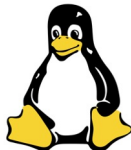
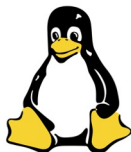


Booting Linux mit UEFI

FraLUG

Laura Liparulo

28.10.2025



Agenda

- Der Bootvorgang
- BIOS and beyond
- UEFI
- UEFI-Shell
- UEFI Troubleshooting - Fehlerbehebung

Der Bootvorgang

Einschalten...

So starten Sie die BIOS/UEFI-Firmware:

1. Drücken Sie den Einschaltknopf.
2. Das Motherboard wird mit Strom versorgt und startet die CPU.
3. Die CPU löscht alle alten Daten aus allen Registern und startet mit folgendem:

```
IP 0xffff0  
CS Selector 0xf000  
CS Base 0xffff0000
```

(Position, an der die CPU die Ausführung des ersten Befehls erwartet)

1. Der Sprungbefehl, der auf einen BIOS/UEFI-Einstiegspunkt verweist, wird ausgeführt
2. **Die Firmware startet den Bootloader des Betriebssystems...**

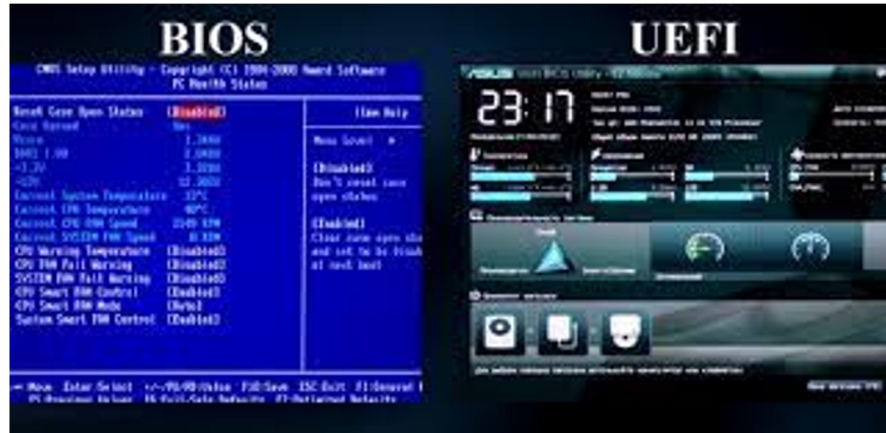
Der Bootvorgang

Eine vereinfachte Darstellung des Bootvorgangs sieht folgendermaßen aus:

1. Die Firmware des Rechners (BIOS, UEFI usw.) lädt einen Bootloader und führt ihn aus.
2. **Der Bootloader findet das Kernel-Image** auf der Festplatte, lädt es in den Arbeitsspeicher und startet es.
3. Der Kernel initialisiert die Geräte und deren Treiber.
4. Der Kernel mountet das Root-Dateisystem.
5. **init/systemd** wird gestartet.
6. Der Benutzerbereich wird gestartet.
7. usw.

Was ist eine Firmware ?

- Eine im non-volatile (nicht flüchtiger) Speicher der Hardware eingebettete Software.
- Die Firmware startet den Bootloader eines Betriebssystems.
- UEFI (Unified Extensible Firmware Interface) und BIOS (Basic Input/Output System) sind Firmware.
- Die UEFI-Firmware-Methode hat das BIOS in den meisten IBM-kompatiblen Computern ersetzt.



Die Firmware läuft

1. Es wird ein schneller **Hardwaretest**, ein sogenannter **Power-On Self-Test (POST)** durchgeführt.
2. Die Firmware sucht nach einem **Bootloader-Programm**, das von einem bootfähigen Gerät ausgeführt werden kann.
3. Der Bootloader wird ausgeführt und ermittelt, welches Linux-Kernel-Programm geladen werden soll.
4. Das **Kernel**-Programm wird in den Speicher geladen.
5. Das System wird vorbereitet, z. B. durch das Mounten der Root-Partition.
6. Das Initialisierungsprogramm wird ausgeführt.
7. Für den Systembetrieb erforderlichen Hintergrundprogramme werden gestartet (grafischer Desktop-Manager für Desktops, Web- und Datenbankanwendungen für Server)

BIOS and beyond

BIOS

Firmware auf älteren PCs, die:

- die Hardwarekomponenten des Systems initialisiert und testet (Power-on self-test)
- einen Bootloader von einem mass-storage Gerät lädt, der dann einen Kernel initialisiert.

Das BIOS prüft zunächst den Bootloader im **Master Boot Record (MBR)**.

Seit 2020 wird es auf Intel nicht mehr unterstützt.

Die letzte Version von Microsoft Windows, die die BIOS-Firmware offiziell unterstützt, ist Windows 10.

