Study-Guide: Booten, Initialisierung, Shutdown, Runlevels

Alles, was mit dem Booten und dem kontrollierten Herunterfahren des Systems zu tun hat, wird in diesen zwei Kapiteln besprochen. Booten des Systems Ändern des Runlevels und Niederfahren oder Neustart des Systems

Seite:	-= LinuxLernSystem =- (http://www.lpi-test.de)
Kurs:	LPIC-1 [102]
Buch:	Study-Guide: Booten, Initialisierung, Shutdown, Runlevels
Gedruckt von:	André Scholz
Datum:	Dienstag, 1 November 2005, 10:36 Uhr

Inhaltsverzeichnis

1.106 - Booten, Initialisierung, Shutdown, Runlevels

Alles, was mit dem Booten und dem kontrollierten Herunterfahren des Systems zu tun hat, wird in diesen zwei Kapiteln besprochen.

- Booten des Systems
- Ändern des Runlevels und Niederfahren oder Neustart des Systems

1.106.1 - Booten des Systems

Beschreibung: Prüfungskandidaten sollten in der Lage sein, das System durch den Bootprozeß zu führen. Dieses Lernziel beinhaltet das Übergeben von Kommandos an den Bootlader und die Übergabe von Optionen an den Kernel zum Bootzeitpunkt sowie das Überprüfen von Ereignissen in den Logdateien.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **dmesg**
- `/var/log/messages`
- `/etc/conf.modules` oder `/etc/modules.conf`
- LILO
- GRUB

Der Bootvorgang von Linux kann auf verschiedene Art und Weise gestartet, beeinflusst und nachvollzogen werden. Für die Anforderungen des LPIC 101 Examens genügt das Wissen um die Techniken, die mit dem Linux Loader (lilo) oder dem alternativen Bootmanager **grub** zu tun haben. Das soll auf dieser Seite näher erläutert werden.

Die Funktion von Bootmanagern wurde bereits in der Vorbereitung auf die erste LPI-Prüfung in Abschnitt 1.102.2 ausführlich dargestellt und wird hier als bekannt vorausgesetzt.

Kernelparameter angeben

Jeder Linux-Kernel kann beim Start, also bevor er geladen wird, noch Parameter übergeben bekommen, die sich auf die Art und Weise auswirken, wie er startet bzw. wie er sich zur Laufzeit verhalten soll. Es kann sich dabei um Parameter für bestimmte Geräte handeln (etwa SCSI-Adressen des Bootlaufwerks) oder um Eigenschaften des Kernels selbst wie z.B. die oben als Beispiel angegebene Anweisung bei einem Reboot einen Warmstart statt einem Kaltstart zu machen.

Kernelparameter für Geräte werden nur notwendig, wenn mit Hardware gearbeitet wird, die der Kernel nicht selbstständig erkennt. Ein Standard-PC mit Standard-Komponenten benötigt keine zusätzlichen Kernelparameter.

Es würde den Rahmen dieser Darstellung bei weitem sprengen, alle denkbaren Kernel-Parameter hier zu beschreiben, das ist für die LPI101 Prüfung auch gar nicht notwendig. Wenn die Kernel-Quellen installiert sind, findet sich eine Beschreibung aller unterstützer Parameter in der Datei `/usr/src/linux/Documentation/kernel-parameters.txt`. Wichtig ist die Tatsache, daß es möglich ist, bestimmte Parameter dem Kernel zu übergeben.

Wo werden diese Parameter übergeben, das ist jetzt die Frage. Es gibt mehrere mögliche Antworten, die hier kurz angefügt werden sollen:

Auf dem Bootprompt von LILO oder GRUB

Wenn der Bootprompt beim Start des Systems erscheint, dann erwartet der Bootmanager die Nennung des Systemnamens, das gestartet werden soll (also des in /etc/lilo.conf angegebenen Labels). Diesem Namen können Kernelparameter mitgegeben werden. Das könnte z.B. folgendermaßen aussehen:

`LILO boot:` linux aha152x=0x300,10,7

Dieser Kernel-Parameter würde dem Kernel also mitteilen, daß ein Adaptec AHA152X SCSI Hostadapter an der Adresse 0x300 sitzt, den IRQ 10 benutzt und die SCSI-ID 7 belegt.

