Peter Ganten • Wulf Alex

Debian GNU/Linux
Grundlagen, Installation, Administration und Anwendung

Stand: 9. Januar 2005 (Test)

Springer
Berlin Heidelberg New York
Hong Kong London
Milan Paris Tokyo

1

Der Startvorgang von Debian GNU/Linux

Nachdem die Startphase des Kerns abgeschlossen ist, bindet dieser das Rootdateisystem ein und ruft das Programm */sbin/init* auf, das den weiteren Verlauf des Systemstarts kontrolliert und aktiv bleibt, bis das System heruntergefahren ist. *init* ist der erste voll ausgebildete Prozess auf einem Linux/UNIX-System, trägt deswegen die Prozess-ID 1 und ist der Urahne aller weiteren Prozesse.

1.1 Runlevel

Debian GNU/Linux verwendet das System-V-Konzept verschiedener Runlevels. Unter einem Runlevel wird ein Zustand des Systems verstanden, in dem bestimmte Prozesse ausgeführt werden. Während der Laufzeit des Systems kann zwischen den Runlevels gewechselt werden, ohne das System neu starten zu müssen. So könnte ein Rechner regelmäßig für zwei verschiedene Aufgaben genutzt werden, nämlich zum einen als HTTP- und FTP-Server und zum anderen als Arbeitsplatzrechner. Im ersten Fall müssen die genannten Serverdienste ausgeführt werden, eine grafische Benutzeroberfläche wird jedoch nicht benötigt. Im zweiten Fall wird eine grafische Benutzeroberfläche gewünscht, jedoch sollen die Serverdienste nicht ausgeführt werden. Für diesen Rechner lassen sich zwei Runlevels definieren, in denen die unterschiedlichen Dienste oder Prozesse ausgeführt werden.

Neben diesen normalen Runlevels gibt es drei besondere, nämlich einen, in dem das System heruntergefahren wird, einen, in dem es neu gestartet wird, und den Single-User-Modus, in dem sich nur der Verwalter anmelden kann und keine weiteren Prozesse wie Serverdienste ausgeführt werden. Normalerweise gibt es insgesamt sieben Runlevels, welche die Nummern 0-6 tragen. 0 entspricht dabei dem Anhalten des Systems, 1 dem Single-User-Modus und 6 dem Neustart des Systems. Die Runlevels 2-5 sind die gewöhnlichen Runlevels. Per Voreinstellung wird nach dem Systemstart in den Runlevel 2 geschaltet.

Bei einer unveränderten Debian GNU/Linux Installation unterscheiden sich die Runlevels 2 und 3 nicht. Es werden dieselben Prozesse ausgeführt, und deswegen macht es keinen Unterschied, in welchem dieser beiden Runlevels sich das System

befindet. Die beiden Runlevels 4 und 5 unterscheiden sich ebenfalls nicht voneinander, wohl aber von 2 und 3. In den Runlevels 2 und 3 stehen die sechs virtuellen Konsolen zur Verfügung, außerdem kann über andere Schnittstellen wie serielle Terminals oder Faxmodems auf den Rechner zugegriffen werden. In den Runlevels 4 und 5 steht nur eine Konsole zur Verfügung; die Programme zur Kommunikation über andere Schnittstellen sind nicht aktiv. Weiterhin aktiv sind in den Runlevels 4 und 5 die Netzverbindungen, sodass der Zugriff auf den Rechner über das Netz möglich ist. Diese Runlevels eignen sich eher für Rechner, mit denen nicht direkt, sondern über das Netz gearbeitet wird.

1.2 Die Datei */etc/inittab*

Das Programm */sbin/init*, vom Kern aufgerufen, arbeitet die Datei */etc/inittab* ab. Dort finden sich Einträge, mit denen konfiguriert wird, welche Programme in welchem Runlevel auszuführen sind. Außerdem lassen sich hier Programme angeben, die bei besonderen Ereignissen wie dem Systemstart, einem Stromausfall oder der Betätigung der Tastenkombination STRG-ALT-ENTF ausgeführt werden sollen.

