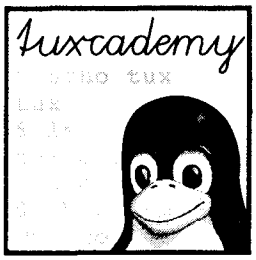

9

Zeitgesteuerte Vorgänge – at und cron

Inhalt

Lernziele

- Kommandos mit at zu einem zukünftigen Zeitpunkt ausführen können
- Kommandos periodisch mit cron ausführen können
- anacron kennen und einsetzen können

Vorkenntnisse

- Umgang mit Linux-Kommandos
- Umgang mit einem Texteditor

9.1 Allgemeines

Ein wichtiger Bestandteil der Systemverwaltung besteht darin, wiederholt ablaufende Vorgänge zu automatisieren. Eine denkbare Aufgabe wäre, dass sich der Mailserver eines Firmennetzwerkes in regelmäßigen Abständen beim Internet-Provider einwählt und die eingegangenen Nachrichten abholt. Ferner könnten alle Mitglieder einer Projektgruppe eine halbe Stunde vor der wöchentlichen Besprechung eine schriftliche Erinnerung erhalten. Auch Administrationsaufgaben wie Dateisystemtests oder Datenarchivierung lassen sich automatisch in der Nacht ausführen, da zu dieser Zeit die Systembelastung naturgemäß deutlich geringer ist.

9.2 Einmalige Ausführung von Kommandos

9.2.1 at und batch

Mit Hilfe des Kommandos at lassen sich beliebige Shell-Befehle zu einem späteren Zeitpunkt, also zeitversetzt, einmalig ausführen. Wenn Kommandos hingegen in regelmäßigen Intervallen wiederholt ausgeführt werden sollen, ist die Verwendung von cron (Abschnitt 9.3) vorzuziehen.

Die Idee hinter at ist, einen Zeitpunkt vorzugeben, zu dem dann ein Kommando oder eine Folge von Kommandos ausgeführt wird. Etwa so:

```
$ at 01:00
warning: commands will be executed using /bin/sh
at> tar cvzf /dev/st0 $HOME
at> echo " Backup fertig" | mail -s Backup $USER
at> Strg+d
Job 123 at 2006-11-08 01:00
```

Hiermit würden Sie um 1 Uhr nachts eine Sicherheitskopie Ihres Heimatverzeichnisses auf Band schreiben (Band einlegen nicht vergessen!) und sich anschließend per Mail eine Vollzugsmeldung schicken lassen.

Zeitangabe

Das Argument von at ist eine Zeitangabe für die Ausführung der Kommandos. Zeiten im Format »⟨*HH*⟩:⟨*MM*⟩« bezeichnen den nächstmöglichen solchen Zeitpunkt: Wird das Kommando »at 14:00« um 8 Uhr früh angegeben, bezieht es sich auf denselben Tag; wird es um 16 Uhr angegeben, auf den Folgetag.

Sie können die Zeitpunkte durch Anhängen von today und tomorrow eindeutig machen: »at 14:00 today«, vor 14 Uhr gegeben, bezieht sich auf heute, »at 14:00 tomorrow« auf morgen.

Möglich sind auch angelsächsische Zeitangaben wie 01:00am oder 02:20pm sowie die symbolischen Namen midnight (0 bzw. 24 Uhr), noon (12 Uhr) und teatime (16 Uhr) (!); der ebenfalls erlaubte Zeitpunkt now ist vor allem in Verbindung mit relativen Zeitangaben (siehe unten) sinnvoll.

Datumsangabe

Neben Uhrzeiten versteht at auch Datumsangaben in der Form »⟨*MM*⟩⟨*TT*⟩⟨*JJ*⟩« und »⟨*MM*⟩/⟨*TT*⟩/⟨*JJ*⟩« (also nach amerikanischem Brauch, mit dem Monat vor dem Tag) sowie »⟨*TT*⟩.⟨*MM*⟩.⟨*JJ*⟩« (für uns Europäer). Daneben sind ausgeschriebene amerikanische Datumsangaben der Form »⟨*Monatsname*⟩ ⟨*Tag*⟩« und »⟨*Monatsname*⟩ ⟨*Tag*⟩ ⟨*Jahr*⟩« erlaubt. Wenn Sie nur ein Datum angeben, werden die Kommandos zur aktuellen Zeit am betreffenden Tag ausgeführt; Sie können auch Zeit- und Datumsangabe kombinieren, müssen dann aber das Datum nach der Zeit anführen:

```
$ at 11:11 November 11
warning: commands will be executed using /bin/sh
at> echo 'Helau!'
```

```
at> Strg+D
Job 124 at 2003-11-11 11:11
```

