Korrekturhinweis

Klaus Schmidt, *Ubuntu Desktop und Server. Ubuntu 11.04: Installation, Anwendung, Administration*, Carl Hanser Verlag 2011 (ISBN: 978-3-446-42792-1)

Berichtigung von Kapitel 9.9, „Ein System-Upgrade durchführen“ (S. 254):

Das Kommando `apt-get dist-upgrade` haben die Entwickler von APT für ein Distributions-Upgrade entwickelt, also für ein Upgrade auf ein neues Release. Unter Debian arbeitet dieses Kommando auch so, wenn in der Datei "/etc/apt/sources.list" Release-Namen wie "stable", "testing" etc. eingetragen sind.

Unter Ubuntu sind diese Release-Namen nicht möglich. Hier müssten erst alle Release-Namen in dieser Datei zum Beispiel von "maverick" auf "natty" geändert werden. Ein Hinweis darauf wurde im Buch leider übersehen. Daher ist diese Option unter Ubuntu nur dann interessant, wenn weitergehende Abhängigkeiten unter den Paketen aufzulösen sind.

Das Kommando `do-release-upgrade` erledigt unter anderem auch die Änderungen in der Datei "/etc/apt/sources.list" automatisch. Deshalb ist dieses Kommando für ein Release-Upgrade, wie es auf S. 254 beschrieben ist, wesentlich besser geeignet. Eine Garantie auf ein problemloses Upgrade gibt es jedoch auch damit nicht (vgl. Praxistipp auf S. 254).

Berichtigung von Kapitel 6, „Arbeiten auf der Kommandozeile“, und Kapitel 11, „Der Bootvorgang“

In Kapitel 6 und 11 sind versehentlich Backticks in Hochkommata gewandelt worden, was dazu führt, dass Befehlseingaben und Skripte nicht funktionieren bzw. Fehlermeldungen produzieren.

Anbei finden Sie die Seiten 115, 125, 128, 155, 265 und 278 zum Austausch.

Gleiches gilt für Dateinamen. Der große Vorteil dabei ist, dass sich der Anwender nicht mehr verschreiben kann. Hiermit verlieren lange Dateinamen ihre Schrecken, da oft nur ein paar Buchstaben genügen, damit die Shell den Namen vervollständigen kann.

Neuere Versionen der Bash ermöglichen eine Suchfunktion über die Tastenkombination [Ctrl] + [R]. Nach deren Eingabe erhält man einen anderen Prompt und kann nun die Buchstaben von einem Befehl oder einer Datei eingeben:

```
reverse-i-search)`a': rm abc
```

Die eingegebenen Buchstaben (im obigen Fall „a“) erscheinen hinter dem Prompt, und der letzte Befehl, der hierzu passt, ist hinter dem Doppelpunkt zu sehen. Das Drücken von [Return] aktiviert diesen Befehl erneut.

Eine mehrfache Eingabe von [Ctrl] + R liefert der Reihe nach eine Rückwärtssuche nach allen vorangegangenen Treffern, die auf die bereits eingegebenen Buchstaben zutreffen. Durch Betätigen von [Ctrl] + C lässt sich die Suchfunktion wieder abschalten, ohne einen Befehl zu aktivieren.

Mit dem Paket `bash-completion` (das unter Ubuntu per Default installiert ist) gelingt eine Vervollständigung auch mit den Optionen von Befehlen, wenn Sie vorher einen Bindestrich eingeben.

Ein kleiner Tipp am Rande: Drücken Sie die Tab-Taste zweimal hintereinander, ohne einen Text einzugeben. Sie erhalten damit die Anzahl aller möglichen Kommandos und die Frage, ob alle angezeigt werden sollen.

Start eines Programms

Zum Start eines Programms oder eines Kommandos wird der Name des Programms nach dem Eingabe-Prompt der Shell in die Kommandozeile getippt (wie z. B. `ls`), oft gefolgt von weiteren Angaben für dieses Programm, die zur Abarbeitung wichtig sind.

Einem Programm können Optionen mitgegeben werden, die die Abarbeitung beeinflussen (wie z. B. die Ausgabe in Langform bei `ls -l`). Sie heißen so, weil es keinen Zwang zu deren Eingabe gibt, sie also optional sind. Sie sind unter UNIX durch ein vorangestelltes „`-`“ oder „`--`“ gekennzeichnet. Optionen sind fest im Programm eingebaut und können zwar benutzt, aber nicht verändert werden.

