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.

Die Autoren haben beschlossen hier diese grundlegenden Linux-Konzepte für Systemverwalter, die bisher nur mit proprietären Servern von Novell und Microsoft gearbeitet haben, einzufügen.

Leser mit Linux-Kenntnissen können getrost weiterblättern.

- Der Abschnitt 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 und
- regelmäßige Vorgänge mit `cron` nehmen Systemverwaltern viele Routinearbeiten ab.
- Der Superdämon `Inetd` führt im Hintergrund viele Kommunikationsdienste an straffen Zügeln.

4.1 Die Run-Level von SuSE-Linux

Als Betriebssystem für mehrere Benutzer kennt Linux verschiedene Betriebszustände (Runlevel). Nach einem normalen Bootvorgang befindet sich Linux im Runlevel 2, bei dem die Netzwerkunterstützung aktiviert ist und mehrere Benutzer gleichzeitig mit dem System arbeiten können.

In vielen Konfigurationsbeschreibungen findet sich die Anweisung, die Netzwerkprogramme mit

```
init 1
init 2
```

neu zu starten. Das wechselt zweimal den System-Zustand (Run-Level). Das Wechseln der Runlevel stoppt und startet Programme.

SuSE-Linux kennt die folgenden Runlevel:

Runlevel	*Bedeutung*
0	Halt
S	Single User Mode
1	Multi User ohne Netzwerk
2	Multi User mit Netzwerk
3	Multi User mit Netzwerk und Xdm (grafischer Anmeldung)
4	Unbenutzt
5	Unbenutzt
6	Reboot

Tabelle 4.1: Die Runlevel von SuSE-Linux

Bei anderen Distributionen können die Nummern abweichen.

Mit dem Befehl

- `init 1` wechselt man in den Modus *Multi User ohne Netzwerk*. Dies stoppt alle Programme, die mit dem Netzwerk zusammenhängen.
- `init 2` wechselt wieder in den Modus *Multi User mit Netzwerk* und startet alle Netzwerkprogramme neu.
- `init 0` hält das System an, hat den gleichen Effekt wie `halt`.
- `init 6` startet das System neu, bewirkt ein `reboot`.

Programme, die auf einen Wechsel des Runlevels reagieren sollen, müssen im Ordner `/sbin/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

```
/sbin/init.d/dhcp start
```

startet man den DHCPD und mit

```
/sbin/init.d/dhcp stop
```

stoppt man ihn wieder.

Das Programm ist einigermaßen lesbar und hat folgenden Inhalt.

`/sbin/init.d/dhcp`:

```
#! /bin/sh
# Copyright (c) 1996 SuSE Gmbh Nuernberg, Germany.
# All rights reserved.
#
# Author: Rolf Haberrecker <rolf@suse.de>, 1997, 1998, 1999
#
# /sbin/init.d/dhcp
#

. /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 = $base && START_DHCPD="yes"
test "$START_DHCPD" = yes || exit 0

# The echo return value for success
# (defined in /etc/rc.config).
return=$rc_done

case "$1" in
    start)
   echo -n "Starting service dhcpd"
   startproc /usr/sbin/dhcpd -q $DHCPD_INTERFACE ||
   ➥ return=$rc_failed
   echo -e "$return"
   ;;
    stop)
   echo -n "Shutting down service dhcpd"
   killproc -TERM /usr/sbin/dhcpd || return=$rc_failed
   echo  -e "$return"
   ;;
    reload|restart)
        $0 stop && sleep 3 && $0 start || return=$rc_failed
   ;;
    status)
        echo -n "Checking for service dhcpd: "
        ## Check status with checkproc(8),
        ## if process is running
```

```
        ## checkproc will return with exit status 0.

        checkproc /usr/sbin/dhcpd && echo OK || echo No
        ➥ process
        ;;
    *)
   echo "Usage: $0 {start|stop|status|reload|restart}"
   exit 1
esac

# Inform the caller not only verbosely and set an exit status.
test "$return" = "$rc_done" || exit 1
exit 0
```

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 Runlevel. Bei einem Start über die Runlevel, 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 Runlevel. Dazu gibt es unterhalb von `/sbin/init.d` für jeden Runlevel ein Verzeichnis, also

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

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

- `S30dhcp` und
- `K30dhcp`.

Der Buchstabe S steht hier für *Start*, der Buchstabe K für *Kill* (Beenden). Beim Wechsel in den Runlevel 2 ruft das Linux-System alle Links, 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 Runlevels kommen die Links zum Einsatz, die mit einem K beginnen. Das zugehörige Programmscript startet dann mit dem Parameter stop.

