Cronjobs einrichten

Internal reference: topics/05-3.md

Einführung

Mithilfe von Cronjobs können auf Unix - und Linux-Systemen Vorgänge automatisiert und zu einem bestimmten Zeitpunkt immer wiederkehrend ausgeführt werden. Diese Vorgänge können einzelne Befehle, Shell-Scripts, Programme, PHP- und sonstige Scriptsprachen-Scripts oder auch eine Anreihung von Linux-Befehlen sein. Beispielsweise werden Backups, die wöchentlich, täglich oder stündlich geschehen sollen meist per Cronjob ausgeführt.

Crontab

Crontab wird die Tabelle genannt, in der die einzelnen Cronjobs definiert und konfiguriert werden. Die Tabelle enthält pro Zeile den Zeitpunkt und die Befehlsfolge, die ausgeführt werden soll. Der Begriff Crontab setzt sich aus dem griechischen Chronos (Zeit) und lateinischen Tabula (die Tafel, das Brett) zusammen.

Gleichzeitig ist crontab auch das Programm, mit dessen Hilfe man die Crontabs bearbeiten kann. Um die Crontab zu bearbeiten, muss folgender Befehl eingegeben werden:

```
crontab -e
```

Crontab Syntax

Jeder Cronjob hat folgendes Format:

```
  * * * * * auszuführender Befehl
┬ ┬ ┬ ┬ ┬
│ │ │ │ │
│ │ │ │ └──── Wochentag (0-7, Sonntag ist 0 oder 7)
│ │ │ └────── Monat (1-12)
│ │ └──────── Tag (1-31)
│ └────────── Stunde (0-23)
└──────────── Minute (0-59)
```

Ein Stern * bedeutet Ausführung wird immer erfolgen, also zu jeder Minute, jeder Stunde, jedem Tag, jedem Monat oder jedem Wochentag. Um die einzelnen Stellen auseinander zu halten, hilft folgendes Diagram:

```
**1 2 3 4 5 Befehl**
```

```
1 = Minute (0-59)
2 = Stunde (0-23)
3 = Tag (0-31)
4 = Monat (1-12)
5 = Wochentag (0-7, Sonntag ist 0 oder 7)
Befehl = Der auszuführende Befehl.
```

Für die ersten fünf Stellen, also die Zeitwerte sind folgende Optionen zusätzlich möglich:

```
* = Ausführung immer (zu jeder…)
*/n = Ausführung aller n, wobei n,x,y = Ausführung um/am n, x und y
```

Cronjob Beispiele

Um zum Beispiel jede Nacht um 5 Uhr morgens das Backup auszuführen, würde man den Cronjob folgendermaßen anlegen:

```
0 5   * * * /usr/bin/backup.sh
```

Einen Sound alle 10 Minuten Abzuspielen könnte wie folgt aussehen:

```
*/10   * * * * /usr/bin/play_sound.sh
```

Eine Erinnerungsmail um 8 und um 17 Uhr zu verschicken, geht z.B. so:

```
0 8,17 * * * /usr/bin/send_reminder_mail.sh
```

An einem bestimmten Tag, beispielsweise am 31. - um 23:59 Uhr, eine Mail zu versenden, könnte so aussehen:

```
59 23 31 12 * (echo Lass uns die Raketen holen | mail -s Gleich knallt es
user@domain.xy)
```

Ausgabe der Cronjobs

Die Ausgabe der Cronjobs wird standardmässig per Mail an den jeweiligen System-User, der den Cronjob eingerichtet hat gesendet. Um dies zu unterdrücken, könnte man die Ausgabe in eine Datei umleiten oder mit Umleitung zu /dev/null komplett verwerfen:

Cronjob-Ausgabe in Logfile umleiten

```
0 8,17 * * * /usr/bin/script.sh >>/var/log/cron/send_reminder_mail 2>&1
```

Cronjob-Ausgabe verwerfen

```
0 8,17 * * * /usr/bin/script.sh >/dev/null 2>&1
```

2>&1 bedeutet, das sowohl die normale Ausgabe als auch Fehler in die vorher angegebene Datei umgeleitet werden.