Außer »expliziten« Zeit- und Datumsangaben können Sie auch »relative« Angaben machen, indem Sie eine Differenz zu einem gegebenen Zeitpunkt bestimmen: »relative« Angaben

```
$ at now + 5 minutes
```

führt die Kommandos in fünf Minuten aus, während

```
$ at noon + 2 days
```

sich auf 12 Uhr am übermorgigen Tag bezieht (jedenfalls solange das at-Kommando vor 12 Uhr angegeben wurde). at unterstützt die Zeiteinheiten minutes, hours, days und weeks.

Eine einzige Verschiebung um eine einzige Zeiteinheit muss reichen: Kombinationen wie

```
$ at noon + 2 hours 30 minutes
```

oder

```
$ at noon + 2 hours + 30 minutes
```

sind leider nicht erlaubt. Natürlich können Sie jede beliebige Verschiebung in Minuten ausdrücken ...

at liest die Kommandos von der Standardeingabe, also normalerweise der Tastatur; mit der Option »-f ⟨*Datei*⟩« können Sie stattdessen eine Datei angeben. Kommandos

at versucht, die Kommandos in einer Umgebung auszuführen, die der zum Zeitpunkt des at-Aufrufs möglichst ähnelt. Das aktuelle Arbeitsverzeichnis, die *umask* und die aktuellen Umgebungsvariablen (außer TERM, DISPLAY und _) werden gesichert und vor der Kommandoausführung wieder aktiviert.

Eine etwaige Ausgabe der per at gestarteten Kommandos – Standardausgabe- und Standardfehlerausgabekanal – bekommen Sie per Mail geschickt. Ausgabe

Wenn Sie vor dem Aufruf von at mit su eine andere Identität angenommen haben, werden die Kommandos mit dieser Identität ausgeführt. Die Ausgabemails gehen aber trotzdem an Sie.

Während Sie mit at Kommandos zu einem bestimmten Zeitpunkt ausführen können, macht der (ansonsten analog funktionierende) Befehl batch es möglich, eine Folge von Kommandos zum »nächstmöglichen Zeitpunkt« auszuführen. Wann das passiert, hängt von der aktuellen Systemlast ab; hat das System gerade viel zu tun, müssen batch-Jobs warten. Ausführung zum »nächstmöglichen Zeitpunkt«

Eine at-artige Zeitangabe ist bei batch erlaubt, aber nicht vorgeschrieben. Wird sie angegeben, so werden die Kommandos »irgendwann nach« dem genannten Zeitpunkt ausgeführt, so als wären sie gerade dann per batch eingereicht worden.

batch ist nicht geeignet für Umgebungen, wo die Benutzer um Ressourcen wie Rechenzeit konkurrieren. Hierfür müssen andere Systeme eingesetzt werden.

Übungen

9.1 [!1] Nehmen wir an, jetzt ist es der 1. März, 15 Uhr. Wann werden die mit den folgenden Kommandos eingereichten Aufträge ausgeführt?

1. `at 17:00`
2. `at 02:00pm`
3. `at teatime tomorrow`
4. `at now + 10 hours`

9.2 [1] Verwenden Sie das Programm `logger`, um in 3 Minuten eine Nachricht ins Systemprotokoll zu schreiben.

