4.1 Datenträger/Partitionen

4.1.1 Von Festplatten und anderen Speichern

Bei der Partitionierung geht es um die Aufteilung von Hintergrundspeichern (auch Massenspeicher genannt). Heute sind das meistens Festplatten und SSD-Speicher. Hintergrundspeicher enthalten Programme und Daten, die bei Bedarf ins RAM geladen werden sollen. Wesentliche Aspekte dabei sind hohe Datenrate (nicht aber geringe Zugriffszeit, denn darin sind sie dem RAM hoffnungslos unterlegen) und Zuverlässigkeit.

4.1.2 Warum (nicht) partitionieren?

Partitionierung ist der Begriff für die Aufteilung von Festplatten und anderen Hintergrundspeichern in feste Bereiche. Es gibt mehrere Gründe dafür:

- Mehrere Betriebssysteme auf einem Computer sind möglich, wobei jedes Betriebssystem eine eigene Partition bekommt
- Trennung Daten und Betriebssystem: Bei Volllaufen der Daten-Partition kann das Betriebssystem weiterlaufen, bei Zerstörung der Betriebssystem-Partition (z. B. durch ein Update) bleiben die Daten erhalten, die Daten-Partition kann bequem gesichert werden
- Anlegen einer Sicherungs-Partition möglich
- Jede Partition kann gezielt optimiert werden (viele kleine Dateien oder wenige große usw.)

Aber es gibt auch Gründe dagegen:

- Zu Beginn braucht man Wissen, wie viel Platz wofür benötigt wird
- Frühe Festlegung, Partitionen können später nicht/nur unter Risiko verändert werden (Abhilfe: LVM)
- Spätere Um-Partitionierung kann Datenbereiche zerstören

Beim PC haben einige kleine Datenträger wie Disketten oder manche USB-Sticks gar keine Partitionstabelle.

4.1.3 Arten der Partitionierung

Für den Anwender soll eine Festplatte möglichst unabhängig vom übrigen Rechnersystem benutzbar sein (und umgekehrt), damit sie auch dann noch verwendet werden kann, wenn der Rest des Rechners ausgetauscht werden muss.

Eine erste Konsequenz daraus ist, dass die Partitionierungsdaten nicht im NVRAM des BIOS oder UEFI gespeichert werden, sondern auf der Festplatte selbst. Sie befinden sich in einem Bereich, den man Partitionstabelle nennt.

Eine zweite Konsequenz ist, dass für die Partitionierung ein einheitliches Schema benutzt werden muss. Das ROM (BIOS oder UEFI) bei PCs ist sehr einheitlich und unterstützt dieses Anliegen:

a) Das ROM nach BIOS-Industrienorm unterstützt das sogenannte MBR-Schema

b) Das ROM nach UEFI-Norm unterstützt neben dem MBR- zusätzlich das GPT-Schema

c) Für andere Hardware gibt es weitere Arten (z. B. BSD/Sun und Irix/SGI) der Partitionierung

Bei PCs befindet sich auch das Betriebssystem auf einem Massenspeicher. Deshalb gibt es bei PCs auch die Notwendigkeit, dass das ROM den Betriebssystem-Kernel oder zumindest ein kleines Ladeprogramm (Bootloader) vom Massenspeicher ins RAM laden und starten kann (sogenannter Boot-Vorgang).

Dafür muss das ROM wissen, wo das zu ladende Programm auf der Festplatte liegt. Auch diese Information ist ein Teil des Partitionierungsschemas.

