at – ein (fast) vergessenes Helferlein

von Markus Schnalke

Den Cron-Daemon kennt und verwendet fast jeder Unix-Benutzer, sein Gegenstück *at* dagegen kennen die meisten nur vom Namen. Die Funktion die *at* anbietet, kann jedoch im Alltag deutlich häufiger eingesetzt werden, als man es erst mal erwartet.

Dies ist eine kurze Einführung in *at* mit Beispielen, in denen vorgestellt wird, was man damit machen kann.

Was ist at?

Es wird kaum jemanden geben, der *at* erklärt, ohne nicht davor cron erklärt zu haben. Hier wird ebenso vorgegangen, denn die beiden Programme sind ein Paar, das sich ergänzt.

Cron ist ein Daemon, der Programme regelmäßig startet; das kann stündlich, tageweise, oder auch um vier Uhr fünfzehn am ersten Tag jedes Monats sein. *at* startet auch Programme, nur eben nicht regelmäßig, sondern genau einmal zu einem bestimmten Zeitpunkt.

Auf Rechnern gibt es üblicherweise eine ganze Reihe von Systemaufgaben, die regelmäßig erledigt werden müssen. Man kann also davon ausgehen, dass ein Cron-Daemon auf quasi jedem System installiert ist und auch läuft. (Die regelmäßigen Systemaufgaben kann man sich auf Debian-Systemen mit `ls /etc/cron.*` auflisten lassen.) Der cron-Daemon kann dann auch von den Nutzern genutzt werden, indem sie sich ihre eigenen Cronjobs anlegen.

at dagegen ist kein Dienst, der vom System genutzt wird. Er ist für die Benutzer da und somit auch nicht auf jedem System von Haus aus installiert. Allerdings sind sowohl *at* als auch cron (bzw. crontab) in POSIX und der SUSv3 (Single Unix Specification Version 3, Nachfolger von POSIX) spezifiziert, somit dürften sie für jedes Unix verfügbar sein.

Struktur

Das at-System besteht aus den zwei Programmen at und atd. Auf vielen Systemen existieren noch weitere Programme, die jedoch nur Aliasse für bestimmte *at*-Aufrufe sind. In der folgenden Tabelle sind die Programme kurz vorgestellt. Nähere Informationen finden sich in der Manpage von *at* oder in der Open Group Base Specification [1].

at und atd	
atd	Ein Daemon, der im Hintergrund läuft und zum passenden Zeitpunkt die Jobs ausführt.
at	Ein Programm, mit dem Jobs angelegt, aufgelistet und gelöscht werden können.
atq	Das gleiche wie `at -l`: die angelegten Jobs des Users auflisten.
atrm	Ein Alias für `at -d`: einen Job entfernen.
batch	Stellvertretend für `at -q b -m now`. Damit wird ein Job anlegt, der gestartet wird, wenn das System unter geringer Last steht.

Von den drei Aliassen ist nur `batch` nach POSIX und SUSv3 spezifiziert, er sollte also verfügbar sein. Die anderen beiden sind auf GNU/Linux-Systemen verbreitet, bei BSD eher nicht. Die oben aufgelisteten Programme liegen natürlich irgendwo im Pfad. (Das Kommando `whereis at` sagt wo.)

Zusätzlich gibt es das Spool-Verzeichnis, in dem die Jobs gespeichert sind. Dort schaut atd regelmäßig, ob einer der Jobs jetzt zu starten ist. Dieses Verzeichnis liegt je nach System an verschiedenen Orten. Es wird üblicherweise entweder unter `/var` oder `/var/spool` ein Verzeichnis mit Namen `at` oder `cron` sein.

Weiterhin können die Dateien `at.allow` und `at.deny` existieren, mit denen Nutzern die Berechtigung für *at* gegeben oder entzogen werden kann. Diese Dateien sollten entweder im jeweiligen Spool-Verzeichnis oder direkt unter `/etc` liegen. Grundsätzlich sollte die Manpage von *at* alle Pfade auflisten.

