Linux booten

- Was passiert beim Einschalten eines PCs?
- Das BIOS
- Viele Bootsektoren
- Bootloader (Grub, LILO)
- Kernel Parameter und InitRD
- Feinheiten beim Plattenausfall

Jörg Ahrens <joerg@hydrops.han.de>

Der Bootvorgang

- Beim Einschalten eines PCs wird das BIOS ausgeführt
- Das BIOS lädt einen Bootloader von einem Speichermedium oder vom Netzwerk
- Der Bootloader lädt ein Betriebssystem (oder einen weiteren Bootloader)

Das BIOS

- Das ***B***asic ***I***nput ***O***utput ***S***ystem ist ein kleines Betriebssystem.
- Es stellt Ein-/Ausgabe Dienste für Tastatur, Grafikkarte, Speichergeräte, Drucker, usw. zur Verfügung.
- Das BIOS „wohnt“ in einem (P)ROM auf dem Motherboard.
- Beim Einschalten des PCs wird das BIOS ausgeführt.
- Es führt den ***P***ower ***O***n ***S***elf ***T***est durch.
- Es liest einen Boot Sektor von einem Speichergerät und startet ihn.

BIOS Interrupt 0x13

- Schnittstelle zum Zugriff auf Speichergeräte
- Gerätecodes des Interrupt 0x13:
  - 0x00 Diskette 1
  - 0x01 Diskette 2
  - 0x80 Platte 1
  - 0x81 Platte 2
  - 0x83 Platte 3 usw.
- Die Gerätecodes werden POST festgelegt ! (z.B.: IDE Ctrl. 1 Master, Slave, IDE Ctrl. 2 Master, Slave)
- Bootloader benutzen int 0x13 zum Zugriff auf Geräte. Sie können also nur auf die Blöcke einer Platte zugreifen, die das BIOS sieht (1024 Zylinder Grenze!)

BIOS boot Priorität

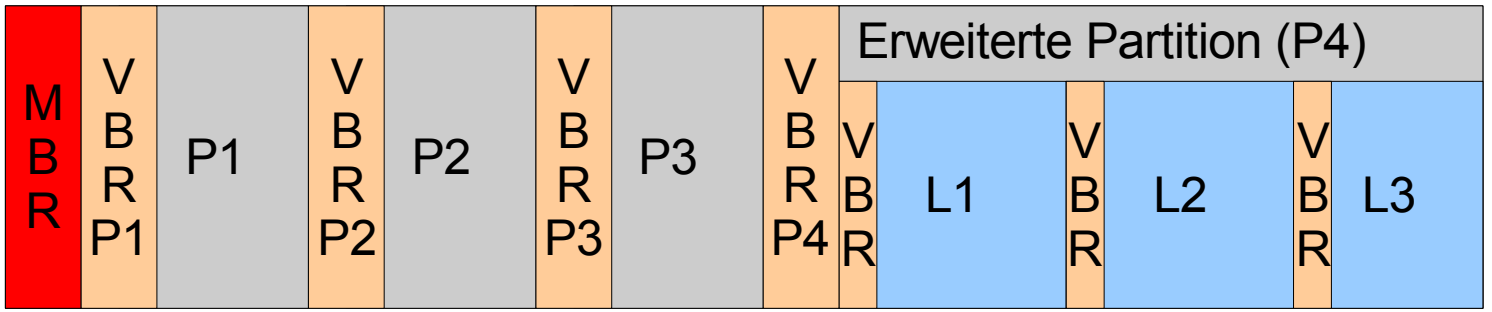
- Das BIOS sucht nach eingestellten Prioritäten die Geräte nach gültigen Bootsektoren ab.
- Im BIOS Setup können diese Prioritäten verändert werden z.B.: „Diskette, CD, Platte1, Platte2“ oder „CD, Platte2, Diskette, Platte1“.
- Der PC Standard sieht vor, dass nur von der ersten Diskette und von der ersten Platte gebootet werden kann. Fast alle BIOS Implementierungen können mehr.
- Soll von der „2. Platte“ gebootet werden, passt das BIOS die Gerätecodes an , die 2. Platte wird 0x80 und die erste Platte 0x81 !

Bootsektoren

- Ein Medium mit Partitionierung (Platte) hat
  - Einen Master Bootsektor (MBR), das ist der 1. Sektor des Mediums
  - Pro Partition einen Volume Bootsektor(VBR), das ist der erste Sektor der Partition
- Ein Medium ohne Partitionierung (Diskette) kann nur einen Volume Bootsektor enthalten.
- Ein Bootsektor ist 512 Bytes groß und muss eine bestimmte Signatur (0xAA55 an Position 0x01FE) tragen, um gültig zu sein.
- Das BIOS startet bei partitionierten Medien nur den MBR (Ausnahmen bestätigen die Regel)

