4 Vorgänge automatisch starten

Systemverwalter, die Linux-Server einrichten und verwalten wollen, sollten sich mit

- Betriebsarten,
- Zeitsteuerung von Prozessen und
- dem hintergründigen Wirken von Dämonen gründlich vertraut machen.

Systemverwalter, die bisher nur mit proprietären Servern von Novell und Microsoft gearbeitet haben, sollten sich spätestens hier mit diesen grundlegenden Linux-Konzepten vertraut machen; Leser mit fundierten Linux-Kenntnissen können getrost weiterblättern.

- Abschnitt 4.1 (Run-Level) beschreibt Betriebsarten zum Starten und Stoppen des Systems, für Verwaltungsarbeiten und für Mehrbenutzerbetrieb mit und ohne Netz oder Dienste.
- Zeitgesteuerte Einzelaufträge mit dem *at*-Befehl finden Sie im Abschnitt 4.2.
- Regelmäßige Vorgänge mit *cron* (Abschnitt 4.3) nehmen Systemverwaltern viele Routinearbeiten ab.
- Der Superdämon *Inetd* (Abschnitt 4.4) kann im Hintergrund viele Kommunikationsdienste an straffen Zügeln lenken.

4.1 Die Run-Level von SuSE-Linux

Das Mehrbenutzer-Betriebssystem Linux kennt verschiedene Betriebszustände (Run-Level) für normales Arbeiten, Wartung und Neustart.

Ein normaler Bootvorgang bringt das Linux-System in den Run-Level 3, bei dem die Netzwerkunterstützung aktiviert ist und mehrere Benutzer gleichzeitig mit dem System arbeiten können.

Hinweis: In den aktuellen Versionen hat SuSE die Benennung der Run-Level stark verändert. Falls Sie Erfahrungen mit älteren Linux-Versionen haben, sollten Sie sich in `/sbin/init.d.README` mit den Änderungen vertraut machen.

Das Wechseln der Run-Level stoppt und startet Programme. So enthalten u.a. viele Konfigurationsbeschreibungen die Anweisung, die Netzwerkprogramme mit

```
init 1
init 3
```

neu zu starten. Dies wechselt zweimal den System-Zustand (Run-Level).

SuSE-Linux 7.3 kennt die folgenden Run-Level:

Run-Level	*Bedeutung*
0	Halt
S	Single User Mode
1	Single User Mode ohne Netzwerk
2	Multi User ohne Netzwerk
3	Multi User mit Netzwerk
4	Unbenutzt
5	Multiuser mit Netzwerk und xdm (grafischer Anmeldung)
6	Reboot

Tabelle 4.1: Die Run-Level von SuSE-Linux

Bei anderen Distributionen können die Nummern abweichen.

Mit dem Befehl

- *init 0* hält man das System an, ebenso wie mit dem Befehl *halt*.
- *init S* wechseln Sie in den Single User Mode, bei dem Ihnen nur eine einzige Konsole zur Verfügung steht. Sie müssen sich nach dem Wechsel auch neu anmelden.
- *init 1* wechselt man in den Modus *Single User ohne Netzwerk*. Dies stoppt u.a. alle Programme, die mit dem Netzwerk zusammenhängen. Sie müssen sich nach dem Wechsel erneut anmelden.
- *init 2* wechselt das System wieder in den Modus *Multi User ohne Netzwerk* und stoppt alle Netzwerkprogramme.
- *init 3* aktiviert man den *Multi User Modus mit Netzwerk* und startet dabei alle Netzwerkprogramme neu.
- *init 6* startet man das System neu, bewirkt also ein *reboot*.

Programme, die auf einen Wechsel des Run-Levels reagieren sollen, müssen im Ordner `/etc/init.d` ein Programmscript besitzen, das auf die Kommandoparameter `start` bzw. `stop` reagieren kann.