Windows 11 erfordert ein UEFI-kompatibles System.

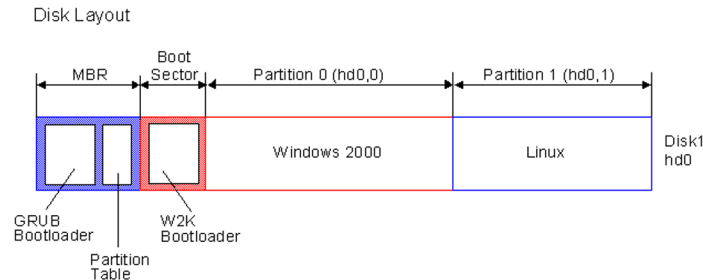
Bootloader

- Ein kleines Programm, das die erforderliche Hardware initialisiert und das vollständige Betriebssystem ausführt.
- Befindet sich normalerweise an einem anderen Speicherort auf derselben Festplatte, manchmal aber auch an einem anderen Ort.
- Verfügt in der Regel über eine Konfigurationsdatei (Betriebssystemspeicherorte, Auswahlmenü usw.).

Beim Booten von einer Festplatte müssen Sie festlegen, von welcher Festplatte und von welcher Partition das BIOS das Bootloader-Programm laden soll. Dies geschieht durch die Definition eines Master Boot Record (MBR).

Master Boot Record

- der erste Sektor auf einer Festplatte, der Informationen zur Festplattenpartitionierung und zum Betriebssystemspeicherort speichert.
- Die maximale Partitionsgröße einer MBR-formatierten Festplatte beträgt 2 TB.
- Die maximale Anzahl primärer Partitionen in einem MBR-Partitionssystem beträgt 4.
- Die geeignete Größe für die `/boot` Partition eines Linux-Systems liegt zwischen 100 und 500 MB.



Wo kann ein Bootloader gefunden werden

Das BIOS/UEFI der Workstation versucht, an den folgenden Orten einen Bootloader zu finden:

- internal hard drive
- external hard drive
- CD / DVD drive
- USB memory stick (z.B. WebFAI für Tuxedo)
- ISO file
- network server (anhand NFS, HTTP, or FTP)



Bootloaders für BIOS

- LILO (alt)
- Grub legacy (seit 1999)
- GRUB2 (aktuell, seit 2005) - auch für UEFI

Chainloading

Ein primäres Bootloader-Programm kann auf ein sekundäres Bootloader-Programm verweisen, das das Laden mehrerer Betriebssysteme ermöglicht (**mehrere Linux-Kernel-Versionen sind zulässig**).

Der MBR ist klein und kann nur auf eine einzelne kleine Linux-Kernel-Programmdatei verweisen.

Die Kernel-Datei des Betriebssystems wird im Bootsektor einer separaten Partition gespeichert.

Es gibt **keine Größenbeschränkung für die Kernel-Boot-Datei**.

Das Bootloader-Programm muss nicht direkt auf eine Kernel-Datei des Betriebssystems verweisen; es kann auf jedes beliebige Programm verweisen, auch auf andere Bootloader-Programme.

Probleme mit BIOS

- Veraltetes System, das nur 8 KB - 32 MB Speicherplatz benötigt.
- Es kann nur die Daten eines Sektors von einer Festplatte in den Speicher lesen, um ausgeführt zu werden.
- Es funktioniert nur mit Festplatten bis zu 2,2 Terabyte.
- Es ist einfach nicht für die heutige Technologie ausgelegt.

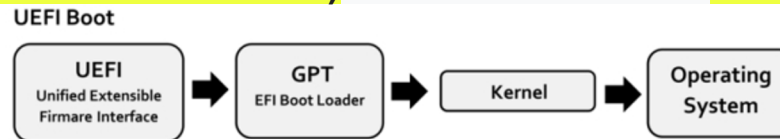
UEFI

Unified Extensible Firmware Interface (UEFI) - Geschichte

- 1998 – Die UEFI-Firmware wurde von Intel entwickelt, um einige der Einschränkungen des BIOS zu beheben.
- 2005 – Das UEFI-Forum wurde als **gemeinnützige Organisation** in Washington gegründet.
- Fördermitglieder: AMD, AMI, Apple, Dell, HP, IBM, Insyde, Intel, Lenovo, Microsoft, Phoenix.
- 2005 – Übernahme als Standard.
- Januar 2006 – Version 2.0 wurde veröffentlicht (Quellcode verfügbar).
- 2017 – Das **UEFI Security Response Team (USRT)** wurde gegründet.
- Dezember 2024 – Die neueste UEFI-Spezifikation, Version 2.11, wurde veröffentlicht.