Die Datei ist folgendermaßen aufgebaut: Leerzeilen und Zeilen, die mit einem Kommentarzeichen (#) beginnen, haben keine Bedeutung. Jede andere Zeile spezifiziert Prozesse, die beim Wechsel in einen neuen Runlevel oder einem anderen Ereignis gestartet werden sollen. Solche Zeilen bestehen aus vier Feldern, die durch Doppelpunkte ohne Leerzeichen voneinander getrennt werden. Die Felder haben folgende Bedeutung:

1. Im ersten Feld wird dem Eintrag eine Kennung (ID) gegeben. Kennungen bestehen aus maximal vier Zeichen. Einträge, mit denen ein *getty*-Prozess gestartet wird, sollten als Kennung die Nummer der virtuellen Konsole tragen, auf welcher der Prozess gestartet werden soll.
2. Im zweiten Feld wird bestimmt, in welchem Runlevel der Prozess laufen soll. Es dürfen mehrere Runlevels angegeben werden. So bedeutet die Zeichenkette *23*, dass der Prozess in den Runlevels *2* und *3* laufen soll.
3. Im dritten Feld wird durch ein Schlüsselwort angegeben, wie der Prozess ausgeführt werden soll. Die wichtigsten der Schlüsselwörter sind:
   wait Der angegebene Prozess wird einmal gestartet, wenn in den Runlevel gewechselt wird. *init* wartet, bis der Prozess beendet ist.
   once Der angegebene Prozess wird einmal ausgeführt, wenn in den Runlevel gewechselt wird. Es wird nicht auf das Ende des Prozesses gewartet.
   sysinit Der Prozess wird zum Systemstart ausgeführt. *init* wartet, bis der Prozess beendet ist, bevor es weitere Programme aufruft.
   boot Der Prozess wird zum Systemstart ausgeführt, nachdem ein mit *sysinit* angegebener Prozess beendet ist. Auf die Beendigung dieses Prozesses wird nicht gewartet.
   bootwait Wie *boot* mit dem Unterschied, dass auf die Beendigung dieses Prozesses gewartet wird.

ctrlaltdel Der Prozess wird ausgeführt, sobald *init* das Signal SIGINT empfängt. Dies passiert dann, wenn die Tastenkombination STRG-ALT-ENTF betätigt wird. Gewöhnlich wird durch diesen Eintrag das Programm *shutdown* (S. **??**) aufgerufen.

respawn Der Prozess wird neu gestartet, sobald er aus irgendeinem Grund gestorben ist.

4. Im vierten Feld wird der Name des zu startenden Programms angegeben. Argumente, die dem Programm beim Aufruf übergeben werden sollen, können durch Leerzeichen getrennt hinter dem Namen des Programms stehen.

1.2.1 Festlegen des Standardrunlevels

Der Runlevel, in den automatisch nach dem Systemstart gewechselt wird, wird durch die folgende Zeile festgelegt:

```
id:2:initdefault:
```

Hier wird mit dem Runlevel-Feld festgelegt, welcher Runlevel als Standard benutzt werden soll (im Beispiel 2).

1.2.2 Systeminitialisierung

Während des Systemstarts wird von *init* ein Skript oder Programm aufgerufen, welches das System initialisiert und für die Benutzung vorbereitet. Dies wird durch den folgenden Eintrag festgelegt:

```
si::sysinit:/etc/init.d/rcS
```

Die Angabe eines Runlevel ist hier nicht notwendig, das zweite Feld ist leer.

1.2.3 Dauerhaft auszuführende Programme (*respawn*)

Zur Laufzeit des Systems muss es möglich sein, sich an den virtuellen Konsolen anzumelden. Deswegen wird nach Eintritt in die Runlevels 2 bis 5 das Programm */sbin/getty* einmal oder mehrmals gestartet:

```
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
...
6:23:respawn:/sbin/getty 38400 tty6
```

Das Programm (*getty*) hat die Aufgabe, ein Terminal zu initialisieren und in diesem Terminal auf eine Benutzeranmeldung zu warten. Nachdem ein Benutzer seinen Namen eingegeben hat, startet *getty* das Programm *login*, welches das Passwort des Benutzers erfragt und überprüft. Wenn die Überprüfung des Passworts erfolgreich ist, startet *login* die Standardshell des Benutzers. Das zu verwendende Terminal sowie die Bitrate, mit der mit dem Terminal kommuniziert werden soll, werden dem

Programm *getty* als Argumente übergeben. Das Programm versteht einige Optionen, die in der Manualseite beschrieben sind.

Beim Start der Programme *login* sowie der Standardshell wird der Programmcode des jeweiligen *getty* bzw. *login*-Prozesses durch den Code des zu startenden Programms ersetzt, das heißt, nach dem Start eines Programms besteht der Prozess weiterhin, jedoch mit einem anderen Programmcode (siehe *exec* (S. **??**)). Wenn die Standardshell beendet wird, weil sich der Benutzer abmeldet, wird der Prozess beendet.