Verwendung

Jobs mit *at* anzulegen ist keine große Sache. Man startet das Programm einfach mit einer Zeitangabe als Argument. Dann gibt man die Befehle ein, die ausgeführt werden sollen und beendet mit ^D. (^D steht für die Tastenkombination Strg + D.) Dieser Vorgang wird nun genauer beschrieben.

Zeitangabe
Die Zeitangabe kann in allerlei verschiedenen Formen erfolgen, in jedem Fall ist sie aber das Pflichtargument beim Aufruf von *at*. Die einfachste Form ist die direkte Angabe einer Uhrzeit oder eines Datums. In der folgenden Tabelle stehen die gängigsten Formate. (Im deutschsprachigen Raum unübliche Zeitangaben wurden weggelassen. Diese können bei Bedarf in der Dokumentation nachgelesen werden.)

Zeitangabe

Beschreibung	Schema	Beispiel
Stunde	HH	18
Stunde + Minute	HHMM	1800
Stunde + Minute	HH:MM	18:00
Tag	DD.MM.YY	24.12.08

Wird nur eine Uhrzeit angegeben, bezieht diese sich auf die folgenden 24 Stunden – also den aktuellen Tag oder, falls sie an diesem schon vergangen sein sollte, auf den nächsten Tag. Uhrzeit- und Tagesangaben können auch kombiniert werden. Dabei muss die Uhrzeit- stets vor der Tagesangabe erfolgen. Ein Beispiel hierfür wäre 18:00 24.12.08.

Aliasse

Aliasname	steht für
midnight	00:00
noon	12:00
now	<jetzt>
today	<Heute>
tomorrow	<Morgen>

Gültige Zeitangaben sind zum Beispiel: `noon tomorrow`, oder `14:59 tomorrow`.

Man könnte meinen, der Alias `today` sei überflüssig, da sich Uhrzeiten automatisch auf den aktuellen Tag beziehen und es nur bei einer bereits zurückliegenden Uhrzeit einen Unterschied macht. Zum Beispiel `08:00` und `08:00 today`. Im ersten Fall wird der Job für acht Uhr am folgenden Tag angesetzt, im zweiten Fall wird er sofort ausgeführt, da die Zeit bereits in der Vergangenheit liegt (d. h. er entspricht `now`).

Tatsächlich wird `today` aber ziemlich oft verwendet. Noch öfter allerdings `now` (aber hauptsächlich, weil es zwei Buchstaben kürzer ist *g*).

Der Grund dafür sind die relativen Inkremente, die zu den Zeitpunkten noch angegeben werden können. Jede beliebige, oben beschriebene, Zeitangabe kann duch ein relatives Inkrement erweitert werden. Die Syntax dafür ist `<zeitpunkt> + <zahl> <einheit>`. Die `<einheit>` ist dabei eine der folgenden Worte: `minutes`, `hours`, `days`, `weeks`, `months`, `years`. (Oder die jeweiligen Einzahlformen.)

So könnten Zeitangaben etwa diese sein: `now + 1 hour` oder `today + 4 days` oder auch `14:30 tomorrow + 1 week`.

Es sollte nun ersichtlich geworden sein, dass es eine Vielzahl möglicher Formen der Zeitangabe gibt. Diese Auflistung ist keineswegs vollständig. Sie ist primär an der SUSv3 orientiert. Je nach System können einzelne Angabemöglichkeiten abweichen oder weitere verfügbar sein. Es ist deshalb empfehlenswert, einen Blick in die lokale Manpage von *at* zu werfen.

Befehle
Die Befehle werden, wie für gute Unix-Programme üblich, von der Standardeingabe gelesen. Das heißt, standardmäßig von der Tastatur. Man kann also Befehle eingeben und beendet diese Eingabe dann mit ^D, das für EOF (= End of File) steht. Das Prinzip ist dasselbe wie zum Beispiel bei `cat` oder `mail` auch.

Da die Befehle von der Standardeingabe gelesen werden, hat man sehr einfach die Möglichkeit, sie alternativ aus einer Datei oder einer Pipe zu lesen. Hier zwei Beispiele dafür.