UEFI - So funktioniert es

- Eine dedizierte Festplattenpartition, die sogenannte **EFI-Systempartition (ESP)**, dient zum Speichern des Bootloaders anstelle eines einzelnen Bootsektors.
- Bootloaderprogramme jeder Größe sind möglich.
- Die Möglichkeit, mehrere Bootloaderprogramme für mehrere Betriebssysteme zu speichern.
- Das alte Microsoft **FAT32-Dateisystem** dient zum Speichern der Bootloaderprogramme.
- EFI wird typischerweise im Verzeichnis **/boot/efi/** eingebunden.
- Die Bootloaderdateien werden üblicherweise mit der Dateinamenerweiterung **.efi** gespeichert.
- Ein Mini-Bootloader dient zum Konfigurieren der zu startenden Bootloaderprogrammdatei.
- **GPT-Partitionen (GUID-Partitionstabelle)** anstelle von MBR – **bis zu 128 Partitionen!**



UEFI - Features

- Kann Festplatten mit einer Kapazität von bis zu **9,4 Zettabyte** ansprechen.
- Zugriff auf die gesamte Hardware des PCs (wie ein Mini-Betriebssystem).
- Mini-Bootloader (**Bootmanager**) zur Konfiguration des zu verwendenden Bootloader-Programms.
- **Beschleunigte Hardware-Initialisierung** für ein sofortiges Einschalten.
- Sicherheit und Authentifizierung vor dem Booten des Betriebssystems.
- **Fernzugriff** auf den PC für Fehlerbehebung und Wartung.
- Ein Linux-Betriebssystemkernel kann direkt ohne speziellen Bootloader geladen werden.
- Das GPT-Partitionierungsschema ist Teil des UEFI-Standards.
- Windows 8 verwendet UEFI bereits standardmäßig!

UEFI-compatible bootloaders

- **systemd-boot** (einfach, textbasiert)
- **GRUB 2**
- **rEFInd** (automatische Erkennung bootfähiger EFI-Anwendungen und Kernel)
- **Windows Boot Manager** (Teil des Windows-Betriebssystems)
- **EFISTUB** (Kernel wird direkt von der UEFI-Firmware ohne separaten Bootloader gebootet)
- **U-Boot** (kann von jedem Datenträgertyp booten und jedes Boot-Image laden)

Systemd-boot loader

- Ursprünglich **gummiboot** genannt, ist ein Systemd-Bootloader-Programm für Linux-Distributionen, die den Systemd-Bootloader verwenden.
- Es generiert ein Menü mit Boot-Image-Optionen und kann jedes UEFI-Boot-Image laden
- Es ist für den Einsatz auf UEFI-Systemen mit **Secure Boot** konzipiert.
- <https://systemd.io/BOOT/>
- <https://github.com/systemd/systemd>
- https://uapi-group.org/specifications/specs/boot_loader_specification/

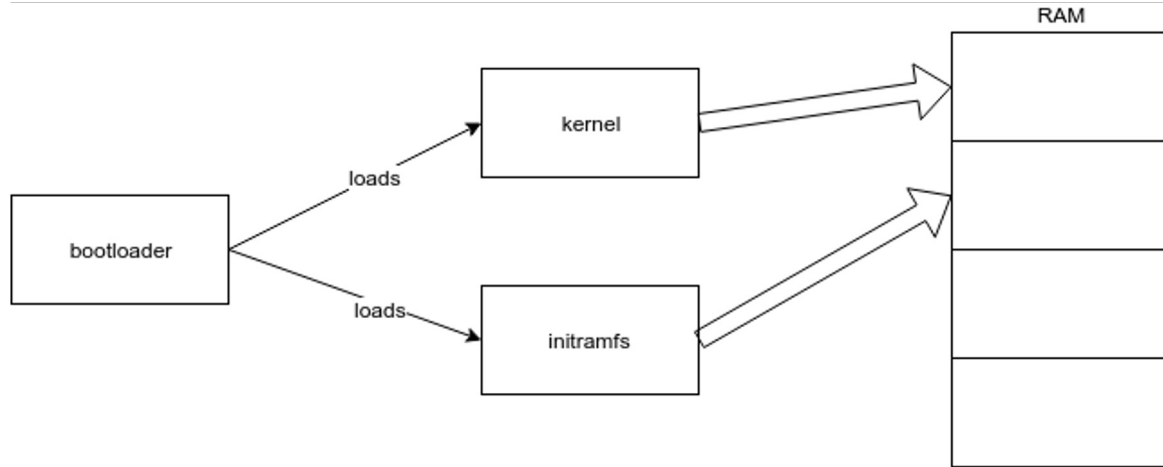
UEFI Bootvorgang

1. Einschalten der Maschine
2. UEFI-Firmware-Initialisierung (Hardware und Boot-Einträge aus dem NVRAM)
3. Bootloader-Auswahl
4. Die UEFI-Firmware identifiziert einen Bootloader (z. B. anhand eines Boot-Eintrags oder des Standardpfads \\EFI\\BOOT\\BOOTx64.EFI).
5. Kernel und initiale RAM-Disk (**initramfs**) werden in den Speicher geladen
6. Kernel-Ausführung - Der Bootloader übergibt die Kontrolle an den Kernel, der dann die Systeminitialisierung startet.

