Jobs ausführen mit cron

Computer wurden in erster Linie entwickelt, um Routineaufgaben automatisch zu erledigen. Wenn Sie täglich morgens um eins Ihre Festplatte sichern müssen, besteht kein Grund, dies jedesmal von Hand zu erledigen - erst recht nicht, wenn Sie dafür aus dem Bett kriechen müßten. Es sollte möglich sein, den Computer einmal anzuweisen, den Job zu erledigen, und ihm die Angelegenheit dann zu überlassen. Auf Unix-Systemen übernimmt *cron* diese Art der Automatisierung für Sie. Kurz gesagt, benutzen Sie *cron*, indem Sie den Befehl *crontab* aufrufen und Zeilen in einem bestimmten Format eingeben, das von *cron* erkannt wird. Jede Zeile enthält einen auszuführenden Befehl sowie die Uhrzeit der Ausführung.

Hinter Ihrem Rücken speichert *crontab* die Befehle in einer Datei im Verzeichnis */var/spool/cron/crontabs*, die Ihren Benutzernamen trägt. (Die *crontab*-Datei für den Benutzer mdw würde also */var/spool/cron/crontabs/mdw* heißen.) Ein Dämon namens *crond* liest diese Datei regelmäßig und führt zu den richtigen Zeiten die Befehle aus. Eine der *rc*-Dateien Ihres Systems startet beim Booten den *crond*. Es gibt eigentlich keinen Befehl namens *cron*, sondern nur das Programm *crontab* und den Dämon *crond*.

Auf einigen Systemen kann *cron* nur von **root** benutzt werden. Wir wollen uns einen nützlichen Befehl ansehen, den Sie als **root** vielleicht ausführen möchten, und dabei zeigen, wie Sie daraus einen Eintrag für *crontab* machen. Nehmen wir an, daß Sie einmal täglich alte Dateien aus dem Verzeichnis */tmp* löschen wollen, das als temporärer Speicherplatz für Dateien dient, die von allen möglichen Programmen erzeugt werden.

Die meisten Systeme leeren */tmp* beim Booten, aber wenn Ihr System für längere Zeit ununterbrochen läuft, ist es vielleicht sinnvoll, mittels *cron* die alten Dateien im Auge zu behalten (zum Beispiel Dateien, auf die drei Tage lang nicht zugegriffen wurde). Der Befehl, den Sie dazu aufrufen, ist:

```
ls -l dateiname
```

Woher wissen Sie aber, welchen `dateinamen` Sie angeben sollen? Sie müssen den Befehl innerhalb eines Suchbefehls (*find*) aufrufen, der alle Dateien in einem Verzeichnis auflistet und mit diesen Dateien die gewünschte Operation durchführt.

Wir sind dem Befehl *find* bereits im Abschnitt »Inkrementelle Backups« begegnet. In diesem Beispiel geben wir */tmp* als das zu durchsuchende Verzeichnis an und benutzen die Option *-atime*, um alle Dateien zu finden, auf die zuletzt vor mehr als drei Tagen zugegriffen wurde. Mit der Option *-exec* bewirken wir, daß der folgende Befehl mit allen gefundenen Dateien ausgeführt wird:

```
find /tmp \! -type d -atime +3 -exec ls -l {} \;
```

Wir haben bereits gesehen, daß der eigentliche Befehl, den *find* für uns ausführen soll, *ls -l* ist. (Viele Leute benutzen einen ähnlichen Eintrag in *crontab*, der die Dateien entfernt, aber das ist schwierig zu bewerkstelligen, ohne ein Sicherheitsloch zu produzieren.) Die geschweiften Klammern {} bedeuten: »Führe den Befehl mit jeder Datei aus, die der Suchlauf findet.« Der String `\;` zeigt *find*, daß hier die Option *-exec* beendet ist.

