Dieser Artikel erschien (leicht modifiziert) im PC Magazin / Weka Verlag in der Ausgabe Februar 2001, Seite 188 bis 191

Startverhalten von Linux

Autor: Dr. Peter Bieringer <pb@bieringer.de>

Bei jedem Betriebssystem läuft das Starten in Stufen ab, so auch bei Linux. Im folgenden werden diese Stufen nun detailliert beschrieben, wobei die Distribution *Red Hat Linux 6.2* als Beispiel dient. Wichtige Unterschiede zu *SuSE Linux 7.0* werden an den entsprechenden Stellen genannt.

Vom BIOS zum LiLo

Nach dem Einschalten des PCs führt das Mainboard-BIOS diverse Tests durch, initialisiert das System und prüft die Existenz der konfigurierten Disketten- und Festplattenlaufwerke.
Danach lädt das BIOS je nach Stellung für die Boot-Reihenfolge beim entsprechenden Laufwerk den im Boot-Sektors (Diskette) oder im Master-Boot-Record (MBR, bei Festplatte) gespeicherten Binär-Code und führt diesen aus.
Dieser MBR kann durch unterschiedliche Methoden erzeugt werden: unter DOS mit `format /mbr`, mit LiLo (wenn `boot=`*MBR*) oder, falls mehrere Betriebssysteme komfortabel gestartet werden sollen, z. B. mit den Freeware-Bootmanager in XFDisk von Florian Painke (http://home.pages.de/~xfdisk/).
Der gestartete Binär-Code lädt nun vom Boot-Sektor der gewählten, voreingestellten oder aktivierten Partition weiteren Code (den sogenannte Bootloader), im Falle von Linux den LiLo.

LiLo

Hier werden nun bei der Installaton von LiLo konfigurierte Parameter aktiv. Als Resultat erscheint die LiLo-Kommandozeile, eventuell mit einem vorangestellten Menü. Die Konfigurationsdatei für das Verhalten ist `/etc/lilo.conf`, deren Änderungen allerdings immer erst nach dem Aufruf von `/sbin/lilo` aktiv werden (d.h. im Boot-Sektor bzw. MBR gespeichert werden).
Wichtige globale Konfigurationsparameter sind in Tabelle LILOGLOBAL gezeigt.

Schlüssel	Beispiel	Erklärung
boot=	/dev/hda	LiLo wird im Master-Boot-Record abgespeichert, schon vorhandener Code (z.B. DOS-MBR oder Bootmanager von XFDisk) wird überschrieben
	/dev/hda2	LiLo wird im Boot-Sektor der Partition 2 abgelegt, Betriebssysteme auf anderen Partitionen werden nicht beeinflußt. Startbar nur durch Bootmanager oder als aktivierte Partition
prompt		Zeigt LiLo-Prompt auch ohne expliziten Tastendruck an
timeout=	50	Wartet 50 x 0,1 Sekunde und startet danach den *default* Eintrag
vga=	ask	Textdarstellung der Konsole ist wählbar
	extended	Textdarstellung der Konsole ist 80 Zeichen x 50 Zeilen (sehr empfehlenswert)
message=	/boot/lilo.msg	Datei wird vor dem LiLo-Prompt angezeigt und kann ein Menü beinhalten
password=	mybootpass-word	Booten der Linuxinstallation durch den installierten LiLo ist paßwortgeschützt (schützt aber nicht vor Booten von CD oder Diskette)
restricted		In Verbindung mit dem Parameter *password* ist die Eingabe am LiLo-Kommando-Prompt ist vor Mißbrauch geschützt (z.B. die Angabe eines anderen Init-Prozesses mit init=/bin/sh und dadurch das lokale Erlangen der root-Berechtigung)
serial=	0,38400n8	Lokale Konsole wird über die serielle Schnittstelle umgeleitet (interessant bei entfernter Serverunterbringung)
default	linux	Image mit dem Label linux wird als Standard gestartet
init	/sbin/init	Init-Prozeß, der vom Kernel nach dem Booten gestartet wird (für Recoveryzwecke siehe Erklärung unter restricted)

Tabelle LILOGLOBAL: Wichtige globale Parameter in /etc/lilo.conf

Welchen Linux-Kernel LiLo booten soll und welche speziellen Parameter dafür verwendet werden, sind mit Parameter im image-Abschnitt spezifiziert (Tabelle LILOIMAGE).
Einige der globalen Konfigurationsparameter können auch pro Kernel-Image verwendet werden, weitergehende Informationen findet man in der Dokumentation zu LiLo (`/usr/doc/lilo-`*`version`*`/README`).