Analog dazu kann ein solcher Parameter auch auf der Boot-Zeile von **grub** eingegeben werden, statt LILO steht hier dann entsprechend GRUB.

Diese Form der Parameterübergabe ist natürlich lästig, weil man jedesmal beim Booten die erforderlichen Parameter angeben muß. Sie eignet sich daher nur für experimentelles booten, also das Ausprobieren einzelner Parameter, um sicherzustellen, daß sie auch richtig funktionieren.

Andere Bootprompts

Auch die Bootprompts von SYSLINUX (Startdisketten vieler Distributionen) oder als Parameter von LOADLIN (Linux von DOS/Windows aus starten) können Kernelparameter mitgegeben werden. Auch hier gilt das oben gesagte, es handelt sich um die Möglichkeit zu experimentieren und sollte keine Lösung für den Alltag sein.

In der Datei /etc/lilo.conf

Sollen Parameter dauerhaft eingerichtet werden, so ist die Datei /etc/lilo.conf der richtige Ort. Wie oben schon gezeigt, gibt es in dieser Datei die Möglichkeit, sowohl in der globalen ersten Sektion, als auch für jeden Kerneleintrag einzeln mit dem Schlüsselwort `append=` Kernelparameter anzugeben.

Wird diese Angabe wie im obigen Beispiel in der ersten, globalen Sektion der lilo.conf vorgenommen, so gilt sie für alle zu bootenden Linux-Kernel, steht sie stattdessen in der zweiten Sektion, so gilt sie nur für den jeweiligen Kernel, in dessen Kontext sie angegeben wurde.

In der Datei /boot/grub/grub.conf

Wird statt **lilo** das Programm **grub** als Bootmanager verwendet, so muß die Angabe der Parameter entsprechend in seiner Konfigurationsdatei vorgenommen werden. In diesem Fall werden die Parameter direkt an die Kernelangabe gehängt, also etwa in der Art

```
kernel /bzImage-2.2.14 ro root=/dev/hda3 aha152x=0x300,10,7
```

In der Datei /etc/conf.modules bzw. modules.conf

Nachträglich ladbare Module können (und sollen) in der Datei /etc/conf.modules konfiguriert werden. Streng genommen handelt es sich hier auch um Kernel-Parameter, aber eben um die für ladbare Module. Eine genaue Beschreibung dieser Datei folgt im nächsten Abschnitt.

Mit Ausnahme der Einstellungen in der Datei /etc/conf.modules (oder modules.conf) gilt für Kernelparameter immer die folgende Regel:

Alle für einen Treiber relevanten Parameter müssen unmittelbar hintereinander, durch Kommas getrennt, angegeben werden. Es darf kein Leerzeichen zwischen den einzelnen Parametern eines Treibers sein. Sollen mehrere Kernelparameter angegeben werden, so werden diese mit Leerzeichen voneinander getrennt. Also z. B.

```
aha152x=0x300,10,7 reboot=warm
```

Die Angaben für die Module und ihre Form wird im nächsten Abschnitt dargestellt.

Kernelmodule konfigurieren

Ein Linux-Kernel kann prinzipiell auf zwei verschiedene Arten aufgebaut werden. Entweder alle notwendigen Gerätetreiber sind fest in den Kernel hineinkompiliert, oder einzelne Gerätetreiber sind als ladbare Module kompiliert und werden zur Laufzeit ge- und entladen. Im ersten Fall spricht man von einem **monolithischen Kernel**, im zweiten von einem **modularisierten Kernel**.

Die Vorteile des zweiten Modells liegen auf der Hand. Müssten alle Gerätetreiber immer fest im Kernel eingebaut sein, dann hätten wir entweder einen riesigen (und demzufolge langsamen) Kernel oder wir müssten wegen jeder Kleinigkeit (etwa einer neuen Netzwerkkarte) den Kernel neu kompilieren.

Durch die Modularisierung haben wir die Möglichkeit, die Treiber unabhängig vom Kernel zu kompilieren und dann zu laden, wenn wir sie brauchen. Es ist sogar möglich, die geladenen Treiber während der Laufzeit wieder abzuhängen, so daß unnötiger Speicherbedarf vermieden werden kann. So kann z.B. ein ZIP-Laufwerk an der parallelen Schnittstelle betrieben werden. Immer dann, wenn wir es einstecken laden wir den dafür notwendigen Treiber, sobald wir es abhängen entfernen wir auch den Treiber wieder...