Cronjob-Dateien und Verzeichnisse

Zusätzlich zum crontab-Befehl gibt es je nach Distribution Dateien, die systemweite Crontabs beinhalten und nur durch den User root bearbeitet werden können: `/etc/crontab`

Die System-Crontab-Datei, in der zusätzlich noch ein System-Benutzer, der den Befehl ausführen soll, mit angegeben werden muss:

```
**1 2 3 4 5 Benutzer Befehl
**1 = Minute (0-59)
2 = Stunde (0-23)
3 = Tag (0-31)
4 = Monat (1-12)
5 = Wochentag (0-7, Sonntag ist 0 oder 7)
Benutzer = Benutzername des Benutzers, unter dem der Befehl ausgeführt wird.
Befehl = Der auszuführende Befehl.
```

Beispiel

```
0 8,17 * * * root /usr/bin/script.sh >>/var/log/cron/send_reminder_mail
2>&10
3 * * * wwwrun /usr/bin/webjobs_nighly.sh >>/var/log/cron/send_reminder_mail
2>&1
```

crontab Verzeichnisse

Eine weitere Möglichkeit Cronjobs anzulegen sind die Verzeichnisse unter `/etc/cron*`, in denen alle enthaltenen Dateien zum bestimmten Zeitpunkt ausgeführt werden. Die Dateien im Einzelnen

```
/etc/cron.d/ = Erweiterungen zur /etc/crontab-Datei, gleiche Syntax.
/etc/cron.daily/ = Einmal irgendwann täglich.
/etc/cron.hourly/ = Einmal irgendwann stündlich.
```

```
/etc/cron.monthly/ = Einmal irgendwann monatlich.
/etc/cron.weekly/ = Einmal irgendwann wöchentlich.
```

Die letzten vier werden oft genutzt, wenn ein Job in einem bestimmten Interval erledigt werden muss, der genaue Zeitpunkt hierfür aber unerheblich ist.

Ein Tipp zur Verwendung des Prozentzeichens in Crontabs

Manchmal kommt es vor, dass man – beispielsweise bei der Verwendung eines Datums – das Prozentzeichen innerhalb eines Crontab-Befehls nutzen möchte:

```
0 8,17 * * * /usr/bin/script.sh >>/var/log/cron/script_$(date +%Y%m%d).log
```

Leider wird dies so nicht funktionieren und Crontab verweigert die Ausführung des Cronjobs mit der folgenden oder einer ähnlichen Fehlermeldung:

```
/bin/sh: -c: line 0: unexpected EOF while looking for matching ''')'/bin/sh:
-c: line 1: syntax error: unexpected end of file
```

Grund hierfür ist, dass Prozentzeichen im sechsten Feld einer jeden Crontab-Zeile generell in einen Zeilenumbruch übersetzt werden. Zitat aus der Crontab Manpage:

```
The sixth field of a line in a crontab file is a string that
is executed by the shell at the specified times. A percent
character in this field (unless escaped by \) is translated
to a NEWLINE character.

Only the first line (up to a ''%‘ or end of line) of the com-
mand field is executed by the shell. Other lines are made
available to the command as standard input. Any blank line
or line beginning with a ''#‘ is a comment and is ignored.
```

Die Lösung ist, das Prozentzeichen zu escapen, daher einen Backslash voranzustellen:

```
0 8,17 * * * /usr/bin/script.sh >>/var/log/cron/script_$(date
+\%Y\%m\%d).log
```

So steht dann auch der Nutzung des Datums mit Prozentzeichen nichts mehr im Wege.

Crontab und gut?

Ich hoffe, ich konnte einen kleinen Einblick in die Cronjobs unter Linux geben und dem ein oder anderen bei der Syntax behilflich sein. Dieser Artikel entsteht auch mit etwas Eigennutz, da ich auch oft überlegen muss, wie die genaue Reihenfolge in der Crontab nun ist. Es gibt natürlich noch eine

Menge anderer Tools wie http://linux.about.com/library/cmd/blcmdl1_at.htm, http://anacron.sourceforge.net/, http://fcron.free.fr/ und so weiter. Cronjob hat für mich jedoch sehr zuverlässig gearbeitet, weshalb ich andere Tools noch nicht in Betracht gezogen habe.

Let the crons work!

Based on: „Nico Puhlmann (https://www.stetic.com/developer/cronjob-linux-tutorial-und-crontab-syntax/)“
