Bootvorgang und Partitionstabelle

Michael Dienert

19. Oktober 2009

Inhaltsverzeichnis

1 BIOS bei PC mit Intel-Architektur

Das BIOS ist die **Firmware** (s.u.) eines PCs. Da das BIOS *sofort* nach dem Einschalten des Rechners zur Verfügung stehen muss, ist es in einem Flash-ROM oder EEPROM (Electrically Erasable Read Only Memory) gespeichert und kann so ohne Festplattenzugriff durch den Power-On-Reset (Reset des Rechners beim Einschalten) gestartet werden.

Die Abkürzung **BIOS** steht für **B**asic **I**nput **O**utput **S**ystem.

1.1 Firmware

Heute besitzt praktisch jedes elektronische Gerät einen Mikroprozessor (Festplatte, USB-Stick, Kamera, Mobiltelefon, Waschmaschine, ...). Die Software dieser Prozesso-

ren wird als **Firmware** bezeichnet. Sie stellt die elementaren Funktionen zur Steuerung des Geräts oder sogar das gesamte Betriebssystem bereit.

Die Firmware ist dauerhaft in einen nicht-flüchtigen Speicher (ROM, Read Only Memory) einprogrammiert. Heutzutage werden dabei oft Flash-ROMs als Speicher verwendet. Diese kann man löschen und wiederprogrammieren, so dass ein Update der Firmware möglich ist.

1.2 Aufgaben des BIOS

Das BIOS eines PCs ist ein Stück Software, dessen Ursprung inzwischen 25 Jahre alt ist.

Als die ersten PCs 1983 auf den Markt kamen, war das BIOS Teil des Betriebssystems und enthielt eine Reihe von Unterprogrammen (Routinen), die von DOS (ursprünglich QDOS - Quick and Dirty Operating System) aufgerufen wurden.

Diese Routinen hatten vielfältige Aufgaben, die wichtigsten davon waren:

- Festplattensektoren lesen und schreiben
- Zeichen auf den Bildschirm schreiben
- Kontrolle des Cursors
- Zeichen von Tastatur lesen
- Einzelne Zeichen von der seriellen Schnittstelle lesen und dorthin senden
- Diskette lesen, schreiben und formatieren

Diese Funktionen werden heute absolut nicht mehr genutzt, sondern vom Betriebssystemkern (NT-, Linux- oder BSD-Kernel) übernommen.

Damit aber überhaupt ein Betriebssystemkern von der Festplatte in den Speicher geladen werden kann, muss das BIOS zumindest folgende Aufgaben auch heute noch übernehmen:

- Power On Self Test - POST
- Initialisierung der Hardware
- Wahl des Boot-Datenträgers (z.B. Festplatte, CDROM, LAN)
- Laden des Boot-Sektors in den Speicher

2 Bootvorgang eines PCs

Der Begriff *Booten* ist eine Abkürzung für *Bootstrap*, womit der Startvorgang eines Computers bezeichnet wird.

Beim Starten eines PCs hat man folgendes Problem:

im normalen Betrieb steht der Betriebssystemkern (Kernel) im RAM. Er enthält die Softwareteile, mit denen auf die Festplatte zugegriffen werden kann.

Wird der Rechner eingeschaltet, ist aber kein Kernel im RAM mit dem man die Festplatte einlesen könnte.

Um jetzt überhaupt starten zu können, enthält das BIOS einen Softwareteil, mit dem man zumindest einen Sektor der Platte ins RAM laden kan.

> Da das BIOS beim Starten noch keinerlei Informationen über die Struktur der Festplatte hat, wurde fest vereinbart, dass nach dem Starten immer der allererste Sektor der Masterplatte am IDE-Port 0 geladen wird. Dieser ganz spezielle Sektor wird **Master Boot Record** (**MBR**) genannt.

