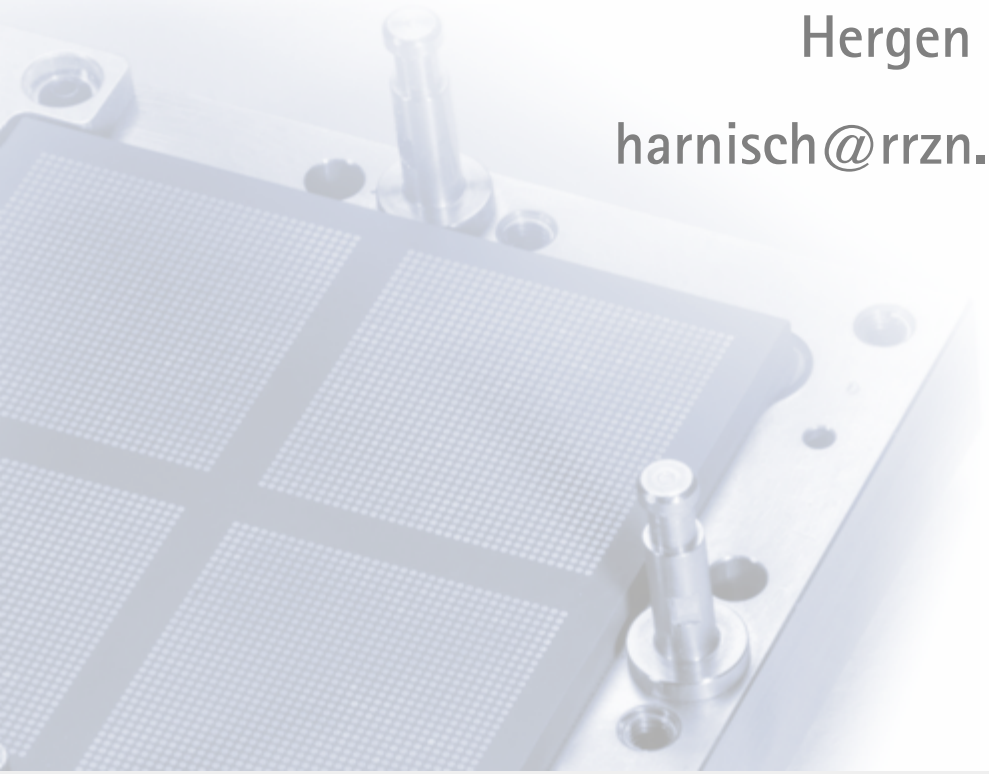
R|R|Z|N

Booten in x86-Architekturen

Hergen Harnisch

harnisch@rrzn.uni-hannover.de



Vorwort

Nicht mehr eingegangen wird auf veraltete Dinge:

- Rom-Basic (Int 18h)
- RPL / Netware-Netboot
- Jahr 2000 Fixups
- kein CD-Rom-Boot im Bios
- 8 GB, 1024 Zylinder u.Ä. Grenzen für Festplatten (-Boot)

Ebenso nicht auf Boot-Umfeld für verschlüsselte Systempartitionen mit Bitlocker oder TrueCrypt.

1. Prä-OS:
   - BIOS
   - Bootmanager
2. OS
   - Windows
   - Linux
3. EFI:
   - GPT
   - EFI-Bios
4. Malware

Ablauf

Als aller erstes wird das Motherboard-BIOS ausgeführt:

- Grundinitialisierung fürs Booten notwendiger Hardware
- Systemtest (u.A. POST)
- Aufruf von Option-ROM (gerätespezifischer BIOS-Erweiterungen)
- Laden & Ausführen des 1. Sektors des Bootmediums

BIOS und geladener Boot-Sektor laufen im Real-Mode (also 16-Bit)

R|R|Z|N

Motherboard-BIOS

- Boot-Block: kann BIOS-Integrität prüfen und ggf. BIOS neu flashen
- Backup-BIOS / DualBIOS manchmal als Fall-Back enthalten
- BIOS ist meist modular, je nach BIOS-Hersteller,
  z.B. EPA-Logo, OEM-Logo, PXE-Rom, eigentliches BIOS, CPU-Microcode
- Einige Einstellungen müssen im BIOS erfolgen, nicht im OS möglich:
  CPU-ID, VMX, TPM, teilweise On-Board-Hardware
- System-Passwort oder Passwort zum Schutz der Bootreihenfolge:
  - nötig zur Sicherung eines Systems
  - teils durch Hersterllerpasswörter, fast immer durch Jumper umgehbar