Aus einer Datei (hierbei muss man den Teil mit `echo` und das ^D auch eingeben):

```
$ cat >at-commands
echo "hello world"
^D
$ <at-commands at now + 5 minutes
```

Aus einer Pipe:

```
$ echo "d_te" | tr _ a | at noon
```

Zugegeben, gerade das zweite Beispiel ist sehr gestellt, jedoch wird der Alltag als Unix-Nutzer und System-Administrator schon Situationen finden, in denen derartige Möglichkeiten Gold wert sind.

Job-Management

Hat man sich nun Jobs angelegt, so interessiert einen vielleicht später noch mal, welche Jobs angelegt sind und wann sie starten werden. Für diesen Zweck gibt es `at -l`, beziehungsweise den Alias `atq`. Damit kann man sich eine Liste von anstehenden Jobs ausgeben lassen. Normalen Benutzern werden ihre eigenen Jobs aufgelistet, root dagegen bekommt alle zu sehen. (Die erste Spalte der Liste ist die Job-Nummer.)

Anhand seiner Nummer kann ein Job auch wieder gelöscht werden. Ein simples `at -d <jobnr>`, und der Job ist weg. Alternativ kann hier der Alias `atrm` verwendet werden – aber ob dieser existiert, hängt vom vorliegenden System ab.

Benachrichtigung

Manch einer hat sich vielleicht schon gefragt, welchen Sinn ein Befehl wie `date` oder `echo` in einem Hintergrund-Job haben soll. Richtig, das ist erst einmal nicht sinnvoll. Zum Testen allerdings ist es ungemein geschickt. Wie cron auch, schickt *at* in Jobs anfallende Ausgaben per System-Mail an den jeweiligen Nutzer. (Ein MTA muss dafür natürlich verfügbar sein.) Um zu zeigen, wie das aussehen kann, folgt hier ein Auszug einer Shellsession.

Mailnachricht eines Jobs:

```
$ at now
at> echo hello world
at> ^D
job 8 at 2008-08-11 14:59
```

```
$ mail
Mail v. 8.1 6/6/93. Type ? for help.
"/var/spool/mail/meillo": 1 message
>N 1 meillo@localhost Mon Aug 11 ↷
14:59 16/727 "Output from your job 8"
& p
Message 1:
From meillo@localhost Mon Aug 11
14:59:13 2008
Date: Mon, 11 Aug 2008 14:59:13 +0200
From: meillo@localhost
Subject: Output from your job 8
To: meillo@localhost

hello world

& d
& q
```

Mail-Benachrichtigungen werden nur versendet, wenn Ausgaben im Job anfallen. Erzeugt keines der Programme im Job eine Ausgabe, wird auch keine Mail verschickt. Falls man es trotzdem wünscht, kann man sich aber mit der Option `-m` in jedem Fall nach Beendigung des Jobs eine Nachricht senden lassen. Gerade bei lange laufenden Programmen ist dies sinnvoll.

Der Alias `batch`, der Jobs automatisch dann ausführt, wenn die Systemlast niedrig ist, hat diese Option standardmäßig gesetzt. Wer also umfangreiche und vielleicht sogar Ressourcen-intensive Aufgaben zu erledigen hat, sollte sich `batch` genauer anschauen.

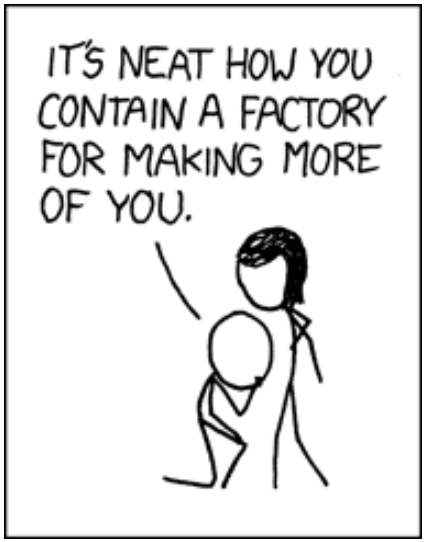
In der nächsten Ausgabe von freiesMagazin werden dann einige Probleme besprochen, die man mit *at* lösen kann.