initramfs

- initramfs (Initial RAM Filesystem) fungiert während des Bootvorgangs **als temporäres initiales Root-Dateisystem.**
- Enthält die notwendigen Treiber für die Interaktion des Kernels mit der Systemhardware.
- **Beschleunigt den Bootvorgang (nur Hardwareerkennung, minimales Laden von Modulen und Geräteerkennung).**
- Es unterstützt den Kernel beim Zugriff auf die echte Root-Partition.
- Es bietet die notwendigen Module und Tools zum Mounten des echten Root-Dateisystems.

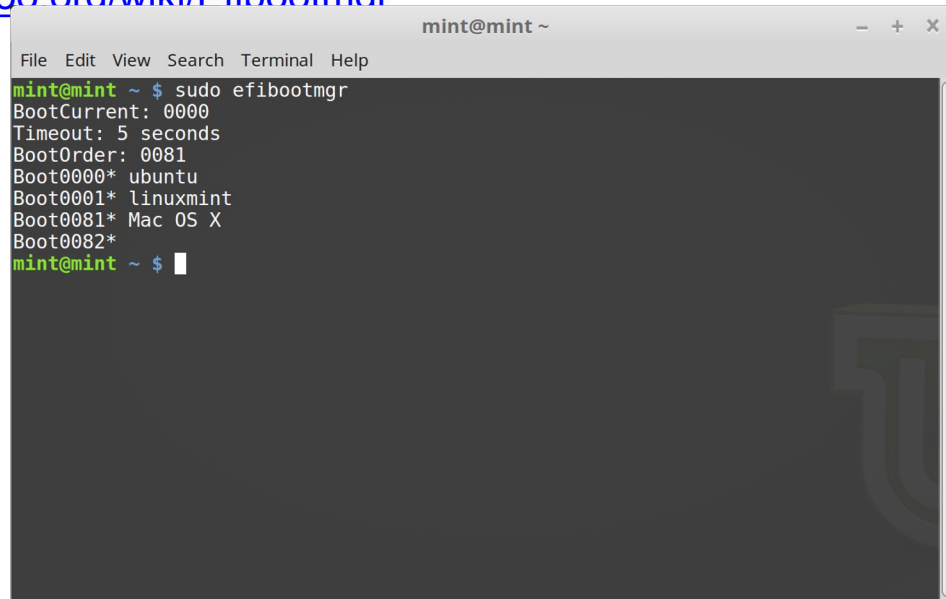
initramfs - how it works



Picture by [Friedemann Liohardt](#)

efibootmgr

- Ein Kommandozeilentool zur Verwaltung von UEFI-Booteinträgen.
- Interagiert mit der EFI-Firmware des Systems, die selbst als Bootmanager fungiert.
- <https://wiki.gentoo.org/wiki/Efiboottmgr>



```
mint@mint ~  
File Edit View Search Terminal Help  
mint@mint ~ $ sudo efibootmgr  
BootCurrent: 0000  
Timeout: 5 seconds  
BootOrder: 0081  
Boot0000* ubuntu  
Boot0001* linuxmint  
Boot0081* Mac OS X  
Boot0082*  
mint@mint ~ $
```

Secure boot

- Eine Sicherheitsfunktion, die die Integrität des Bootvorgangs überprüft.
- Sie stellt sicher, dass das System nur mit Software bootet, die mit einer anerkannten Signatur signiert ist.
- Eine Sicherheitsmaßnahme, die sicherstellt, dass Bootloader, Kernel und Treiber, die nicht manipuliert wurden und von vertrauenswürdigen Anbietern stammen, **nur signierte Bootloader zugelassen sind.**
- Details zur Signierung dieser Treiber finden Sie in der UEFI-Spezifikation.
- Auch im benutzerdefinierten Modus verfügbar, um zusätzliche öffentliche Schlüssel hinzuzufügen.
- Zusätzliche Schlüsselaustauschschlüssel (**Key Exchange Keys - KEK**) können einer im Speicher abgelegten Datenbank hinzugefügt werden.
- Unterstützt seit Windows 8, Fedora 18, Suse 12.3, Debian 10, Ubuntu 12.04 usw.