9.2.2 at-Hilfsprogramme

Warteschlange anschauen

Das System reiht die mit `at` registrierten Aufträge in eine Warteschlange ein. Diese können Sie mit `atq` inspizieren (Sie sehen aber nur Ihre eigenen Aufträge, es sei denn, Sie sind `root`):

```
$ atq
123     2003-11-08 01:00 a hugo
124     2003-11-11 11:11 a hugo
125     2003-11-08 21:05 a hugo
```

Das »a« in der Liste bezeichnet die »Auftragsklasse«, einen Buchstaben zwischen »a« und »z«. Sie können mit der Option `-q` für `at` eine Auftragsklasse bestimmen; Aufträge in Klassen mit »größeren« Buchstaben werden mit höheren `nice`-Werten ausgeführt. Standard ist »a« für `at`- und »b« für `batch`-Aufträge.

Ein gerade laufender Auftrag ist in der besonderen Auftragsklasse »=«.

Auftrag stornieren

Mit `atrm` können Sie einen Auftrag stornieren. Hierzu müssen Sie dessen Auftragsnummer angeben, die Sie bei der Einreichung genannt oder mit `atq` gezeigt bekommen haben. Wenn Sie nachschauen wollen, aus welchen Kommandos der Auftrag besteht, können Sie das mit »`at -c` ⟨*Auftragsnummer*⟩«.

Daemon

Zuständig für die tatsächliche Ausführung der `at`-Aufträge ist ein Daemon namens `atd`. Dieser wird üblicherweise beim Systemstart geladen und wartet im Hintergrund auf Arbeit. Beim Start von `atd` sind einige Optionen möglich:

`-b` (engl. *batch*, Stapel) Legt das minimale Intervall für den Start zweier `batch`-Aufträge fest. Voreinstellung ist hier 60 Sekunden.

`-l` (engl. *load*, Last) Legt einen Grenzwert für die Systembelastung fest, oberhalb der `batch`-Aufträge nicht ausgeführt werden. Voreinstellung ist hier ein Wert von 0,8.

`-d` (engl. *debug*, »Entwanzen«) aktiviert den Debug-Modus, d. h., alle Fehlermeldungen werden nicht per `syslogd` protokolliert, sondern auf die Standardfehlerausgabe geschrieben.

Der Daemon `atd` benötigt zu seiner Arbeit die folgenden Verzeichnisse:

- Im Verzeichnis `/var/spool/atjobs` werden die `at`-Aufträge abgelegt. Der Zugriffsmodus sollte 700 sein, der Eigentümer ist `at`.
- Das Verzeichnis `/var/spool/atspool` dient zur Zwischenspeicherung von Ausgaben. Auch hier sollten der Eigentümer `at` und der Zugriffsmodus 700 sein.

Übungen

9.3 [1] Registrieren Sie mit at einige Aufträge und lassen Sie sich eine Auftragsliste anzeigen. Entfernen Sie die Aufträge wieder.

9.4 [2] Wie würden Sie eine Liste von at-Aufträgen erzeugen, die nicht nach der Auftragsnummer, sondern nach dem vorgesehenen Ausführungszeitpunkt sortiert ist?

9.2.3 Zugangskontrolle

Die Dateien /etc/at.allow und /etc/at.deny bestimmen, wer mit at und batch Aufträge einreichen darf. Wenn die Datei /etc/at.allow existiert, sind nur die darin eingetragenen Benutzer berechtigt, Aufträge einzureichen. Gibt es die Datei /etc/at.allow nicht, aber die Datei /etc/at.deny, dann dürfen diejenigen Benutzer Aufträge einreichen, die *nicht* in der Datei stehen. Existiert weder die eine noch die andere, dann stehen at und batch nur root zur Verfügung.

/etc/at.allow
/etc/at.deny

Debian GNU/Linux liefert eine /etc/at.deny-Datei aus, in der die Namen diverser Systembenutzer stehen (etwa alias, backup, guest und www-data). Damit werden diese Benutzeridentitäten von der Verwendung von at ausgeschlossen.

Die Voreinstellung von Ubuntu entspricht auch hier der von Debian GNU/Linux.

Red Hat liefert eine leere /etc/at.deny-Datei aus; damit darf jeder Benutzer Aufträge einreichen.

Die OpenSUSE-Voreinstellung gleicht (interessanterweise) der von Debian GNU/Linux und Ubuntu – diverse Systembenutzer dürfen at nicht verwenden. (Den explizit ausgeschlossenen Benutzer www-data zum Beispiel gibt es bei OpenSUSE überhaupt nicht; Apache läuft mit der Identität von wwwrun.)