Schlüssel	Beispiel	Erklärung
image=	/boot/vmlinuz-2.2.14-5.0	Zeigt auf das Kernel-Image
label=	linux	Name für das Image, einzugeben am LiLo-Kommandoprompt
alias=	1	Alias für das Label, hilfreich, falls mit /boot/lilo.msg ein Menü angezeigt wird
initrd=	/boot/initrd-2.2.14-5.0.img	Zeigt auf eine für das angegebene Kernel-Image notwendige RAM-Disk (meist nötig, wenn ein SCSI-Treiber der Boot-Festplatte nur als Modul vorhanden ist)
read-only		Dateisystem wird zuerst im Nur-Lesemodus betrieben (spätere Umschaltung in den Schreib-/Lesemodus erfolgt durch ein Init-Skript)
root=	/dev/hda2	Partition, die das Root-Dateisystem beinhaltet
append=	...	Diverse Kernelparameter, z.B. für Netzwerkkarten, IDE-Kanäle, serielle Schnittstellen, etc.

Tabelle LILOIMAGE: Parameter zur Spezifizierung eines Kernel-Image und zusätzlicher Parameter in /etc/lilo.conf

Vom LiLo zum Kernel

Nach der Auswahl des Kernels werden entsprechende Parameter (`init`, `append`) gesetzt und danach der Kernel geladen. Ein Beispiel mit konfiguriertem `message`-Parameter resultiert in:

```
LILO

[1] 2.2.14-5.0 RedHat 6.2
[2] 2.2.16-IPv6-1
boot: 1  Image 1 ausgewählt

Loading 1...........
```

Kernel

LiLo initialisiert und startet nun den Kernel, das Herz von Linux. Er detektiert (abhängig von den Features, die einkompiliert wurden) die vorhandene Hardware (Speicher, CPU, Laufwerke etc.) und initialisiert diese. Dann werden Treiber für die Laufwerke geladen, die Partitionen erkannt und als *root*-(/)-Verzeichnis das von LiLo im übergebenen Parameter `root` definierte im read-only-Modus gemountet.

Falls die *root*-Partition auf einem SCSI-Laufwerk liegt und der Treiber als Modul kompiliert wurde, muß dieser in der initialen RAM-Disk vorliegen (definiert in `/etc/lilo.conf`, Parameter `initrd`), ansonsten schlägt das Booten mit der Meldung „`Kernel panic: VFS: unable to mount root fs on ...`“ fehl.

Während des ganzen Vorgangs werden interessante wie auch wichtige Meldungen über gefundene Hardware ausgegeben, die in Bild KERNELBOOT in Teilen dargestellt sind.

```
Linux version 2.2.14-5.0 (root@porky.devel.redhat.com) (gcc version egcs-2.91.66
 19990314/Linux (egcs-1.1.2 release)) #1 Tue Mar 7 20:        Kernel-Version
...
Memory: 30532k/32768k available (1096k kernel code, 408k reserved, 668k data, 64
k init, 0k bigmem)                                            Erkannter Speicher
...
CPU: 486        CPU-Typ
...
Serial driver version 4.27 with MANY_PORTS MULTIPORT SHARE_IRQ enabled
ttyS00 at 0x03f8 (irq = 4) is a 16450       Serielle Schnittstellen
ttyS01 at 0x02f8 (irq = 3) is a 16450
...
hda: Miniscribe 7080 AT, ATA DISK drive                 Festplatten-Laufwerke
hdb: WDC AC2540F, ATA DISK drive
hdc: Conner Peripherals 170MB - CFA170B, ATA DISK drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
ide1 at 0x170-0x177,0x376 on irq 15
hda: Miniscribe 7080 AT, 81MB w/18kB Cache, CHS=980/10/17
hdb: WDC AC2540F, 515MB w/64kB Cache, CHS=1048/16/63
hdc: Conner Peripherals 170MB - CFA170B, 163MB w/32kB Cache, CHS=332/16/63
Floppy drive(s): fd0 is 1.44M     Floppy-Laufwerk
...
Partition check:
 hda: hda1 hda2
 hdb: hdb1          Partitionen
 hdc: hdc1
...
VFS: Mounted root (ext2 filesystem) readonly.
Freeing unused kernel memory: 64k freed
```

Bild KERNELBOOT: Meldungen des Kernels beim Booten über erkannte Hardware (Auszug)

Vom Kernel zum Init

Nachdem der Kernel alle Geräte (die sogenannten *devices*) bereitgestellt und den Scheduler (zuständig für das Handling von Prozessen) gestartet hat, ist das Multiuser/Multitasking-System einsatzbereit. Der Kernel startet jetzt das Programm, welches im Kernel-Parameter `init` (normalerweise bzw. voreingestellt: `/sbin/init`) angegeben ist:

```
INIT: version 2.78 booting
```