Ubuntu - Check secure boot is enabled

> mokutil --sb-state

```
laura@laura-VMware20-1:~$ mokutil --sb-state
SecureBoot disabled
laura@laura-VMware20-1:~$ █
```

> sudo apt-get install shim-signed grub-efi-amd64-signed

Konfigurieren Sie dann den sicheren Start in den UEFI-Einstellungen beim Booten

GRUB2 mit UEFI

- **BIOS**: Die Datei **grub.cfg** befindet sich im Verzeichnis **/boot/grub/** oder **/boot/grub2/**. (So können Sie GRUB Legacy und GRUB2 gleichzeitig installieren.)
- **UEFI**: Die Datei grub.cfg befindet sich meistens im Verzeichnis **/boot/efi/EFI/<distro-name>/**.

Kernel-Update: Eine neue Konfigurationsdatei wird generiert.

Der alte Kernel ist weiterhin als Boot-Option verfügbar.

GRUB 2 - Konfiguration für UEFI

Für GRUB2 sind folgende Konfigurationselemente zu beachten:

- **Menuentry**: Erste Zeile und Name jeder Boot-Option.
- **set root**: Festplatte und Partition, auf der sich die GRUB2-/boot-Verzeichnispartition im System befindet.
- **linux, linux16**: Kernel-Image-Datei im /boot-Verzeichnis zum Laden für das BIOS.
- **linuxefi**: Kernel-Image-Datei im /boot-Verzeichnis zum Laden für UEFI.
- **initrd**: Initiales RAM-Dateisystem für das BIOS.
- **initrdefi**: Initiales RAM-Dateisystem für UEFI.

Chainloading mit UEFI und GRUB

- Das Laden anderer Boot-Loaders anstelle von Kernel multiple boot wird problemlos unterstützt.
- Mehrere Bootloader in der EFI-Partition.
- Einzelne Partition mit einem MBR-Bootloader weiterhin möglich.

Beispiel: Die Option +1 weist Chainloader an, den ersten Sektor einer Partition zu laden. Sie können auch direkt eine Datei angeben.

```
menuentry "Windows" {  
    insmod chain  
    insmod ntfs  
    set root=(hd0,3)  
    chainloader +1  
}
```

Troubleshooting UEFI

Troubleshooting tips

- Boot logs prüfen (**dmesg**, **journalctl**)
- Suchen Sie nach **/sys/firmware/efi**: Wenn dieses Verzeichnis existiert, zeigt dies an, dass das System im UEFI-Modus gestartet ist.
- Boot-entries prüfen: Verwenden Sie **efibootmgr -v**, um die aktuellen UEFI-Boot-Einträge anzuzeigen.
- **Kernel-Parameter** überprüfen: Stellen Sie sicher, dass der Kernel nicht mit dem Parameter *noefi* gestartet wird, da dieser die EFI-Laufzeitdienste deaktiviert.

Wo kann man boot logs finden

Die aktuellsten Bootmeldungen finden Sie hier:

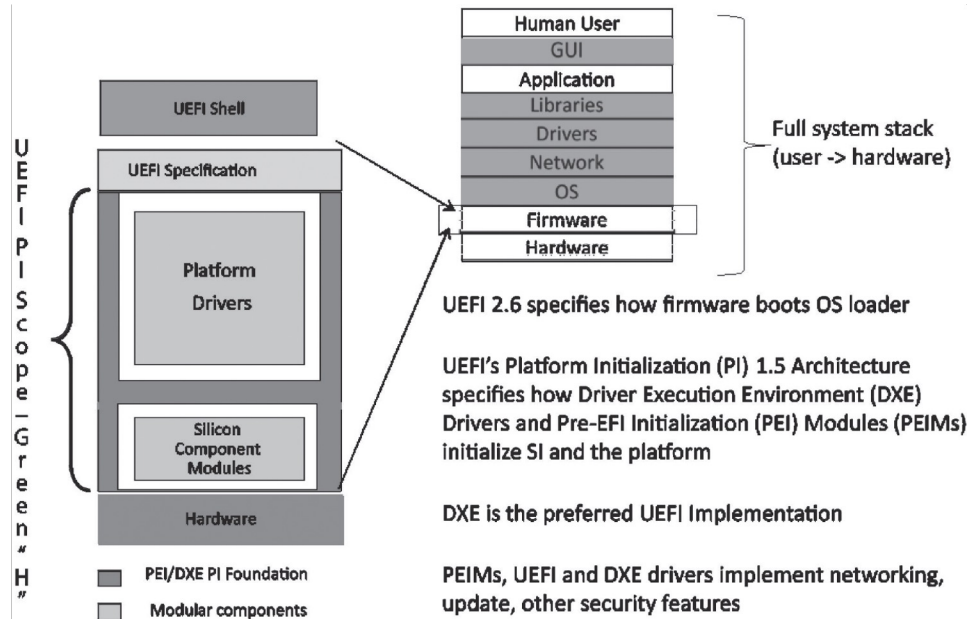
- `less /var/log/boot` (ex. Debian)
- `less /var/log/boot.log` (ex. Red-hat)
- **dmesg** (kernel ring buffer - neueste Nachrichten)
- **journalctl** (systems employing systemd-journald)
- Drücken Sie während des Bootvorgangs entweder die ESC-Taste oder Strg+Alt+F1