Übungen

9.5 [1] Wer darf auf Ihrem System at und batch benutzen?

9.3 Wiederholte Ausführung von Kommandos

9.3.1 Aufgabenlisten für Benutzer

Im Unterschied zu den at-Kommandos dient der Daemon cron zur automatischen Ausführung von sich regelmäßig wiederholenden Aufgaben. Gestartet werden sollte cron – wie auch der atd – beim Systemstart mit einem Init-Skript; Sie müssen sich darum normalerweise nicht kümmern, da cron und atd so wichtige Bestandteile eines Linux-Systems sind, dass keine der wichtigen Distributionen auf sie verzichtet.

Jeder Anwender hat seine eigene Aufgabenliste (vulgo crontab), die im Verzeichnis /var/spool/cron/crontabs (bei Debian GNU/Linux und Ubuntu; bei SUSE: /var/spool/cron/tabs, bei Red Hat: /var/spool/cron) unter dem jeweiligen Benutzernamen abgelegt ist. Die dort beschriebenen Kommandos werden mit den Rechten des betreffenden Anwenders ausgeführt.

Aufgabenliste

Auf Ihre Aufgabenliste im cron-Verzeichnis haben Sie keinen direkten Zugriff, sondern müssen das Programm crontab bemühen (siehe unten). Beachten Sie hierzu auch Übung 9.6.

Syntax

crontab-Dateien sind zeilenweise aufgebaut; jede Zeile beschreibt einen (wiederkehrenden) Zeitpunkt und ein Kommando, das zu dem betreffenden Termin ausgeführt werden soll. Leerzeilen und Kommentarzeilen (mit einem »#« am Anfang) werden ignoriert. Die übrigen Zeilen bestehen aus fünf Zeitfeldern und dem auszuführenden Kommando; die Zeitfelder beschreiben respektive die Minute (0–59), die Stunde (0–23), der Tag im Monat (1–31), der Monat (1–12 oder der englische Name) und den Wochentag (0–7, 0 und 7 stehen für Sonntag, oder der englische Name), zu denen das Kommando ausgeführt werden soll. Alternativ ist jeweils ein Sternchen (»*«) erlaubt, das für »egal« steht. Zum Beispiel bedeutet

Zeitfelder
Kommando

```
58 19 * * * echo "Gleich kommt die Tagesschau"
```

dass das Kommando täglich um 19.58 Uhr ausgeführt wird (Tag, Monat und Wochentag sind egal).

Das Kommando wird ausgeführt, wenn Stunde, Minute und Monat genau stimmen und *mindestens eine* der beiden Tagesangaben – Tag im Monat oder Wochentag – zutreffen. Die Spezifikation

```
1 0 13 * 5 echo "Kurz nach Mitternacht"
```

gibt also an, dass die Meldung an jedem Monats-Dreizehnten *und* an jedem Freitag ausgegeben wird, nicht nur am Freitag, dem 13.

Die letzte Zeile in einer crontab-Datei *muss* mit einem Zeilentrenner aufhören, sonst wird sie ignoriert.

Listen

cron akzeptiert in den Zeitfeldern nicht nur einzelne Zahlen, sondern erlaubt auch kommaseparierte Listen. Die Angabe »0,30« im Minutenfeld würde also dazu führen, dass das Kommando zu jeder »vollen halben« Stunde ausgeführt wird. Außerdem sind Bereiche erlaubt: »8-11« ist äquivalent zu »8,9,10,11«, »8-10,14-16« entspricht »8,9,10,14,15,16«. Ebenfalls gestattet ist die Angabe einer »Schrittweite« in Bereichen. Die Spezifikation »0-59/10« im Minutenfeld ist gleichbedeutend mit »0,10,20,30,40,50«. Wird – wie hier – der komplette Wertebereich abgedeckt, so könnten Sie auch »*/10« schreiben.

Schrittweite

Monats- und Wochentagsangaben

Die bei Monats- und Wochentagsangaben erlaubten Namen bestehen jeweils aus den ersten drei Buchstaben des englischen Monats- oder Wochentagsnamen (also zum Beispiel may, oct, sun oder wed). Bereiche und Listen von Namen sind nicht erlaubt.

