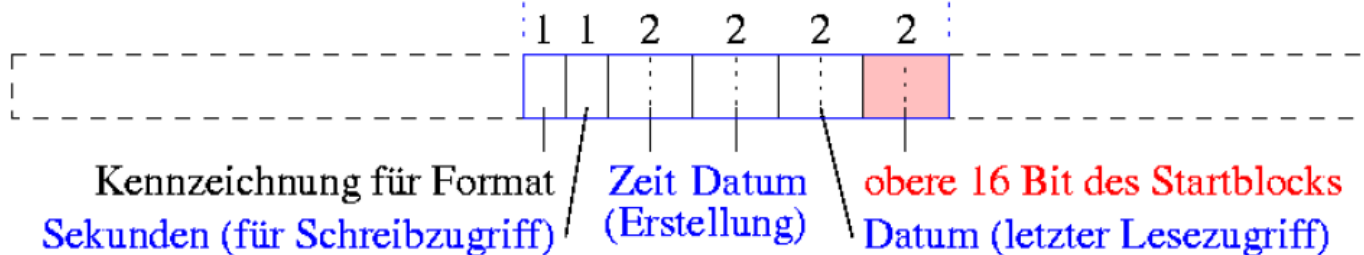
# FAT32 Dateisystem 3 - Aufbau

- unsortierte 32-Byte Einträge werden hintereinander in Liste gespeichert
- aus langen Dateinamen (bis zu 255 Zeichen) wird ein neuer eindeutiger Name aus 8+3 Zeichen generiert und eingetragen; der vollständige Name wird in zusätzlichen vorangestellten 32-Byte Feldern gespeichert

ursprünglicher Eintrag (MS-DOS mit FAT16):



erweiterter Eintrag (Windows 98 mit FAT32):



# FAT32 Dateisystem 4

- die Verkettung der Cluster wird in der FAT festgehalten:
  - enthält Eintrag für jedes Cluster auf der Festplatte
  - für jedes Cluster einer Datei ist die Nummer des nachfolgenden Clusters als 32-Bit Zahl eingetragen
    - die Nummer des Startblocks kann dem Verzeichnis entnommen werden
    - Dateiende wird durch Eintrag -1 markiert
  - freie Cluster werden durch Eintrag 0 markiert

Auszug aus einer FAT:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	...
-	-	0	8	5	6	20	0	9	15	11	17	0	0	0	16	18	4	-1	0	-1	0	...

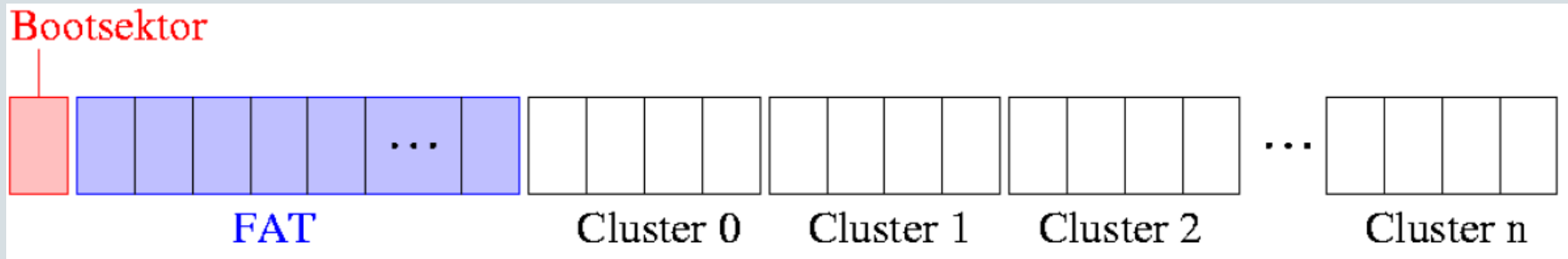
Plattenblöcke (Cluster) für Datei A:

10	11	17	4	5	6	20
----	----	----	---	---	---	----

Plattenblöcke (Cluster) für Datei B:

3	8	9	15	16	18
---	---	---	----	----	----

# FAT32 Dateisystem 5 - Blockorganisation



- **Bootsektor** enthält neben dem Bootloader noch einige Angaben über das Dateisystem, z. B.:
  - Gesamtanzahl der Sektoren (4 Byte)
  - Bytes je Sektor (2 Byte)
  - Sektoren je Cluster (1 Byte, nur Zweierpotenzen von 1 bis 128 erlaubt)
  - Startposition des Hauptverzeichnisses (4 Byte)
  - Label (10 Byte) und Serien-Nummer (4 Byte)
  - Anzahl FATs (1 Byte) und Sektoren je FAT (4 Byte)
- FAT kann zur Erhöhung der Sicherheit auch **mehrfach** auf Festplatte gespeichert sein

# FAT32 Dateisystem 6 - Nachteile

- umständliche Datenstrukturen (wegen Kompatibilität zu MS-DOS)
- sehr große FAT bei modernen Festplatten hoher Kapazität
- Positionieren eines Dateizeigers bei großen Dateien sehr zeitaufwendig
- für jeden Dateizugriff muss mindestens ein Plattenblock mit einem Teil der FAT von der Festplatte geladen werden
- FAT enthält Verkettungen für alle Dateien => es werden stets auch viele nicht benötigte Verkettungsinformationen geladen
- langsame Suche nach freien Clustern
- sehr viele Kopfbewegungen, wenn Cluster einer Datei verstreut sind
- (=> regelmäßiger Aufruf eines Defragmentierungsprogramms sinnvoll; es versucht die Cluster jeder Datei zusammenhängend anzuordnen)
- FAT32 wird nicht mehr weiterentwickelt!

# NTFS Dateisystem

- Dateisystem für Windows NT
- einige Unterschiede zum FAT32 Dateisystem:
  - Unterstützung mehrerer Benutzer und Gruppen mit umfangreichen Zugriffsrechten
  - jede Partition wird als Volume bezeichnet und besteht aus einer linearen Sequenz von Clustern (mit z. Zt. 512, 1024, 2048 oder 4096 Byte)
  - Adressierung eines Clusters erfolgt über 64-Bit Cluster-Nummern (=> sehr große Dateien möglich)
  - zentrales Element der Dateiorganisation ist die **Master File Table (MFT)**, die für jede Datei einen Eintrag enthält
  - Unterstützung von Hard Links
  - Dateien können automatisch komprimiert abgespeichert werden
  - Konsistenzüberprüfung nötig



# NTFS Dateisystem - 2

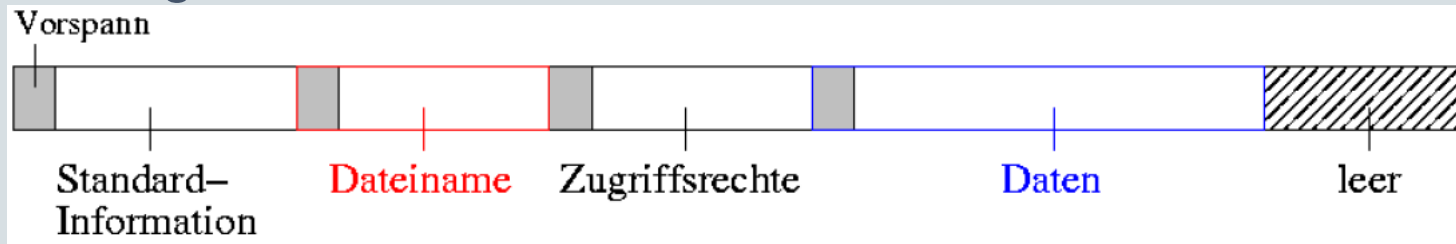
- ➔ Zugriffsrechte einer NTFS-Datei:
  - ➔ no access : kein Zugriff (---)
  - ➔ list : Anzeige von Verzeichnisinhalt erlaubt (r-- )
  - ➔ read : Lesen und Ausführen von Dateien erlaubt (rw-)
  - ➔ add : Hinzufügen von Einträgen in einem Verzeichnis erlaubt (-wx)
  - ➔ change : Ändern und Löschen von Dateien erlaubt (rwx)
  - ➔ full : zusätzlich Ändern von Eigentümer und Zugriffsrechten erlaubt
- ➔ jede Datei wird eindeutig durch eine 64-Bit Dateireferenz (File reference) bezeichnet; sie besteht aus:
  - ➔ 48-Bit Dateinummer (File ID), die einen eindeutigen Index in der MFT darstellt
  - ➔ 16-Bit Folgenummer (Sequence ID), die bei jeder Wiederverwendung der Dateinummer hochgezählt wird