Init

Der *init*-Prozeß wird auch *Vater aller Prozesse* genannt. Er hat üblicherweise die Prozeß-ID 1 und ist für den Start aller anderen Prozesse verantwortlich. Als zentrale Konfigurationsdatei wird `/etc/inittab` benutzt. Diese Datei ist zeilenweise aufgeteilt und besteht aus 4 durch Doppelpunkte getrennte Spalten (Beispiel in Bild INITTAB, Erklärung der Spalten in Tabelle INITTAB).

```
# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
id:3:initdefault:                                   Default runlevel
# Things to run in every runlevel.
ud::once:/sbin/update                 Zuständig für das Zurückschreiben des Disk-Cache
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now   Aktion beim Drücken von CTRL-ALT-DEL
# When our UPS tells us power has failed, assume we have a few minutes
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit   Start der System-Initialisierung
```

Bild INITTAB: Runlevel-unabhängige Einträge in /etc/inittab (Ausschnitt)

Spalte	Bezeichung	Beschreibung
1	id	Eindeutiger Identitätskürzel aus 2 Buchstaben, darf nicht doppelt vorkommen
2	runlevels	Angabe der runlevels, in denen die folgende action durchgeführt werden soll
3	action	Kennzeichung, was durchgeführt werden soll, siehe nächste Tabelle
4	process	Auszuführender Prozeß (Datei) und evtl. benötigte Optionen

Tabelle INITTAB: Beschreibung der Spalten von /etc/inittab

Wichtige Einträge im Felds `action` sind in Tabelle INITABACTION kurz erklärt (für weitergehende Information siehe `man inittab`).

action	**Beschreibung**
initdefault	Festlegung des runlevels
sysinit	Prozeß, der vor Einträge mit action „boot„ oder „bootwait„ gestartet wird
wait	Prozeß wird einmal im angegebenen runlevel gestartet und init wartet auf die Beendigung
respawn	Prozeß wird nach Beendigung immer wieder gestartet (z.B. für getty)

Tabelle: INITTABACTION: Beschreibung der Aktionen im Feld action von /etc/inittab

Als erstes ist der sogenannte *runlevel* definiert, von dem abhängt, welche Dienste später gestartet werden. *runlevel* 0 und 6 sind normalerweise für *halt* und *reboot* reserviert, 1 für den *single user mode*. Bei Kernel-Start-Tests kann, um Zeit zu sparen, auf *runlevel* 1 geschaltet werden. Wer anstatt Konsolen-Login lieber das graphische bevorzugt, sollte (nach erfolgreicher Konfiguration von X-Windows) den *runlevel* auf 5 ändern.

Bei *SuSE* sind die runlevels etwas anders gegliedert, dort bedeutet S *single user mode*, 1 normal ohne Netzwerk, 2 mit Netzwerk und 3 graphisches Login.
Mit Hilfe von `telinit` kann auch im laufenden Betrieb der *runlevel* gewechselt werden (z.B. `telinit 1` wechselt bei *Red Hat* in den `single user mode`).
Zudem können Prozesse definiert werden, die in jedem *runlevel* gestartet werden sollen oder auf Ereignisse von außen wie das Drükken von CTRL-ALT-DEL oder Signale einer unterbrechungsfreien Stromversorgung (USV, engl. UPS) reagieren.
Eine Deaktivierung kann entweder durch Auskommentieren der Zeile (ein # am Anfang) oder durch die *action off* vorgenommen werden.
Nun startet *init* bei *Red Hat* das Programm `/etc/rc.d/rc.sysinit` (ein Shell-Skript), welches das System weiter initialisiert, soweit es unabhängig von den *runlevels* ist. *SuSE* verwendet hier `/sbin/init.d/boot`. Dieses führt einen Dateisystemcheck der *root*-Partition durch und wechselt bei Erfolg den Modus von *ReadOnly* auf *ReadWrite*. Danach wird das */proc*-Dateisystem gemountet. Nun erfolgt der Check und Mount (abhängig von den Parametern) der in `/etc/fstab` angegebenen restlichen Partitionen (siehe auch Bild FSTAB).

Bild FSTAB: Für Dateisystemckeck und Mounten beim Booten wichtige Einträge in /etc/fstab

Zudem führt das Script das Setzen der Zeitzone, Laden der Tastaturtabelle und Zeichensätze, Aktivieren von Swap und RAID, Konfigurieren von PnP, Laden von Kernelmoduln (SCSI, Sound) und Löschen von temporären Dateien durch. In dieser Datei kann man selbst auch eigene weitergehende Systemkonfiguration einfügen. Ein Beispiel für einen erfolgreichen Startablauf ist in Bild INITSCRIPTS zu sehen.