BIOS-Update

- ein fehlschlagendes Update kann den Rechner zerstören

  früher Wechsel des gesockelten Eprom-Bausteins möglich

  heute auf Motherboard höchstens Programmieranschluss, aber Boot-Block mit Recovery von Floppy / USB
- Update teilweise wichtig für Hardware-Support (insb. CPU)
- manchmal Behebung von RAID-Firmware-Fehlern o.Ä.

Option-ROM

- als Modul im Motherboard-BIOS oder separat als Speicherenblendung
- im Speicher erkennbar an einer Signatur
  2kB-Grenze: 55h, aah, Size-Byte (512 Byte-Blöcke), Einsprung, ...
  Option-Roms liegen im Speicherbereich 0xC0000 – 0xF0000
- Einklinken in den Bootprozess:
  früher Umbiegen von Interrupt 18h oder 19h
  heute über BBS-API (BIOS boot specification)
- z.B. Raid-Firmware, PXE-Bootrom, Server-Management-Erweiterungen

Int 13h

Das BIOS bietet für grundsätzliche I/O-Operationen eine API:

- Per Int13h-Calls wird der Massenspeicher (Floppy, Festplatte) angesprochen.
- USB-Storage wird als Festplatte emuliert
- Option-Rom für RAID, SAS etc. klinkt Support in Int13h ein

Da der erste Sektor per Int13h gelesen wird und Boot-Manager wie MS-Dos die BIOS-API verwenden:

- gebootet wird meist von 1. Festplatte (DL=80h)
- daher wird Boot-Medium zur 1. Festplatte,
  z.B. USB-Stick 1. HD, Master-SATA-Platte dann 2. HD

⟶ Bootreihenfolge kann

  → Boot-Manager verwirren

  → Laufwerksbenennungen im OS ändern (mind. in MS-Dos)

MBR

- ein Sektor, 512 (000h-1ffh) Bytes lang
- vom BIOS nur akzeptiert, wenn Bytes 1feh = 55h & 1ffh = aah
- wird vom BIOS nach 0000:7c00 geladen & dort ausgeführt
- enthält individuelle Disk-Signatur in Bytes 1b8h–1bbh
- enthält Partitions-Tabelle (4 primäre) in Bytes 1beh–1fdh

normalerweise üblicher Festplatten-MBR:

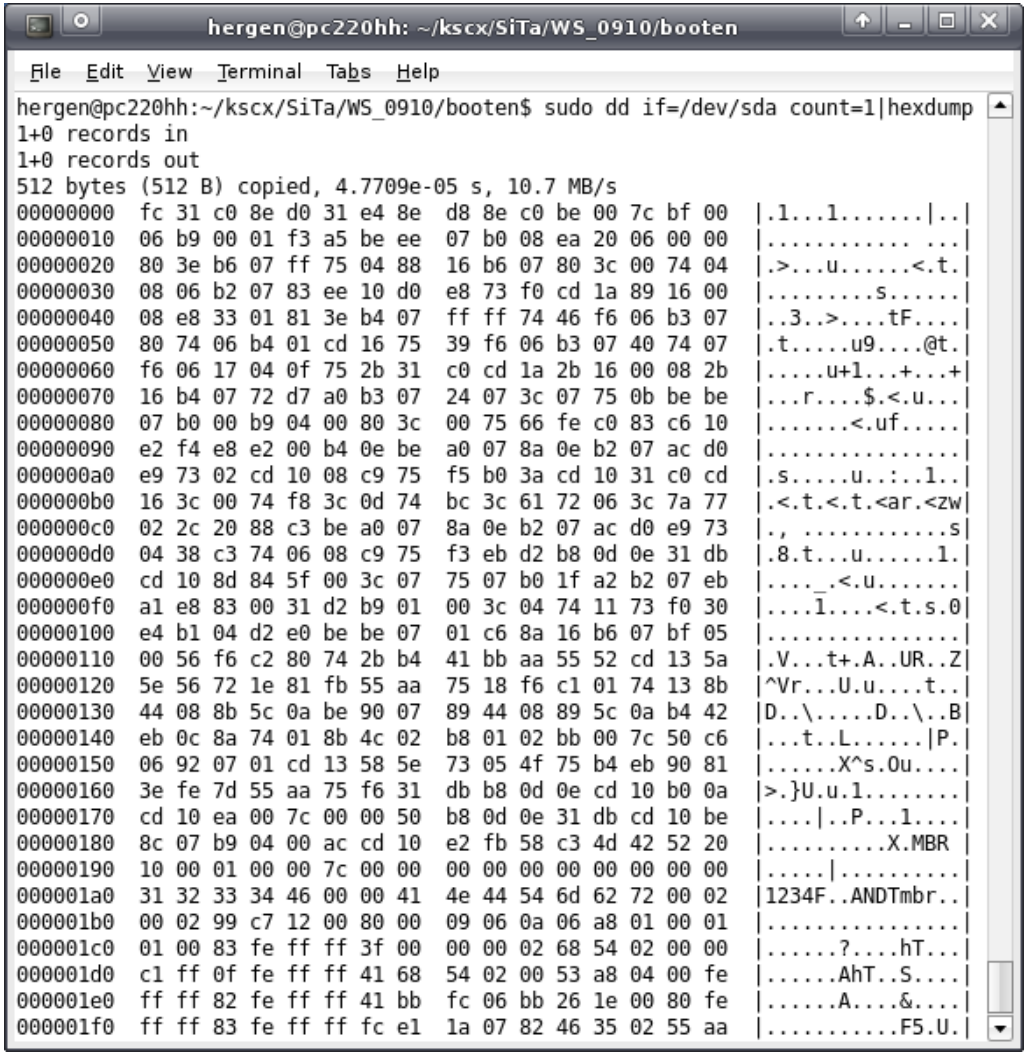
1. verschiebt sich im Speicher, damit nachfolgender Partitions-Bootsektor wieder an 0000:7c00 laufen kann
2. sucht aktiv-markierten Partitionseintrag

3a. lädt 1. Sektor der aktiven Partition nach 0000:7c00,
prüft auf aa55h am Ende des Sektors und führt ihn aus

3b. Fehlermeldung oder Int 18h bei fehlender aktiver Partition

Prä-OS: Bootmanager

MBR



GPL mbr

- Paket `mbr` in Debian & Ubuntu
- Installation mit `install-mbr`
- erlaubt Auswahl Boot-Partition 1-4 zur Bootzeit
- Auswahl einschränkbar

http://www.chiark.greenend.org.uk/~neilt/mbr/

Prä-OS: Bootmanager

Echte Bootmanager mit größerem Funktionsumfang passen nicht in den MBR:

- Im 1. Sektor nur erster Schritt, lädt eigentlichen Boot-Manager nach
- eigentlicher Bootmanager und Zusatzdaten liegen in
  - ungenutzten Sektoren (z.B. bis Zylinder 1)
  - in extra Bootmanager-Partition
  - in einer Partition mit Dateisystem
    - Kern an festem Block
    - Zusatzdaten häufig als normale Dateien

1. Sektor des Bootmanagers muss meist nicht MBR sein, kann auch erster Sektor einer Partition sein.

Empfehlung

- MBR: normalen MBR der Platte oder GPL-MBR
- Bootmanager in den ersten Sektor einer Partition

Vorteile

- mehrere Bootmanager möglich (Systemupgrade, Failover)
- MBR und Partitionstabelle leicht wiederherstellbar
- größter Vorteil in Dual-Boot-Umgebung:
  1. Windows-Installation überschreibt MBR
  2. Wahl der aktiven Partition ermöglicht wieder Zugriff auf Linux

... kann allerdings Linux nicht von erweiterter Partition booten

Ist leider nicht (mehr) Standard bei den Linux-Distributionen.