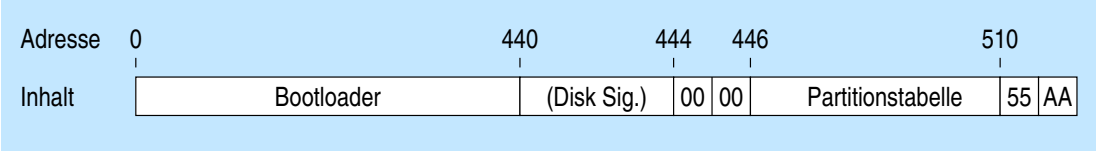

Abbildung 1: Aufbau des MBR

4.1.4 Exkurs: CHS und LBA

Hintergrundspeicher wie Festplatten und SSDs werden nicht byteweise, sondern Sektor für Sektor angesprochen. Ein Sektor besteht oft aus 512 Bytes. Er ist ein Teil einer Spur auf einer Ebene der Festplatte. Wie beim Hauptspeicher muss hier jeder Speicherinhalt (beim RAM jedes Datenwort, hier jeder Sektor) eindeutig adressiert werden können.

In der Anfangszeit der PCs gab es dafür das dreidimensionale CHS-Schema (cylinder x head x sector). Mit Cylinder ist die Nummer der Spur gemeint, mit Sector der Abschnitt in dieser Spur und mit Head der Schreibkopf und damit die Ebene der Festplatte. Das Schema sah maximal 1024 Cylinder, 63 Sectoren und (unrealistische) 255 Schreibköpfe vor, so dass 10+6+8=24 Bit für die Adressierung eines Sektors ausreichten. Sobald es Festplatten mit mehr als 8 GiB gab, musste man sich etwas ausdenken. Man wählte statt der getrennten Werte für C, H und S eine einzige Zahl und überließ die Umrechnung dem Festplatten-Controller. Dies war sinnvoll, denn mit dem Übergang von MFM- zu (Parallel-) ATA-Platten hatte man ohnehin eingeführt, dass die Anzahl der Sektoren variabel sein konnte: Im Innern der Platte hatten auf einer Spur weniger Sektoren Platz als außen. Dieses neue Schema nannte man LBA. Mit einer 32-Bit-Zahl konnte man nun bis zu 2 TiB große Platten adressieren.

4.1.5 MBR-Partitionierungsschema

4.1.5.1 MBR und Primäre Partitionen Der vorderste Sektor der Festplatte (512 Bytes) wird MBR (*Master Boot Record*) genannt. Abbildung 1 zeigt seinen Aufbau. Er kann in den vorderen 440 Byts entweder einen Bootlader enthalten oder aber lauter Null-Bytes. Ab dem Byte Nr. 446 beginnt (meistens[1]) die Partitionstabelle. Sie ist 64 Bytes groß und hat genau vier Einträge. Jeder Eintrag besteht aus den in Tabelle 1 genannten Daten. Man sieht, dass sowohl CHS als auch LBA unterstützt werden können. Das Boot-Flag sagt dem BIOS-ROM (oder einem Bootloader),

Adresse	Größe	Inhalt
0	1 Byte	Boot-Flag (128=ja, 0=nein)
1	3 Byte	Startsektor (CHS)
4	1 Byte	Partitionstyp
5	3 Byte	Letzter Sektor (CHS)
8	4 Byte	Startsektor (LBA)
12	4 Byte	Anzahl Sektoren (LBA)

Tabelle 1: Eintrag in der MBR-Partitionstabelle

ob die jeweilige Partition wiederum ein Bootprogramm in seinem vordersten Sektor enthält oder nicht. Zwei Werte bestimmen, von wo bis wo die Partition liegt (entweder per LBA oder per CHS). Und schließlich gibt es noch den Partitionstyp. Mit ihm kann man dem Betriebssystem einen Hinweis geben, ob es diese Partition benutzen sollte oder nicht.

[1] Eigentlich muss hier keine Partitionstabelle sein. Der Sektor 0 wird durch das BIOS gelesen und ausgeführt. Nur dann, wenn er in den vorderen 440 Bytes nur Null-Bytes enthält, liest das BIOS die Partitionstabelle. Aber jede normale Bootloader geht ebenfalls davon aus, dass im MBR eine Partitionstabelle steht.