Jetzt haben wir also den Befehl, der alte Dateien in */tmp* anzeigt. Wir müssen aber noch festlegen, wie oft das geschehen soll. *crontab* benutzt dazu folgende sechs Felder:

```
Minute   Stunde   Tag   Monat   Wochentag   Befehl
```

Und so füllen Sie die Felder aus:

1. Minute (geben Sie einen Wert zwischen 0 und 59 an)
2. Stunde (geben Sie einen Wert zwischen 0 und 23 an)
3. Tag des Monats (geben Sie einen Wert zwischen 1 und 31 an)
4. Monat (geben Sie einen Wert zwischen 1 und 12 oder einen Namen wie jan, feb usw. an)
5. Wochentag (geben Sie einen Wert zwischen 0 und 6 an, wobei 0 der Sonntag ist, oder einen Namen wie mon, tue usw.)
6. Befehl (kann aus mehreren Wörtern bestehen)

Abbildung 8-1 zeigt einen *cron*-Eintrag, in dem alle Felder ausgefüllt sind. Der Befehl ist ein Shell-Skript, das unter der Bourne-Shell *sh* aufgerufen wird. Allerdings ist dies kein allzu realistischer Eintrag - das Skript wird nur ausgeführt, wenn alle Bedingungen in den ersten fünf Feldern erfüllt sind. Es würde also nur an einem Sonntag aufgerufen, der auf den 15. im Januar oder Juli fällt - das kommt wohl nicht sehr oft vor! Dies ist also kein besonders praktisches Beispiel.

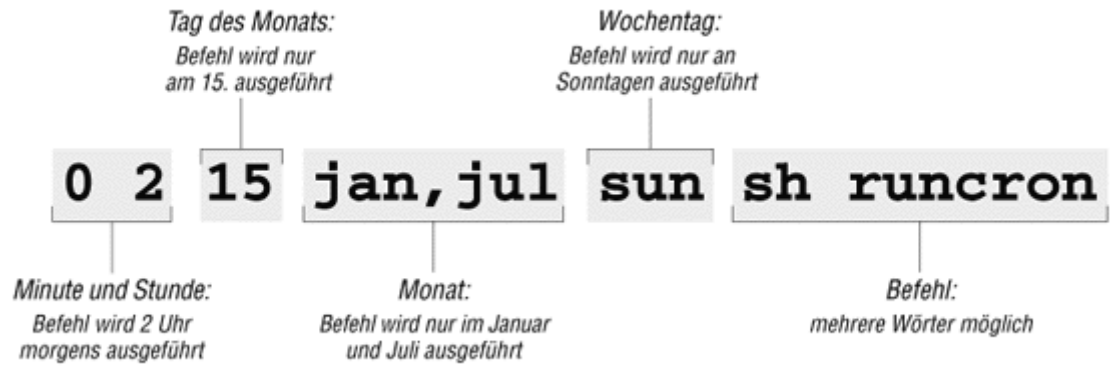

Abbildung 8-1: Beispieleintrag in crontab

Wenn Sie einen Befehl jeden Tag um ein Uhr morgens aufrufen wollen, geben Sie 0 Minuten und die Stunde 1 an. Die anderen drei Felder sollten nur einen Stern enthalten, was dann bedeutet: »jeden Tag und jeden Monat zur angegebenen Zeit«. Die vollständige Zeile in *crontab* sieht dann so aus:

```
0 1 * * * find /tmp -atime 3 -exec ls -l {} \;
```

Da es eine ganze Menge netter Dinge gibt, die Sie mit den Zeitfeldern anstellen können, wollen wir noch ein bißchen mehr mit diesem Befehl herumspielen. Nehmen wir an, daß der Befehl nur am ersten Tag eines jeden Monats ausgeführt werden soll. Sie würden dann die ersten beiden Felder beibehalten, aber in das dritte Feld eine 1 schreiben:

```
0 1 1 * * find /tmp -atime 3 -exec ls -l {} \;
```

Wenn Sie's einmal in der Woche am Montag tun wollen, schreiben Sie wieder einen Stern in das dritte Feld und geben im fünften Feld entweder eine 1 oder mon ein:

```
0 1 * * mon find /tmp -atime 3 -exec ls -l {} \;
```

Wir können die Sache noch anspruchsvoller gestalten, weil in jedem Feld mehrere Zeiten stehen können. Ein Komma an dieser Stelle bedeutet: »Führe den Befehl am 1. und 15. jeden Monats aus.«

```
0 1 1,15 * * find /tmp -atime 3 -exec ls -l {} \;
```

Während ein Bindestrich besagt: »Führe den Befehl jeden Tag vom 1. bis zum 15. einschließlich aus.«

```
0 1 1-15 * * find /tmp -atime 3 -exec ls -l {} \;
```

Und ein Schrägstrich mit einer 5 bedeutet: »Führe den Befehl an jedem 5. Tag aus«, also am 1., 6., 11. usw.

```
0 1 */5 * * find /tmp -atime 3 -exec ls -l {} \;
```