R|R|Z|N

syslinux / isolinux / pxelinux

- trotz Namen überhaupt nicht auf Linux beschränkt
- üblich im OSS-Umfeld für CDs, USB-Sticks, Netzwerk-PXE-Boot
- benutzt FAT oder ext2-Partition
- Menü-Auswahl möglich
- bootbar sind:
  - Chain-Booting in Laufwerke und Partitionen
  - Booten von Linux-Kerneln
  - Booten von Floppy-Images via memdisk-„Kernel“, das auch in anderen Bootmanagern wie grub nutzbar
  - .c32-Programme (quasi .com-Programme bei minimaler API)

syslinux-Erweiterungen

Comboot modules, Dateiendung `.c32`

menu	Implementation des Boot-Menüs
hdt	Hardware-Detection-Tool
chain	Chainbooting von anderen Partitionen, Dateien, Swapping von BIOS-Plattenkennungen
ifcpu	Boot in Abhängigkeit von 64Bit, VMX, smp

R|R|Z|N

Bootmanger für CD-Rom / USB

Eigentlich veraltet

Es gibt Bootmanager, die für Systeme ohne CD-Rom oder USB-Support im BIOS gedacht sind (unterstützen Chain-Booting dieser Medien).

SBM `http://sourceforge.net/projects/btmgr`

PLoP `http://www.plop.at/de/bootmanager.html`

Veränderter Einsatzzweck

Einbindung in einen Bootmanager wie grub oder insbesondere im Netz bei pxelinux kann sinnvoll sein:

- recht freie Auswahl des Bootmediums ohne vorherige Menüdefinition
- nachträgliches Chain-Booten auf CD-Rom oder USB

Bootloader

Wann das OS-spezifische Booten des Betriebssystems selbst anfängt oder ob es noch ein Bootmanager ist ... die Grenzen sind fließend:

Bei MS-Dos enthält der 1. Sektor der Partition oder Floppy einen Boot-Code, der nur die Dateien `IO.SYS` & `MSDOS.SYS` laden kann.
⟶ OS-Bestandteil

Bootloader moderner Betriebssysteme können eigentlich alle auch mind. Chain-Booting anderer Bootloader.
⟶ „OS-spezifische Bootmanager"

NT-Loader NTLDR (NT bis einschl. Server 2003)

Konfiguration in der Datei `C:\boot.ini`:

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="MS Win XP Pro" /fastdetect
C:\ubuntu.bin="Ubuntu Linux"
```

- Plattennumerierung bezieht sich auf BIOS-Reihenfolge
- Chain-Booting nach Linux mit Kopie Startsektor in Datei möglich
- NTLDR & `boot.ini` auf separatem Medium (Floppy, per PXE) möglich

http://www.winfaq.de/faq_html/Content/tip0000/onlinefaq.php?h=tip0408.htm

BootMGR (Vista, 2008, Windows-7)

Konfiguration:

- abgelegt in Registry-artiger Datei Boot Configuration Data (BCD)
- veränderbar über Kommandozeilen-Tool `bcdedit`
- oder Freeware EasyBCD (`http://neosmart.net/dl.php?id=1`)
- relativ schlecht dokumentiert

BootMGR ersetzt den ersten Teil des NTLDR (Boot-Manager-Funktionalität), weitere Lade-Funktionalitäten separiert:

`winload.exe` starten des Windows-Kernels, HAL etc.

`winresume.exe` fortfahren nach Hibernation

lilo

traditioneller LInuxLOader, der nur noch wenig Verwendung findet

- kann auch andere Dinge als Linux laden
- hat eine Menüauswahl
- versteht keine Dateisysteme
  - kann nur vorgefertigte Einträge booten
  - Konfiguration normalerweise in `/etc/lilo.conf`
  - benötigt fixe Lage (in Sektoren/LBA) der Kernel etc.
  - deshalb nach Änderungen immer Aufruf von `lilo` notwendig
  - kann aber deshalb auch von ungewöhnlichen Dateisystemen booten

⟶ auf üblichen Dateisystemen eher nicht mehr verwenden

Grub

wohl der derzeit gebräuchlichste Bootloader für Linux-Systeme

- Stage 1 kann im MBR oder auch in der Partition installiert werden
- Versteht im Stage 1.5 gängige Dateisysteme (u.A. ext,fat)
  - Stage 1.5 muss feste Lage haben (hinter MBR oder Dateisystem)
  - Stage 2 dann nicht mehr
- Zusatz- & Config-Dateien normalerweise in `/boot/grub` oder in eigener Boot-Partition
- Konfiguration in Datei `menu.lst`
- während des Bootens dynamisch eingeb- & änderbare Booteinträge
- Chain-Booting, Disk-Swap
- separate Anpassungen: PXE, Boot-Logo, Grub4Dos

Grub kann nicht mit UUIDs umgehen, benutzt BIOS-Reihenfolge der Platten.

Grub in Debian/Ubuntu

Die Datei `menu.lst` wird von Debian z.T. mitverwaltet, z.B. bei Kernel-Updates:

- Programm `update-grub` verwaltet für Kernels `/boot/vmlinuz-*` Booteinträge in `menu.lst`
- Optionen für `update-grub` stehen in `menu.lst` hinter
  `## ## Start Default Options ##`
- diese sind für grub selbst auskommentiert, z.B.
  ```
  # kopt=root=LABEL=DebRoot
  # defoptions=vga=794 splash
  ```

grub-2

Eigentlich noch nicht als stabil released, aber z.B. in Ubuntu 9.10 der Standard.

Änderungen/Zusätze:

- Stage 1 in den MBR oder 1. Partitionsektor installierbar (Warnung?)
- Stage 2 besteht aus `kernel.img` und Modulen `*.mod`, Notwendiges (inkl. Rescue) wird bei Installation als `core.img` zusammengepackt, was dann über die fixe Block-Adresse geladen wird
- weitere Module & Konfiguration dann als Dateien über das Dateisystem
- Konfiguration in Datei `/boot/grub/grub.cfg`
- echte Skriptsprache (lua-Integration)
- Unterstützung für Boot-Logos, Themes und mehrere Sprachen
- wohl UUID- und LABEL-Unterstützung für Zusatzmodule, Linux-Kernel und InitRD über `search`-Befehl

Partitionsangabe

Für die root-Partitionsangabe für den Kernel und in `/etc/fstab`:

device Geräte-Nodes in `/dev`:
`/dev/sda1`, `/dev/hdb2`