Ablauf des Bootvorgangs Im Detail läuft der Bootvorgang in folgenden Schritten ab (siehe auch Abb. 1):

- Das BIOS lädt die 512 Bytes des MBR (s.o.) in den Hauptspeicher. Diese haben folgende Bedeutung:
  - Die Bytes 0 - 445 enthalten einen sog. *Bootloader*, das ist ein Programm, mit dem man weitere Teile der Festplatte in den Speicher laden kann
  - Die Bytes 446-510 enthalten die *Partitionstabelle* (s.u.) der Festplatte.
- Das in den Speicher geladenen Bootloader-Programm wird gestartet.
- Das Bootloader-Programm sucht die als aktiv markierte Partition in der Partitionstabelle und lädt den Bootsektor dieser Partition in den Speicher und startet das darin enhaltene Programm. Dieses Programm wird oft als Stage 1 bezeichnet.
- Das Stage 1 - Programm (nur 512 Bytes gross) lädt den Stage 2 und startet ihn.
- Der Stage 2 lädt schliesslich und endlich den Betriebbssystemkern (Kernel) in den Hauptspeicher.

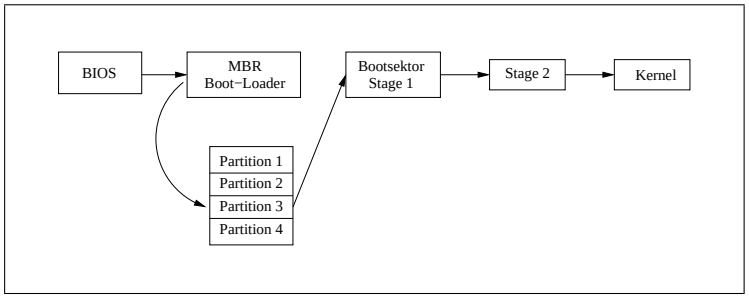

Abbildung 1: Ablauf des Bootvorgangs eines PCs

3 Festplatten

3.1 Physikalischer Aufbau

3.2 Die CHS-Koordinaten

Die kleinste adressierbare Dateneinheit auf einer Festplatte ist ein Sektor. Dieser enthält 512 Bytes. Um nun gezielt Sektoren von der Platte einlesen oder auf die Platte schrei-

ben zu können, wurden die Sektoren früher mit den drei Angaben

Cylinder - eine Spur mit bestimmtem Radius

Head - welcher der Schreib-/Leseköpfe wird verwendet

Sector - der Kreis, den eine Spur bildet wird in gleichgrosse Abschnitte unterteilt. Diese nennt man Sektoren.

Da man mit der Angabe von Cylinder, Head und Sector durch Begrenzungen im BIOS und der IDE-Schnittstelle nur Platten bis zu einer Grösse von nur 504MB verwenden kann, werden die CHS-Werte nicht mehr für die Adressierung eines Sektors verwendet.

3.3 Logical Block Adressing

Beim *Logical Block Adressing* (LBA) werden alle Sektoren der Platte von 0 bis zur höchsten Sektornummer durchnummeriert.

Um einen bestimmten Sektor zu schreiben oder zu lesen genügt es, wenn man der Platte die Nummer dieses Sektors sendet.

Die Elektronik und Firmware der Festplatte rechnet diese Sektornummer dann selbst in den richtigen, internen CHS-Wert um.