Die Kernel-Module benötigen - genauso, als wären sie fest im Kernel eingebaut - zum Teil Angaben über ihre verwendeten Adressen, IRQs, DMA-Kanäle und ähnliches. Damit diese Angaben nicht jedesmal beim Laden der Module von Hand eingegeben werden müssen (und damit so auch ein automatisches Laden der Module ermöglicht werden kann), müssen wir diese Parameter fest in einer Datei eintragen. Diese Datei heißt /etc/conf.modules oder /etc/modules.conf. Meist ist modules.conf ein Link auf conf.modules oder umgekehrt.

Diese Datei beinhaltet die notwendigen Parameter für die verwendeten Module, allerdings werden diese Parameter hier auf eine andere Art - mit einer anderen Syntax - als bei der direkten Übergabe an den Kernel angegeben.

Die vollständige Dokumentation über diese Datei und deren Format ist in der Handbuchseite modules.conf(5) nachzulesen. Hier werden nur die wichtigsten Elemente beschrieben.

Ein einfaches Beispiel für eine solche Datei (reduziert auf die Beschreibung der Netzwerkkarte) könnte wie folgt aussehen:

```
alias eth0 ne
alias eth1 off

options ne io=0x320 irq=7
```

Wir definieren mit der ersten Zeile einen Alias eth0 (erste Ethernetkarte) und weisen ihm den Wert ne (der Name des passenden Moduls) zu. Die zweite Zeile definiert den Alias eth1 mit dem Wert off. Wird dieser Wert gegeben, so wird dieses Modul nie geladen, sogar dann nicht, wenn es explizit von Hand aufgerufen werden würde.

Die eigentliche Definition der Parameter für das Modul ne wird mit der dritten Zeile vorgenommen. Das Schlüsselwort options legt fest, daß das Modul ne die Parameter io=0x320 und irq=7 bekommt.

Im Gegensatz zur Übergabe der Parameter an den Kernel werden hier die Parameter genau bezeichnet. Das führt häufig zu Verwechslungen, weil ein und das selbe Modul eine unterschiedliche Sytax für die Übergabe der Parameter benutzt, es ist aber nicht zu ändern.

Ein weiterer interessanter Aspekt der Datei /etc/conf.modules ist die Möglichkeit, mit den Schlüsselwörtern **pre-install** und **post-install** festzulegen, daß bestimmte Kommandos vor (pre-install) bzw. nach (post-install) dem Einhängen eines Moduls ausgeführt werden. Bei den Kommandos handelt es sich in den häufigsten Fällen wiederum um Kommandos zum Laden (oder Entladen) von Modulen.

Die Syntax dieser beiden Befehle ist dabei sehr einfach:

```
pre-install Modul Kommando
post-install Modul Kommando
```

Den gleichen Mechanismus gibt es genauso wiederum zum Entladen von Modulen, hier heißen dann die Schlüsselwörter:

```
pre-remove Modul Kommando
post-remove Modul Kommando
```

Wenn also ein Stück neue Hardware installiert werden soll und diese neue Hardware über ein Modul

angesprochen wird, so ist die Datei /etc/conf.modules der Ort, an dem die entsprechenden Parameter für die Module eingetragen werden. Die meisten Distributionen liefern bereits eine solche Datei mit, die schon verschiedenste Einstellungen beispielhaft gesetzt hat. Diese Einstellungen sind mit dem #-Zeichen auskommentiert und können durch das Entfernen dieses Zeichens aktiviert werden.

Die Meldungen des Bootvorgangs nachlesen

Jedes Linux-System protokolliert alle Vorkommnisse in einem Logbuch mit. Der Daemon, der dieses Logbuch schreibt ist der **syslogd** über den im Abschnitt 1.111.3 noch mehr zu lernen sein wird. Für die Meldungen des Kernels selbst steht noch ein weiterer Daemon zur Verfügung, der **klogd**, also der Kernel Log Daemon. Auch er schreibt Logbuchmeldungen, allerdings die des Kernels selbst. Beide diese Logbuchdaemonen schreiben ihre Meldungen in der Regel in die Datei **/var/log/messages**.