LINKS

[1] http://www.opengroup.org/onlinepubs/009695399/utilities/at.html

Autoreninformation

Markus Schnalke ist ein überzeugter Fan der Unix-Philosophie und interessiert sich somit stark für Programme, die nach ihren „Geboten“ programmiert wurden. Dies sind in erster Linie die alten Unix-Tools, die auf nahezu jedem System verfügbar sind.



„Advanced Technology“ © by Randall Munroe (CC-BY-NC-2.5), http://xkcd.com/387

Um Musik des Künstlers foo abzuspielen, muß man einfach `play foo` in *GNOME Do* eintippen. Natürlich können auch die üblichen Player-Funktionen so gesteuert werden. Es existieren Plugins für die Musikplayer Rhythmbox und Amarok, die den Funktionsumfang erweitern.

Es stehen noch jede Menge anderer Aktionen zur Verfügung, etwa markierten Text zu einer Pastebin-Website schicken, Screenshots erstellen, sich über SSH oder VNC an einem Server anmelden, sich abmelden, den Benutzer wechseln oder neustarten.

Im *GNOME-Do*-Wiki [1] findet sich eine schöne Übersicht über alle Plugins und deren Einsatz. Es ist natürlich auch möglich, selbst Plugins zu schreiben – allerdings nur für die, die der Sprache Mono mächtig sind.

Alles in allem ist *GNOME Do* eine klare Unterstützung im Produktions-Fluss am Desktop, weil Aktionen mit Programmen und Dateien schnell ausgeführt werden können, ohne dafür umständlich die Maus oder das Trackpad einsetzen zu müssen. KDE-Benutzer können Katapult verwenden, das ganz ähnlich funktioniert.

Weiterführende Informationen finden sich auf den Entwicklerseiten [2] und im Ubuntu Team Wiki [3] und ein englischsprachiges Video-Tutorial [4] demonstriert den Einsatz von *GNOME Do*.

LINKS

[1] https://wiki.ubuntu.com/GNOMEDo/Plugins
[2] http://do.davebsd.com/
[3] https://wiki.ubuntu.com/GNOMEDo
[4] http://video.google.com/videoplay?docid=-9110909248380195562&hl=en

Autoreninformation

Bernd Essl verwendet GNOME seit knapp 2 Jahren und war davor ein Minimal-Desktop-Benutzer unter anderem mit ICE-WM und Fluxbox. Um die Produktivität beim täglichen Arbeiten zu steigern, verwendet er Programme wie GNOME Do.

at – Beispiele aus dem Alltag

von Markus Schnalke

Den Cron-Daemon kennt und verwendet fast jeder Unix-Benutzer, sein Gegenstück *at* dagegen kennen die meisten nur vom Namen. Die Funktion, die *at* anbietet, kann jedoch im Alltag deutlich häufiger eingesetzt werden, als man es erst mal erwartet.

Dieser Artikel zeigt anhand von einigen Beispielen, was man mit *at* alles machen kann. Es empfiehlt sich, vorher den Einführungsartikel „at – ein (fast) vergessenes Helferlein“ aus freiesMagazin 10/2008 [1] gelesen zu haben.

Erinnerungen per Mail

Es passiert oft, dass man in ein paar Tagen eine E-Mail an jemanden senden will oder dass man an einem bestimmten Datum eine Aktion starten muss oder andere Dinge dieser Art. An dieser Stelle ist *at* wohl am meisten wert. Ein simples

```
$ at 01.12.08
at> mail meillo -s ↷
"Weihnachtsfeier organisieren"
at> (weitere Infos einfügen)
at> ^D
```

und schon muss man nicht mehr daran denken. Am entsprechenden Tag wird man dann die Mail in seinem Postfach finden und weiß: Jetzt ist es so weit. Dies ist einfacher und vor allem schneller, als zum Beispiel Programme wie „remind“ [2] dafür zu verwenden.