Damit dies nicht dazu führt, dass an dem Terminal keine erneute Anmeldung möglich ist, überwacht *init* die mit dem Schlüsselwort *respawn* gestarteten Prozesse und startet sie erneut, falls sie beendet wurden. Die Folge ist, dass nach Beendigung der Shell des angemeldeten Benutzers ein neuer *getty*-Prozess gestartet wird und eine erneute Anmeldung möglich ist.

Neben dem normalen *getty*-Programm, das für die Verwendung an der Konsole gedacht ist, stehen weitere *getty*-ähnliche Programme zur Verfügung, die serielle Schnittstellen überwachen und somit die Anmeldung über ein Modem ermöglichen. Aber auch Aufgaben wie Anrufbeantworter (Paket: *mgetty-voice*) oder Faxempfang (Pakete *mgetty-fax* oder *hylafax-server*) werden durch *getty*-ähnliche Prozesse gesteuert. Hier ist es ebenfalls notwendig, eine spezifische Schnittstelle des Rechners zu initialisieren, sie zu überwachen und bestimmte Aktionen auszulösen, sobald ein Zugriff auf den Rechner über diese Schnittstellen erfolgt.

Debian-Pakete, die zusätzliche *getty*-Programme beinhalten, fügen der Datei */etc/inittab* normalerweise Beispieleinträge zu, die jedoch auskommentiert sind. Um die Programme zu verwenden, ist die Datei zu editieren.

1.2.4 Aktionen beim Wechsel des Runlevel

Beim Wechsel von einem Runlevel in einen anderen wird das Skript */etc/init.d/rc* aufgerufen. Dem Skript wird als Argument die Nummer des Runlevel übergeben, in den gewechselt werden soll. Dies wird durch folgende Zeilen in */etc/inittab* festgelegt:

```
l0:0:wait:/etc/init.d/rc 0
...
l6:6:wait:/etc/init.d/rc 6
```

Programme, die während eines Runlevel dauerhaft ausgeführt werden sollen und mit dem Schlüsselwort *respawn* versehen sind, werden beim Wechsel von einem Runlevel in den nächsten beendet, sofern sie für den neuen nicht vorgesehen sind. Programme, die für beide Runlevels definiert sind, werden nicht unterbrochen.

1.2.5 Single-User-Modus

Für den Single-User-Modus befindet sich folgender Eintrag in der Datei */etc/inittab*:

```
~~:S:wait:/sbin/sulogin
```

Der Eintrag bewirkt, dass nach Eintritt in diesen Runlevel das Programm */sbin/sulogin* aufgerufen wird. Das Programm erfragt das Passwort des Verwalters und ruft eine Shell auf, wenn dieses richtig eingegeben wird. Wenn der Prozess beendet wird, wird wieder zurück in den Standardrunlevel geschaltet.

Im Single-User-Modus werden nur wenige Programme im Hintergrund ausgeführt, alle Serverdienste sind gestoppt, und es besteht keine Möglichkeit, dass außer dem Verwalter irgendjemand mit dem Rechner arbeiten kann. Der Modus ist deswegen geeignet, kritische Arbeiten am System auszuführen. Wenn die Festplattenpartition, auf der sich die Home-Verzeichnisse der Benutzer befinden, überprüft werden muss, sollte das System zunächst in den Single-User-Modus gebracht werden. Die Partition kann dann sicher aus dem Dateisystem ausgehängt und überprüft werden.

Gelegentlich ist es notwendig, das System so zu starten, dass es in den Single-User-Modus und nicht in einen gewöhnlichen Runlevel bootet. Hierzu dient der Bootparameter *single*. Der Parameter wird vom Kern an *init* übergeben und weist das Programm an, direkt in den Single-User-Modus zu starten.

1.3 Benutzung von *init*

Zur Kommunikation mit dem Prozess *init* steht das Kommando *telinit* zur Verfügung. Mit *telinit* kann *init* veranlasst werden, den Runlevel zu wechseln oder die Konfigurationsdatei */etc/inittab* erneut zu lesen. Die Datei */sbin/telinit* ist ein symbolischer Link auf das Programm *init* selbst. Es ist auch möglich, das Programm *init* direkt zur Kommunikation mit dem eigentlichen *init*-Prozess zu verwenden.