Kommando

Der Rest der Zeile bestimmt das auszuführende Kommando, das von cron an /bin/sh (bzw. die in der Variablen SHELL benannte Shell, siehe unten) übergeben wird.

Prozentzeichen (%) im Kommando müssen mit einem Rückstrich versteckt werden (also »\%«), sonst werden sie in Zeilentrenner umgewandelt. Dann gilt das Kommando als am ersten Prozentzeichen beendet; die folgenden »Zeilen« werden dem Kommando auf der Standardeingabe verfüttert.

Übrigens: Wenn Sie als Systemadministrator möchten, dass bestimmte Kommandos nicht, wie sonst bei cron üblich, bei der Ausführung per syslogd protokolliert werden, können Sie dies durch die Angabe eines »-« als erstes Zeichen der Zeile unterdrücken.

Zuweisungen an Umgebungsvariablen

Außer Kommandos mit Wiederholungsangaben können die crontab-Zeilen auch Zuweisungen an Umgebungsvariablen enthalten. Diese haben die Form »⟨*Variablenname*⟩=⟨*Wert*⟩« (wobei im Gegensatz zur Shell vor und nach dem »=« auch Leerplatz stehen darf). Enthält der ⟨*Wert*⟩ Leerzeichen, sollte er mit Anführungszeichen eingefasst werden. Die folgenden Variablen werden automatisch voreingestellt:

SHELL Mit dieser Shell werden die eingegebenen Befehle ausgeführt. Voreingestellt ist dabei `/bin/sh`, aber auch die Angabe anderer Shells ist erlaubt.

LOGNAME Der Benutzername wird aus `/etc/passwd` übernommen und kann nicht geändert werden.

HOME Das Heimatverzeichnis wird ebenfalls aus `/etc/passwd` übernommen. Hier ist jedoch eine Änderung des Variablenwerts zulässig.

MAILTO An diese Adresse schickt `cron` Nachrichten mit der Ausgabe des aufgerufenen Kommandos (sonst gehen sie an den Eigentümer der `crontab`-Datei). Soll `cron` überhaupt keine Nachrichten verschicken, muss die Variable leer gesetzt sein, d. h. `MAILTO=""`.

9.3.2 Systemweite Aufgabenlisten

Neben den benutzerbezogenen Dateien existiert noch eine systemweite Aufgabenliste. Diese findet sich in `/etc/crontab` und gehört dem Systemadministrator `root`, der sie als einziger ändern darf. Die Syntax von `/etc/crontab` ist geringfügig anders als die der benutzereigenen `crontab`-Dateien; zwischen den Zeitangaben und dem auszuführenden Kommando steht hier noch der Name des Benutzers, mit dessen Rechten das Kommando ausgeführt werden soll.

`/etc/crontab`

Diverse Linux-Distributionen haben ein `/etc/cron.d`-Verzeichnis; in diesem Verzeichnis können Dateien stehen, die als »Erweiterungen« von `/etc/crontab` interpretiert werden. Per Paketmanagement installierte Softwarepakete können so leichter den `cron`-Dienst benutzen, als wenn sie Zeilen zur `/etc/crontab` hinzufügen müßten.

Populär sind auch Verzeichnisse `/etc/cron.hourly`, `/etc/cron.daily` und so weiter. In diesen Verzeichnissen können Softwarepakete (oder der Systemadministrator) Dateien ablegen, deren Inhalt dann stündlich, täglich, ... ausgeführt wird. Diese Dateien sind keine `crontab`-Dateien, sondern »normale« Shellskripte.

`cron` liest die Aufgabenlisten – aus benutzereigenen `crontab`-Dateien, der systemweiten `/etc/crontab` und den Dateien in `/etc/cron.d`, sofern vorhanden – nur einmal beim Start ein und behält sie danach im Speicher. Das Programm schaut allerdings jede Minute nach, ob die `crontab`-Dateien geändert wurden. Zu diesem Zweck wird einfach die *mtime*, der Zeitpunkt der letzten Änderung, herangezogen. Wenn `cron` hier eine Veränderung bemerkt, wird die Aufgabenliste automatisch neu aufgebaut. In diesem Fall ist also kein expliziter Neustart des Daemons erforderlich.