uuid über im Dateisystem verankerte IDs:
`UUID=9921a850-8f94-4502-a553-586807a8861c`

label im Dateisystem gesetzte Namen (beim Formatieren, `e2label`):
`LABEL=swap`, `LABEL=ub910-root`

UUID inzwischen häufig der Standard, LABEL einfacher für den Admin.
Beides im Gegensatz zum Device unabhängig von Reihenfolge der Geräteerkennung.
aber: Bootloader nutzt BIOS-Bennung, kennt keine UUID oder LABEL

Änderung auf LABEL=

Für Systeme, die noch mit /dev-Bezeichnern arbeiten, bietet sich die Migration zu Labeln an:

1. Benennung (hier DebRoot,Swap) der Partitionen, z.B.:

   root `e2label /dev/sda1 DebRoot`

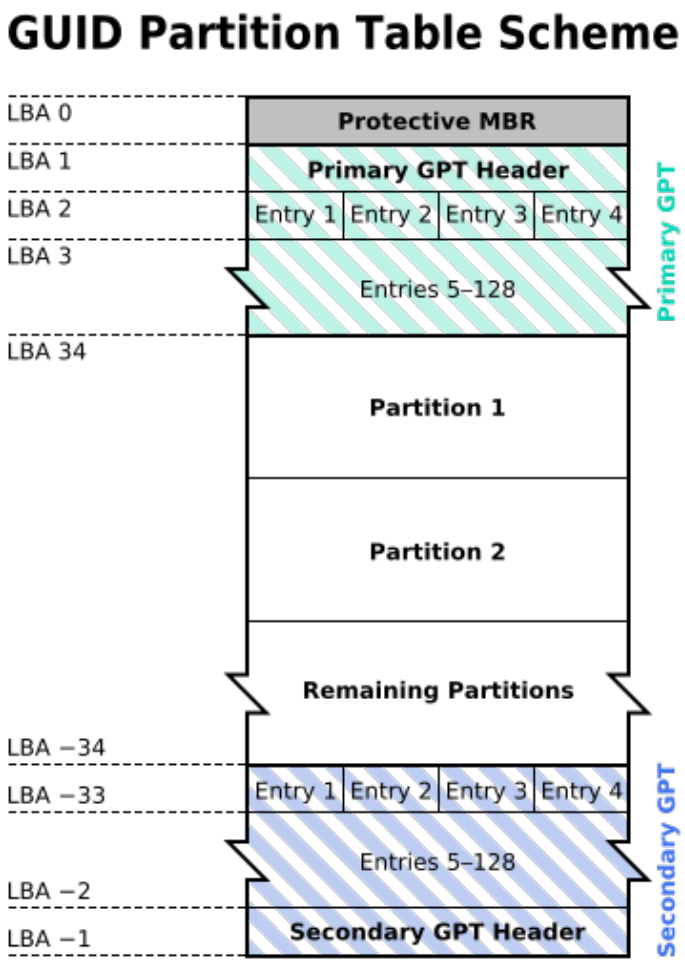
   swap swapoff -a; mkswap -L Swap /dev/sda3; swapon -a

2. Änderung in der Datei `/etc/fstab`
   Ersetzen der Bezeichner `/dev/sda?` in der ersten Spalte durch `LABEL=DebRoot` bzw. `LABEL=Swap`

3. Anpassung des Bootloaders, Angabe der Kernel-Root-Partition als `root=LABEL=DebRoot`
   - für grub in `/boot/grub/menu.lst`
     (Debian/Ubuntu: Einträge & `kopt`-Variable anpassen)
   - für grub-2 in `/boot/grub/grub.cfg`

GUID Partition Table

- manchmal auch als EFI-Partitionierung bezeichnet,
  da zunächst zusammen mit EFI-Bios (s.u.) spezifiziert
- enthält bedingt Kompatibilität zu bisheriger Partitionierung (s.u.)
- Belegt die ersten 34 Sektoren, zudem Kopie am Plattenende
- GPT erforderlich für Storage $>$ 2 TByte
- bis zu 128 Partitionen, 18 ExaBytes (EB: 10^6 bzw. 2^{20} TB)
- unterstützt auch Sektorgrößen $>$ 512 Bytes
  (erste Consumer-Platten mit intern 4 kB-Blöcken in 512 Byte-Emulation)
- Partitionstypen nicht mehr ein Byte sondern GUID
- Partitionen haben individuelle GUIDs, mit Namen versehbar



Quelle: Wikipedia-Artikel zu GPT

MBR in GPT-Definition

In der GPT-Definition ist MBR weiterhin vorgesehen, um eine gewisse Kompatibilität zu wahren.

Protective MBR

Im GPT-Standard ist die ganze Platte oder 2TByte mit einer Partition im MBR belegt (Typ EEh). Dadurch erscheint Platte herkömmlichen Partitionierungstools und Betriebssystemen als belegt.

An dem Typ EEh erkennt GPT-unterstützende Software, dass die Platte nicht im MBR-Modus sondern GPT-Modus partitioniert ist.

Hybrid MBR

Bis zu drei der GPT-Partitionen sind zusätzlich wie herkömmlich primäre Partitionen im MBR verzeichnet. GPT-Teil ab Sektor 2 ist meist als Reservierungs-Partition mit Typ EEh geführt.

- auch konventionell definierte durch Legacy-OS nutzbar
- diese natürlich nur innerhalb der ersten 2 TByte
- nie mit Legacy-Partitionierer bearbeiten
  - Definitionen müssen übereinstimmen
  - GPT-belegter Platz kann herkömmlich frei scheinen
- herkömmliche erweiterte Partitionen theoretisch möglich, von keinem Tool unterstützt
- Tools: Boot Camp (Mac), gptsync (von rEFIt), GPT-Fdisk (gdisk)
  Stichworte/Bezeichnungen: sync / map gpt to msdos

aber: GPT-Partitionierer machen das evt. kaputt, nicht GPT-Standard!

`http://www.rodsbooks.com/gdisk/hybrid.html`

Betriebssystem-Unterstützung

MacOSX auf Intel-Plattform (GPT dort Standard)

Windows
- Win XP-64 Bit, Server 2003 als Datenpartition (nicht Boot)
- Vista, Server 2008, Win 7 nativ; Boot bei EFI-Systemen

Linux ja, aber richtige Tools verwenden (nicht normales fdisk)