Orte für Bootsektoren

Diskette



Festplatte

Jörg Ahrens <joerg@hydrops.han.de>

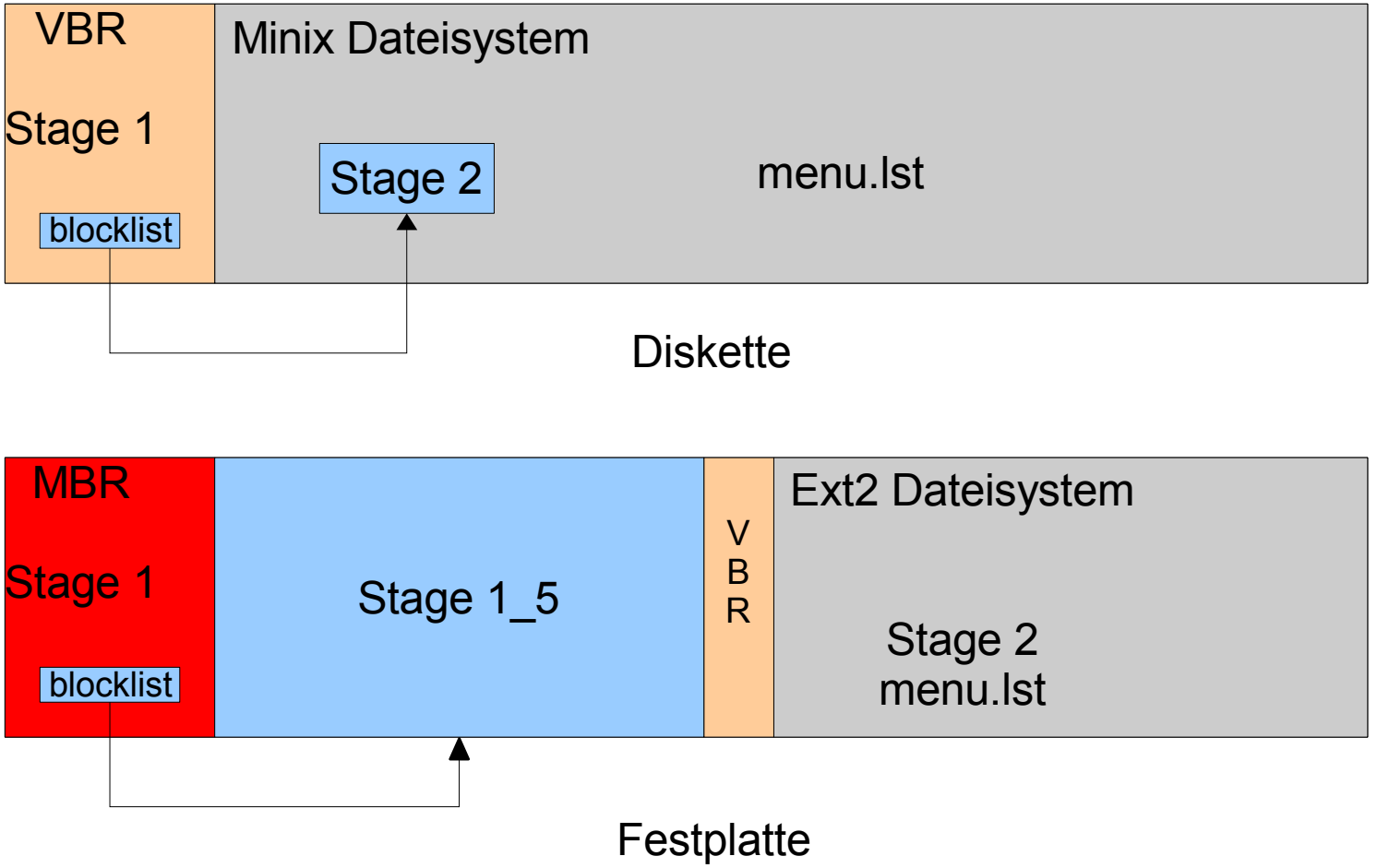
Struktur eine Bootloaders

- Ein Bootloader besteht aus mehreren Teilen (stages), weil in einem Bootsektor nicht genug Platz ist.
- Der Code für Stage 1 ist im Bootsektor enthalten. Seine Aufgabe ist es, die nächste Stage zu laden und zu starten.
- Stage 1 kennt keine Strukturen wie Dateisysteme. Daher muss die Information, wo die nächste Stage zu finden ist in Blocknummern (oder CHS Adressen) im Stage 1 Code eingebaut werden.

Grub

- **Stage 1** auf einem VBR
  - Lädt **Stage 2** per Blockliste von einem Volume(Plattenpartition)
  - **Stage 2** kennt Dateisysteme und kann einen Linux Kernel per Dateinamen laden
- **Stage 1** auf einem MBR
  - Lädt **Stage 1.5** aus den Blöcken zwischen MBR und erster Partition.
  - **Stage 1.5** kennt Dateisysteme und kann **Stage 2** per Dateinamen laden
  - **Stage 2** kennt Dateisysteme und kann einen Linux Kernel per Dateinamen laden

Grub



Jörg Ahrens <joerg@hydrops.han.de>

Grub

- Die Grub hat eine „shell“ (Stage 2) mit der man Grub im Bootvorgang interaktiv bedienen kann.
- Grub Grundlagen:
  - Gerätenamen in Klammern: (fd0) (hd0) ...
  - Platten werden von 0 an hochgezählt:
    (hd0), (hd1) ...
  - Partitionen werden von 0 hochgezählt und mit , an die Plattenbezeichnung angehängt:
    (hd0,0) (hd0,1) ...
  - Grub arbeitet immer im Kontext eines root Gerätes.