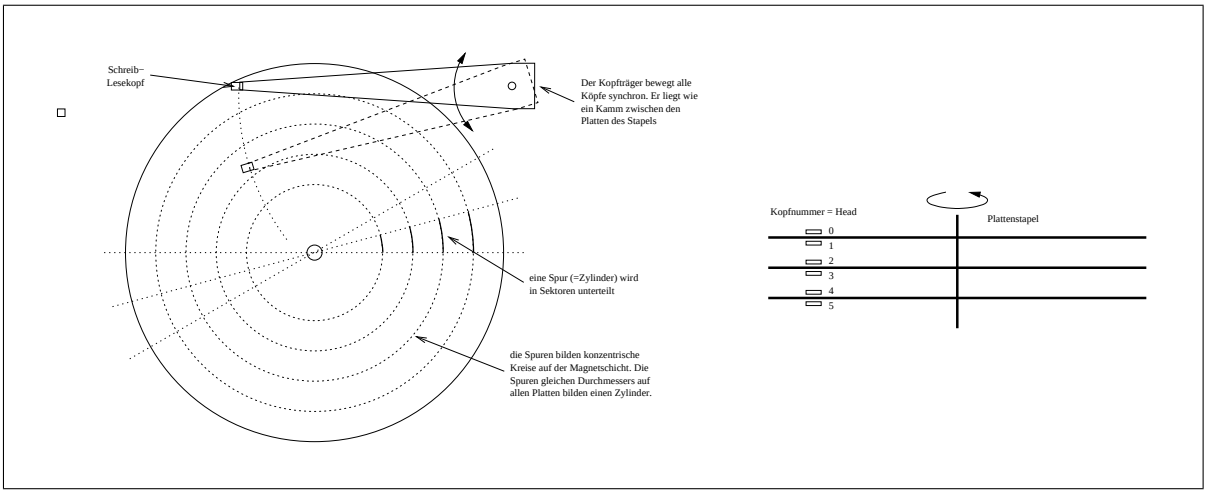

Abbildung 2: Schema einer Festplatte

3.4 Partitionierung und Partitionstabelle

Bei der Partitionierung wird ein einzelnes, *physikalisches* Laufwerk in mehrere *Partitionen* unterteilt. Eine Partition kann man sich wiederum wie ein *virtuelles* Laufwerk vorstellen.

Die Partitionierung ist dabei einfach nur eine Unterteilung der Festplatte in mehrere Bereiche. Diese Unterteilung wird in der *Partitionstabelle* festgelegt.

- Die Partitionstabelle ist 64 Bytes lang und steht im Master Boot Record.
- Sie besteht aus 4 Zeilen zu je 16 Bytes mit denen jeweils 1 Partition bestimmt wird
- Eine Partition wird durch Angabe ihrer **Startsektornummer** und die **Anzahl** der Sektoren bestimmt. Beide Werte werden als 4 Byte lange Zahlen in den Zeilen der Tabelle gespeichert.
- Eine solche Partition, deren Startsektornummer in der Partitionstabelle des mbr steht wird **primäre Partition** genannt.
- Am Ende der Partitionstabelle steht die *Magic Number*, die aus den beiden Bytes 0x55 und 0xaa besteht.

Früher wurden in den Partitionstabellen auch noch CHS-Werte eingetragen. Da mit diesen aber nur 8GB grosse Platten verwaltet werden können, stehen dort heutzutage bei Partitionen, die oberhalb der ersten 8GB liegen, die festen Werte C=254, H=63 und S=1023 drin.

Mit den 4Byte = 32 bit langen Sektornummer lassen sich bis $2^32 \cdot 512Bytes = 2TB$ grosse Platten partitionieren.

Die Abbildung 3 zeigt einen Teil des MBRs einer Festplatte mit 3 Partitionen. Die Partitionstabelle beginnt bei Byte 446=0x1be.

```
Device: /dev/sda
0x000: EB 48 90 10 8E D0 BC 00 B0 B8 00 00 8E D8 8E C0
0x010: FB BE 00 7C BF 00 06 B9 00 02 F3 A4 EA 21 06 00

                  ................

0x190: 69 73 6B 00 52 65 61 64 00 20 45 72 72 6F 72 00
0x1A0: BB 01 00 B4 0E CD 10 AC 3C 00 75 F4 C3 00 00 00
0x1B0: 00 00 00 00 00 00 00 00 82 0A 0B 00 00 00 80 01
0x1C0: 01 00 83 FE FF E0 3F 00 00 00 62 6A F3 00 00 00
0x1D0: C1 E1 05 FE FF FF A1 6A F3 00 73 80 0C 00 00 00
0x1E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA
```