Diese Datei ist also das Systemlogbuch, ein Fundus von Informationen, über alle möglichen Vorkommnisse, die das System betreffen. Auch die Startmeldungen des Kernels sind hier größtenteils zu finden. Allerdings eben nur größtenteils, denn um die entsprechenden Daemonen zu starten, muß der Kernel schon geladen, und seine Dateisysteme müssen eingehängt sein. Die ersten - und für die Fehlersuche meist interessantesten - Meldungen des Kernels sind in dieser Datei also nicht zu finden.

Der Kernel hält aber einen internen Ringbuffer aufrecht, in den er alle Meldungen hineinschreibt, die etwa beim Start ausgegeben werden. Um diesen Ringbuffer zu lesen, gibt es das Programm **dmesg**. Es ließt den Buffer und gibt den Inhalt auf der Standard-Ausgabe aus. Hier finden sich dann alle Meldungen des Kernels, die er seit dem Booten erstellt hat, bzw. wenn der Rinbuffer voll ist, die letzten Meldungen, die in den Buffer passen. Eine typische Form der Ausgabe des dmesg-Programms ist hier einsehbar:

Ausgabe des dmesg-Programms:

```
Linux version 2.2.16 (root@Pentium.suse.de)\
gcc version 2.95.2 19991024 (release)) #1 Wed Aug 2 20:22:26 GMT

Detected 300687 kHz processor.
Console: colour VGA+ 80x25
Calibrating delay loop... 599.65 BogoMIPS
Memory: 256892k/262080k available (1448k kernel code, 412k reserved, 2912k
data, 68k init, 0k bigmem)
Dentry hash table entries: 32768 (order 6, 256k)
Buffer cache hash table entries: 262144 (order 8, 1024k)
Page cache hash table entries: 65536 (order 6, 256k)
VFS: Diskquotas version dquot_6.4.0 initialized
CPU: Intel Pentium II (Klamath) stepping 03
Checking 386/387 coupling... OK, FPU using exception 16 error reporting.
Checking 'hlt' instruction... OK.
POSIX conformance testing by UNIFIX
mtrr: v1.35a (19990819) Richard Gooch (rgooch@atnf.csiro.au)
PCI: PCI BIOS revision 2.10 entry at 0xf0720, last bus=1
PCI: Using configuration type 1
PCI: Probing PCI hardware
Linux NET4.0 for Linux 2.2
Based upon Swansea University Computer Society NET3.039
NET4: Unix domain sockets 1.0 for Linux NET4.0.
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP, IGMP
TCP: Hash tables configured (ehash 262144 bhash 65536)
Initializing RT netlink socket
Starting kswapd v 1.5
Detected PS/2 Mouse Port.
```

```
pty: 256 Unix98 ptys configured
Real Time Clock Driver v1.09
RAM disk driver initialized:  16 RAM disks of 64000K size
loop: registered device at major 7
Uniform Multi-Platform E-IDE driver Revision: 6.30
ide: Assuming 33MHz system bus speed for PIO modes; override with idebus=xx
PIIX4: IDE controller on PCI bus 00 dev 21
PIIX4: chipset revision 1
PIIX4: not 100% native mode: will probe irqs later
    ide0: BM-DMA at 0xb800-0xb807, BIOS settings: hda:DMA, hdb:DMA
hda: Maxtor 32049H2, ATA DISK drive
hdb: IBM-DHEA-36481, ATA DISK drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
hda: Maxtor 32049H2, 19541MB w/2048kB Cache, CHS=2491/255/63
hdb: IBM-DHEA-36481, 6197MB w/472kB Cache, CHS=790/255/63
Floppy drive(s): fd0 is 1.44M
FDC 0 is a post-1991 82077
LVM version 0.8e  by Heinz Mauelshagen  (4/1/2000)
lvm -- Driver successfully initialized
md driver 0.36.6 MAX_MD_DEV=4, MAX_REAL=8
linear personality registered
raid0 personality registered
raid1 personality registered
raid5 personality registered
scsi : 0 hosts.
scsi : detected total.
Partition check:
 hda: hda1 hda2 hda3 hda4 < hda5 hda6 hda7 hda8 hda9 >