Für den im zweiten Kapitel nachinstallierten DHCPD heißt das Programmscript `dhcp`.

Mit

```
/etc/init.d/dhcpd start
```

startet man den DHCPD und mit

```
/etc/init.d/dhcpd stop
```

stoppt man ihn wieder.

In früheren SuSE-Versionen lagen diese Programmscripte im Verzeichnis `/sbin/init.d/`. Wer also schon Erfahrungen mit älteren Versionen hat, der muss sich hier umstellen. Hilfreich hierbei können die symbolischen Links sein, die SuSE jeweils im Verzeichnis `/usr/sbin` ablegt und bei denen jeweils die Buchstaben `rc` dem Namen vorangestellt sind. Für den DHCPD finden Sie also in `/usr/sbin` den Link `rcdhcpd`. Da das Verzeichnis `/usr/sbin` im Suchpfad aller Benutzer liegt, können Sie den DHCPD auch ohne Pfadangabe mit

```
rcdhcpd start
```

starten und mit

```
rcdhcpd stop
```

wieder stoppen.

Das Programmscript selber ist einigermaßen lesbar und hat folgenden Inhalt:

`/sbin/init.d/dhcp` (Auszug):

```
#! /bin/sh
# Copyright (c) 1996 SuSE Gmbh Nuernberg, Germany.
# All rights reserved.
#
# Author: Rolf Haberrecker <rolf@suse.de>, 1997, 1998, 1999
#         Peter Poeml <poeml@suse.de>, 2000, 2001
#
# /etc/init.d/dhcpd
#   and its symbolic link
# /usr/sbin/rcdhcpd
#
### BEGIN INIT INFO
# Provides:            dhcpd
# Required-Start:      $network $named $syslog
# Required-Stop:      $network $named $syslog
# Default-Start:       3 5
# Default-Stop:        0 1 2 6
# Description:         DHCP server
```

```
### END INIT INFO
..........

case "$1" in
    start)
  echo -n "Starting $DAEMON "

        ## If there is no conf file, skip starting of dhcpd
        ## and return with "program not configured"
  if ! [ -f $DAEMON_CONF ]; then
      echo -e -n "... no configuration file found";
      # Tell the user this has skipped
      rc_status -s
      # service is not configured
      exit 6;
  fi

  ............

  # Remember status and be verbose
  rc_status -v
  ;;
   stop)
  echo -n "Shutting down $DAEMON "

  ## Stop daemon with killproc(8) and if this fails
  ## set echo the echo return value.

  killproc -p $CHROOT_PREFIX/$DAEMON_PIDFILE -TERM
  ➥ $DAEMON_BIN

  # Remember status and be verbose
  rc_status -v
  ;;

  ............

    restart)
  ## Stop the service and regardless of whether it was
  ## running or not, start it again.
  $0 stop
  sleep 3
```

```
    $0 start

    # Remember status and be quiet
    rc_status
    ;;
```

In dem Listing sind die erlaubten Parameter fett hervorgehoben. Neben `start` und `stop` kennt das Programm auch noch die Parameter `reload`, `restart` und `status`. Die Kommandos `reload` und `restart` stoppen den dhcpd und starten ihn nach 3 Sekunden wieder, mit `status` testet das Programm, ob der dhcpd läuft oder nicht. Alle anderen Parameter führen zur Ausgabe eines kleinen Hilfstextes.

Am Anfang wertet das Programm aus, ob es direkt von der Konsole aus gestartet wurde oder über einen Wechsel der Run-Level. Bei einem Start über die Run-Level, wertet es die Variable `START_DHCPD` aus der Konfigurationsdatei `/etc/rc.config` aus. Nur wenn diese den Wert `yes` hat, startet das Programm.