Jetzt sind wir soweit, daß wir tatsächlich einen Eintrag in *crontab* machen können. Loggen Sie sich als **root** ein (weil dies eine typische Aufgabe für **root** ist), und rufen Sie den Befehl *crontab* mit der Option *-e* auf; *-e* steht für »editieren«:

```
rutabaga# crontab -e
```

Per Voreinstellung wird mit diesem Befehl der Editor *vi* aufgerufen. Wenn Sie statt dessen lieber Emacs benutzen möchten, können Sie das vor der Anpassung von *crontab* einstellen. Bei einer Bourne-kompatiblen Shell geben Sie dazu ein:

```
rutabaga# export VISUAL=emacs
```

Für die C-Shell lautet der entsprechende Befehl:

```
rutabaga# setenv VISUAL emacs
```

Die Umgebungsvariable `EDITOR` kann in einigen Versionen von *crontab* statt der Variable `VISUAL` benutzt werden. Geben Sie zunächst eine oder zwei Kommentarzeilen ein, die mit einem Doppelkreuz (#) beginnen, und dann den *crontab*-Eintrag:

```
# List files on /tmp that are 3 or more days old.  Runs at 1:00 AM
# each morning.
0 1 * * * find /tmp -atime 3 -exec ls -l {} \;
```

Beim Verlassen von *vi* werden die Befehle gespeichert. Mit folgendem Befehl lassen Sie sich Ihre *crontab*-Einträge anzeigen:

```
rutabaga# crontab -l
```

Bei der Benutzung von *cron* sollten Sie auch überlegen, was mit den Meldungen geschieht, die von den Befehlen erzeugt werden. *cron* hebt alle Meldungen aus der Standardausgabe und der Standardfehlerausgabe auf und verschickt sie als Mail an den Benutzer. Im obigen Beispiel geht die Mail an **root**, aber Sie sollten das automatisch an sich selbst als Systemverwalter umlenken lassen. Tragen Sie auf jeden Fall diese Zeile in */usr/lib/aliases* ein (*/etc/aliases* auf Debian-Systemen):

```
root: ihr-account-name
```

Wir zeigen Ihnen gleich, was Sie machen müssen, wenn Sie die Ausgaben lieber in einer Datei als in Form einer Mail haben möchten.

Wir zeigen Ihnen ein weiteres Beispiel für einen Befehl, der häufig in *crontab*-Dateien steht, um ein Verzeichnis auf ein Magnetband zu sichern. Wir gehen davon aus, daß jemand ein Magnetband eingelegt hat, bevor der Befehl ausgeführt wird. Zunächst sorgt der Befehl *mt* dafür, daß das Band im Gerät */dev/rft0* an den Anfang zurückgespult wird. Anschließend überträgt ein *tar*-Befehl alle Dateien aus dem Verzeichnis */src* auf das Band. Die Befehle werden durch ein Semikolon getrennt, wie es die Shell-Syntax verlangt.

```
# back up the /src directory once every two months.

0 2 1 */2 * mt -f /dev/rft0 rewind; tar cf /dev/rft0 /src
```

Die ersten beiden Felder bewirken, daß der Befehl um zwei Uhr morgens aufgerufen wird; das dritte Feld bestimmt den ersten Tag des Monats. Das vierte Feld legt jeden zweiten Monat fest. Dasselbe Ergebnis würden wir mit folgendem Befehl erzielen, der vielleicht einfacher zu lesen ist:

```
0 2 1 jan,mar,may,jul,sep,nov * mt -f /dev/rft0 rewind; \
          tar cf /dev/rft0 /src
```

Im Abschnitt »Backups erstellen« erklären wir, wie Backups regelmäßig durchgeführt werden können.