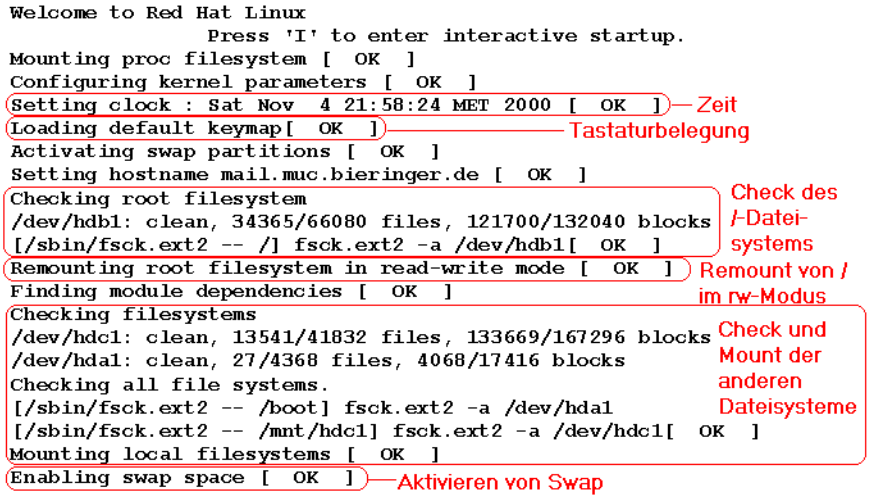

Bild INITSCRIPTS: Erfolgreicher Durchlauf des von init gestarteten /etc/rc.d/rc.sysinit beim Booten

Init und die Startup-Skripts

Nachdem der eben gezeigte Initialisierungsprozeß erfolgreich durchlaufen wurde, wird der *runlevel* bezogene Teil in `/etc/inittab` abgearbeitet:

```
10:0:wait:/etc/rc.d/rc 0
11:1:wait:/etc/rc.d/rc 1
12:2:wait:/etc/rc.d/rc 2
13:3:wait:/etc/rc.d/rc 3
14:4:wait:/etc/rc.d/rc 4
15:5:wait:/etc/rc.d/rc 5
16:6:wait:/etc/rc.d/rc 6
```

Dieses geschieht durch den Aufruf des Shell-Skripts `/etc/rc.d/rc` mit dem Parameter *runlevel* (*SuSE*: `/sbin/init.d/rc`). Im Beispiel *runlevel* 3 durchsucht das eben genannte Skript das Verzeichnis `/etc/rc.d/rc3.d/` (*SuSE*: `/sbin/init.d/rc3.d`) nach Dateien bzw. Softlinks, die mit *S* beginnen. Das Skript startet nun jedes einzelne in aufsteigender Reihenfolge. Normalerweise sind dies nur Softlinks, die auf Dateien im zentralen Verzeichnis `/etc/rc.d/init.d/` zeigen (*SuSE*: `/sbin/init.d/`).

Um nun einen Dienst in einem gewünschten *runlevel* zu starten bzw. zu deaktivieren, ist es nur nötig, entsprechende Softlinks zu erstellen oder zu löschen. Dies kann per Hand oder mit Hilfe entsprechender Werkzeuge durchgeführt werden. Bei *Red Hat* sind dies entweder das bildschirmorientierte `/bin/linuxconf` (*Config ? Control ? Control panel ? Control service activity*) oder das zeilenorientierte `/sbin/chkconfig`. Die Ausgabe von letzterem (Service und jeweilige runlevel-Konfiguration) ist in Bild CHKCONFIG dargestellt.

on=aktiv off=deaktiviert im jeweiligen runlevel

```
                # chkconfig --list
X-Font-Server   xfs             0:off   1:off   2:on    3:on    4:on    5:on    6:off
HTTP-Server     httpd           0:off   1:off   2:off   3:on    4:on    5:on    6:off
Secure SHell-S. sshd            0:off   1:off   2:off   3:on    4:on    5:on    6:off
Tastatur und Font keytable      0:off   1:off   2:on    3:on    4:on    5:on    6:off
Konsolen-Maus-Tr. gpm           0:off   1:off   2:on    3:on    4:on    5:on    6:off
InternetSuperDae. inet          0:off   1:off   2:off   3:on    4:on    5:on    6:off
Mount von NetzLW netfs          0:off   1:off   2:off   3:off   4:off   5:off   6:off
Netzwerk-Konfig. network        0:off   1:off   2:on    3:on    4:on    5:on    6:off
Zufallsgenerator random         0:off   1:on    2:on    3:on    4:on    5:on    6:off
Paketfilter-Firewall ipchains   0:off   1:off   2:off   3:off   4:off   5:off   6:off
Kerberos-DateiRot. kdcrotate    0:off   1:off   2:off   3:off   4:off   5:off   6:off
HardwareErkenn. kudzu           0:off   1:off   2:off   3:on    4:on    5:on    6:off
Initial. Admintool linuxconf    0:off   1:off   2:on    3:on    4:on    5:on    6:off
Druckerspooler  lpd             0:off   1:off   2:on    3:on    4:on    5:on    6:off
Portmapper (NFS) portmap        0:off   1:off   2:off   3:on    4:on    5:on    6:off
Samba (Windows) smb             0:off   1:off   2:off   3:on    4:on    5:on    6:off
Syslog          syslog          0:off   1:off   2:on    3:on    4:on    5:on    6:off
Zeitplandienst  crond           0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

Bild CHKCONFIG: Ausgabe von chkconfig --list zeigt an, welcher Service in welchem runlevel automatisch gestartet wird

Mit diesem Werkzeug läßt sich sehr einfach ein Dienst für den jeweiligen *runlevel* konfigurieren. Dabei ist anzumerken, daß dies *Red Hat* spezifisch ist und sich an einer Zeile in den jeweiligen Startup-Skripts orientiert, die wie folgt aussieht:

```
# chkconfig: 2345 80 20 Kill
    aktiv im runlevel  Start
```

Über die Handhabung siehe `man chkconfig`. In Bild INITSCRIPTSRUNLEVEL3 ist ein erfolgreicher Boot im *runlevel 3* zu sehen.

```
INIT: Entering runlevel: 3   Definierter runlevel
Entering non-interactive startup
Setting network parameters [  OK  ]
Bringing up interface lo [  OK  ]
Bringing up interface eth0 [  OK  ]
Starting portmapper: [  OK  ]
Initializing random number generator [  OK  ]
Starting system logger: [  OK  ]
Starting kernel logger: [  OK  ]
Starting cron daemon: [  OK  ]
Starting INET services: [  OK  ]
Starting sshd [  OK  ]
Starting lpd: [  OK  ]
Starting keytable [  OK  ]
Starting httpd: [  OK  ]
Starting X Font Server: [  OK  ]
Starting linuxconf [  OK  ]
```

Bild INITSCRIPTSRUNLEVEL3: Erfolgreiches Starten der runlevel-spezifischen Services durch /etc/rc.d/rc beim Booten

SuSE-Benutzer können Dienste generell durch Editieren von `/etc/rc.config` oder mit Hilfe von `/sbin/yast` (*System administration ? Change configuration file*) schalten (nach Variablen mit `START_`.. suchen). Für *runlevel*-abhängige Konfigurationen müssen die Softlinks von Hand gelöscht oder erstellt werden.

Startup-Skripts

Die jeweiligen Startup-Skrips werden abhängig von *S*.. oder *K*.. mit die Optionen *start* und *stop* aufgerufen. Für die Eigenentwicklung solcher Skripts ist ein bereits bestehendes immer eine gute Vorlage. Jedes vorhandene Skript ist für einen bestimmten Service oder eine spezielle Konfiguration zuständig.

Daemons

Daemons werden im Normalfall durch eben genannte Skripts gestartet. Falls einer davon nicht benötigt wird, so kann er wie bereits beschrieben ab dem nächsten Reboot deaktivert werden. Eine sofortige Abschaltung kann auch per Hand durch Ausführen des jeweiligen Startup-Skript erfolgen, z.B. `/etc/rc.d/init.d/lpd stop` (stoppen des Druckerspoolers). Natürlich kann ein Daemon auch per Hand nachträglich gestartet werden, z.B, `/etc/rc.d/init.d/named start` (startet den DNS-Server) oder - je nach Programmierung in dem Skript – auch direkt einen Restart vollziehen mit der Option `restart`.
In dem schon gezeigten Bild CHKCONFIG ist auch die Bedeutung einzelner Services kurz angedeutet. Je nach Einsatzgebiet der Linux-Installation sind diese zum Teil (un-)nötig. Z. B. braucht eine Workstation im einfachen Fall keinen Web- (*httpd*), Samba- (*smb*) oder NFS-Server (*portmap, nfs, nfslock*). Bei Server dagegen ist der X-Font- (*xfs*) oder Konsolen-Maus-Server (*gpm*) eventuell unnötig. Auf alle Fälle sollte nach einer Installation die Liste der aktiven Diense überprüft und gegebenenfalls hier Änderungen

vorgenommen werden. Genauere Hinweise, für welchen Dienst ein Service-Skript zuständig ist, steht meist am Anfang eines solchen.
Zum Beispiel führt `/etc/rc.d/init.d/netfs` einen Mount aller in `/etc/fstab` angegebenen Netzwerklaufwerke (NFS, SMB, NCP) durch.
Zu guter letzt wird noch `/etc/rc.d/rc.local` gestartet (durch einen Softlink mit dem Namen `S99local`). Das Skript setzt diverse global gültige Variablen und erzeugt die Dateien `/etc/issue` und `/etc/issue.net`. Falls deren Inhalt geändert werden soll, kann hier der Administrator Modifikationen vornehmen (bei *SuSE* existiert so ein Skript nicht).

Internet-Super-Daemon inetd

Der Internet Super Daemon *inetd* wird hauptsächlich für das Anbieten von Internetdiensten verwendet, welche nicht so häufig nachgefragt werden bzw. für Daemons, die selbst *standalone* nicht starten können. Dies sind üblicherweise die Services TELNET, FTP oder auch R-Dienste. Welche Dienste der Super-Daemon *inetd* bereithält und bei Nachfrage startet, wird in der Konfigurationsdatei `/etc/inetd.conf` definiert. Nach einer Linux-Installation ist es sehr angeraten, diese Datei durchzusehen und nicht benötigte Dienste durch Löschen oder Auskommentieren (ein # am Anfang) der entsprechenden Zeile (Bild INETDCONF) abzuschalten.