Jetzt fehlt noch die Kopplung an den Wechsel der Run-Level. Dazu gibt es unterhalb von `/etc/init.d` für jeden Run-Level ein Verzeichnis, also

- `/etc/init.d/rc0.d`,
- `/etc/init.d/rc1.d`,
- `/etc/init.d/rc2.d`,
- `/etc/init.d/rc3.d`,
- `/etc/init.d/rc4.d`,
- `/etc/init.d/rc5.d`,
- `/etc/init.d/rc6.d`,
- `/etc/init.d/rcS.d`.

In diesen Ordnern befinden sich Verweise (Softlinks) auf die Start-/Stopp-Dateien im Ordner `/etc/init.d`, für den DHCPD sind dies die Links

- `S12dhcpd` und
- `K13Odhcpd`.

Der Buchstabe `S` steht hier für *Start*, der Buchstabe `K` für *Kill* (Beenden). Beim Wechsel in den Run-Level 3 ruft das Linux-System alle Links im Verzeichnis `rc3.d`, die mit einem *S* beginnen, mit dem Parameter `start` auf. Die Zahl gibt eine Reihenfolge an; je höher die Zahl, desto später startet das zugehörige Programm.

Beim Verlassen eines Run-Levels kommen die Links zum Einsatz, die mit einem *K* beginnen. Das zugehörige Programmscript startet dann mit dem Parameter `stop`.

Die Distribution der SuSE-CD installiert die Startscripte und Links der Programme automatisch. Bei Programmen, die vor ihrem Start noch konfiguriert werden müssen, stehen in der Konfigurationsdatei `/etc/rc.config` die entsprechenden Startschalter (z.B. *DHCP_START*) noch auf `no`.

Im Kapitel 14 über Masquerading und Firewalling werden Sie ein eigenes Programmscript `/etc/init.d/maske` finden. Wenn dieses im Run-Level 3 aktiv sein soll, müssen Sie in `/etc/init.d/rc3.d` folgende Links anlegen:

```
ln -s  /etc/init.d/maske  /etc/init.d/rc3.d/S40maske
ln -s  /etc/init.d/maske  /etc/init.d/rc3.d/K40maske
```

Damit startet das Programm beim Wechsel in den Run-Level und stoppt beim Verlassen des Run-Levels 3.

Ein Muster für eigene Startprogramme finden Sie in der Datei `/sbin/init.d/skeleton`.

`/sbin/init.d/skeleton` (Auszug, Dateianfang):

```
#! /bin/sh
# Copyright (c) 1995-2000 SuSE GmbH Nuernberg, Germany.
#
# Author: Kurt Garloff <feedback@suse.de>
#
# init.d/FOO
#
#   and symbolic its link
#
# /sbin/rcFOO
#
# System startup script for the nessus backend nessusd
#
### BEGIN INIT INFO
# Provides: FOO
# Required-Start: $remote_fs $syslog
# Required-Stop:  $remote_fs $syslog
# Default-Start:  3 5
# Default-Stop:   0 1 2 6
# Description:    Start FOO to allow XY and provide YZ.
### END INIT INFO
```

```
# Source SuSE config
. /etc/rc.config

# Determine the base and follow a runlevel link name.
base=${0##*/}
link=${base#*[SK][0-9][0-9]}

# Force execution if not called by a runlevel directory.
test $link = $base && START_FOO=yes
test "$START_FOO" = yes || exit 0

FOO_BIN=/usr/sbin/FOO
test -x $FOO_BIN || exit 5

# Shell functions sourced from /etc/rc.status:
#      rc_check         check and set local and overall rc
       ➥ status
#      rc_status        check and set local and overall rc
       ➥ status
#      rc_status -v     ditto but be verbose in local rc
       ➥ status
#      rc_status -v -r  ditto and clear the local rc status
#      rc_failed        set local and overall rc status to
       ➥ failed
#      rc_failed <num>  set local and overall rc status to
<num><num>
#      rc_reset         clear local rc status (overall
       ➥ remains)
#      rc_exit          exit appropriate to overall rc status
. /etc/rc.status

# First reset status of this service
rc_reset

# Return values acc. to LSB for all commands but status:
# 0 - success
# 1 - generic or unspecified error
# 2 - invalid or excess argument(s)
# 3 - unimplemented feature (e.g. "reload")
# 4 - insufficient privilege
# 5 - program is not installed
# 6 - program is not configured
```

```
# 7 - program is not running
#
# Note that starting an already running service, stopping
# or restarting a not-running service as well as the restart
# with force-reload (in case signalling is not supported) are
# considered a success.

case "$1" in
    start)
   echo -n "Starting FOO"
   ## Start daemon with startproc(8). If this fails
   ## the echo return value is set appropriate.

   # NOTE: startproc return 0, even if service is
   # already running to match LSB spec.
   startproc $FOO_BIN

   # Remember status and be verbose
   rc_status -v
   ;;
    stop)
   echo -n "Shutting down FOO"
   ## Stop daemon with killproc(8) and if this fails
   ## set echo the echo return value.
```