UEFI Shell

UEFI shell

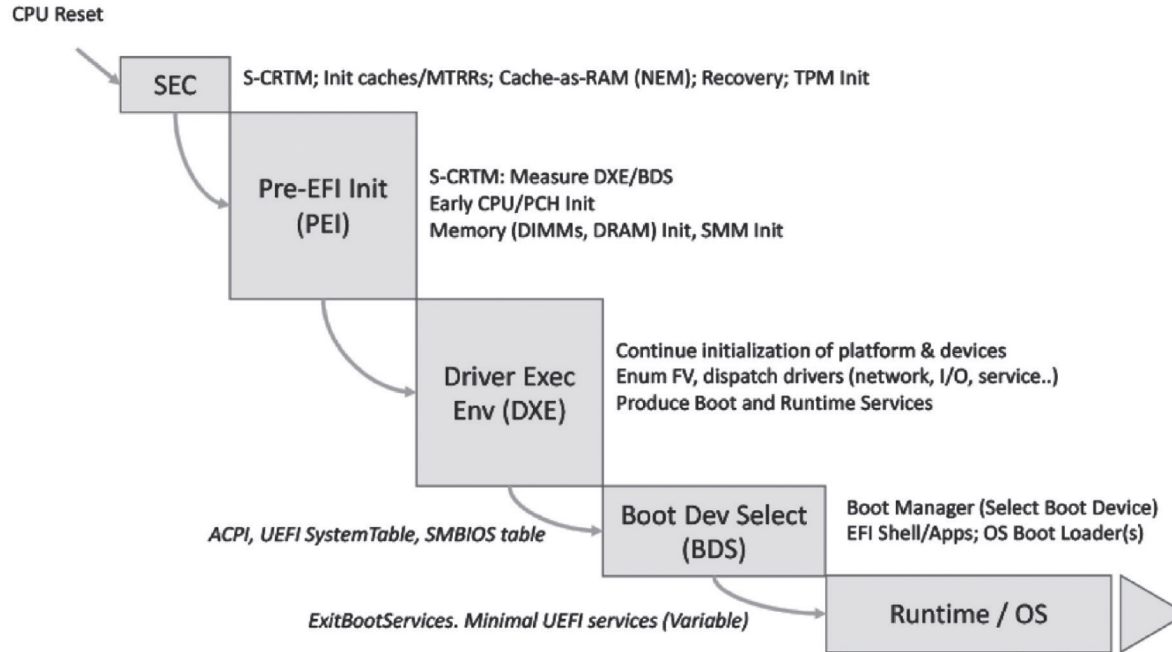
- **APIs + Kommandozeilenschnittstelle**
- Klein und nicht intrusiv
- Für die **Bare-Metal**-Umgebung
- Rohe Systemressourcen können eingesehen und abgerufen werden (z. B. mit MM- und PCI-Shell-Befehlen).
- Wichtig für die Systemwartung
- Bearbeitung, Vernetzung und andere Funktionen
- Universelle Umgebung für **Diagnose** und andere betriebssystemunabhängige Anwendungen.
- Anwendungen werden als PE/COFF-Programme formatiert und über den UEFI-Dienst LoadImage() in den Speicher geladen und verschoben sowie über den UEFI-Dienst StartImage() aufgerufen.
- Zwei Eingabeparameter
- Image-Handler
- Zeiger auf die UEFI-Systemtabelle
- Die Anwendung ist nicht mehr betriebsbereit, sobald das Hauptbetriebssystem oder der Hypervisor die Kontrolle übernimmt.

Zwischen der hardware and und dem OS



Picture from the book "Harnessing the UEFI shell"

UEFI Shell - wann ist sie verfügbar?



Picture from the book "Harnessing the UEFI shell"

Uefi shell - bcfg command

Der Befehl `bcfg` dient zur Verwaltung von Boot- und Treiberoptionen im NVRAM.

Sie können `bcfg` verwenden, um

- Boot-Optionen mit `bcfg boot dump` oder `bcfg boot dump -v` für eine ausführliche Ausgabe anzuzeigen,
- Boot-Einträge hinzuzufügen, zu entfernen oder zu ändern
- usw.

Beispiel:

Um eine neue Boot-Option hinzuzufügen, können Sie `bcfg boot add` mit einer Nummer, dem Dateipfad zum Loader und einem beschreibenden Namen verwenden.