Die Startscripte und Links der Programme der SuSE-CD installiert die Distribution 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 /sbin/init.d/maske finden. Wenn dieses im Runlevel 2 aktiv sein soll, müssen Sie in /sbin/init.d /rc2.d folgende Links anlegen:

```
ln -s  /sbin/init.d/maske  /sbin/init.d/rc2.d/S40maske
ln -s  /sbin/init.d/maske  /sbin/init.d/rc2.d/K40maske
```

Damit startet das Programm beim Wechsel in den Runlevel 2 und stoppt beim Verlassen des Runlevels 2.

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:
#
# /sbin/init.d/<skeleton>
#
#   and symbolic its link
#
# /sbin/rc<skeleton>
#

.  /etc/rc.status
.  /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

# 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_reset         clear local rc status (overall remains)
#   rc_exit          exit appropriate to overall rc status

# First reset status of this service
rc_reset
case "$1" in
    start)
        echo -n "Starting service foo"
        ## Start daemon with startproc(8). If this fails
        ## the echo return value is set appropriate.

        #startproc /usr/sbin/foo

        # Remember status and be verbose
        rc_status -v
        ;;
    stop)
        echo -n "Shutting down service foo"
        ## Stop daemon with killproc(8) and if this fails
```

Diese Datei müssen Sie für Ihre Bedürfnisse nur noch geeignet anpassen.

Über den Wechsel der Runlevel 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.

4.2 Zeitgesteuerte Einzel-Aufträge

Der Befehl `at` soll Programme zu einer bestimmten Zeit ausführen, z.B. ein zeitaufwendiges Programm nachts starten.

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

```
find / -nouser
```

Der Suchvorgang ist recht zeitaufwendig, da `find` alle Dateien untersucht. Dateien ohne Benutzer entstehen, wenn man Benutzer löscht und diese Dateien außerhalb ihres Home-Verzeichnisses 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
at> find / -nouser
at> <EOT>
warning: commands will be executed using /bin/sh
job 4 at 2000-03-22 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` möglich.

Für uns eher exotisch sind Zeitangaben wie `teatime` (16:00 Uhr) und `midnight`.

Unerledigte Aufträge zeigt `atq` an:

```
boss:~ # atq
4        2000-03-22 22:00 a
```

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

```
boss:~ # atrm 4
```

Daten gibt der `at`-Dämon nicht auf dem Bildschirm, sondern in einer Mail an den Auftraggeber aus.

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.

Will man diese Suche nach herrenlosen Dateien regelmäßig ausführen, so ist die Arbeit mit `at` zu aufwendig. Hier benutzt man besser `cron`.

4.3 Regelmäßige Vorgänge mit Cron

Für regelmäßige Vorgänge gibt es ein besseres Werkzeug als `at`: `cron`. Für dieses erstellt man eine Tabelle (`crontab`), welche die Vorgänge und die Zeiten aufführt, an denen man sie ausführen will. Eine derartige 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 im Kapitel 2.

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.

4.4 Der Super-Dämon Inetd für Internetdienste

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

Das spart Ressourcen, da die zugehörigen Programme erst bei Bedarf starten. Der Super-Dämon `inetd` wird über das Startscript `/sbin/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.
telnetstream  tcp     nowait  root    /usr/sbin/tcpd
  ➥ in.telnetd
# nntpstreamtcp   nowaitnews  /usr/sbin/tcpd
  ➥ /usr/sbin/leafnode
# smtpstream  tcp     nowait  root    /usr/sbin/sendmail
  ➥ sendmail -bs
# printer   streamtcp   nowaitroot  /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 streamtcp   nowaitroot  /usr/sbin/tcpd    in.rshd -L
# shell     streamtcp   nowaitroot  /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 streamtcp   nowaitroot  /usr/sbin/tcpd    in.rlogind
# login     streamtcp   nowaitroot  /usr/sbin/tcpd
  ➥ in.rlogind -a
# execstreamtcp   nowaitroot  /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
#
# pop2streamtcp   nowaitroot  /usr/sbin/tcpd    in.pop2d
pop3  streamtcp   nowaitroot  /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 ist in der ersten Spalte der Dienst und am Ende der Zeile das zugehörige Programm und die Parameter mit denen es startet.

Der Parameter `-s` beim Programm `popper`, den POP3-Diensten, bewirkt dass Ausgaben in die Datei `/var/log/messages` geschrieben werden.

Die Angaben in der Mitte der Zeile geben 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 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.