Abbildung 3: Hexdump eines MBR

```
Platte /dev/sda: 8589 MByte, 8589934592 Byte
255 Köpfe, 63 Sektoren/Spuren, 1044 Zylinder
Einheiten = Zylinder von 16065  512 = 8225280 Bytes
Disk identifier: 0x000b0a82

   Gerät  boot.     Anfang        Ende     Blöcke   Id  System
/dev/sda1   *           1         993     7976241   83  Linux
/dev/sda2             994        1044      409657+   5  Erweiterte
/dev/sda5             994        1044      409626   82  Linux Swap / Solaris
```

Abbildung 4: Partitionen eines GNU/Linux-Systems

Der erste Eintrag in die Partionstabelle ist hier nochmals herauskopiert:

```
80 01 01 00 83 FE FF E0 3F 00 00 00 62 6A F3 00
```

Die einzelnen Werte (alle Hexadezimal) haben folgende Bedeutung:

80 - diese Partion ist **bootfähig**. Wäre sie nicht bootfähig, stünde hier 0x00.

01 01 00 - CHS-Koordinaten des ersten Sektors

83 - Typ der Partion; 0x83 = Linuxpartition; 0x82 = Swappartition

fe ff e0 - CHS-Koordinaten des letzten Sektors

3f 00 00 00 - Nummer des Startsektors

62 6a f3 00 - Anzahl der Sektoren dieser Partion

4 Erweiterte Partitionen

Da die Partitionstabelle nur Platz für 4 Einträge hat, kann man damit nur 4 Partitionen auf einem Laufwerk adressieren. Um diese Einschränkung zu umgehen, ohne die Struktur des **mbr** ändern zu müssen (um also kompatibel zum alten Standard zu bleiben), wurden *erweiterte* und *logische* Partitionen eingeführt.

Erweiterte Partition - unter Linux haben erweiterte Partitonen den Partitionstyp 0x05 oder auch 0x85. Eine erweiterte Partition ist eine Partion, die auf einen Festplattensektor zeigt, der eine *Partitionstabelle* enthält.

Diese Partitionstabelle hat einen Eintrag auf den Startsektor einer Daten-Partition (Typ 0x83) und sie kann noch einen weiteren Eintrag mit dem Typ 0x85 haben, der nochmal auf eine Partitionstabelle zeigt.

Da diese Partitionstabelle wiederum einen Eintrag vom Typ 0x05 (evtl. auch 0x85) und 0x83 haben darf, erhält man eine einfach verkettete Liste, mit der man beliebig viele Partitionen anlegen kann.

In allen Partitionstabellen (auch in der im mbr) darf aber immer nur genau ein Eintrag mit Typ 0x85 stehen!

logische Partition - logische Partitionen sind alle Datenpartitionen, deren Startsektoren in den Partitionstabellen der verketteten Liste stehen, die also keine primären Partitionen sind.

Die Abb. 5 zeigt die mit `fdisk` ausgelesenen Partitionstabellen. Blöcke, die nur Nullen entahlten sind durch einen '*' abgekürzt.

```
Platte /dev/sdb: 2147 MByte, 2147483648 Byte
255 Koepfe, 63 Sektoren/Spuren, 261 Zylinder, zusammen 4194304 Sektoren
Einheiten = Sektoren von 1 x 512 = 512 Bytes
Disk identifier: 0x2008b531

Geraet     boot.   Anfang      Ende        Bloecke  Id  System
/dev/sdb1              63      401624      200781   83  Linux
/dev/sdb3          401625      610469      104422+   5  Erweiterte
/dev/sdb5          401688      514079       56196   83  Linux
/dev/sdb6          514143      610469       48163+  83  Linux

Geraet /dev/sdb

allererster Sektor der Platte /dev/sdb
Adressen 0x000 bis 0x1f0 sind absolut

0x000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
0x1A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1B0: 00 00 00 00 00 00 00 00 31 B5 08 20 00 00 00 01
0x1C0: 01 00 83 FE 3F 18 3F 00 00 00 9A 20 06 00 00 00
0x1D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1E0: 01 19 05 FE 3F 25 D9 20 06 00 CD 2F 03 00 00 00
0x1F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA


Sektor mit Datenpartition und Zeiger auf naechste Partitionstabelle
Adressen 0x000 bis 0x1f0 sind relativ zum Sektor 401688

0x000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
0x1A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01
0x1C0: 01 19 83 FE 3F 1F 3F 00 00 00 08 B7 01 00 00 00
0x1D0: 01 20 05 FE 3F 25 47 B7 01 00 86 78 01 00 00 00
0x1E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA

Sektor mit einer Datenpartition
Adressen 0x000 bis 0x1f0 sind relativ zum Sektor 514143

0x000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
0x1A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01
0x1C0: 01 20 83 FE 3F 25 3F 00 00 00 47 78 01 00 00 00
0x1D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA
```