Diese Datei müssen Sie an Ihre Bedürfnisse anpassen. Relativ neu in dieser Datei ist der Block `INIT INFO`. Hier geben Sie an, in welchen Run-Leveln Ihr Programm aktiv sein soll und welche anderen Dienste bereits gestartet bzw. gestoppt sein müssen. Mit diesen Informationen kann das Programm `insserv` im Verzeichnis `/etc/init.d/` automatisch symbolische Links anlegen.

Über den Wechsel der Run-Level startet das System Programme, die ständig aktiv sein sollen. Daneben gibt es auch Anwendungsfälle, bei denen ein Programm zu einem ganz bestimmten Zeitpunkt aktiv sein soll. Hierzu gibt es die Zeitsteuerung über `at` und `cron`:

- Mit `at` startet man ein Programm einmalig zu einem bestimmten Zeitpunkt. Eine typische Anwendung sind Wartungsarbeiten, die dann ausgeführt werden sollen, wenn keine Benutzer angemeldet sind.
- Will man ein Programm regelmäßig zu einem bestimmten Zeitpunkt aufrufen, so bietet sich `cron` an. Alle regelmäßigen Wartungsarbeiten und statistische Auswertungen gehören zu den möglichen Anwendungsfällen.

Lesen Sie in den nächsten beiden Abschnitten mehr über Zeitsteuerung.

4.2 Zeitgesteuerte Einzel-Aufträge

Mit dem Befehl `at` können dafür Berechtigte zu einem bestimmten Zeitpunkt Programme ausführen, z.B. ein zeitaufwändiges Programm nachts starten.

Tipp: Mit `at` kann man nur Programme starten, für deren Ausführung man auch die notwendigen Rechte besitzt. Für das Beispiel sollte man als *root* angemeldet sein, da normale Benutzer mit `find` nicht in allen Verzeichnisen suchen dürfen.

Um z.B. alle Dateien zu finden, die keinem Benutzer gehören, kann man den `find`-Befehl in der folgenden Form einsetzen:

```
find  /  -nouser
```

Der Suchvorgang ist recht zeitaufwändig, da `find` alle Dateien untersucht. Dateien ohne Benutzer entstehen, wenn man Benutzer löscht und diese Dateien außerhalb ihrer Home-Verzeichnisse abgelegt haben. Da die Suche in größeren Systemen recht lange dauern kann, sollte man diese Suche auf einen ruhigen Zeitpunkt, z.B. 22:00 Uhr, verschieben. Dazu gibt man ein:

```
at  22:00
```

Am veränderten Eingabezeichen gibt man den eigentlichen Befehl ein

```
at> find  /  -nouser
```

und schließt die Eingabe dann mit Strg D ab.

```
boss:~ # at 22:00
warning: commands will be executed using /bin/sh
at> find / -nouser
at> <EOT>
job 1 at 2001-12-22 22:00
boss:~ #
```

Die Zeitpunkte für die Ausführung kann man auf verschiedene Arten angeben, wie hier im Beispiel über *HH:MM*, aber auch mit `now +2 hours`. Damit würde das Programm in zwei Stunden starten. Statt `hours` sind auch die Angaben `minutes`, `days` und `weeks` und absolute Zeitangaben wie `teatime` (16:00 Uhr) und `midnight` möglich.