4.1.5.2 Erweitere Primäre Partition In vielen Fällen kommt man mit vier Partitionen nicht aus. Deshalb hat man bald eine Erweiterung beschrieben: Eine der vier primären Partitionen darf einen bestimmten Partitionstyp haben (Typ 5). Dieser Typ sagt aus, dass es sich um eine *erweiterte* (primäre) Partition handelt. Diese erweiterte (primäre) Partition kann nun wiederum beliebig viele Partitionen in sich aufnehmen. Diese Partitionen heißen dann *logische Partitionen*.

Die technische Realisierung erfolgt mit Hilfe einer verketteten Liste. Die erweiterte Partition beginnt mit einem EBR (extended boot record), der nichts als eine Partitionstabelle enthält. Diese besitzt einen normalen Eintrag für die erste logische Partition und gegebenenfalls einen zweiten Eintrag (Typ 5) mit einem Zeiger (Sektornummer als CHS oder LBA) auf den nächsten EBR. Der enthält wiederum eine Partitionstabelle mit einem normalen Eintrag für die zweite logische Partition und ggf. einen zweiten (Typ 5) mit einem Zeiger auf den folgenden EBR.

4.1.6 MBR und Linux, Programm fdisk

Bei Linux sind die vier primären Partitionen mit den Nummern 1 bis 4 bezeichnet. Die logischen Partitionen erhalten Nummern von 5 an aufwärts. Die zugehörigen Gerätedateien haben folgendes Schema:

- /dev/sda – erste Platte (Serial-ATA)
- /dev/sdb – zweite Platte
- /dev/sda1 – erste Platte, primäre Partition 1
- /dev/sda2 – erste Platte, primäre Partition 2 usw.

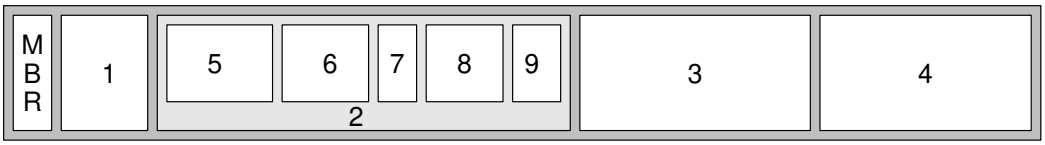

Abbildung 2: Aufteilung einer Festplatte mit MBR (Beispiel)

Abbildung 2 zeigt eine typische Festplatte mit Master Boot Record, den primären Partitionen 1, 2, 3 und 4 (von denen Nr. 2 eine erweiterte Partition ist) sowie die logischen Partitionen 5 bis 9 innerhalb der erweiterten Partition.

Zum Partitionieren dient unter Linux das Programm `fdisk` (es gibt noch weitere, auch mit GUI). Als Parameter muss man den Namen der Festplatte mitgeben.

Terminal

```
root@debian964:~# fdisk /dev/sdb
   Gerät  boot.     Anfang        Ende     Blöcke   Id  System
/dev/sdb1   *           1          16      128488+  83  Linux
/dev/sdb2              17       30401   244067512+   5  Erweiterte
/dev/sdb5              17         140      995998+  83  Linux
/dev/sdb6             141         724     4690948+  83  Linux
/dev/sdb7             725        4708    32001448+  83  Linux
/dev/sdb8            4709        5206     4000153+  82  Linux Swap
/dev/sdb9            5207        9190    32001448+  83  Linux
/dev/sdb10           9191       30401   170377326   83  Linux
```

Nun kann man mit `n` eine neue Partition anlegen (wenn Platz ist), sich mit `l` mögliche Partitionstypen anzeigen lassen und schließlich mit `t` den Typ einer Partition ändern. Mit `w` kann man speichern und verlassen (am besten danach neu starten), mit `q` verlässt man das Programm ohne Ändern der Tabelle.