`crontab`-Änderungen und `cron`

Übungen

9.6 [2] Wieso können Sie als Benutzer nicht auf Ihre Aufgabenliste in `/var/spool/cron/crontabs` (oder dem Äquivalent für Ihre Distribution) zugreifen? Wie wird arrangiert, dass `crontab` das kann?

9.7 [2] Wie können Sie dafür sorgen, dass eine Aufgabe nur am Freitag, dem 13., ausgeführt wird?

9.8 [3] Wie stellt das System sicher, dass die Aufgaben in `/etc/cron.hourly`, `/etc/cron.daily`, ... tatsächlich einmal in der Stunde, einmal am Tag usw. ausgeführt werden?

9.3.3 Zugangskontrolle

Welche Anwender überhaupt mit cron arbeiten dürfen, ist ähnlich wie bei at in zwei Dateien festgelegt. In /etc/cron.allow (gelegentlich /var/spool/cron/allow) sind die Benutzer aufgeführt, die cron verwenden dürfen. Wenn die Datei nicht existiert, aber es dafür die Datei /etc/cron.deny (manchmal /var/spool/cron/deny) gibt, sind in letzterer diejenigen Benutzer aufgelistet, die *nicht* in den Genuss der automatisierten Befehlsausführung kommen dürfen. Sind beide Dateien nicht vorhanden, hängt es von der jeweiligen Konfiguration ab, ob nur root die Dienste von cron beanspruchen darf oder sozusagen »gleiches Recht für alle« herrscht und cron jedem Benutzer zur Verfügung steht.

9.3.4 Das Kommando crontab

Die einzelnen Benutzer können ihre crontab-Datei nicht von Hand ändern, da das System sie vor ihnen versteckt. Lediglich die systemweite Aufgabenliste /etc/crontab ist ein Fall für den Lieblingseditor von root.

Aufgabenlisten verwalten

Statt einen Editor direkt aufzurufen, sollten alle Benutzer das Kommando crontab verwenden. Hiermit können Aufgabenlisten erstellt, eingesehen, verändert und auch wieder entfernt werden. Mit

```
$ crontab -e
```

können Sie Ihre crontab-Datei mit dem Editor bearbeiten, dessen Name in den Umgebungsvariablen VISUAL bzw. EDITOR festgelegt ist – ersatzweise dem Editor vi. Nach dem Verlassen des Editors wird die modifizierte crontab-Datei automatisch installiert. Statt der Option -e können Sie auch den Namen einer Datei angeben, deren Inhalt dann als Aufgabenliste installiert wird. Der Dateiname »-« steht hierbei stellvertretend für die Standardeingabe.

Mit der Option -l gibt crontab Ihre crontab-Datei auf der Standardausgabe aus; mit der Option -r wird eine installierte Aufgabenliste ersatzlos gelöscht.

Mit der Option »-u ⟨*Benutzername*⟩« können Sie sich auf einen anderen Benutzer beziehen (wofür Sie in der Regel root sein müssen). Dies ist vor allem wichtig, wenn Sie su benutzen; in diesem Fall sollten Sie immer mit der -u-Option operieren, um sicherzustellen, dass Sie die richtige crontab-Datei erwischen.

Übungen

9.9 [1] Registrieren Sie mit dem Programm crontab eine Aufgabe, die jede Minute das aktuelle Datum an die Datei /tmp/date.log anhängt. Wie können Sie bequem erreichen, dass dasselbe alle zwei Minuten passiert?

9.10 [1] Verwenden Sie crontab, um den Inhalt Ihrer Aufgabenliste auf der Standardausgabe auszugeben und anschließend wieder zu löschen.

9.11 [2] (Für Administratoren.) Sorgen Sie dafür, dass der Benutzer hugo den cron-Dienst *nicht* verwenden darf. Vergewissern Sie sich, dass Ihre Maßnahme funktioniert.