1.3.1 Wechsel des Runlevel

Um in einen anderen Runlevel zu wechseln, ist *telinit* oder *init* mit der Bezeichnung des Runlevel, in den gewechselt werden soll, aufzurufen. Beispielsweise wird mit dem folgenden Kommando in den Runlevel 4 geschaltet:

```
debian:~# init 4
```

Auf diese Weise kann auch ein Systemabschluss (Runlevel 0) oder ein Neustart (Runlevel 6) durchgeführt werden. Der Single-User-Modus (Runlevel 1) kann alternativ mit der Bezeichnung *S* angegeben werden. Es wird dann direkt in den Single-User-Modus geschaltet. Dienste und Programme, die in dem vorherigen Runlevel ausgeführt wurden, werden dann nicht beendet:

```
debian:~# init S
```

Normalerweise sollte in den Single-User-Modus mit dem Kommando *init 1* gewechselt werden.

1.3.2 Erneutes Einlesen der Konfiguration

Die meisten Programme lesen ihre Konfigurationsdatei nur beim Start. Um solche Programme mit einer neuen Konfiguration zu verwenden, beendet man sie und startet sie dann erneut. Dies ist bei *init* nur möglich, wenn das System neu gestartet wird. Um dies im Falle einer Veränderung der Datei */etc/inittab* zu vermeiden, wird *init* durch folgendes Kommando dazu gebracht, die Konfigurationsdatei erneut zu lesen:

```
debian:~# init Q
```

Programme, die nach der Veränderung dieser Datei als im aktuellen Runlevel dauerhaft ausführbar eingetragen sind, also mit dem Schlüsselwort *respawn* versehen sind, werden durch dieses Kommando automatisch gestartet, wenn sie nicht schon aufgrund der alten Konfiguration gestartet waren. Programme, die in der alten Konfiguration als dauerhaft ausführbar eingetragen waren, in der neuen jedoch fehlen, werden beendet.

1.3.3 Ersetzen des Programms *init*

Während der Software-Aktualisierung kommt es vor, dass auch für *init* eine neue Programmversion installiert wird. Damit diese nicht erst beim nächsten Systemstart ausgeführt wird, sondern sofort nach der Installation, muss folgendes Kommando benutzt werden:

```
debian:~# init U
```

Die Folge ist, dass das laufende *init*-Programm sich selbst durch das neu installierte ersetzt. Dieser Schritt wird bei der Aktualisierung des Debian-Pakets (*sysvinit*) automatisch vollzogen.

1.3.4 Fehler

Wie beschrieben versucht *init* dauerhaft auszuführende Programme sofort neu zu starten, wenn sie beendet wurden. Wenn jedoch ein Programm nicht gestartet werden kann, etwa weil eine von diesem Programm benötigte Bibliothek auf dem System nicht vorhanden ist, ist die Folge, dass *init* immer wieder versucht, das Programm zu starten. Hierdurch wird unnötigerweise eine hohe Systemlast erzeugt.

Aus diesem Grund beobachtet *init*, wie oft es Programme neu starten muss, und gibt gegebenenfalls die folgende Fehlermeldung aus:

```
init: Id "xx" respawning too fast: disabled for 5
  minutes
```

Dabei wird für *xx* die ID aus dem Eintrag in der Datei */etc/inittab* ausgegeben. Daraufhin versucht *init* für fünf Minuten nicht mehr, dass Programm zu starten. Der Verwalter hat dann die Möglichkeit, das Problem zu beheben, indem er entweder den Eintrag aus der Konfigurationsdatei entfernt oder sicherstellt, dass das angegebene Programm ausführbar ist.

1.4 Start- und Stopskripte

Wie erwähnt führt *init* beim Systemstart das Programm */etc/init.d/rcS* aus. Hierbei handelt es sich um ein Shellskript, das die Aufgabe hat, alle Skripte oder Programme im Verzeichnis */etc/rcS.d* aufzurufen[1]. In diesem Verzeichnis befindet sich eine Reihe von Skripten, von denen jedes für einen bestimmten Teil des Systemstarts zuständig ist. Die Skripte werden in ASCII-alphabetischer Reihenfolge ihrer Namen ausgeführt. Die Namen dieser Skriptdateien müssen mit dem Buchstaben *S* (für Start) beginnen, damit sie von *rcS* aufgerufen werden. Dem Buchstaben *S* folgt eine zweistellige Zahl, die die Reihenfolge des Aurufs bestimmt.