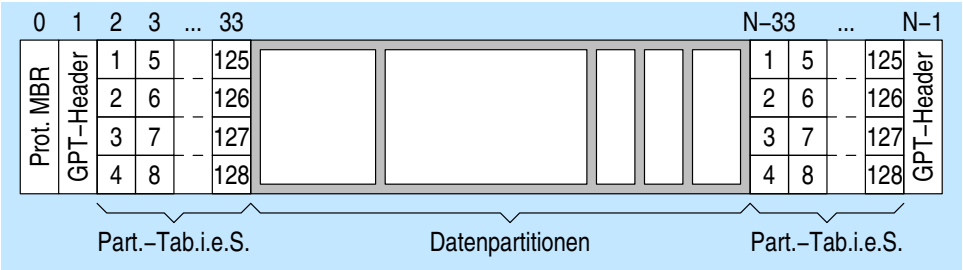

Abbildung 3: Aufteilung einer Festplatte mit GPT

4.1.7 GPT-Partitionierungsschema

Aus Tabelle 1 sieht man zwei Begrenzungen des MBR-Schemas: Eine Partition kann nicht hinter einer 2-TiB-Grenze liegen und nicht größer als 2 TiB sein. Damit können Festplatten mit mehr als 4 TiB nicht vollständig benutzt werden.

Beim GPT-Schema dagegen werden für Start- und Endsektor jeweils 64 statt 32 Bit verwendet. Damit sind Partitionsgrößen bis 8 ZiB möglich[2], bei NTFS immerhin noch bis 256 TiB[3].

Die CHS-Adressierung der Sektoren entfällt bei GPT vollständig; alle Sektoren werden hier mit LBA adressiert.

GPT steht für *GUID partition table* und GUID für *global unique identifier*, was wiederum ein anderes Wort für UUID ist (universally unique identifier). Das soll andeuten, dass sowohl die Festplatten als auch die Partitionen (und selbst die Partitionstypen) nicht durch Buchstaben oder Nummern, sondern durch 128-Bit-IDs eindeutig gekennzeichnet sind.

Abbildung 3 zeigt eine den Aufbau einer Festplatte mit GPT. Sektor 0 ist mit einem Pseudo-MBR ausgestattet. Für die alten Werkzeugen, die GPT nicht kennen, soll es so aussehen, als ob die ganze Festplatte mit einer einzigen Partition vom Partitionstyp 0xEE belegt ist. Man spricht daher von einem „protective MBR“, und so wird auch der MBR-Partitionstyp 0xEE benannt. In Sektor 1 befindet sich der sogenannte primäre GPT-Header. Er beinhaltet Informationen über die Partitionstabelle[4]. Hinter dem GPT-Header folgen 32 Sektoren mit insgesamt 128 Partitionseinträgen (zu je 128 Byte), und damit ist die GPT zu Ende.

Am Schluss der Platte liegen aus Gründen der Sicherheit noch einmal eine Kopie der 128 Partitionseinträge und des GPT-Headers (der sogenannte sekundäre GPT-Header).

Von den 512 Bytes des Sektors Nr. 1 werden in der momentanen Version 1.00 nur 92 Bytes benutzt. Er beginnt mit der Signatur `"EFI PART"` und der Versionsnummer, enthält u. a. eine UUID für die Festplatte und zum Schluss eine Checksumme.

Interessanter ist der Eintrag für eine Partition (Tabelle 2).

Adresse	Größe	Inhalt
0	16 Byte	UUID des Partitionstyps
16	16 Byte	UUID der Partition
32	8 Byte	Vorderster Sektor
40	8 Byte	Letzter Sektor
48	8 Byte	Attribute
56	72 Byte	Partitionsname (UTF16, Little-Endian)

Tabelle 2: Eintrag in der GPT-Partitionstabelle

[2] Bei einer geschätzten Verdoppelung der Festplattenkapazität alle 1,5 Jahre müsste das für etwa 50 Jahre reichen.
[3] reicht immerhin noch für 10 Jahre
[4] Dadurch kann man in zukünftigen GPT-Versionen Erweiterungen anbringen, die man im GPT-Header beschreibt.

4.1.8 GPT und Linux, Programm gdisk