Unerledigte Aufträge zeigt `atq` an:

```
boss:~ # atq
1        2001-12-22 22:00 a
```

Wichtig hierbei ist die Jobnummer eines Auftrages, da man hierüber den Auftrag auch wieder löschen kann:

```
boss:~ # atrm 1
```

Der `at`-Dämon gibt Daten statt auf den Bildschirm in eine Mail an den Auftraggeber aus.

Um diese Suche nach herrenlosen Dateien regelmäßig auszuführen, benutzt man besser `cron`.

4.3 Regelmäßige Vorgänge mit cron

Für regelmäßige Vorgänge gibt es ein besseres Werkzeug als `at`: `cron`. Für diese erstellt man eine Tabelle (*crontab*), welche die Vorgänge und die Zeiten aufführt, an denen man ausführen will.

In der Grundeinstellung dürfen alle Benutzer, die nicht in der Datei `/var/spool/cron/deny` verzeichnet sind, eine `crontab` anlegen. In der Grundinstallation sind hier nur *gast* und *guest* eingetragen.

Eine derartige `crontab`-Tabelle könnte folgendermaßen aussehen:

```
# roots crontab
#
# min hour day month dayofweek (1=Mo,7=Su) command
15 22 * * * /usr/bin/find / -nouser
```

So startet der Suchbefehl jeden Tag um 22:15 Uhr. Ein Stern steht als Jokerzeichen für alle Zeiten. Es startet also an jedem Tag, in jedem Monat und an jedem Wochentag um 22:15 Uhr der Suchbefehl.

Eingeben kann man diese Tabelle als Benutzer *root* durch:

```
crontab  -e
```

Die Möglichkeiten der Zeitangabe sind recht vielfältig. Mit z.B.

```
# roots crontab
#
# min hour day month dayofweek (1=Mo,7=Su) command
15 22 * * 1-5 /usr/bin/find / -nouser
```

würde die Suche nur an Werktagen ablaufen.

Zu den Anwendungen, die Sie regelmäßig per `cron` ausführen sollten, gehört die Datensicherung – das Backup. Hinweise hierzu finden Sie oben im Kapitel 2.

4.4 Der Super-Dämon inetd für Internetdienste

Für viele Internetdienste, wie POP, SMTP, FTP und Telnet findet man weder ein Startscript in `/etc/init.d`, noch einen zeitgesteuerten Aufruf.

Das spart Ressourcen, da die zugehörigen Programme erst bei Bedarf starten. Der dafür zuständige Super-Dämon `inetd` wird über das Startscript `/etc/init.d/inetd` gestartet. Danach soll er auf Anforderungen an die Internetdienste warten und dann das Programm starten.

Konfigurieren können Sie `inetd` über die Datei `/etc/inetd.conf`.