So ähnlich ist es bei folgendem Sachverhalt: Viele lassen ihre Daten von automatisierten Backup-Skripten (gesteuert von cron) sichern. Nun gibt es aber Tätigkeiten, bei denen man doch selbst Hand anlegen muss. Das ist etwa das Einlegen eines Rohlings in den DVD-Brenner, um das Backup offline zu haben. Also lässt man sich in seinem Backup-Intervall von cron eine Erinnerungsmail schicken, dass es jetzt wieder an der Zeit ist, einen Rohling einzulegen und das (automatisch erstellte) Backup darauf zu brennen.

Es kann vorkommen, dass man beim Eintreffen der Mail ziemlich beschäftigt ist und auch in den nächsten Tagen nicht dazu kommen wird, das Backup zu brennen. Vielleicht ist man ja nicht zuhause oder es könnte sein, dass man gerade keine Rohlinge da hat. Anstatt die Mail jetzt in der Inbox liegen zu lassen, wo sie vermutlich bald untergehen würde, lässt man sich von *at* einfach zwei, drei Tage später eine neue Mail schicken:

```
$ at now + 2 days
```

Eigentlich `today + 2 days`, aber `now` ist kürzer.

Erinnerungen im Terminal

Neben Erinnerungen per E-Mail, die man üblicherweise erst Tage später erhalten will, kann es sinnvoll sein, sich nur Stunden oder Minuten später an etwas erinnern zu lassen.

Hierfür sind E-Mails nicht besonders geeignet, da sie nicht minutengenau abgerufen und schon gar nicht so regelmäßig gelesen werden. Viel sinnvoller ist es, sich direkt im Terminal eine Hinweismeldung anzeigen zu lassen.

Das geeignete Tool dafür ist `write`. Es schickt eine Meldung an das Terminal in dem der Nutzer gerade arbeitet. Für Konsolenbenutzer ist das eine sehr praktische Sache.

Es sei angenommen, dass man um 16 Uhr aus dem Haus möchte und deshalb rechtzeitig mit Programmieren aufhören muss. Anstatt eine Eieruhr zu stellen, könnte man alternativ Folgendes verwenden:

```
$ at 15:40
at> write meillo
at> Programmieren beenden!
at> ^D
```

Zur angegebenen Uhrzeit wird auf dem aktuellen Terminal folgende Meldung erscheinen:

```
Message from meillo@localhost on
(none) at 15:40 ...
Programmieren beenden!
EOF
```

Wer vorwiegend grafische Programme nutzt, wird davon nicht besonders viel haben, aber für Kommandozeilen-Freunde ist es eine ziemlich nette Sache.

Zeitgesteuerte Downloads

Auch in der heutigen Zeit soll es Leute geben, denen nur eine sehr langsame Internetverbindung zur Verfügung steht. (Der Autor des Artikels gehört leider zu ihnen.) Diese versuchen größere Downloads möglichst auf Zeiten zu verlagern, an denen sie nicht interaktiv im Netz tätig sind. Wer einen Server hat, der dauerhaft online ist oder seinen Desktop-Rechner die Nacht durch laufen lässt, hat da kein Problem.

Fans von „ncftp“ [3] sind da ziemlich verwöhnt, denn mit ncftpbatch ist es ohne weiteres möglich, Downloads auf nachts zu verlagern. Tatsächlich ist ncftpbatch nichts anderes als ein eingebauter *at*-Daemon. Statt ihn zu verwenden, kann man aber genau so gut jedes beliebige Programm in Verbindung mit *at* einsetzen. Eine große Datei lädt man sich mit folgenden Anweisungen über Nacht runter:

```
$ at -m midnight
at> cd ~/downloads
at> wget -c ftp://example.com/some-↷
big-file.ext
at> ^D
```

Durch das Einfügen von `sudo halt` als weitere Zeile, kann man seinen Rechner sogar automatisch herunterfahren lassen. (Das Programm „sudo“ muss dazu natürlich installiert und passend konfiguriert sein.)

Hangup vermeiden

Wer nicht nur auf seinem Desktop-Rechner, sondern auch auf entfernten Servern arbeitet, startet lang laufende Aufgaben gerne am Ende seiner Arbeitszeit. Das Problem ist erst mal, dass gestartete Prozesse Kinder der Login-Shell sind und deshalb automatisch beendet werden, wenn

(beim Logout) die Shell beendet wird. Das Programm in Hintergrund (mit &) zu starten, hilft da nicht.