Debian-Pakete, die Software für eine bestimmte Systemkomponente beinhalten, die beim Systemstart initialisiert werden muss, können diesem Verzeichnis ein Skript hinzufügen, das durch den beschriebenen Mechanismus beim Systemstart automatisch ausgeführt wird. Eine Veränderung der Datei */etc/inittab* ist normalerweise nicht notwendig.

Alle Startskripte befinden sich in Wirklichkeit im Verzeichnis */etc/init.d*. Im Verzeichnis */etc/rcS.d* sind lediglich symbolische Links auf die Skripte vorhanden. Wenn ein bestimmtes Skript nicht mehr während des Systemstarts ausgeführt werden soll, reicht es aus, den symbolischen Link im Verzeichnis */etc/rcS.d* zu löschen. Auf keinen Fall soll das Skript selbst aus dem Verzeichnis */etc/init.d* gelöscht werden.

Die meisten Skripte im Verzeichnis */etc/init.d* müssen mit einem der Parameter *start*, *stop* oder *restart* aufgerufen werden. Wie die Bezeichnungen der Parameter sagen, wird eine bestimmte Systemeigenschaft mit dem Parameter *start* gestartet, mit *stop* wird sie beendet, und mit *restart* wird sie zunächst beendet und dann erneut gestartet. Dies dient zum Starten von Server-Programmen mit veränderter Konfiguration. Man findet auch den Parameter *reload* zum erneuten Einlesen der Konfiguration ohne Unterbrechung des Prozesses.

1.4.1 Startskripte für den Systemstart

Die folgenden symbolischen Links auf die Skripte im Verzeichnis */etc/init.d* sind im Verzeichnis */etc/rcS.d* vorhanden:

S10checkroot.sh Das Skript bindet die Swapbereiche ein und überprüft das Rootdateisystem. Nach erfolgreicher Prüfung wird das Rootdateisystem zu Lesen und Schreiben geöffnet.

S20modutils Ruft *depmod* (S. **??**) auf, um Modulabhängigkeiten zu berechnen und lädt Module, die in der Datei */etc/modules* eingetragen sind.

S30checkfs.sh Prüft alle Dateisysteme außer dem Rootdateisystem.

S35mountall.sh Bindet alle Dateisysteme, die sich auf dem lokalen Rechner befinden, mit Ausnahme des Rootdateisystems ein.

S40hostname.sh Setzt den Rechnernamen dem Eintrag in der Datei */etc/hostname* gleich.

[1] Aus Gründen der Kompatibilität zu älteren Versionen von Debian GNU/Linux werden die Skripte auch im Verzeichnis */etc/rc.boot* aufgeführt.

S40network Initialisiert das Netz. Es stehen zwei Methoden zur Verfügung. Zum einen dieses relativ simple Skript und zum anderen das Skript *S40networking* aus dem Paket *netbase*, das die Netzkonfiguration den Angaben in der Konfigurationsdatei */etc/network/interfaces* vornimmt. Wenn Sie das zweite Verfahren verwenden (empfohlen), sollten alle Einträge in *S40network* auskommentiert sein.

S45mountnfs.sh Bindet über NFS Dateisysteme ein, die sich physikalisch auf anderen Rechnern im Netz befinden.

S55bootmisc.sh Führt einige Aufräumarbeiten durch und speichert beim Booten erzeugte Kernmeldungen in der Datei */var/log/dmesg*.

S55urandom Initialisiert den Zufallzahlengenerator des Kerns.

Welche Skripte zusätzlich ausgeführt werden, hängt davon ab, welche Pakete auf dem System installiert sind. Viele Startskripte können über die Datei */etc/default/rcS* konfiguriert werden. Dort wird eingestellt, ob die Startskripte während ihrer Ausführung anzeigen, was sie gerade tun. Das beruhigt manche Verwalter.

1.4.2 Start- und Stopskripte für einzelne Runlevel

Neben dem Verzeichnis */etc/rcS.d* existieren im Verzeichnis */etc* sieben weitere Verzeichnisse, die eine ähnliche Aufgabe haben. Ihre Namen lauten *rc0.d*, *rc1.d* bis *rc6.d*. In diesen Verzeichnissen befinden sich ebenfalls symbolische Links auf Skripte im Verzeichnis */etc/init.d*, die beim Wechsel in den jeweiligen Runlevel ausgeführt werden.

Beim Wechsel von einen Runlevel in einen anderen wird durch *init* das Skript */etc/init.d/rc* ausgeführt. Dieses Skript verhält sich ähnlich wie das beim Systemstart ausgeführte Skript */etc/init.d/rcS*. Es ruft die Skripte in dem zugehörigen *rc?.d*-Verzeichnis auf.