```
####### Aktivierte Dienste
#
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd -l -a
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd   Telnet & FTP
#
pop-3   stream  tcp     nowait  root    /usr/sbin/tcpd ipop3d
imap    stream  tcp     nowait  root    /usr/sbin/tcpd imapd          POP & IMAP
#
swat    stream  tcp     nowait.400      root    /usr/sbin/swat  swat
#                                                 Samba Web Administration Tool
####### Abgeschaltete Dienste
#
#shell   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd
#login   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind    R-Dienste
#exec    stream  tcp     nowait  root    /usr/sbin/tcpd  in.rexecd
#
#linuxconf      stream  tcp     wait    root    /bin/linuxconf  linuxconf --http
```

Bild INETDCONF Ausschnitte einer /etc/inetd.conf, der Konfigurationsdatei des Super-Inet-Daemons inetd

Aktiviert wird die Konfigurationsänderung mit `/etc/rc.d/init.d/inet reload`. Eine Kontrolle über aktive Netzwerkdienste kann mit `netstat -nlp` oder mit `lsof -i` erfolgen (Bild NETSTAT).

```
[root@gate rc.d]# netstat -lp --ip -u -t
Active Internet connections (only servers)
Proto Local Address     State       PID/Program name
tcp   *:www             LISTEN      895/httpd   Webserver
tcp   *:smtp            LISTEN      819/sendmail: accep Mailserver
tcp   *:printer         LISTEN      693/lpd     Druckerspooler
tcp   *:exec            LISTEN      654/inetd
tcp   *:login           LISTEN      654/inetd   Internet Super Daemon
tcp   *:shell           LISTEN      654/inetd   startet bei Bedarf
tcp   *:telnet          LISTEN      654/inetd   entsprechende Daemons
tcp   *:ftp             LISTEN      654/inetd
tcp   *:domain          LISTEN      668/named
udp   *:1057                        668/named   Nameserver
udp   *:domain                      668/named
tcp   *:netbios-ssn     LISTEN      963/smbd
udp   *:netbios-dgm                 972/nmbd    Samba
udp   *:netbios-ns                  972/nmbd
```

Bild NETSTAT netstat -lp --ip -u -t liefert Information über die laufenden Internet-Server-Dienste (Ausschnitt)

Vom Init zum getty

Zum Schluß startet *init* noch spezielle Daemons (sogenannte *getty*), um eine interaktive Eingabe via Konsole oder serielle (direkt, analoge oder digital) Schnittstellen zu ermöglichen. (Bild GETTY). Darin sind *getty* mit verschiedenen Eigenschaften gezeigt.
Für lokale Tastatur/Bildschirm ist `/sbin/mingetty` zuständig, der in dem angegebenen Beispiel auf 6 virtuellen Konsolen (zu wechseln mit ALT-F1 bis ALT-F6) gestartet wird. Wer weniger oder mehr benötigt, kann in `/etc/inittab` durch Auskommentieren, *action off* oder Hinzufügen von Zeilen mit den jeweiligen Geräteschnittstellen (`ttyX`) das Verhalten anpassen.
Zu bemerken ist, daß Änderungen im laufenden Betrieb erst nach einem HangUP-Signal an den init-Prozeß (`kill -HUP 1`) aktiv werden. Bei der Deaktivierung von einzelnen *getty* kann es zudem vorkommen, daß diese nachträglich auch noch extra beendet werden müssen (`kill -TERM` *`pid`*).
Im *runlevel* 5 wird auch auch ein X-Windows Display-Manager gestartet, der dann für das graphische Login zuständig ist.
Einträge für Remote Logins via Modem, seriellem Kabel oder ISDN sind als weitere Beispiele in Bild GETTY gezeigt. Dabei werden unterschiedliche *getty* benutzt (`/sbin/getty` für serielle Direktverbindungen, `/sbin/mgetty` für Verbindungen via Modem oder gleichwertigen).

mingetty

`/sbin/mingetty` ist der *getty*, mit dem der Benutzer nach der Installation als erstes „Kontakt aufnimmt“ - außer, das graphische Login ist aktiv. In der Standardkonfiguration (`/etc/inittab`) löscht *mingetty* den Bildschirm bevor er aktiv wird. Wem dies nicht zusagt, weil er z. B. die Meldungen vom Booten noch lesen will, kann dieses Verhalten durch den Parameter `--noclear` deaktivieren (siehe auch Bild GETTY).