`/etc/inetd.conf` (Dateianfang):

```
# See "man 8 inetd" for more information.
#
# If you make changes to this file, either reboot your machine
  ➥ or send the
# inetd a HUP signal with "/sbin/init.d/inetd reload" or by
  ➥ hand:
# Do a "ps x" as root and look up the pid of inetd. Then do a
# "kill -HUP <pid of inetd>".
# The inetd will re-read this file whenever it gets that
  ➥ signal.
#
# <service_name> <sock_type> <proto> <flags> <user>
  ➥ <server_path> <args>
#
# echo stream tcp   nowait root  internal
# echo dgram  udp   wait  root  internal
# discard   stream tcp   nowait root  internal
# discard   dgram  udp   wait  root  internal
# daytime   stream tcp   nowait root  internal
# daytime   dgram  udp   wait  root  internal
# chargen   stream tcp   nowait root  internal
# chargen   dgram  udp   wait  root  internal
time  stream tcp   nowait root  internal
time  dgram  udp   wait  root  internal
#
# These are standard services.
#
# ftp  stream tcp   nowait root  /usr/sbin/tcpd     wu.ftpd -a
# ftp  stream tcp   nowait root  /usr/sbin/tcpd     proftpd
ftp   stream tcp   nowait root  /usr/sbin/tcpd     in.ftpd
#
```

```
# If you want telnetd not to "keep-alives" (e.g. if it runs
  ➥ over a ISDN
# uplink), add "-n".  See 'man telnetd' for more details.
telnet stream  tcp     nowait  root    /usr/sbin/tcpd
  ➥ in.telnetd
# nntp stream tcp    nowait news  /usr/sbin/tcpd
  ➥ /usr/sbin/leafnode
# smtp stream  tcp     nowait  root    /usr/sbin/sendmail
  ➥ sendmail -bs
# printer    stream tcp    nowait root  /usr/sbin/tcpd
  ➥ /usr/bin/lpd -i
#
# Shell, login, exec and talk are BSD protocols.
#  The option "-h" permits ``.rhosts'' files for the
   ➥ superuser. Please look at
#  man-page of rlogind and rshd to see more configuration
possibilities about
#  .rhosts files.
shell stream tcp    nowait root  /usr/sbin/tcpd     in.rshd -L
# shell      stream tcp    nowait root  /usr/sbin/tcpd
  ➥ in.rshd -aL
#
# If you want rlogind not to "keep-alives" (e.g. if it runs
  ➥ over a ISDN
# uplink), add "-n".  See 'man rlogind' for more details.
login stream tcp    nowait root  /usr/sbin/tcpd     in.rlogind
# login      stream tcp    nowait root  /usr/sbin/tcpd
  ➥ in.rlogind -a
# exec stream tcp    nowait root  /usr/sbin/tcpd     in.rexecd
talk  dgram udp    wait  root  /usr/sbin/tcpd     in.talkd
ntalk dgram udp    wait  root  /usr/sbin/tcpd     in.talkd
#
#
# Pop et al
#
# pop2 stream tcp    nowait root  /usr/sbin/tcpd     in.pop2d
# pop3 stream tcp    nowait root  /usr/sbin/tcpd
  ➥ /usr/sbin/popper -s
#
```

Mit dem #-Zeichen beginnen Zeilen, die kommentieren oder den Dienst deaktivieren.

SuSE hat die Konfiguration für verschiede FTP-Server vorgesehen, daher finden Sie in der Datei drei Zeilen für FTP-Dienste. Nur einer der Server kann aktiviert sein, hier der `in.ftpd`, die anderen Zeilen sind deaktiviert. Sie werden später im FTP-Kapitel (Kapitel 7) den `wu.ftpd` aktivieren.

Angegeben sind in der ersten Spalte der Dienst und am Ende der Zeile das zugehörige Programm und die Parameter, mit denen es startet.

Die Angaben in der Mitte der Zeile liefern Informationen über die Art des Datenaustausches, z.B. Protokoll `tcp`, und den Benutzer, mit dessen Rechten der Dienst gestartet werden soll. Der Aufruf über `/usr/sbin/tcpd` aktiviert eine Zugriffskontrolle für den jeweiligen Dienst. Einerseits protokollieren Dienste die Zugriffe in der `/var/log/messages`, andererseits kann man über die Dateien `/etc/hosts.allow` und `/etc/hosts.deny` festlegen, welche Rechner auf den jeweiligen Dienst zugreifen können.

Die meisten Einträge in dieser Datei sind nicht aktiviert. Man sollte nur Dienste aktivieren, die man auch wirklich benötigt.