# NTFS Dateisystem - 3

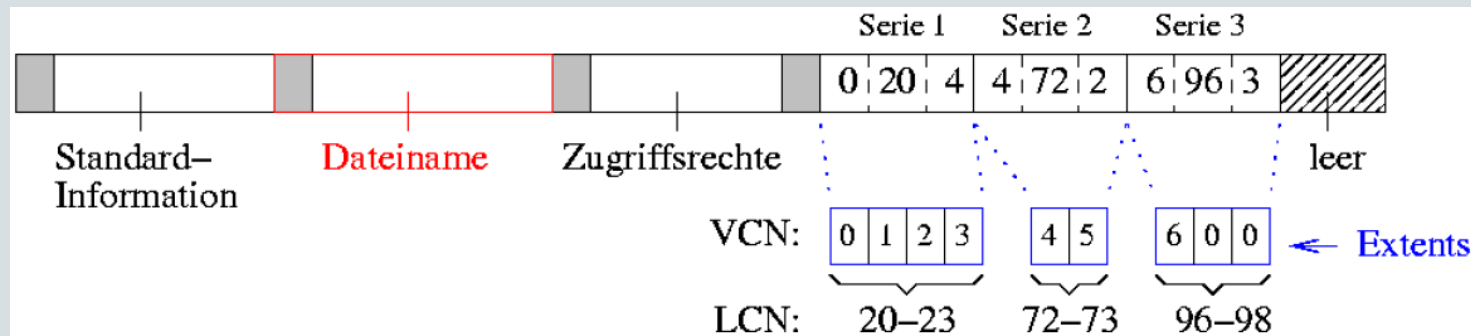
- jede Datei besteht aus mehreren Strömen, z.B.:
  - Standard-Information (zu MS-DOS kompatibler Dateiname sowie klassische MS-DOS Attribute wie Dateilänge, Zeitstempel, Typ, ...)
  - Dateiname (in Unicode mit 16-Bit Zeichen)
  - Dateireferenz (64-Bit Wert)
  - Sicherheits-Beschreibung (enthält Eigentümer und Zugriffsrechte)
  - eigentliche Daten
- Dateiorganisation erfolgt mittels Master File Table (MFT):
  - enthält für jede Datei genau einen Eintrag
  - Größe jedes Eintrags entspricht der Cluster-Größe
  - Index in Tabelle wird durch die Datei-Nummer festgelegt
  - Eintrag in Bootsektor verweist auf Beginn der MFT
  - ein Eintrag besteht aus Hintereinanderreihung mehrerer Ströme, die jeweils durch einen kurzen Vorspann (mit Länge, ...) eingeleitet werden

# NTFS Dateisystem - 4

→ Eintrag in MFT für eine kurze Datei:



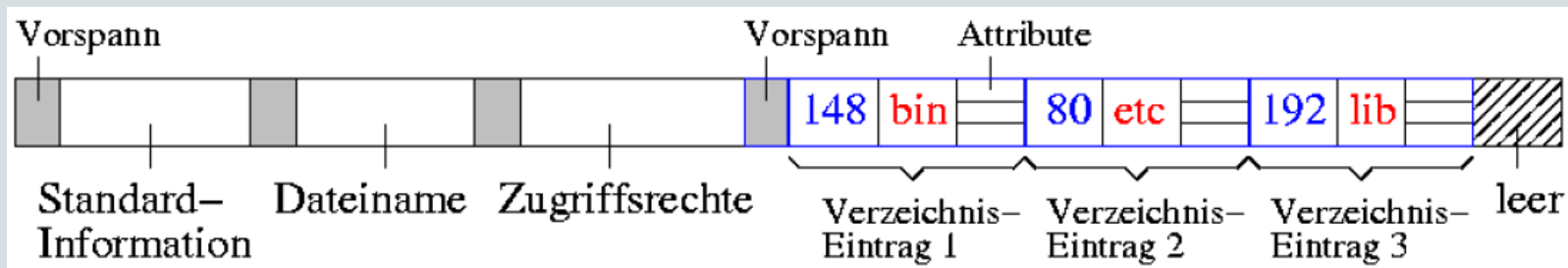
→ Eintrag in MFT für eine lange Datei (Beispiel):