UEFI - shell configuration commands

- **drvcfg** – ruft die Konfigurationsinfrastruktur der Plattform auf.
- **drvdiag** – ruft das Treiberdiagnoseprotokoll auf.
- **memmap** - zeigt die Speicherzuordnung der UEFI-Umgebung an.
- **dblk** – ermöglicht einem Skript die Interaktion mit dem zugrunde liegenden Blockspeichergerät, um den Inhalt eines oder mehrerer seiner Blöcke/Sektoren anzuzeigen.
- **dmem** – zeigt den Inhalt des System- oder Gerätespeichers an. Wenn keine Adresse angegeben ist, wird der Inhalt der EFI-Systemtabelle angezeigt. Andernfalls wird der Speicher ab einer bestimmten Adresse angezeigt. Dies ist besonders nützlich, um den Inhalt bestimmter Speicherbereiche anzuzeigen, wie z. B. den PCI-Konfigurationsbereich eines Geräts.
- **mm** – der Benutzer das E/A-Register, den Speicherinhalt oder den PCI-Konfigurationsbereich anzeigen oder ändern.
- usw.

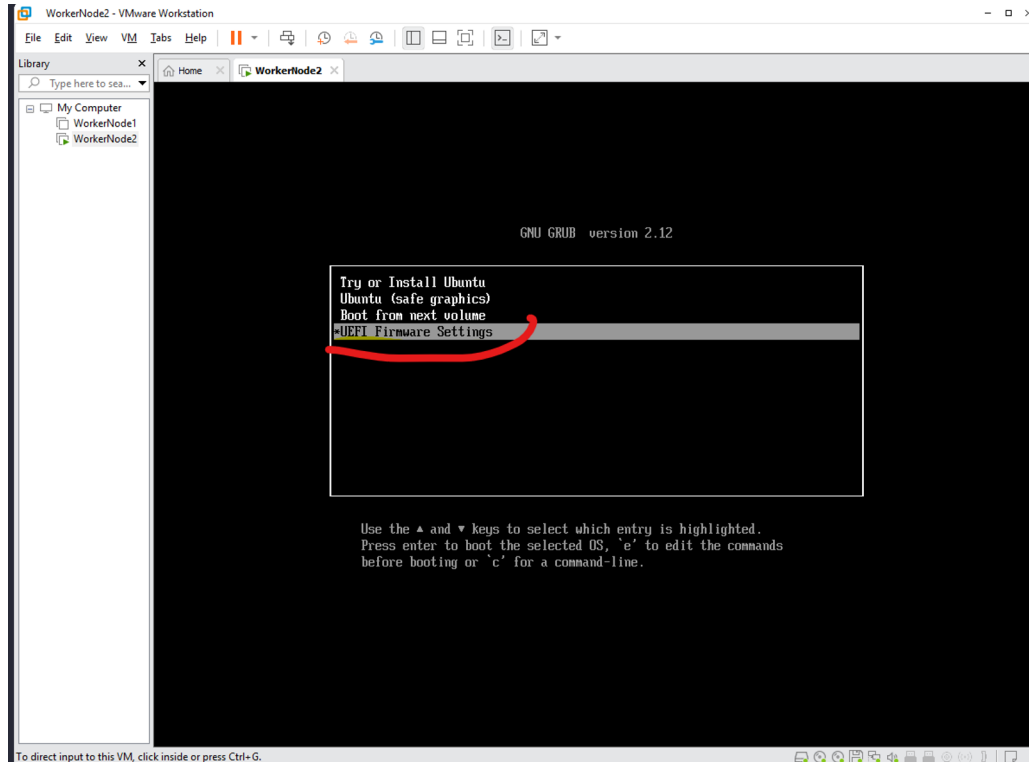
UEFI Shell Scripts

- UEFI interpretierte Programme (textbasierten Sprache)
- Dateierweiterung `.nsh`
- direkt von der UEFI-Shell unterstützt wurden
- Es enthält außerdem einzigartige Funktionen der Vor-Betriebssystem-Umgebung, wie beispielsweise:
 - standardized command output
 - redirection to and from environment variables.