Dagegen sind Parameter bestimmte Angaben vom Benutzer, die vom Programm nicht vorhersehbar sind, wie z. B. welche Datei bzw. welches Verzeichnis geöffnet werden soll. Oft sind bestimmte Parameter in einem Programm vorbelegt, wenn sie vom Benutzer keinen Wert erhalten. Sie bekommen dann einen Default-Wert. Ein Beispiel ist das Programm `ls`, welches das aktuelle Verzeichnis ausgibt, wenn kein Verzeichnis vom Benutzer angegeben ist. Der Wert „`.`“ ist damit der Default-Parameter von `ls`.

Ein Beispiel:

```
user@ubuntu:~$ ls -l /home
```

Hier ist das Programm `ls` mit der Option `-l` (die Ausgabe soll in Langformat erfolgen) und dem Parameter `/home` (welches Verzeichnis soll vom Programm bearbeitet werden) aufgerufen. Ein Backslash hat eine komplett andere Bedeutung als unter DOS! Er leitet keine Option ein, sondern er entwertet ein Zeichen.

`[!0-2]` durch das Ausrufezeichen wird die Menge verneint (hier: nicht 0, 1 oder 2)

Beispiele:

Alle Dateien und Verzeichnisse im aktuellen Verzeichnis ausgeben:

```
user@ubuntu:~$ ls *
```

Alle Dateien und Verzeichnisse im aktuellen Verzeichnis ausgeben, die mit dem Kleinbuchstaben `c` beginnen:

```
user@ubuntu:~$ ls c*
```

Alle Dateien und Verzeichnisse im aktuellen Verzeichnis ausgeben, die mindestens drei Zeichen enthalten:

```
user@ubuntu:~$ ls ???
```

Alle Dateien und Verzeichnisse im aktuellen Verzeichnis ausgeben, die entweder mit `c` oder `C` beginnen:

```
user@ubuntu:~$ ls [cC]*
```

Alle Dateien und Verzeichnisse im aktuellen Verzeichnis ausgeben, die mit den Buchstaben `a`, `b`, `c`, `r` oder `s` beginnen:

```
user@ubuntu:~$ ls [a-c,rs]*
```

Alle Dateien und Verzeichnisse im aktuellen Verzeichnis ausgeben, die nicht mit `f` beginnen:

```
user@ubuntu:~$ ls [!f]*
```

PRAXISTIPP: Passt das Suchmuster auch auf Verzeichnisse, so wird mit dem Kommando `ls` deren Inhalt mit ausgegeben.

Beispiel:

Unter `/etc` alle Verzeichnisse mit drei Buchstaben ausgeben zusammen mit dem Inhalt.

```
user@ubuntu:~$ ls /etc/???
```

Ist dieses Verhalten nicht gewünscht, kann es mit der Option `-d` abgeschaltet werden. Dann zeigt `ls` nur noch die Namen von passenden Dateien und Verzeichnissen an, aber nicht mehr den Inhalt der Verzeichnisse:

```
user@ubuntu:~$ ls -d /etc/???
```

Namen von Dateien und Verzeichnissen können maximal 255 Zeichen lang sein.

6.3.12 Sonderzeichen

Sonderzeichen sind das Leerzeichen (Blank) „ “ und folgende Zeichen:

? * $ & ( ) [ ] { } < > ! | ~ ; # - `

Sie haben in Linux eine besondere Bedeutung (Blank als Trennzeichen; * als Wildcard etc.) und sollten in Dateinamen nicht verwendet werden! In der Regel interpretiert die Shell Sonderzeichen. Sind sie in Namen enthalten, muss entweder der Name in Anführungszeichen

Globale Variablen

Damit eine Umgebungsvariable auch weitergegeben bzw. vererbt wird, muss sie mit dem Kommando `export` exportiert werden. Nur dann ist sie in den in dieser Shell gestarteten Programmen und in danach gestarteten Shells (Subshells) über die Vererbung bekannt, aber nicht in anderen Shells!

Damit auch andere Shells diese Variable kennen, muss sie in der Datei `.bashrc` bzw. bei Login-Shells in der Datei `.profile` im Homeverzeichnis eines Benutzers oder für alle Benutzer unter `/etc/profile` gesetzt werden. Mit dem Kommando `export` ohne Parameter aufgerufen können alle globalen Variablen angezeigt werden; mit `set` werden zudem alle anderen Shell-Variablen angezeigt.

Beispiel:

```
user@ubuntu:~$ export var7="Das ist ein Test"
user@ubuntu:~$ echo $var7
Das ist ein Test
```

6.3.15 Kommandosubstitution

Wenn Sie beim Aufruf eines Programms dessen Ausgabe weiter verwenden wollen, benötigen Sie diese Technik. Sie müssen hierzu das Programm entweder in Backticks (bzw. Graves `...`) oder in eine runde Klammer mit Dollarzeichen $(...) einschließen. Dann kann die Ausgabe von diesem Programm von anderen Programmen weiter benutzt werden oder dazu dienen, den Inhalt einer Umgebungsvariablen zu füllen.