Grub Kommandos

- **`root`** `<Gerät>` : bestimmt das aktive Volume (Partition), auf der sich stage 2 und Kernel befinden.
- **`kernel`** `<Dateiname>` : bestimmt die Datei, die als Kernel geladen werden soll.
- **`initrd`** `<Dateiname>` : bestimmt die Init Ramdisk, die geladen werden soll.
- **`boot`** : startet den Bootvorgang (wenn mindestens root und kernel angegeben sind)
- **`setup`** `<Gerät>` : installiert eine Grub stage 1 auf einem Bootsektor

Grub Praxis

- Wir bauen eine Grub Diskette auf einem System, auf dem Grub als Software installiert ist:
  - Minix Dateisystem auf Diskette anlegen mit
    `mkfs.minix /dev/fd0`
  - Dateisystem mounten mit
    `mount /dev/fd0 /mnt`
  - Grub Stages kopieren mit
    `cp -Rp /boot/grub /mnt`
  - Dateisystem unmounten mit
    `umount /mnt`
  - Grub Bootsektor auf Diskette schreiben mit
    $ grub
    > root (fd0)
    > setup (fd0)
    > quit

Grub Praxis

- Wir booten mit dieser Diskette eine System (hda1 = /boot und hda2 = /)
  - > root (hd0,0)
  - > kernel /<tab> Der Tab Ergänzungsmechanismus, der aus der bash bekannt ist.
  - > boot

Grub Menü

- Alle Grub Kommandos können in einer Datei „menu.lst“ benutzt werden.
- Zusätzlich:
  - timeout : legt eine Wartezeit fest
  - default : legt den default Menüeintrag fest
  - title : leitet einen Menü Eintrag ein
- Praxis: wir bauen ein Menü und machen die Platte bootfähig.

Kernel Command Line

- Dient dazu, dem Kernel Parameter zu übergeben. Näheres siehe BootPrompt HowTo und `Documentation/kernel-parameters.txt` im Kernel Quellverzeichis.
- Einige wichtige Kernel Parameter:
  - root=/dev/hd... sagt den Kernel, auf welcher Partition sein root Dateisystem liegt
  - init=... sagt den Kernel, welches Programm als Init zu starten ist.

Initrd

- Ist ein vorbereitetes Dateisystem, dass vom Bootloader in eine Ramdisk geladen wird. Es wird vom Kernel kurz nach dem Start gemountet. Es muss ein Script mit dem Namen „linuxrc“ enthalten, das vom Kernel ausgeführt wird.
- Es hauptsächlich dazu, Module zu laden, bevor der Kernel Zugriff auf die Festplatte hat (z.B. Treiber für RAID Controller, ...)
- Es wird von Distributoren gern benutzt, um mit einem stark modularisierten Kernel jede Hardware bedienen zu können.
- Ist im „Normalbetrieb“ mit einem eigenen an die Hardware angepassten Kernel nicht notwendig.

Bootloader: Chain Loading

- Dient dazu, Bootsektoren zu laden und starten (z.B. eine Linux Installation auf einer zweiten Festplatte oder ein ganz fremdes System ;-)
- Mit Grub:
  > rootnoverify (hd1)
  > chainloader +1
  > boot
  um den MBR der zweiten Festplatte zu booten.
- Oder
  > rootnoverify (hd0,3)
  > chainloader +1
  > boot
  um den VBR der 4. Partition der ersten Platte zu booten.

Chain Loading: Praxis

- Booten der zweiten Platte von der ersten (und umgekert).
- Booten eines Volume Boot Sektors

LILO

- Der **LI**nux **Lo**ader arbeitet nur mit Blocklisten (map file)
- Das bedeutet: nach **jeder** Veränderung an einer Kernel-, InitRD- oder Menü-Datei muss LILO neu installiert werden.
- Keine „shell“, daher auch nur wenig Interaktion möglich.
- Funktioniert ebenso sicher und zuverlässig wie Grub.

Plattenausfall des Primary Masters

- Kann man die verbliebene Platte noch vollautomatisch booten? **Jein!**
- Praxis: Vorbereitung auf den Ernstfall
  - Mounten der Dateisysteme über Label
  - Kernel Parameter root= als Label angeben

Links

- BIOS
  http://en.wikipedia.org/wiki/BIOS
- PC BIOS Boot Standard
  http://www.phoenix.com/en/Customer+Services/White+Papers-Specs/pc+industry+specifications.htm
- Booting
  http://en.wikipedia.org/wiki/Booting
- Grub manual
  http://www.gnu.org/software/grub/manual/
- BootPrompt Howto
  http://www.tldp.org/HOWTO/BootPrompt-HOWTO.html