- Daten befinden sich in Serien aus zusammengehörigen Clustern (Extents)
- Zuordnung von virtuellen Cluster-Nummern (VCN) zu logischen Cluster-Nummern (LCN) wird als weiterer Strom gespeichert

# NTFS Dateisystem - 5

→ Eintrag in MFT für ein **kurzes Verzeichnis** (Beispiel):



- Inhalt eines Verzeichnisses wird als eigener Strom gespeichert
- jeder Verzeichniseintrag enthält Dateireferenz, Dateiname und einige ausgewählte Attribute (z.B. Dateilänge, Datum der letzten Modifikation)
- Sortierung in lexikographischer Reihenfolge
- Eintrag in MFT für **ein langes Verzeichnis**:
  - Verzeichniseinträge werden nicht in MFT, sondern in separaten **Extents** gespeichert
  - Organisation als **B<sup>+</sup>-Baum** ermöglicht eine schnelle Suche in großen Verzeichnissen

# NTFS Dateisystem - 6

→ die ersten 16 Dateien in der MFT sind Metadateien, die für das System reserviert sind:

Index	Bedeutung
0	MFT
1	Kopie der MFT
2	Journal-Datei (protokolliert die Änderungen am Dateisystem)
3	Volume-Informationen (z.B. Name, Größe des Volumes)
4	Attribut-Tabelle (definiert erlaubte Ströme in den Einträgen)
5	Wurzelverzeichnis
6	Cluster-Bitmap (kennzeichnet alle freien und belegten Cluster)
7	Bootloader
8	Bad Cluster List (enthält die Indizes aller fehlerhaften Cluster)
9 bis 15	... (reserviert für weitere Systemdateien)
16	erste Benutzerdatei

# Vorzüge NTFS

- Sicherheitskonzept nach C2
- Benutzer und Gruppen mit umfangreichen Zugriffsrechten (no access, list, read, add, change, full)
- Clusteradressierung über 64-bit
- Automatische Komprimierung möglich
- Journalprüfung
- Tool zum Lesen/Schreiben von NTFS ohne NTFS: `ntfsdos.exe`

# Vergleich der Dateisysteme

	FAT	HPFS	NTFS
Dateiname	8+3 Zeichen (durch Punkt getrennt)	254 Bytes Mehrere Punkte zulässig	Z55 Unicode-Zeichen, Punkte zulässig
Dateigröße	2 <sup>32</sup> Byte	2 <sup>32</sup> Byte	2 <sup>64</sup> Byte
Partition	2 <sup>32</sup> Byte	2 <sup>41</sup> Byte	2 <sup>64</sup> Byte
Max. Länge des Suchweges (Pfad- länge)	64	Unbegrenzt	Unbegrenzt
Attribute	Einige Bitflags	Bitflags u. bis zu 64 k an erweiterten Attributen	Alles, einschl. der Daten wird als Dateiattribut behandelt
Verzeichnisse	Unsortiert	B-Baum	B-Baum
Konzept	Einfach	Schnell	schnell, mit Datenwiederherst./ Sicherheit
Eingebaute Sicherheit	Nein	Nein	Ja