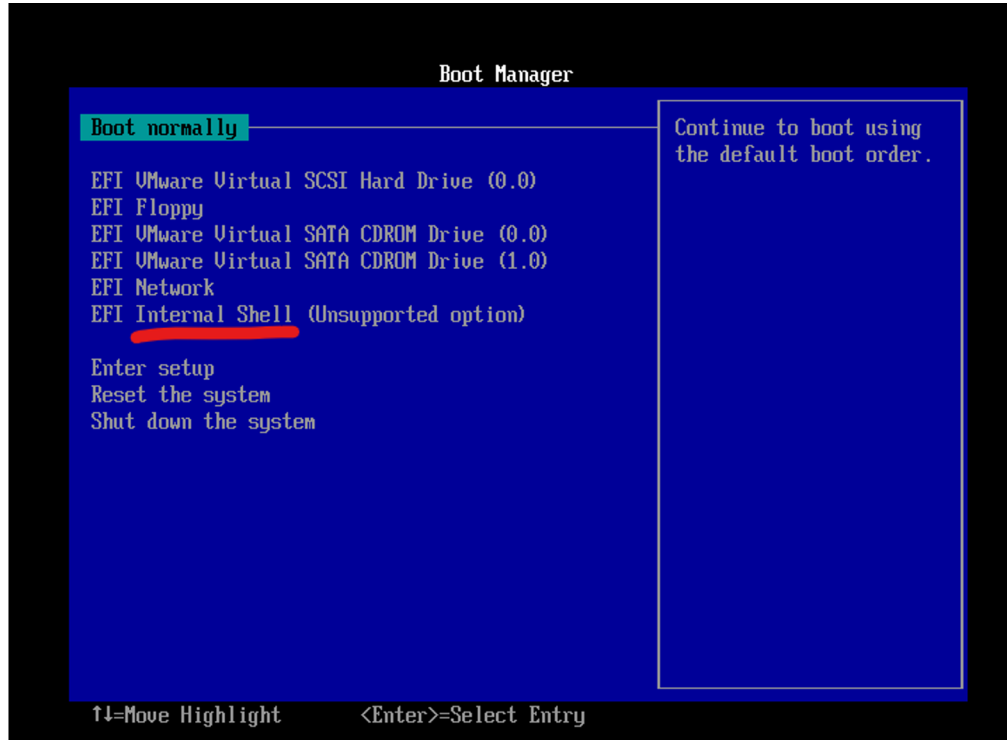
Demo on VM Ware - Boot mit UEFI

The image shows two overlapping windows from VMware Workstation. The background window is the main interface for 'WorkerNode2', displaying its configuration: 17.1 GB memory, 2 processors, 20 GB hard disk, and various other hardware settings. The foreground window is the 'Virtual Machine Settings' dialog, specifically the 'Hardware' tab. In this dialog, the 'Advanced' option is selected, and the 'Firmware type' is set to 'UEFI'. The 'Enable secure boot' checkbox is also visible and checked. The 'File locations' section shows the configuration path as 'C:\Users\liparu\Documents\Virtual Machines\Ubuntu 64-bit (3)\Ubuntu 64-bit (3)'. The 'Log' field is currently empty, indicating the VM is not powered on.

Demo on VMWare - Zugriff auf die UEFI settings im Menu



Demo on VMWare - UEFI Shell auswählen



Demo on VMWare - UEFI Shell benutzen

```
0FBDE138: CD E2 B8 OF 00 00 00 00-E3 BC B8 OF 00 00 00 00 *.....*
0FBDE148: E7 E2 B8 OF 00 00 00 00-4D BD B8 OF 00 00 00 00 *.....M.....*
0FBDE158: F3 E2 B8 OF 00 00 00 00-9A BC B8 OF 00 00 00 00 *.....*
0FBDE168: 0C E3 B8 OF 00 00 00 00-FE BC B8 OF 00 00 00 00 *.....*
0FBDE178: 19 E3 B8 OF 00 00 00 00-68 BD B8 OF 00 00 00 00 *.....h.....*
0FBDE188: 33 E3 B8 OF 00 00 00 00-80 BD B8 OF 00 00 00 00 *3.....*
0FBDE198: 46 E3 B8 OF 00 00 00 00-A1 BD B8 OF 00 00 00 00 *F.....*
0FBDE1A8: 5A E3 B8 OF 00 00 00 00-CB BD B8 OF 00 00 00 00 *Z.....*
0FBDE1B8: 67 E3 B8 OF 00 00 00 00-AA BB B8 OF 00 00 00 00 *g.....*
0FBDE1C8: 7F E3 B8 OF 00 00 00 00-1A BC B8 OF 00 00 00 00 *.....*
0FBDE1D8: 8E E3 B8 OF 00 00 00 00-D6 BB B8 OF 00 00 00 00 *.....*
0FBDE1E8: AB E3 B8 OF 00 00 00 00-46 BC B8 OF 00 00 00 00 *.....F.....*
0FBDE1F8: B6 E3 B8 OF 00 00 00 00-81 BB B8 OF 00 00 00 00 *.....*
0FBDE208: CF E3 B8 OF 00 00 00 00-F1 BB B8 OF 00 00 00 00 *.....*

Valid EFI Header at Address 00000000FBDE018
-----
System: Table Structure size 00000078 revision 00020046
ConLn (0FA05C9B) ConOut (0FA01E18) StdErr (0FA0179B)
Runtime Services      000000000FBDEB98
Boot Services        000000000FFE48A0
ACPI 2.0 Table       000000000DFD6000
SMBIOS Table         000000000E03C000

Shell> _
```

Bibliographie