9.3.5 Anacron

Mit cron können Sie Kommandos wiederholt zu vorgegebenen Zeitpunkten ausführen. Offensichtlich funktioniert das aber nur, wenn der Computer zum betreffenden Zeitpunkt eingeschaltet ist – es hat keinen großen Zweck, auf einem Arbeitsplatzrechner einen cron-Job für 2 Uhr morgens zu konfigurieren, wenn der Rechner außerhalb der üblichen Bürozeiten aus Stromspargründen ausgeschaltet ist. Auch mobile Rechner sind oft zu ungewöhnlichen Zeiten ein- oder ausgeschaltet, so dass es schwierig ist, die periodischen automatischen Aufräumungsarbeiten einzuplanen, die für ein Linux-System nötig sind.

Das Programm anacron (von Itai Tzur, aktuell gewartet von Pascal Hakim) kann ähnlich wie cron Jobs ausführen, die im täglichen, wöchentlichen oder monatlichen Rhythmus laufen sollen. (Tatsächlich gehen beliebige Abstände von *n* Tagen.) Dabei ist es nur wichtig, dass der Rechner am betreffenden Tag lange genug eingeschaltet ist, dass die Jobs ausgeführt werden können – es ist egal, wann am Tag. Allerdings wird anacron maximal einmal am Tag aktiv; wenn Sie eine höhere Frequenz brauchen (Stunden oder Minuten), führt an cron kein Weg vorbei.

Im Gegensatz zu cron ist anacron relativ primitiv, was die Verwaltung von Jobs angeht. Mit cron kann potentiell jeder Benutzer Jobs anlegen; bei anacron ist das das Privileg des Systemverwalters root.

Die Jobs für anacron werden in der Datei /etc/anacrontab festgelegt. Neben den üblichen Kommentar- und Leerzeilen (die ignoriert werden) enthält sie Zuweisungen an Umgebungsvariablen der Form

```
SHELL=/bin/sh
```

und Job-Beschreibungen der Form

```
7 10 weekly run-parts /etc/cron.weekly
```

Dabei steht die erste Zahl (hier 7) für den Zeitabstand (in Tagen), in dem einzelne Aufrufe des Jobs stattfinden sollen. Die zweite Zahl (10) gibt an, wieviele Minuten nach dem Start von anacron der Job gestartet werden soll. Als Nächstes kommen ein Name für den Job (hier weekly) und schließlich das auszuführende Kommando. Überlange Zeilen können mit einem »\« am Zeilenende umbrochen werden.

Der Jobname darf alle Zeichen enthalten außer Freiplatz und dem Schrägstrich. Er wird verwendet, um den Job in Protokollnachrichten zu identifizieren, und anacron benutzt ihn auch als Namen für die Datei, mit der er den Zeitstempel der letzten Ausführung protokolliert. (Diese Dateien stehen normalerweise in /var/spool/anacron.)

Wenn anacron gestartet wird, liest es /etc/anacrontab und prüft für jeden Job, ob er während der letzten *t* Tage ausgeführt wurde, wobei *t* der Zeitabstand aus der Jobdefinition ist. Wenn nein, dann wartet anacron die in der Jobdefinition angegebene Anzahl von Minuten ab und startet das Shell-Kommando.

Sie können auf der Kommandozeile von anacron einen Jobnamen angeben, um (gegebenenfalls) nur diesen Job auszuführen. Alternativ sind auf der Kommandozeile auch Shell-Suchmuster erlaubt, damit Sie Gruppen von (geschickt vergebenen) Jobnamen mit einem anacron-Aufruf ausführen können. Beim Aufruf von anacron gar keinen Jobnamen anzugeben ist dasselbe wie der Jobname »*«.

Den Zeitabstand zwischen Ausführungen eines Jobs können Sie auch symbolisch angeben; gültige Werte sind @daily, @weekly, @monthly, @yearly und @annually (die beiden letzten sind äquivalent).

In der Definition einer Umgebungsvariablen werden Leerzeichen links vom »=« ignoriert. Rechts vom »=« sind sie Teil des Wertes der Variablen. Definitionen gelten bis zum Dateiende oder bis zu einer neuen Definition derselben Variablen.