# Betriebssystem - Dateisystem

	DOS	W95a	W95b/98/ ME	WinNT4	W2k	Wxp>
FAT16	X	X	X	X	X	X
VFAT		X	X	X	X	X
FAT32			X	X	X	X
NTFS4				X	X	X
NTFS5					X	X



# Weitere Dateisysteme

## → **ext2**

→ Linux Standard

## → **ext3**

→ Linux Standard mit Journaling

## → **Ext4**

→ Weiterentwicklung v. ext3 - mit erweiterten Grenzen

## → **riserfs**

→ Linux, Unix mit verbesserter Wiederherstellung

## → **HPFS (NTFS)**

→ OS/2 von IBM

## → **btrFS**

→ Linux/Unix, neu, sehr stabil und sicher

## → **ZFS**

→ Linux/Unix, SICHERHEIT

→ <https://www.ionos.de/digitalguide/server/knowhow/dateisysteme/>

→ [https://de.wikipedia.org/wiki/Liste\\_von\\_Dateisystemen](https://de.wikipedia.org/wiki/Liste_von_Dateisystemen)

# Fragmentierung und Defragmentierung

- ➡ Bei neuen/leeren Partitionen werden die Dateien hintereinander geschrieben
- ➡ Dateien werden gelöscht und Cluster werden frei gegeben
- ➡ Neue Dateien werden in die frei gewordenen Cluster geschrieben und dabei zerlegt (fragmentiert)
- ➡ Um Dateien wieder zusammenhängend zu erhalten muss man defragmentieren (Tool defrag.exe)

# Hinweise zum Defragmentieren

- Auf der Partition muss ausreichend Platz für das Zwischenspeichern verschobener Cluster sein
- Bei zu wenig Platz dauert der Prozess sehr lange
- Defragmentieren nur nötig, wenn ständig gelöscht und wieder neu beschrieben wird
- Partitionen, die nicht verändert werden brauchen nicht defragmentiert werden

# Bootmanager 01 – Warum?

- Bootmanager ermöglichen mehrere Betriebssysteme auf einer Festplatte
- Im Normalbetrieb benötigen Nutzer zur Erfüllung ihrer Aufgaben (Sekretärin, Buchhalter Datenbankbenutzer etc.) keinen Bootmanager
- Für "Computerpioniere", Tester, Programmierer und andere ist ein Bootmanager eine interessante Lösung mehrere Betriebssysteme auf einem Computer zu benutzen (nicht gleichzeitig -> dann VM).

# Bootmanager 02

- Linux und ab Windows 2000 haben eigenen Bootmanager
- Andere Bootmanager platzieren sich in MBR oder / und Bootsektor der Bootpartition (primäre Partition oder andere Partition)
- Windows ab W2k benötigt immer eine aktive primäre Partition für den Bootmanager (**ntloader und boot.ini**)

# Bootmanager 03 - Auswahl

- NT-Bootmanager (WNT / W2k / W2003 / WXP)
- LINUX (GRUB, GRUB2)
- PTS Boot Manager ME2 ([www.bhv.net](http://www.bhv.net))
- Bootmagic (Partition Magic [www.powerquest.de](http://www.powerquest.de))
- Xosl (Freeware) [www.xosl.org](http://www.xosl.org)
- **Beachte:** W9x überschreibt MBR und kann vorhandenen Bootmanager in den Partitionen überschreiben
- Reparatur: Supergrub, Rescatux, easyBCD, easyUEFI -> Internet

# Abkürzungen

Abkürzung	Bedeutung
BS / OS	Betriebssystem
HW	Hardware
SW	Software
API	Applications Programmers Interface
BIOS	Basic Input Output System
UEFI	Unified Extensible Firmware Interface
MMU	Memory Management Unit
SMP	<b>symmetrisches Multiprozessorsystem</b>