Das folgende Beispiel ruft *mailq* jeden zweiten Tag auf, um zu testen, ob in der Mail-Warteschlange irgendwelche Nachrichten hängengeblieben sind, und verschickt das Ergebnis des Tests als Mail an den Mail-Verwalter. Falls Nachrichten in der Warteschlange

zurückgeblieben sind, enthält die Benachrichtigung Details zu Adressierungs- und Zustellproblemen, ansonsten ist die Benachrichtigung leer:

```
0 6 */2 * * mailq -v | \
          mail -s "Tested Mail Queue for Stuck Email" postmaster
```

Wahrscheinlich möchten Sie nicht jeden Tag eine Nachricht erhalten, solange alles problemlos läuft. In den Beispielen, die wir bisher gezeigt haben, erzeugen die Befehle nur dann Meldungen, wenn Fehler auftreten. Vielleicht möchten Sie es aber auch zur Regel machen, die Standardausgabe nach */dev/null* umzulenken oder sie folgendermaßen in eine Datei zu schreiben (beachten Sie die zwei >-Zeichen, damit der alte Inhalt nicht überschrieben wird):

```
0 1 * * * find /tmp -atime 3 -exec ls -l {} \; >> /home/mdw/log
```

Mit diesem Eintrag lenken wir die Standardausgabe um, aber die Standardfehlerausgabe kann als Mail verschickt werden. Dies kann ganz praktisch sein, weil wir eine Nachricht erhalten, wenn etwas schiefgeht. Wenn Sie sicherstellen möchten, daß Sie auf keinen Fall Mail erhalten, lenken Sie sowohl die Standardausgabe als auch die Standardfehlerausgabe um:

```
0 1 * * * find /tmp -atime 3 -exec ls -l {} \; >> /home/mdw/log 2>&1
```

Falls Sie die Meldungen in eine Logdatei schreiben, sehen Sie sich mit einer ständig größer werdenden Datei konfrontiert. Eventuell sollten Sie einen weiteren *cron*-Eintrag erzeugen, der diese Datei beispielsweise einmal pro Woche löscht.

In den Einträgen für *crontab* lassen sich nur die Befehle der Bourne-Shell benutzen. Sie können also nicht auf die komfortablen Erweiterungen zurückgreifen, die *bash* und andere moderne Shells bieten - etwa Aliasnamen oder die Abkürzung ~ für das Home-Verzeichnis. Allerdings können Sie `$HOME` benutzen - *cron* kennt die Umgebungsvariablen `$USER`, `$HOME` und `$SHELL`. Alle Befehle benutzen Ihr Home-Verzeichnis als das aktuelle Verzeichnis.

Einige Leute ziehen es vor, in *crontab*-Einträgen immer die absoluten Pfadnamen der Befehle anzugeben, wie zum Beispiel */usr/bin/find* und */bin/rm*. Damit stellen sie sicher, daß immer der richtige Befehl gefunden wird, statt sich darauf zu verlassen, daß die Variable `PATH` richtig gesetzt ist.

Wenn ein Befehl so lang und kompliziert wird, daß er nicht mehr in eine Zeile paßt, sollten Sie ein Shell-Skript schreiben und dieses von *cron* aufrufen lassen. Sorgen Sie dafür, daß das Skript ausführbar ist (mit *chmod +x*), oder starten Sie für den Aufruf eine Shell:

```
0 1 * * * sh runcron
```

Als Systemverwalter werden Sie häufig *crontab*-Dateien für fiktive Benutzer wie **news** oder **uucp** anlegen. Es wäre unnötig und potentiell gefährlich, alle Utilities als **root** ausführen zu lassen, deshalb gibt es diese speziellen Benutzer.

Die Wahl des Benutzers hat auch Einfluß auf die Dateieignerschaft: Eine *crontab*-Datei für **news** sollte Dateien ausführen, die **news** gehören usw. Sie sollten ganz allgemein sicherstellen, daß die Utilities dem Benutzer gehören, in dessen Namen Sie die *crontab*-Datei erstellen.

Als root können Sie die *crontab*-Dateien anderer Benutzer mit

```
tigger# crontab -u benutzername -e
```

editieren.

Kapitel 4

Bedenken Sie auch, wer die entstehenden Dateien mit den Systemmeldungen nutzen soll. Wenn ein *cron*-Eintrag, der unter **news** aufgerufen wird, eine Datei erzeugt, werden Sie eventuell Schwierigkeiten haben, diese Datei später unter einer anderen Benutzerkennung zu lesen. Eventuell müssen Sie in Ihrem *cron*-Skript *chown* oder *chmod* benutzen, um die Datei hinterher auswerten zu können. Im Abschnitt »Dateiberechtigungen« in Kapitel 4, *Grundlegende Unix-Befehle und -Konzepte,* besprechen wir diese Befehle.

Da Sie sich nicht als **news** einloggen können, müssen Sie die *crontab*-Datei von **news** als **root** mit dem Befehl

```
rutabaga# crontab -e -u news
```

editieren.