Achtung: Bei einigen Shells sind nicht alle Schreibweisen möglich!

Beispiele mit gleicher Wirkung:

```
user@ubuntu:~$ echo `date`
user@ubuntu:~$ echo $(date)
```

Das Ergebnis in eine Variable speichern:

```
user@ubuntu:~$ datum=$(date)
user@ubuntu:~$ echo $(datum)
Di 8. Mär 12:55:46 CET 2011
```

6.3.16 Der Alias-Befehl

Ein Alias ist ein Befehl (ein eingebautes Shell-Kommando), der unter einem anderen Namen (Alias) benutzt wird. Es kann auch eine Befehlsfolge sein. Damit können Sie sich eine Menge Tipparbeit ersparen, wenn Sie sich solche Befehle selbst eintragen.

Beispiel: neues Alias `"gross_erst"` einfügen:

```
user@ubuntu:~$ alias gross_erst="ls -lS"
```

alle Alias-Befehle anzeigen mit:

```
user@ubuntu:~$ alias
```

Im Kommando eingebaute Hilfe

Die meisten Kommandos bringen eine kurze Hilfe mit. Sie wird in der Regel ausgegeben entweder über die Option `-h` oder `--help`.

Beispiel:

```
user@ubuntu:~$ ls --help
```

Die Datei `/etc/issue`

In der Datei `/etc/issue` wird vom Distributor die Information über die Version dieser Linux-Distribution abgelegt. Sie wird vor jedem Login auf einer Konsole vom Prozess `mingetty` ausgewertet und angezeigt.

Beispiel für den Inhalt der Datei `/etc/issue`:

```
Ubuntu 10.10 \n \l
```

Nach der Anmeldung in einer Konsole erscheint z. B. dieser Text:

```
Ubuntu 10.10 Rechnername tty1
```

Das Zeichen `\n` wird durch den Rechnernamen ersetzt und `\l` durch den Namen der Konsole. Dagegen wird die Datei `/etc/issue.net` nur bei einer Anmeldung über das Netzwerk (z. B. über `telnet` oder `ssh`) benutzt. Sie enthält meist weniger Informationen als `/etc/issue`, um Angreifern nicht zu helfen.

Ein Beispiel für den Inhalt der Datei `/etc/issue.net`:

```
Ubuntu 10.10
```

Das Kommando `wall`

Über das Programm `wall` kann als Nachricht an alle angemeldeten Benutzer der Inhalt einer Datei oder interaktiv (bis zur Eingabe von CTRL D) geschickt werden. Die Mitteilung wird in allen Konsolen eingeblendet; bei einer grafischen Oberfläche in einem Popup-Fenster.

Beispiel: Den Inhalt der Datei `test` an alle Benutzer schicken:

```
user@ubuntu:~$ wall < test
```

Das Programm `uname`

Es gibt Informationen über den gestarteten Kernel und weitere Systemeigenschaften aus.

Beispiel:

In das Modul-Verzeichnis `misc` des aktiven Kernels wechseln:

```
root@ubuntu:~# cd /lib/modules/`uname -r`/misc
```

6.7.2 Suchen nach Programmen

`which`

Diese Shell-Funktion (bzw. Programm) durchsucht die Einträge im Suchpfad und gibt den kompletten Pfad des ersten gefundenen Programms im Filesystem ohne Manual-Seiten aus. Oft ist die Option `-a` erforderlich.

fügt. Es wird auch bei jedem Kernel-Update automatisch gestartet und aktualisiert dabei den Grub-Startmechanismus.

Einträge in der Datei `/etc/default/grub` bestimmen das generelle Verhalten von Grub2. Wenn man Änderungen am Boot-Verhalten einbringen will, dann sollte dies nur in dieser Datei erfolgen.

Ein Beispiel:

```
GRUB_DEFAULT=0
#GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""
```

Hier sind im Wesentlichen die erste Zeile und die vierte Zeile interessant. Die Nummer nach "`GRUB_DEFAULT`" in der ersten Zeile legt das Betriebssystem fest, das zu starten ist, wenn der Benutzer nicht eingreift. Die Zählweise beginnt hier mit „0“. Es bedeutet „0“, dass der erste Eintrag in `/boot/grub/grub.cfg` benutzt werden soll; „1! steht für den zweiten Eintrag usw.

In Zeile 4 bedeutet die Zahl hinter „`GRUB_TIMEOUT`“ die Anzahl von Sekunden, die Grub auf eine Eingabe vom Benutzer wartet. Falls Sie immer die Bootmeldungen sehen wollen, können Sie in der vorletzten Zeile die Einträge `quiet` und `splash` entfernen.