IPC	Interprozesskommunikation

# Quellenhinweise

- <https://gastack.com/de/superuser/299391/what-are-the-differences-between-firmware-and-softwareos>
- <https://de.wikipedia.org/wiki/Software>
- [https://www.thomas-krenn.com/de/wiki/UEFI\\_Einf%C3%BChrung](https://www.thomas-krenn.com/de/wiki/UEFI_Einf%C3%BChrung)
- <https://www.slideshare.net/k33a/uefi>
- <https://slideplayer.com/slide/4703738/>
- <http://www.softselect.de/business-software-glossar/software>
- [https://www.operating-system.org/betriebssystem/\\_german/w-wissen.htm](https://www.operating-system.org/betriebssystem/_german/w-wissen.htm)
- <https://www.microsoft.com/en-us/windows/windows-10-specifications>
- [https://www.google.com/search?q=Definition+Software&client=firefox-b-d&tbm=isch&source=iu&ictx=1&fir=gwjlIvPySGzctM%252CZg1VVNb0NWjRxM%252C%252Fm%252F01mf0&vet=1&usg=AI4\\_-kQP4--vggJjVkgDisLJ3MJDeTMHwA&sa=X&ved=2ahUKEwIU0sqV6vrvAhUzDGMBHbM4BKYQ\\_B16BAgrEAE#imgsrc=gwjlIvPySGzctM](https://www.google.com/search?q=Definition+Software&client=firefox-b-d&tbm=isch&source=iu&ictx=1&fir=gwjlIvPySGzctM%252CZg1VVNb0NWjRxM%252C%252Fm%252F01mf0&vet=1&usg=AI4_-kQP4--vggJjVkgDisLJ3MJDeTMHwA&sa=X&ved=2ahUKEwIU0sqV6vrvAhUzDGMBHbM4BKYQ_B16BAgrEAE#imgsrc=gwjlIvPySGzctM)
- <http://www.softselect.de/business-software-glossar/software>
- <https://gastack.com/de/superuser/299391/what-are-the-differences-between-firmware-and-softwareos>
- <https://de.wikipedia.org/wiki/Software>
- <https://docs.microsoft.com/en-us/windows-hardware/drivers/bringup/boot-and-uefi>
- <https://slideplayer.com/slide/4703738/>
- [https://www.thomas-krenn.com/de/wiki/UEFI\\_Einf%C3%BChrung](https://www.thomas-krenn.com/de/wiki/UEFI_Einf%C3%BChrung)
- <https://de.wikipedia.org/wiki/Firmware>
- <https://slideplayer.com/slide/4703738/>
- <https://www.heise.de/tipps-tricks/Was-ist-ein-Betriebssystem-4938579.html>
- <https://www.studydrive.net/en/flashcards/betriebssysteme-prozesse/11583>
- [https://www.sachsen.schule/~dvt/lpe13/bmv\\_pv.htm](https://www.sachsen.schule/~dvt/lpe13/bmv_pv.htm)
- [https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiUk8nGvYfxAhXslMUKHSDRC0QQFjATegQIGhAD&url=http%3A%2F%2Fais.informatik.uni-freiburg.de%2Fteaching%2Fws16%2Fsystems1%2Fslides%2Fkap04-prozesse.pdf&usg=AOvVaw17F7Lybvl7\\_cHP5XTepWY](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiUk8nGvYfxAhXslMUKHSDRC0QQFjATegQIGhAD&url=http%3A%2F%2Fais.informatik.uni-freiburg.de%2Fteaching%2Fws16%2Fsystems1%2Fslides%2Fkap04-prozesse.pdf&usg=AOvVaw17F7Lybvl7_cHP5XTepWY)
- <https://www.informatik.uni-leipzig.de/~meiler/Schuelerseiten.dir/MSchmidt/allgemein.html>
- [https://about.google/intl/ALL\\_de/stories/betriebssysteme/](https://about.google/intl/ALL_de/stories/betriebssysteme/)
- <http://www.netzmafia.de/skripten/bs/bs1.html>
- <https://www.tu-chemnitz.de/informatik/friz/Grundl-Inf/Betriebssysteme/Script/index.html>
- [https://www.sachsen.schule/~gdb/daten\\_verarbeiten/BS/Betriebssysteme..html](https://www.sachsen.schule/~gdb/daten_verarbeiten/BS/Betriebssysteme..html)
- [https://www.sachsen.schule/~dvt/lpe13/bmv\\_pv.htm](https://www.sachsen.schule/~dvt/lpe13/bmv_pv.htm)
- <https://www.sachsen.schule/~dvt/lpe13/134.htm>
- <https://www.sachsen.schule/~dvt/lpe13/131w.htm>