```
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty --noclear tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

getty für lokale Konsolen

Bildschirm an Konsole 1 wird nicht gelöscht

```
# Run xdm in runlevel 5
# xdm is now a separate service
x:5:respawn:/etc/X11/prefdm -nodaemon
```

X11-Login

```
# Remote login via Modem
s0:2345:respawn:/sbin/mgetty -D -p "Login: " -i /dev/null /dev/ttyS0

# Remote login via serial
s1:2345:respawn:/sbin/getty ttyS1 DT9600 vt100

# Remote login via ISDN X.75
m0:35:respawn:/sbin/mgetty -i /etc/issue.isdn -a ttyI0
```

getty über seriell, analog oder ISDN

Bild GETTY: Konfiguration der für interaktive Eingaben notwendigen getty in in /etc/inittab

Weitere Optionen (auch die anderer *getty*) würden den Rahmen dieses Artikels sprengen. Die man-pages sind hier eine hilfreiche Informationsquelle.

Von mingetty *zum* login

Beim Start gibt der (*min*)*getty* zuerst den Inhalt der Datei `/etc/issue` (generiert von `/etc/rc.d/rc.local`) und den Hostnamen aus. Danach startet er den Prozeß `/bin/login` und kümmert sich nun fortlaufend um die Übergabe der eingetippten Zeichen, gennannt *stdin* (*standard-in*) an die Applikation und deren Ausgaben *stdout* und *stderr* zur Darstellung auf den Bildschirm:

aus /etc/issue

```
Red Hat Linux release 6.2 (Zoot)
Kernel 2.2.17+IPv6-1 on an i586
login:
```

login

Nachdem `/bin/login` das Kommando übernommen hat, frägt dieser ein Login und ein Paßwort ab. Falls die Datei `/etc/nologin` existiert und die Benutzerkennung ungleich *root* ist, wird das Einloggen verhindert und der Inhalt dieser Datei angezeigt (Bild NOLOGIN).

aus /etc/issue

```
Red Hat Linux release 6.2 (Zoot)
Kernel 2.2.17+IPv6-1 on an i586
login: peter
Password: *********
[/etc/nologin]
Sorry, nur root darf sich einloggen
```

aus /etc/nologin

Bild NOLOGIN: Erfolgloser Login-Versuch eines Benutzers, da die Datei /etc/nologin existiert (und deren Inhalt angezeigt wird)

root dagegen darf sich wiederum nur von den Schnittstellen einloggen, die in `/etc/securetty` spezifiziert sind. Normalerweise sind hier nur die virtuellen lokalen Konsolen aufgelistet (`ttyX`). Falls jedoch auch über eine serielle Verbindung ein *root*-Login erlaubt werden soll, so ist die Datei dementsprechend zu erweitern. Falls es verhindert werden soll, daß sich *root* direkt einloggen darf, sind die entsprechenden Zeilen zu löschen (nicht jedoch die Datei).

```
tty1
tty2
tty3
tty4
tty5
tty6
tty7
tty8
ttyS0
```

Lokale virtuelle Konsolen

Serielle Schnittstelle COM1

/etc/securetty

Achtung: Wenn Datei `/etc/securetty` nicht existiert, dann ist ein *root*-Login von allen Schnittstellen aus erlaubt, auch via *telnet*!

Vom login zur shell

Nach erfolgreichem Login durch Überprüfung des angegebenen Paßworts startet `/bin/login` die in `/etc/passwd` angegebene Shell. Dies ist für gewöhnlich `/bin/bash` (Variante der Bourne-Shell) oder `/bin/tcsh` (erweiterte C-Shell). Änderungen können entweder per Hand (`/usr/sbin/vipw`), via Kommandozeile (`/usr/sbin/usermod` oder `/usr/bin/chsh`) oder über GUI (RedHat: `/bin/linuxconf`, SuSE: `/sbin/yast`) durchgeführt werden. Einträge für die *shell* wie `/bin/false` oder `/bin/true` verhindern ein interaktives Login, da diese Programme sofort wieder beendet sind. Bei leergelassenen Einträgen wird als Standard `/bin/sh` verwendet. Ausschnittsweise ist das in Bild PASSWD dargestellt.

```
root:x:0:0:root:/root:/bin/bash    Shell von root
bin:x:1:1:bin:/bin:
...
xfs:x:43:43:X Font Server:/etc/X11/fs:/bin/false    Deaktivierte Shell
...
peter:x:500:100:Peter Bieringer:/home/peter:/bin/bash    Shell der Benutzer
test:x:501:100:tcshell test:/home/test:/bin/tcsh
```