 hdb: hdb1 hdb2 hdb3 hdb4 < hdb5 hdb6 hdb7 hdb8 >
RAMDISK: Compressed image found at block 0
Uncompressing.............................done.
VFS: Mounted root (ext2 filesystem).
(scsi0)  found at PCI 0/12/0
(scsi0) Narrow Channel, SCSI ID=7, 3/255 SCBs
(scsi0) Downloading sequencer code... 415 instructions downloaded
scsi0 : Adaptec AHA274x/284x/294x (EISA/VLB/PCI-Fast SCSI) 5.1.31/3.2.4

scsi : 1 host.
(scsi0:0:1:0) Synchronous at 10.0 Mbyte/sec, offset 15.
  Vendor: TEAC      Model: CD-R55S           Rev: 1.0K
  Type:   CD-ROM                             ANSI SCSI revision: 02
Detected scsi CD-ROM sr0 at scsi0, channel 0, id 1, lun 0
(scsi0:0:2:0) Synchronous at 10.0 Mbyte/sec, offset 15.
  Vendor: IBM       Model: DFHS              Rev: 17
  Type:   Direct-Access                      ANSI SCSI revision: 02
Detected scsi disk sda at scsi0, channel 0, id 2, lun 0
(scsi0:0:5:0) Synchronous at 10.0 Mbyte/sec, offset 15.
  Vendor: PLEXTOR   Model: CD-ROM PX-40TS    Rev: 1.01
  Type:   CD-ROM                             ANSI SCSI revision: 02
Detected scsi CD-ROM sr1 at scsi0, channel 0, id 5, lun 0
Uniform CD-ROM driver Revision: 3.11
sda: Spinning up disk...........ready
SCSI device sda: hdwr sector= 512 bytes. Sectors= 2055035 [1003 MB] [1.0 GB]
 sda: sda1
VFS: Mounted root (ext2 filesystem) readonly.
change_root: old root has d_count=1
Trying to unmount old root ... okay
Freeing unused kernel memory: 68k freed
Adding Swap: 265064k swap-space (priority -1)
```

```
Serial driver version 4.27 with HUB-6 MANY_PORTS MULTIPORT SHARE_IRQ enabled
ttyS00 at 0x03f8 (irq = 4) is a 16550A
ttyS01 at 0x02f8 (irq = 3) is a 16550A
usb.c: registered new driver usbdevfs
usb.c: registered new driver hub
usb-uhci.c: $Revision: 1.232 $ time 20:30:31 Aug  2 2000
usb-uhci.c: High bandwidth mode enabled
usb-uhci.c: Intel USB controller: setting latency timer to 0
usb-uhci.c: USB UHCI at I/O 0xb400, IRQ 15
usb-uhci.c: Detected 2 ports
usb.c: new USB bus registered, assigned bus number 1
usb.c: USB new device connect, assigned device number 1
hub.c: USB hub found
hub.c: 2 ports detected
Soundblaster audio driver Copyright (C) by Hannu Savolainen 1993-1996
SB 4.13 detected OK (220)
eth0: 3c509 at 0x300 tag 1, 10baseT port, address  00 a0 24 14 82 4a, IRQ 5.
3c509.c:1.16 (2.2) 2/3/98 becker@cesdis.gsfc.nasa.gov.
eth0: Setting Rx mode to 1 addresses.
Installing knfsd (copyright (C) 1996 okir@monad.swb.de)
nfsd_fh_init : initialized fhcache, entries=512
parport0: PC-style at 0x378 (0x778) [SPP,ECP,ECPEPP,ECPPS2]
parport_probe: failed
parport0: no IEEE-1284 device present.
ppa: Version 2.03 (for Linux 2.2.x)
ppa: Found device at ID 6, Attempting to use EPP 32 bit
ppa: Communication established with ID 6 using EPP 32 bit
scsi1 : Iomega VPI0 (ppa) interface
scsi : 2 hosts.
  Vendor: IOMEGA    Model: ZIP 100           Rev: L.01
  Type:   Direct-Access                      ANSI SCSI revision: 02
Detected scsi removable disk sdb at scsi1, channel 0, id 6, lun 0
SCSI device sdb: hdwr sector= 512 bytes. Sectors= 196608 [96 MB] [0.1 GB]
sdb: Write Protect is off
 sdb: sdb1
```

Mit diesem Programm stehen uns also auch die Kernel-Meldungen zur Verfügung, die vom Syslog-Daemon bzw. Klog-Daemon noch nicht protokolliert werden konnten, weil diese Daemonen noch gar nicht geladen waren als die Meldungen anfielen.