Abbildung 5: Primäre und erweiterte Partitionstabellen

5 Die Zukunft: GUID-Partitionen

Das ganze, komplizierte Szenario mit den erweiterten und logischen Partitionen haette man nicht, wenn die Partitionstabelle grösser wäre und man dort mehr Einträge für primäre Partitionen hätte.

Dies wurde beim Nachfolger des mbr realiesiert: die GUID-Partitionstabellen haben Platz für 128 primäre Partitionen und verwenden eine 64Bit-Sektoradresse. Bei einer Sektorgrösse von 512 Bytes, kann man damit

$$2^{64} \cdot 512 = 2^{73} = 8192 \text{ Milliarden Terabyte}$$

adressieren.

GUID steht für **G**lobally **U**nique **ID**entifier, das ist eine 128bit lange ID, mit der die Partitionstypen und die Partition selbst eindeutig gekennzeichnet werden.

6 Praktische Übung: Auslesen des mbr auf einem Linux-System

6.1 Mit dem Kommando dd

Arbeitet man mit Unix, gilt immer:

> Alles ist eine Datei

Das heisst, auch die Laufwerke der Festplatten sind auf einem Unix-System als sog. *Gerätedateien* abgebildet. Das sind keine echten Dateien, sondern die Geräte werden vom Kernel in ein besonderes Verzeichnis (/dev) als Pseudodatei abgebildet.

Die erste Festplatte, auf der sich der mbr befindet heisst nun:

/dev/hda : bei älteren Platten mit Parallel-ATA-Schnittstelle

/dev/sda : bei allen anderen Platten (USB, IEEE1394, SCSI, SAS, SATA, RAID, ...)

Welche Platte denn nun genau verwendet wird zeigt folgendes Kommando:

```
root@xubbie:mount
/dev/sda1 on / type ext3 (rw,errors=remount-ro)
          ....
```

Die Ausgabe des Kommandos zeigt, dass die Root-Partition dieses Rechners auf der ersten Partition (sda1) der ersten Platte (sda) liegt: /dev/sda1.

Mit dem Kommando

```
root@xubbie:dd if=/dev/sda bs=512 count=1 | hexdump
          ....
```

lässt sich nun der mbr auslesen und mit einem Programm zu Darstellung von Binärwerten (hexdump) darstellen.

> Dabei muss man unbedingt beachten, dass Intel-Maschinen **little-endian** sind! Das bedeutet, dass die Bytes eines jeden 16-Bit-Worts vertauscht sind: das niederwertige Byte steht vorne.

6.2 Mit dem Kommando fdisk

Das Kommando fdisk dient eigentlich dazu, eine Platte zu partitionieren. Es bietet aber auch ein Funktion, mit der man sich die Rohdaten der Bootsektoren einer Festplatte anzeigen lassen kann.

Dabei dreht fdisk die Bytereihenfolge so um, dass das höherwertige Byte eines 16-Bit-Worts vorne steht. Man erkennt das daran, dass die Magic-Number mit 0x55AA richtig angezeigt wird.

Gestartet wird fdisk so:

```
root@xubbie:fdisk /dev/sda
```

Im Menue für die Expertenkommandos gibt es den Menuepunkt 'd' (dump), der die Rohdaten anzeigt.