Einige »Umgebungsvariable« haben eine Sonderbedeutung für anacron. Mit RANDOM_DELAY können Sie eine zusätzliche zufällige Verzögerung für die Jobstarts festlegen: Wenn Sie die Variable auf eine Zahl *t* setzen, dann wird eine zufällige Anzahl von Minuten zwischen 0 und *t* zur in der Job-Beschreibung angegebenen Verzögerung addiert. Mit START_HOURS_RANGE können Sie einen Bereich von Stunden (auf der Uhr) angeben, während dem die Jobs gestartet werden sollen. Etwas wie

```
START_HOURS_RANGE=10-12
```

erlaubt Job-Starts nur zwischen 10 und 12 Uhr. Wie cron schickt anacron die Ausgabe von Jobs an die Adresse, die mit der Variablen MAILTO angegeben wurde, ersatzweise den Benutzer, der anacron ausführt (in der Regel root).

Normalerweise führt anacron die Jobs unabhängig voneinander und ohne Rücksicht auf Überlappungen aus. Mit der Option -s werden die Jobs »seriell« ausgeführt, das heißt, anacron startet einen neuen Job erst, wenn der vorige fertig ist.

Im Gegensatz zu cron ist anacron kein Hintergrunddienst, sondern wird beim Systemstart angestoßen, um allfällige liegengebliebene Jobs auszuführen (die Verzögerung in Minuten dient dazu, die Jobs zu verschieben, bis das System vernünftig läuft, um den Systemstart nicht noch weiter zu verlangsamen). Später können Sie anacron einmal am Tag mit cron ausführen, um dafür zu sorgen, das es auch funktioniert, wenn das System einmal unvorhergesehen lange läuft.

Sie können durchaus cron und anacron auf demselben System installiert haben. Während anacron normalerweise die eigentlich für cron gedachten Jobs in /etc/cron.daily, /etc/cron.weekly und /etc/cron.monthly ausführt, wird dafür gesorgt, dass anacron nichts macht, wenn cron aktiv ist. (Siehe hierzu Übung 9.13.)

Übungen

9.12 [!2] Überzeugen Sie sich, dass anacron so funktioniert wie behauptet. (*Tipp:* Wenn Sie nicht tagelang warten wollen, dann manipulieren Sie dazu kreativ die Zeitstempel-Dateien in /var/spool/anacron.)

9.13 [2] Wie vermeidet man, dass auf einem lang laufenden System, das sowohl cron und anacron installiert hat, anacron dem regulären cron in die Quere kommt? (*Tipp:* Untersuchen Sie den Inhalt von /etc/cron.daily & Co.)

Kommandos in diesem Kapitel

anacron	Führt periodische Jobs aus, wenn der Computer nicht immer läuft	anacron(8)	148
at	Registriert Kommandos zur zeitversetzten Ausführung	at(1)	142
atd	Daemon für die zeitversetzte Ausführung von Kommandos über at	atd(8)	144
atq	Programm zur Abfrage der Warteschlangen für zeitversetzte Kommandoausführung	atq(1)	144
atrm	Storniert zeitversetzt auszuführende Kommandos	atrm(1)	144
crontab	Programm zur Verwaltung von regelmäßig auszuführenden Kommandos	crontab(1)	148

Zusammenfassung

- Mit `at` kann man Kommandos registrieren, die zu einem festen späteren Zeitpunkt ausgeführt werden.
- Das Kommando `batch` erlaubt die Ausführung von Kommandos, wenn das System dafür Kapazität frei hat.
- `atq` und `atrm` dienen zur Verwaltung der Auftragswarteschlangen. Der Daemon `atd` kümmert sich um die tatsächliche Ausführung der Aufträge.
- Der Zugang zu `at` und `batch` wird über die Dateien `/etc/at.allow` und `/etc/at.deny` gesteuert.
- Der `cron`-Daemon dient zur periodischen Wiederholung von Kommandos.
- Benutzer können eigene Aufgabenlisten (`crontabs`) haben.
- Eine systemweite Aufgabenliste existiert in `/etc/crontab` und – bei vielen Distributionen – im Verzeichnis `/etc/cron.d`.
- Der Zugriff auf `cron` wird über die Dateien `/etc/cron.allow` und `/etc/cron.deny` ähnlich geregelt wie der Zugriff auf `at`.
- Das Kommando `crontab` dient zur Verwaltung von `crontab`-Dateien.