Der Bootvorgang

Phase 1: Der Bootloader grub

Pakete	grub
Services	-
Konfigurationsdateien	/boot/grub/grub.conf

RHEL 6 nutzt den grub Version 1 um auf einem x86 System einen Linux Kernel zu booten. Ein Teil des grub bootloaders wird in den Bootsektor einer Disk oder Partition kopiert, welcher im Grunde genommen nur aus einer einzigen LONGJMP Assembleranweisung besteht um aus einem unterstützten Filesystem weitere Stages nachzuladen.
Der bootloader schaut in die /boot/grub/grub.conf Konfigurationsdatei nach globalen Einstellungen und einer nachfolgenden Liste von Booteinträgen.

/boot/grub/grub.conf

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You do not have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /, eg.
#          root (hd0,0)
#          kernel /boot/vmlinuz-version ro root=/dev/hda1
#          initrd /boot/initrd-version.img
#boot=/dev/hda
serial --unit=0 --speed=115200
terminal serial

default=0
timeout=5


title CentOS (2.6.32-358.18.1.el6.x86_64)
    root (hd0,0)
    kernel /vmlinuz-2.6.32-358.18.1.el6.x86_64 root=/dev/md1 console=tty0 console=ttyS0,115200 nomodeset
crashkernel=512M SYSFONT=latarcyrheb-sun16 LANG=en_GB.UTF-8 KEYTABLE=uk
    initrd /initramfs-2.6.32-358.18.1.el6.x86_64.img

title CentOS (2.6.32-358.6.1.el6.x86_64)
    root (hd0,0)
    kernel /vmlinuz-2.6.32-358.6.1.el6.x86_64 root=/dev/md1 console=tty0 console=ttyS0,115200 nomodeset
crashkernel=auto SYSFONT=latarcyrheb-sun16 LANG=en_GB.UTF-8 KEYTABLE=uk
    initrd /initramfs-2.6.32-358.6.1.el6.x86_64.img
```

Die globalen Optionen sind schnell zusammengefasst: `default` legt fest welcher Eintrag standardmäßig gebootet wird, wenn der `timeout` von 5 sekunden ohne eine manuelle Auswahl verstrich. `serial` legt die erste serielle Schnittstelle als Standard mit einer Geschwindigkeit von 115200 baud fest und legt das `terminal` auf diese Schnittstelle.

Nach den globalen Optionen folgen die einzelnen Menüeinträge, die aus 4 Direktiven bestehen muss. Der `title` gibt einen String an, der exakt als auswählbarer Eintrag im Bootmenü erscheint. Gemäß Konvention sollte der Titel das folgende Format haben:

```
title OSNAME (KERNELVERSION-REVISION.OSRELEASE.ARCH)
```

Darunter folgen die `root, kernel` und `initrd` Direktiven, die zum booten eines Linux Systems zwingend erforderlich sind.

```
      root(hd0,0)
```

besagt, dass das Bootlaufwerk auf der ersten Festplatte in der ersten Partition liegt. Mit Bootlaufwerk ist, sofern vorhanden, eine getrennte /boot Partition oder falls diese nicht vorhanden ist die / Partition gemeint. Dies ist wichtig, da der Pfad zum Kernel und zur Inititalen Ramdisk relativ zum Bootlaufwerk angegeben wird;

```
      kernel /vmlinuz-KERNELVERSION-REVISION.OSRELEASE.ARCH [KERNELPARAMETER]
```

kann also das Kernelimage entweder unter /boot/vmlinuz-* oder aber eben genausogut unter /vmlinuz-* liegen. Gleiches gilt für den Pfad zur initialen RAM Disk (`initrd`), nicht jedoch für die Pfade innerhalb der KERNELPARAMETER.

```
      initrd /initramfs-KERNELVERSION-REVISION.OSRELEASE.ARCH.img
```

Man kann wenn das Bootmenü angezeigt wird einzelne Einträge on-the-fly editieren und so z.B. bestimmte Kernelfeature wie z.B. SELinux abschalten oder das System durch hintenanfügen eines runlevels bzw. eines s oder single für den rescue Modus (lediglich Kernel und Module werden geladen und eine Shell gestartet aber keine services ausgeführt) das System bis zu einem fest definierten Punkt starten.

Relevante Kommandos

Installation des grub in den Bootsektor einer Disk oder Partition