Was Prozesse von der Login-Shell entkoppelt, ist das Kommando `nohup`, das für „no hangup“, also kein Beenden des Programms beim Beenden der Verbindung, steht. Meldet man sich ab, wird der Prozess automatisch ein Kind des init-Prozesses und lebt weiter.

Alternativ zu `nohup` kann das gleiche Verhalten auch mit `at now` erreicht werden. Ob hier *at* sinnvoll eingesetzt wird oder ob `nohup`, `dtach` oder `screen` besser geeignet sind, soll nicht beurteilt werden. Darüber zu wissen ist aber sicher nicht schlecht.

Den Computer beenden

Das letztes Beispiel beschreibt das automatische Herunterfahren des Rechners. Manch einer hört gerne Abends im Bett noch ein bisschen Musik. Wenn diese vom Computer kommt, dann sollte sich dieser aber auch automatisch ausschalten, wenn die Musik geendet hat. Hierfür gibt es auch wieder mehrere Ansätze, und auch wenn dieser Artikel *at* beschreibt, werden auch die anderen kurz vorgestellt. Es ist entscheidend, die Alternativen zu kennen, um eine vernünftige Wahl treffen zu können.

Zuerst die *at*-Variante: Man startet den Player, schaut, wie lange die Musik noch läuft, addiert eine „Toleranzminute“ hinzu und ruft dann

```
$ at now + <ANZAHL> minutes
```

mit dem Shutdown-Befehl (z. B. `sudo /sbin/halt`) auf. Sollte man sich später doch noch anders entscheiden und will den Rechner doch nicht zu dieser Zeit herunterfahren, kann man den Job ja mit `at -d` entfernen.

Alternativ kann man `sleep` verwenden. Der passende Befehl wäre dann in etwa:

```
$ sleep <ANZAHL>m && sudo /sbin/halt
```

Das && statt ; dient dazu, dass man den Timer noch abbrechen kann. Wird `sleep` mit Strg + C beendet, dann gibt es einen Wert ungleich Null zurück und das nachfolgende Kommando wird nicht ausgeführt – genau das, was man in diesem Fall will.

Statt `sleep` kann man auch einfach seinen Musikplayer vor das Shutdown-Kommando spannen. Ein Beispiel hierfür wäre:

```
$ mpg321 *.mp3 && sudo /sbin/halt
```

Das Gute daran ist, dass der Shutdown eben gerade dann eingeleitet wird, wenn die Lieder vorbei sind. Das Problem ist aber, dass mpg321 nicht mit einem Fehlercode endet, wenn er abgebrochen wird. Man kann sich also nicht mehr umentscheiden, wenn der Befehl einmal gestartet ist. Denn der Rechner wird auch heruntergefahren, wenn der Player abgebrochen wird. Ein anderer Player reagiert an dieser Stelle vielleicht anders. Man sollte sich dieses Verhaltens bewusst sein!

Ich verwende meist die Variante mit `sleep`, da sie wohl die einfachste ist. Nichtsdestotrotz ist *at* genauso gut geeignet und vor allem noch flexibler, da man auch eine absolute Uhrzeit angeben kann.

Fazit

Mit diesen Beispielen schließt der Artikel zu *at* ab. Seine Struktur und Bedienung wurden erklärt und Beispiele für seinen Einsatz im Alltag eines Unix-Benutzers gegeben. Ziel war es, *at* mal aus seinem Schattendasein herauszuholen und ins Rampenlicht zu rücken.

LINKS

[1] http://www.freiesmagazin.de/freiesMagazin-2008-10

[2] http://www.roaringpenguin.com/products/remind

[3] http://www.ncftp.com/ncftp/

Autoreninformation

Markus Schnalke ist ein überzeugter Fan der Unix-Philosophie und interessiert sich somit stark für Programme, die nach ihren „Geboten“ programmiert wurden. Dies sind in erster Linie die alten Unix-Tools, die auf nahezu jedem System verfügbar sind. So auch at, das Inhalt dieses Artikels ist.