Das ist eine gerne benutzte Möglichkeit, wenn z.B. beim Start etwas nicht funktioniert und man Rat sucht. Mit dem Befehl

```
dmesg > Bootmeldungen
```

können die Meldungen in eine Datei geschrieben werden, die dann ausgedruckt oder per Mail an einen Supporter geschickt werden kann.

1.106.2 - Ändern des Runlevels und Niederfahren oder Neustart des Systems

Beschreibung: Prüfungskandidaten sollten in der Lage sein, den Runlevel des Systems zu verwalten. Dieses Lernziel beinhaltet den Wechsel in den Single-User Modus und das Niederfahren oder Rebooten des Systems. Kandidaten sollten auch in der Lage sein, Benutzer vor dem Wechsel des Runlevels zu benachrichtigen und Prozesse sauber zu beenden. Dieses Lernziel beinhaltet auch das Setzen des Standard-Runlevels.

Die wichtigsten Dateien, Bezeichnungen und Anwendungen:

- **shutdown**
- **init**
- `/etc/inittab`

Runlevel und ihr Wechsel

Linux bietet verschiedene Betriebszustände an, sogenannte Runlevel (manchmal auch init-level genannt). Im Wesentlichen geht es dabei um die Frage, welche Dienste zur Verfügung gestellt werden und welche nicht.

Der erste Prozess eines Linux-Systems (der Prozess mit der ProzessID 1) ist der init-Prozess. Dieser Prozess startet (initialisiert) sowohl die Benutzerprozesse (bzw. die getty-Prozesse, bei denen sich die Benutzer dann später anmelden können), als auch die Daemon-Prozesse, also die, die kein eigenes Termonal benutzen sondern im Hintergrund als Dienst arbeiten.

Um nicht nur einen Betriebszustand zu ermöglichen, der immer die selben Dienste anbietet oder nicht, gibt es eben die sogenannten Runlevel. Welche Dienste in welchem Runlevel gestartet werden sollen, lässt sich in der Datei `/etc/inittab` einstellen. Das Format und die verschiedenen Möglichkeiten dieser Datei können auf der Handbuchseite von inittab(5) nachgelesen werden.

Grundsätzlich stehen die Runlevel 0 bis 6 und S (oder s) zur Verfügung. Dabei haben die Runlevel 0, 6 und S eine festgelegte Bedeutung, alle anderen stehen frei zur Verfügung. Verschiedene Linux-Distributionen belegen diese übriggebliebenen Runlevel mit verschiedenen Bedeutungen, es existiert kein Standard. Meist werden mindestens drei Runlevel definiert, einer für Multiuser ohne Netz, einer für Multiuser mit Netz und einer für Multiuser mit Netz und graphischem Login (xdm). Multiuser heißt in diesem Fall, daß das System in der Lage ist, mehrere User gleichzeitig zu bedienen.

Die drei festgelegten Runlevel sind:

Runlevel	Bedeutung
0	System herunterfahren ohne anschließenden Neustart. (halt)
6	System herunterfahren und neu starten. (reboot)
S	System im Single User Mode betreiben

Der Single User Modus ist ein Modus, der in der Systemverwaltung immer dann benötigt wird, wenn Arbeiten durchgeführt werden sollen, bei denen Aktivitäten anderer Benutzer stören könnten. Zum Beispiel die Reparatur eines Dateisystems oder die Spiegelung einer Platte.

Die Datei /etc/inittab enthält einen Eintrag, der dem init-Prozess mitteilt, welcher Runlevel der voreingestellte ist, in dem das System hochfahren soll (initdefault). Das wird in der Regel ein benutzerdefinierter Runlevel (1-5) sein.

Um einen Runlevel manuell zu wechseln, kann der Systemverwalter (und nur er!) das Programm init mit dem gewünschten Runlevel aufrufen. Der Befehl

```
init S
```

würde also das System in den Single User Modus bringen, der Befehl

```
init 3
```

würde es in den Runlevel 3 bringen. Statt init kann auch das Programm **telinit** benutzt werden, das exakt genauso funktioniert. In der Regel ist telinit nur ein symbolischer Link auf init, der aus Kompatibilitätsgründen existiert.