Bild PASSWD: Beispielhafte Einträge aus /etc/passw

shell: bash

Zum Schluß wird noch auf ein paar Konfigurationsdateien der unter Linux meistgenutzen Shell `bash` eingegangen. Beim Start von `/bin/bash` wird - falls existent - zuerst die Datei `/etc/profile` ausgeführt (Bild PROFILE). Darin werden Parameter spezifiziert, die für alle Login-Shells (benutzerunabhängig) gültig sind. Erklärungen über die einzelnen Parameters findet man im Manual zur bash (`man bash`).

```
PATH="/usr/local/bin:$PATH:/usr/X11R6/bin"    Pfad-Erweiterung
ulimit -c 0    Verhindert das Erstellen von core-Dateien

if [ `id -gn` = `id -un` -a `id -u` -gt 14 ]; then
        umask 002
else                                  umask wird abhängig von der
        umask 022                     Benutzer-ID gesetzt
fi

...

if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
        INPUTRC=/etc/inputrc          weitere globale Konfigurationsdatei
fi                                    wird abgearbeitet

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC    Variablen werden global gültig

for i in /etc/profile.d/*.sh ; do
        if [ -x $i ]; then
                . $i
        fi
done            Paket-installationsspezifische Konfigurationsdateien werden abgearbeitet

unset i

http_proxy=http://proxy.muc.bieringer.de:3128
ftp_proxy=http://proxy.muc.bieringer.de:3128    Lokale Proxy-Einstellungen (für wget und lynx)
export http_proxy
export ftp_proxy
```

Bild PROFILE: Wichtige Teile der /etc/profile

Hier können zum Beispiel lokale Proxy-Einstellungen gesetzt oder das Erstellen von core-Dateien (erzeugt bei Programmabstürzen) deaktiviert werden. Diese zentrale Datei lädt auch noch Dateien (`*.sh`) aus dem Verzeichnis `/etc/profile.d/` (diverse Pakete legen dort Initialisierungsdateien ab) und zentrale Einstellungen aus `/etc/inputrc`. Auch global gültige Pfaderweiterungen können hier (Variable `PATH`) gesetzt werden.
Bei *SuSE* besteht zudem die Möglichkeit, alle lokalen Änderungen in `/etc/profile.local` zu plazieren)
Nach `/etc/profile` wird aus dem jeweiligen Heimatverzeichnis die Datei (bei Existenz) `~/.bash_profile` bzw. `~/.bash_login` oder `~/.profile` geladen und ausgeführt.
Unter *Red Hat* existiert `~/.bash_profile`, welche wiederum dann `~/.bashrc` lädt. Diese wiederum lädt systemweite Einstellungen aus `/etc/bashrc`. Bei *SuSE* wird zuerst `~./profile` und danach `~/.bashrc` geladen. In diesen Dateien kann der jeweilige Benutzer eigene Variablen oder Kommando-Aliases definieren.
Danach endlich erscheint der Shell-Prompt (dieser ist bei *Red Hat* in `/etc/bashrc`, bei *SuSE* in `/etc/profile` spezifiziert) und der Benutzer kann endlich interaktiv mit dem System arbeiten.

Nach dem Verlassen der Shell (mit `exit` oder `logout`) wird noch, falls existent, `~/.bash_logout` abgearbeitet.

Zusammenfassung

Tabelle ZUSAMMENFASSUNG zeigt nochmal eine Übersicht, wo der Administrator bzw. Benutzer Änderungen vornehmen kann.

Was	**bei Red Hat**	**bei SuSE**
Startparameter beim Booten und für Kernel	/etc/lilo.conf	
Änderung des Runlevels	/etc/inittab	
Erweiterung der systemweiten Konfiguration	/etc/rc.d/rc.sysinit	/sbin/init.d/boot
Installation eigener Runlevel-Services	/etc/rc.d/init.d + Softlinks	/sbin/init.d + Softlinks
Globale Einstellungen (z.B. /etc/issue)	/etc/rc.d/rc.local	
(De-)Aktivieren von Services	linuxconf, chkconfig, Softlinks löschen	yast, /etc/rc.config, Softlinks löschen
Für alle bash-Benutzer gültige Parameters (z.B. core-Dateigröße, Pfade)	/etc/profile /etc/bashrc	/etc/profile /etc/profile.local
Individuelle Parameters für bash-Benutzer	~/.bash_profile ~/.bashrc ~/.bash_logout	~/.profile ~/.bashrc ~/.bash_logout

Tabelle ZUSAMMENFASSUNG: Kurze Zusammenfassung, welche Änderungen/Erweiterungen wo zu plazieren sind