Es gibt einen Unterschied zwischen dem Systemstart und dem Wechsel zwischen Runlevels. Während beim Systemstart nur Programme gestartet oder Teile des Systems initialisiert werden, ist es beim Wechsel des Runlevel erforderlich, bestimmte Programme zu beenden. In den Verzeichnissen *rc0.d* bis *rc6.d* befinden sich deswegen zum einen symbolische Links, deren Namen mit *K* (für Kill) beginnen und zum anderen solche, deren Namen mit *S* (für Start) beginnen. Beim Wechsel in einen neuen Runlevel werden zunächst alle Skripte, deren Link-Namen mit *K* beginnen, mit dem Parameter *stop* aufgerufen und danach die *S*-Skripte mit dem Parameter *start*. Beide Typen von Links zeigen auf dieselben Skripte im Verzeichnis */etc/init.d*.

Wenn das Paket *gpm* zur Mausunterstützung an der Konsole installiert ist, befindet sich im Verzeichnis */etc/init.d* ein Skript mit dem Namen *gpm*. Dieses Skript kann – wie üblich – mit den Parametern *start* oder *stop* aufgerufen werden, wodurch die Mausunterstützung aktiviert bzw. deaktiviert wird. In den Verzeichnissen */etc/rc2.d* bis */etc/rc5.d* befinden sich symbolische Links auf dieses Skript, die den Namen *S20gpm* tragen. Diese Skripte werden beim Wechsel in die Runlevels mit dem Parameter *start* aufgerufen, sodass die Mausunterstützung in diesen Runlevels

zur Verfügung steht. In den Verzeichnissen *rc0.d*, *rc1.d* und *rc6.d* befinden sich hingegen Links auf das gleiche Skript, die den Namen *K20gpm* tragen. Beim Wechsel in diese Runlevels wird das Skript mit dem Parameter *stop* aufgerufen, der Dienst beendet.

Wichtige Startskripte, die beim Wechsel in den Standardrunlevel (2) ausgeführt werden, sind:

S10sysklogd Startet den Systemlog-Dämon und den Kernlog-Dämon. Die beiden Programme protokollieren Systemereignisse und Kernmeldungen.
S11pcmcia Startet und initialisiert Programme für PCMCIA-Hardware (PC-Cards).
S18portmap Startet den Portmap-Dämon.
S19nfs-common Startet Programme, die für die Verwendung von NFS benötigt werden.
S19nis Startet – je nach Konfiguration – NIS-Server- und/oder NIS-Klientprogramme für NIS-Cluster.
S20acct Startet die Protokollierung der Systembenutzung (Accounting).
S20exim Startet das Mail-Transport-Programm *exim*.
S20gpm Startet die Mausunterstützung für die Konsole.
S20hylafax Startet das Hylafaxsystem zum Faxen.
S20inetd Startet den Internet-Dämon *inetd*.
S20isdnutils Initialisiert ISDN-Karten und startet ISDN-Programme.
S20logoutd Startet das Programm zum automatischen Abmelden von Benutzern.
S20lprng Startet das Druck- und Spoolsystem *lprng*.
S20ppp Startet den PPP-(Einwahl-)Dienst, falls dieser für den automatischen Start konfiguriert ist.
S20quota Startet Programme zur Überwachung von Speicherplatzbeschränkungen für Benutzer.
S20samba Startet den SMB-Server, der Drucker und Verzeichnisse für MS-Windows-Rechner zur Verfügung stellt.
S20xfs Startet den X11-Font-Server.
S25nfs-server Startet den NFS-Server, wodurch andere Rechner nach Konfiguration auf Verzeichnisse dieses Rechners zugreifen können.
S89atd Startet den *at*-Dämon, der mit *at* (S. **??**) oder *batch* (S. **??**) in Auftrag gegebene Prozesse zu gegebener Zeit ausführt.
S89cron Startet den *cron*-Dämon, der Prozesse startet, die regelmäßig wiederkehrend ausgeführt werden sollen.
S91apache Startet den HTTP-Server *apache*.
S99rmnologin Ermöglicht gewöhnlichen Benutzern die Anmeldung an der Konsole oder an virtuellen Terminals.

Welche Skripte auf Ihrem System vorhanden sind, ist abhängig davon, welche Pakete installiert sind.