```
~]# grub-install <DEVICE>
```

Generieren eines grub Kennwortes

```
~]# grub-crypt [OPTION]
```

Phase 2: Die initiale RAM Disk

Pakete	dracut
Services	-
Konfigurationsdateien	/etc/dracut.conf /etc/logrotate.d/dracut /var/log/dracut.log

Nach dem Bootmenü wird nicht, wie man vielleicht vermuten könnte, direkt der Kernel gestartet, sondern ein sehr kleiner Kernel mit einem eigenen kleinen Filesystem in den Speicher geschrieben. Dazu wird ein kleiner Bereich mit dem initramfs formatiert und eingehängt. Aus diesem Filesystem heraus wird der spätere, echte root Filesystem einem fsck unterzogen und vorläufig readonly gemountet. Früher, wo die mögliche Hardware zum booten aus einer Festplatte oder sogar noch Diskette bestand, war eine initiale RAM Disk nicht nötig, heutzutage wo man von Devices verschiedenster Machart starten kann ist es wichtig die Hardware dynamisch konfigurieren zu können, was mit diesem Verfahren problemlos möglich ist bevor das eigentliche System darauf später zugreift.

Wichtig ist, dass das Image der initialen Ramdisk immer dem aktuellen Kernel und den zum booten notwendigen Modulen entspricht. Erfolgt die Installation des neuen Kernels mit den gängigen Paketwerkzeugen wird diese immer automatisch mit erstellt, ansonsten kann man diese auch mit den nachfolgenden Kommandos manuell erstellen.

Relevante Kommandos

Initramfs (initial ramdisk) generieren

```
~]# dracut
~]# dracut „initramfs-$(uname -r).img“ $(uname -r)
```

Phase 3: Kernel und Kernelmodule

Pakete	module-init-tools, Kernel siehe unten
Services	-
Konfigurationsdateien	/etc/sysconfig/modules/*.modules /etc/modprobe.d/*.conf

Anschließend wird der eigentliche Kernel gestartet und das root Filesystem normal, also readwrite, eingehängt. Der Kernel lädt dann je nach gefundener bzw. persistent konfigurierter Hardware bestimmte Treibermodule in den Speicher, initialisiert das System und übergibt die Steuerung danach an den init Prozess in Phase 5. Vorher wird in den allermeisten Fällen aber über den KERNELPARAMETER `crashkernel=` eine bestimmte Größe an Speicher ausgelassen, in dem dann ein sogenannter Crashkernel kopiert wird. Stürzt der Kernel eines Systems unwiederrufbar ab. Näheres zu den kleinen Ameisen, äh Bugs, die die Welt, äh, den Host in einen Friedhof verwandeln wollen finden wir passenderweise in Phase IV.

Für einen Kernel gibt es nicht nur ein einzelnes Paket, sondern eine handvoll spezifischer Teile, wie in der nachfolgenden Tabelle dargestellt. Das Kernel Image wird entweder via `yum` oder als eine der wenigen Ausnahmen mit „`rpm -i`“ installiert.

`kernel`	Paket mit Kernelimage
`kernel-debug`	Paket mit Kernelimage inkl. aller Debuggingsymbole
`kernel-devel`	Header und Makefiles um Kernelmodule kompilieren zu können
`kernel-debug-devel`	Header und Makefiles um Kernelmodule mit Debuggingsymbolen kompilieren zu können
`kernel-doc`	Kernel Dokumentation
`kernel-headers`	Include Dateien für die Applikationsentwicklung unter Linux
`kernel-firmware`	Firmware Dateien
`perf`	Support Skripte für das perf tool.

/etc/sysconfig/modules/bluez-uinput.modules

```
#!/bin/sh
if [ ! -c /dev/input/uinput ] ; then
        exec /sbin/modprobe uinput >/dev/null 2>&1
fi
```

Um Kernelmodule persistent zu laden empfiehlt es sich ein ausführbares Shell-Skript wie nachfolgend exemplarisch dargestellt unter `/etc/sysconfig/modules/*.modules` einzurichten, welches das Laden des Moduls übernimmt.

Modulparameter und aliase können in spezifischen `/etc/modprobe.d/*.conf` Dateien persistent gesetzt werden.

Relevante Kommandos

Aktuell geladene Module anzeigen

```
~]#  lsmod
```

Informationen über ein Modul anzeigen

```
~]#  modinfo <MODUL>
```

Modulparameter sortiert ausgeben

```
~]#  modinfo <MODUL> | grep ^parm | sort
```

Module mitsamt Abhängigkeiten laden