Partitionen, die im kompatiblem MBR ebenfalls verzeichnet sind, sind auch von nicht-GPT-kompatiblen Betriebssystemen benutztbar.

- Das ist aber eigentlich nicht GPT-Standard & anfällig (s.o.)
- Windows erkennt dann wohl reine GPT-Partitionen nicht.

Einsetzbarkeit

- für Nutzung mit EFI-BIOS (Mac) zwingend
- Durchaus schon vorkommend auch bei kleineren Platten (ab Werk), dann aber nur in Hybrid-Partitionierung
- eher kein Problem, wenn Partitionierung kompatibel zu MS-DOS-MBR:

Achtung: Partitionierungstool muss GPT sprechen

- diskpart von Windows
- parted, GPT-disk (gdisk) von Linux

$\longrightarrow$ außer bei EFI lieber vermeiden, Vorsicht bei hybrid

$\longrightarrow$ RAID-/SAN-Storage in Laufwerke $\leq$ 2 TB aufteilen

Extensible Firmware Interface

seit 2005 eigentlich *Unified EFI (UEFI)* genannt, EFI aber weiterhin vorherrschende Bezeichnung.

Verbreitung

- Itanium-Plattform
- alle x86-basierten Apple-Macintosh
- für Gäste seit Virtualbox 3.1 (experimentell)
  `VBoxManage modifyvm` *`vmname`* `-firmware efi`
- als Alternativ-BIOS für Tests auf wenigen Motherboards, teilweise wählbar in BIOS-Setup (eher Server-Hardware)

⟶ eher noch kaum zu finden, Ausnahme

Eigenschaften

technisch:

- 64-Bit-Modus
- Netzwerk-Support: Fernwartung & PXE
- Hardware-API für Betriebssysteme, Treiberintegration als Module
- integrierte Shell & Ausführung von EFI-Anwendungen
- EFI-Traps für I/O-Operations: OS-aushebelnd, u.A. für DRM
- Boot-Auswahl, die Boot-Loader überflüssig macht
- nur noch GPT-Support, nicht mehr traditionellen „MS-Dos"-MBR
- Kompatibilitätsmodul zur PC-BIOS-Emulation

wichtig fürs Marketing:

- Grafik-Modus und Maus-Unterstützung
- schnellere Boot-Zeit (erhofft)

Erweiterungen / Boot-Loader auf Festplatte:

- extra EFI-Partition, (V)FAT als Dateisystem

Betriebssystem-Support

nativ:

Mac	Intel-Macs benötigen EFI, daher alle x86-MacOSX-Versionen, ebenso „Windows-Bootloader“ Bootcamp
Windows	alle 64-Bit-Versionen seit Vista (Vista, 2008, Win-7)
Linux	seit Kernel 2.6.25 mit Bootloader elilo oder Grub-2 (EFI-Build)
?	mir unklar: Hypervisors (VMWare, Hyper-V, XEN); BSD-Varianten; Opensolaris

Und mit BIOS-Kompatiblitätsschicht (CSM; eigentlich in allen x86-UEFIs vorhanden) alle PC-Betriebssysteme.

MBR

Malware kann überall an- und eingreifen. Hauptsächlich relevant ist derzeit der Bootsektor (meist im MBR):

früher Bootsektor-Viren zu DOS-Zeiten

heute Rootkit-Hooks, die vor dem Betriebssystem selbst aktiv werden oder bereits vor dem Booten patchen

MebRoot: derzeit relativ weit verbreitetes, kaum zu detektierendes Rootkit

Torpig: häufig in Verbindung mit MebRoot auftretend

$\longrightarrow$ Booten von schreibgeschütztem Medium (z.B. SD-Card)

Hypervisor

Rootkits können auch in Form einer Virtualisierung auftreten. Rootkit läuft außerhalb des eigentlich infizierten Systems, das nur noch als Gast auf dem Hypervisor des Rootkits ausgeführt wird.

Bisher so kaum in der „freien Wildbahn“, Beispielimplementationen gibt es aber (u.A. Blue Pill).

EFI / Firmware

Malware im BIOS oder Erweiterungen bereits beispielhaft realisiert. Deutlich einfacher wird es bei EFI dank integrierten Treibern, API und Erweiterbarkeit & Persistenz in der EFI System Partition (ESP).

⟶ TPM könnte helfen