1.4.3 Manuelles Verwenden von Startskripten

Wenn Sie einen bestimmten Dienst neu konfiguriert haben, ist es in der Regel notwendig, die Programme, die diesen Dienst zur Verfügung stellen, erneut zu starten. Dies lässt sich auf verschiedene Arten durchführen. Die aufwendigste besteht darin, das ganze System neu zu starten. Einfacher ist es jedoch, das Startskript dieses Dienstes aus dem Verzeichnis */etc/init.d* zu verwenden und es mit dem Parameter *restart* aufzurufen. Angenommen, Sie haben die Pakete *lprng* und *magicfilter* installiert, um Ihren Drucker zu betreiben. Nun haben Sie eine Änderung an der Datei */etc/printcap* vorgenommen, in der die verfügbaren Drucker aufgelistet sind (siehe Kap. **??**, S. **??**). Sie können das Drucksystem danach neu starten, indem Sie folgendes Kommando eingeben:

```
debian:~# /etc/init.d/lprng restart
```

Selbstverständlich können Sie auch die Parameter *stop* und *start* verwenden, um während des Betriebs einen bestimmten Dienst zu beenden oder einen Dienst zu starten, der normalerweise nicht ausgeführt wird. Einige Startskripte stellen zusätzlich den Parameter *reload* zur Verfügung. Der Parameter bewirkt, dass die durch solche Skripte gestarteten Programme ihre Konfigurationsdateien erneut lesen, ohne gestoppt und wieder gestartet zu werden.

1.5 Verwalten der Runlevels

1.5.1 Einrichten von Start-Stop-Links

Die Verzeichnisse */etc/rc0.d* bis *rc6.d* bestimmen, welche Dienste beim Wechsel in einen Runlevel gestartet oder beendet werden. Standardmäßig werden alle Startskripte so installiert, dass die Dienste in den Runlevel *2* bis *5* gestartet und in den Runlevel *0*, *1* und *6* beendet werden. Die Dienste stehen zur Laufzeit des Systems zur Verfügung und werden beendet, wenn das System angehalten, neu gestartet oder in den Single-User-Modus gebracht wird.

Um einen Dienst in einem Runlevel nicht mehr auszuführen, reicht es, den Start-Link aus dem *rc?.d* Verzeichnis zu entfernen. Außerdem sollte dann in diesem Verzeichnis ein Stop-Link angelegt werden, damit der Dienst beendet wird, wenn in den Runlevel gewechselt wird.

Angenommen, Sie möchten in Runlevel 3 den X Display Manager *xdm* benutzen, der im Standardrunlevel 2 nicht aktiv sein soll. Außerdem gibt es eine Unverträglichkeit zwischen dem Paket *gpm* (Mausunterstützung für die Konsole) und dem X-Server. Der Dienst *gpm* darf nicht ausgeführt werden, wenn *xdm* läuft. Um zu verhindern, dass *xdm* in Runlevel 2 ausgeführt wird, ist folgendes Kommando einzugeben:

```
debian:~# rm /etc/rc2.d/S99xdm
```

Analog dazu wird mit dem nächsten Kommando verhindert, dass *gpm* in Runlevel 3 gestartet wird:

```
debian:~# rm /etc/rc3.d/S20gpm
```

Nun muss ein Stop-Link für Runlevel 2 erzeugt werden, damit *xdm* beendet wird, wenn in diesen Runlevel gewechselt wird. Dies geschieht mit dem Kommando *ln* (S. **??**):

```
debian:~# ln -s /etc/init.d/xdm /etc/rc2.d/K01xdm
```

Ein weiterer Stop-Link muss eingerichtet werden, damit *gpm* beendet wird, wenn in den Runlevel 3 gewechselt wird:

```
debian:~# ln -s /etc/init.d/gpm /etc/rc3.d/K20gpm
```

Wie die zu löschenden Links und zu verknüpfenden Skripte heißen, können Sie leicht herausfinden, indem Sie sich den Inhalt der Verzeichnisse mit *ls* (S. **??**) anzeigen lassen. Wenn Sie sich nicht sicher sind, in welcher Reihenfolge Dienste beendet werden sollen, empfiehlt es sich, den Inhalt des Verzeichnisses */etc/rc0.d* zu untersuchen. Aus den Namen der hier befindlichen Links lässt sich auf die Reihenfolge schließen, mit der alle Dienste ordnungsgemäß beendet werden, wenn das System angehalten wird.

KDE-Benutzer können zum Editieren von Runlevels den SysV-Runlevel-Editor benutzen. Das Programm ist in dem KDE-Paket *kdeadmin* enthalten (Programmname: *ksysv*). Eine angenehme Eigenschaft dieses Programms ist, dass es schnell einen Überblick über alle Runlevels bietet.