Erst nach folgendem Kommando werden geänderte Einträge in das Boot-Menü übernommen:

```
root@ubuntu:~# update-grub
```

Unter Ubuntu kann das Boot-Menü recht unübersichtlich werden, wenn mehr als eine Linux-Installation sich zusammen mit Windows auf dem Rechner befinden. Zudem kann es vorkommen, dass man eine Installation nur über eine bestimmte Boot-Option starten kann. Diese Option bei jedem neuen Start von Hand einzugeben, ist kaum praktikabel.

Die Datei `/boot/grub/grub.cfg`, die für die Einträge im Boot-Menü zuständig ist, sollte nicht editiert werden. Sie würde beim nächsten Kernel-Update sowieso wieder überschrieben werden. Stattdessen ist es möglich, eine eigene Datei für das Boot-Menü zu erstellen, deren Inhalt am Ende des Boot-Menüs automatisch angehängt wird. Sie muss den Namen `/boot/grub/custom.cfg` haben und darf nur Einträge der Art „`menuentry ...`" enthalten. Damit einer dieser Einträge als Default gestartet wird, ist eine Anpassung in `/etc/default/grub` erforderlich, z. B „`GRUB_DEFAULT=5`“.

Damit Sie den Eintrag nicht nach jedem Kernel-Update zu ändern brauchen, können Sie die symbolischen Links benutzen, die auf die aktuellen Dateien von Kernel (hinter dem Schlüsselwort „`linux`“) und `Initrd` (der Initial RAM-Disk hinter dem Wort „`initrd`“) zeigen.

Hier ein Beispiel für die Datei `/boot/grub/custom.cfg`:

```
menuentry 'Ubuntu' --class ubuntu --class gnu-linux --class gnu --class os {
   recordfail
   insmod part_msdos
   insmod ext2
   set root='(hd0,msdos14)'
   search --no-floppy --fs-uuid --set 022b3657-b7b3-48a2-8788-8e0e71ef7401
   linux   /vmlinuz root=UUID=022b3657-b7b3-48a2-8788-8e0e71ef7401 ro nomodeset
   initrd /initrd.img
}
```

Siehe auch: *http://wiki.ubuntuusers.de/dienste*
man update-rc.d
/usr/share/doc/upstart/README:Debian.gz

Weitere Programme, die im Hintergrund aktiv sind, können Sie über die grafische Oberfläche Gnome steuern. Über *System / Einstellungen / Startprogramme* erhalten Sie das entsprechende Programm. Dies sind zwar im weiteren Sinn auch Dienste, sie sind aber nicht systemweit, sondern nur für den angemeldeten Benutzer aktiv. Siehe hierzu Kapitel 3.

In UNIX und damit auch in Linux und Ubuntu werden derartige Hintergrundprozesse oft auch „Daemon“ genannt. Daher kommt das „d“ am Ende des Namens vieler derartiger Prozesse, wie `powerd`, `xinetd`, `saned` etc. Dies hat nichts mit Dämonen zu tun, sondern der Name „Daemon“ bezeichnet dabei einen dienstbaren Geist.

Besonders im Zusammenhang mit der Einrichtung von Server-Programmen bzw. dem Start von Diensten im folgenden Kapitel ist ein zumindest ansatzweises Verständnis der Startmechanismen von Vorteil.

11.8 Check von Diensten

Upstart ist vor allem im Server-Einsatz noch umstritten. Hier geht es weniger um schnelles Booten, sondern um den zuverlässigen Start wichtiger Dienste. Falls es einmal vorkommen sollte, dass ein Dienst nicht startet oder sich nach einiger Zeit beendet, können Sie sich ein Checkprogramm erstellen.

Eine Möglichkeit ist die Erstellung eines Skripts, das in regelmäßigen Abständen von `cron` aufgerufen wird und prüft, ob ein Prozess noch läuft. Hier ein Beispiel, das beim Booten des Rechners gestartet werden sollte und beliebig erweiterbar ist. Wenn der Start eines Hintergrundprozesses nicht klappt, erfolgt ein Eintrag in der Datei `/tmp/start` und der Benutzer `root` erhält die Fehlermeldung per Mail:

```
#!/bin/bash
# Dieses Script prueft regelmaessig, ob bestimmte Prozesse noch laufen.
# Falls nicht, startet es diese neu.
#
# Pruefung vom Web-Server
HTTPD=`ps ax|grep -v grep| grep -c http`
if [ $HTTPD == "0" ]; then
    echo "`date`: Apache nicht da" >> /tmp/start
    echo "" >> /tmp/start
    /etc/init.d/apache start >> /tmp/err >/dev/null
    mail root@localhost -s Startproblem mit Apache < /tmp/err
fi
#
# Pruefung vom Mail-Server
...
```