Sauberes Herunterfahren des Systems

Um ein Linux-System jetzt herunterzufahren oder neu zu starten könnten wir jetzt einfach schreiben `init 0` bzw. `init 6`. Diese Lösung funktioniert zwar, ist aber aus verschiedenen Gründen nicht die beste Art.

Es sind verschiedene Aufgaben zu erledigen, um ein System sauber herunterzufahren. Diese Aufgaben beinhalten die saubere Beendigung laufender Dienste und Benutzerprogramme, die Synchronisation der Plattenlaufwerke (das physikalische Schreiben des Cache-Inhalts auf die Platte) und nicht zuletzt die Benachrichtigung der noch eingeloggten User, daß das System jetzt heruntergefahren wird.

Es existieren heute verschiedene Programme, die diese Aufgaben zum Teil erledigen, die aber den für die Systemverwaltung wichtigsten Teil weglassen - die Benachrichtigung der User. Die Programme **reboot**, **poweroff**, **suspend** und **halt** dienen alle dazu, das System auf verschiedene Arten herunterzufahren (in Wahrheit sind reboot, poweroff und suspend nur Links auf halt). Diese Programme teilen dem Kernel mit, daß er eben das System runterfahren, neu starten usw. soll. Moderne Versionen dieser Programme ermitteln, ob sie im Runlevel 0 oder 6 ausgeführt wurden und wenn nicht, rufen sie das Programm **shutdown** auf.

Das Programm **shutdown** ist der beste Weg, ein System herunterzufahren oder neu zu starten. Shutdown tut dies auf einem sicheren Weg. Alle eingeloggten User bekommen eine Nachricht, die sie auf das Herunterfahren hinweist und ein einloggen wird blockiert. Es ist möglich, das System sofort herunterzufahren, oder zu einer anzugebenden Uhrzeit oder Zeitdifferenz.

Shutdown schickt allen Prozessen vor dem Herunterfahren zunächst ein TERM-Signal. Das gibt Programmen wie z.B. vi oder joe die Möglichkeit, die ungesicherten Dokumente noch zu speichern, bevor die Programme beendet werden.

Shutdown wird folgendermaßen aufgerufen:

```
shutdown [Optionen] Zeit
```

Die wichtigsten Optionen sind

-r
: Reboot. Shutdown fährt das System herunter und startet es dann erneut. Genau genommen wechselt shutdown in den Runlevel 6 nachdem es alle anderen Arbeiten erledigt hat.

-h
: Halt. Shutdown fährt das System nur herunter ohne einen Neustart danach. Also ein Wechsel in den Runlevel 0.

-c
: Cancel. Ein schon laufender Shutdown wird unterbrochen und somit das System nicht heruntergefahren.

Das Argument *Zeit* kann in mehreren Formen angegeben werden. Entweder es enthält eine Uhrzeit in der Form `hh:mm` also Stunde und Minute. Dann wird das System zur angegebenen Zeit heruntergefahren. Die Warnmeldung erscheint aber trotzdem schon gleich nach dem Programmaufruf und vom Zeitpunkt des

Programmaufrufs bis zum eigentlichen Shutdown kann sich niemand mehr einloggen. Die zweite Form der Zeitangabe ist die Form `+m` was einfach die Anzahl der Minuten ist, bis das System heruntergefahren wird. Steht das Wort `now` statt einer Zeitangabe, so wird es als `+0m` interpretiert.

Die häufigste Anwendung von Shutdown ist das sofortige Herunterfahren mit

```
shutdown -h now
```

oder das gleiche mit reboot statt halt:

```
shutdown -r now
```

Linux kennt die Möglichkeit, der Tastenkombination Strg-Alt-Entf einen Befehl zuzuordnen (einstellbar in /etc/inittab). In den meisten Fällen ist dieser Befehl ein `shutdown -r now`. Das bedeutet aber, daß jeder User, auch ein Normaluser, die Möglichkeit hat, das System herunterzufahren, wenn er Zugriff auf die Tastatur der Systemkonsole hat. Um das einzuschränken kann shutdown in /etc/inittab mit der Option `-a` aufgerufen werden. In diesem Fall überprüft shutdown, ob ein User, der in der Datei `/etc/shutdown.allow` aufgeführt ist, an einer der Consolen arbeitet. Nur dann wird shutdown ausgeführt. Das Format der Datei /etc/shutdown.allow ist einfach ein Username pro Zeile.