Die Start- und Stop-Skripte im Verzeichnis */etc/init.d* sind Konfigurationsdateien. Es ist möglich, sie zu verändern, ohne dass sie bei einer Aktualisierung des Systems automatisch überschrieben werden. Grundsätzlich ist dies jedoch in den meisten Fällen nicht zu empfehlen. Wenn Programme gestartet werden sollen, die normalerweise nicht gestartet werden und für die keine Startskripte zur Verfügung stehen, ist in Erwägung zu ziehen, ein eigenes Startskript zu schreiben (kopieren, editieren) und zu installieren. Ein Beispiel ist in Kapitel **??**, S. **??** dargestellt.

1.5.2 Einrichten von Start-Stop-Skripten

Wenn Sie ein eigenes Startskript erstellt haben, muss dieses dorthin kopiert werden, wo sich auch alle anderen Skripte dieser Art befinden, also in das Verzeichnis */etc/init.d*. Daraufhin müssen die symbolischen Links in den verschiedenen */etc/rc?.d*-Verzeichnisse erzeugt werden. Dies kann manuell geschehen oder mit dem Programm *update-rc.d*, das auch bei der Installation von Debian-Paketen verwendet wird. Das Programm *update-rc.d* kann auf folgende Arten aufgerufen werden:

```
update-rc.d Skriptname defaults [Zahl | Zahl-start
Zahl-stop]
```

In diesem Fall werden Startlinks in den Verzeichnissen *rc2.d* bis *rc5.d* sowie Stoplinks in den Verzeichnissen *rc0.d*, *rc1.d* und *rc6.d* auf das mit *Skriptname* bezeichnete Startskript gelegt. *Skriptname* ist ohne Verzeichnisnamen anzugeben. Optional kann mit *Zahl* angegeben werden, welche Zahl die Namen der Links nach

den Buchstaben *S* bzw. *K* erhalten. *Zahl* muss als zweistellige Dezimalzahl angegeben werden. Damit wird festgelegt, an welcher Stelle das Skript beim Wechsel in einen Runlevel aufgerufen wird. Wenn unterschiedliche Stellen in der Reihenfolge des Aufrufs beim Starten und beim Beenden des Dienstes benötigt werden, ist die Form *Zahl-start* und *Zahl-stop* zu verwenden. Ist *Zahl* nicht angegeben, wird der Standardwert *20* benutzt.

Wenn das selbsterstellte Skript *midid* in das Verzeichnis */etc/init.d* kopiert worden ist, werden die notwendigen Links mit diesem Kommando erzeugt:

```
debian:~# update-rc.d midid defaults
```

Wenn Start- oder Stoplinks nur für einige Runlevels angelegt werden sollen, wird *update-rc.d* in dieser Form aufgerufen:

```
update-rc.d Skriptname start|stop Zahl Runlevel ...
start|stop
Zahl Runlevel ...
```

Dabei ist mit *start* oder *stop* anzugeben, ob Start- oder Stoplinks erzeugt werden sollen, mit *Zahl* die Stelle in der Reihenfolge des Aufrufs und mit *Runlevel* den oder die Runlevels, für die der Link erzeugt werden soll. Wenn das Skript *midid* in den Runlevels 0-3 und 6 als erstes gestoppt und in den Runlevels 4 und 5 als letztes gestartet werden soll, ist das Kommando folgendermaßen zu verwenden:

```
debian:~# update-rc.d midid start 99 4 5 stop 01 0 1 2
  3 6
```

Der Skriptname *midid* ist dabei durch den tatsächlichen Skriptnamen zu ersetzen.

Achtung: Wenn sich bereits ein Link auf das Startskript in den *rc?.d*-Verzeichnissen befindet, bleibt der Aufruf von *update-rc.d* wirkungslos. Es wird dann davon ausgegangen, dass bereits manuell lokale Einstellungen vorgenommen wurden, die nicht überschrieben werden sollen. In diesem Fall müssen Sie weitere Links ebenfalls manuell erzeugen.

Um ein Startskript vollständig zu entfernen, muss es zunächst aus dem Verzeichnis */etc/init.d* gelöscht werden. Die Verweise werden dann mit folgendem Kommando entfernt:

```
debian:~# update-rc.d midid remove
```

Auch hier muss *midid* durch den tatsächlichen Skriptnamen ersetzt werden.