```
~]#  modprobe [-v] <MODUL>
```

Module mitsamt Abhängigkeiten entladen

```
~]#  modprobe -r [-v] <MODUL>
```

!!! Von der direkten Nutzung des insmod und des rmmod Kommandos wird dringend abgeraten. !!!

Phase IV: kdump Crash Recovery Service

Pakete	`kexec-tools`
Services	`kdump`

Konfigurationsdateien	`/etc/kdump.conf`
	`/etc/rc.d/init.d/kdump`
	`/etc/sysconfig/kdump`
	`/etc/udev/rules.d/98-kexec.rules`

In der `/boot/grub/grub.conf` wird in den Kernelparametern angegeben wieviel Speicher für den Crashkernel reserviert wird. Empfohlen sind 128MB + 64MB/TB Memory.
!!! Wird die Größe auf `auto` *gesetzt muss das System über mindestens 4GB Memory verfügen, ansonsten wird kein Memory reserviert. !!!*

!!! Um auf Intel Systemen einen verlässlichen Crashdump zu erzeugen muss IOMMU im BIOS abgeschaltet werden. !!!

Die Konfiguration des kdump Mechanismus wird in der Datei /etc/kdump.conf vorgenommen, welche mehr als ausführlich kommentiert ist. Es sollte mindestens angegeben werden, wohin der coredump geschrieben werden soll (lokales Verzeichnis, eine raw partition oder eine remote Verbindung) und sinnvollerweise auch noch ein Corecollector, der unnötige Bestandteile im Corefile herausfiltert und damit die Größe wesentlich reduziert.

/etc/kdump.conf

```
path /var/crash
core_collector makedumpfile -c --message-level 1 -d 23
default shell
```

Die Filter werden addiert, mögliche Werte sind:

```
1       Zero pages
2       Cache pages
4       Cache private
8       User pages
16      Free pages
```

Eine sehr nützliche Direktive ist `default`, welche ein Fallbackverhalten vorgibt, falls es zu Problemen bei dem dump gibt. Neben den Optionen reboot, halt und poweroff gibt es auch eine msh shell session um den core manuell zu erstellen zu können.

Phase 5: System V Init Skripte

Pakete	initscripts, chkconfig
Services	-
Konfigurationsdateien	/etc/X11/prefdm /etc/adjtime /etc/inittab /etc/networks /etc/rc.d/rc.local /etc/rc.d/rc0.d/S00killall /etc/rc.d/rc0.d/S01halt /etc/rc.d/rc1.d/S99single /etc/rc.d/rc2.d/S99local /etc/rc.d/rc3.d/S99local /etc/rc.d/rc4.d/S99local /etc/rc.d/rc5.d/S99local /etc/rc.d/rc6.d/S00killall /etc/rc.d/rc6.d/S01reboot /etc/sysconfig/init /etc/sysconfig/netconsole /etc/sysconfig/network-scripts/ifcfg-lo /etc/sysconfig/readonly-root /etc/sysctl.conf

Ist der Kernel mit seiner Aufgabe fertig startet er den init Prozess und fährt, sofern in der Kernel Zeile des grub Booteintrages nichts Anderes angegeben wurde in den default runlevel, der in der `/etc/inittab` eingetragen wurde.

```
id:3:initdefault:
```

RedHat Enterprise Linux kennt 7 runlevel, sowie einen zusätzlichen rescue Modus, welcher ausschliesslich über einen Kernelparameter im grub Menü ausgewählt werden kann.

s, S, single	Rescue Modus, keine Services o. Startskripte
0	Halt bzw. Poweroff
1	Single User Mode
2	
3	Multiuser Mode CLI
4	
5	Multiuser Mode GUI
6	Reboot

Relevante Kommandos

Aktuellen runlevel anzeigen

```
~]# runlevel
```

Services Konfigurationstools

```
~]# ntsysv [--level <RUNLEVEL>]
~]# system-config-services
```

Services starten, stoppen, restarten

```
~]# service <SERVICE> start
~]# service <SERVICE> stop
~]# service <SERVICE> restart
```

Aktuellen Status eines oder aller Services anzeigen

```
~]# service <SERVICE> status
~]# service --status-all
```

Automatischen Start von Services (SysV Init & xinetd) konfigurieren

```
~]# chkconfig --list [SERVICE]
~]# chkconfig <SERVICE> on [--level <RUNLEVEL>]
~]# chkconfig <SERVICE> off [--level <RUNLEVEL>]
```