The Times They Are A Changin'

(Un)regelmäßige Aufgaben mit Systemd planen

Das Automatisieren von Aufgaben unter Linux ist üblicherweise der Job der Daemons Cron und At. Die Timer von Systemd können beide ersetzen und komplexere Aufgabenstellungen mit Bordmitteln lösen.

Von Merlin Schumacher

Der System- und Sitzungsmanager Systemd hat sich bei allen Mainstream-Distributionen etabliert. Einst als Init-System entwickelt, ist Systemd inzwischen weit mehr als nur ein Dienst, der den Systemstart koordiniert. Eine interessante Fähigkeit von Systemd ist es, mit Hilfe von Timer-Units Dienste zu festgelegten Zeiten zu starten. Die Timer können flexibler als Cron und At auf Ereignisse wie Neustarts reagieren und nutzen die Infrastruktur von Systemd, um Aufgaben effizient und koordiniert auszuführen. Mit ihnen lassen sich auch Aufgaben erledigen, die bisher eine Kombination von Cron und einem weiteren Shell-Skript erforderten. Systemd [1] weiß recht genau, in welchem Zustand sich das System gerade befindet und kann das Abarbeiten von Aufgaben präzise koordinieren. Gleich vorweg: Das Anlegen eines Timers macht mehr Arbeit als das Eintragen einer einzelnen Zeile in die Crontab. Wer also nur eben schnell ein Skript regelmäßig ausführen will, ist mit Cron oftmals besser bedient. Es spricht aber nichts dagegen, beide zu benutzen.

Die Timer können zum Beispiel auf den Start anderer Dienste warten, ohne dass externe Tools prüfen müssen, ob diese Dienste bereits aktiv sind. Dann wartet beispielsweise das automatische Datenbank-Backup, bis die Datenbank auch wirklich läuft. Auch das schlichte Ausführen von Aufgaben im Sekundentakt bedarf keiner Sleep-Verkettungen mehr. Oder man lässt den Trim-Wartungsvorgang für die SSD zehn Minuten nach jedem Boot-Vorgang und anschließend regelmäßig alle siebe Tage starten.

Das Debuggen vereinfacht sich, da man die Aufgaben in einer definierten Shell-Umgebung startet und ihre Ausgaben gesammelt im Journal [2] landen, der zentralen Protokollfunktion von Systemd. Per Timer ausgeführte Aufgaben lassen sich auch einer Control Group (cgroup) zuweisen, um den Verbrauch von Rechenzeit, Arbeitsspeicher und anderen Systemressourcen zu begrenzen. Die Timer können den Rechner zur Ausführung der Aufgabe sogar aus dem Standby aufwecken. Timer können auch nicht nur systemweit, sondern auch auf User-Ebene arbeiten. Was Systemd allerdings nicht von Haus aus bewältigt, ist das Versenden von E-Mails im Fall eines Fehlers, wie man es von Cron typischerweise kennt.

Units ...

Systemd-Timer werden, wie andere von Systemd erledigte Aufgaben, über Unit-Files konfiguriert. Ein Timer kann nur Service Units ausführen, das heißt: Wenn das auszuführende Programm keine Unit mitbringt, müssen Sie eine schreiben. Units sind simple Textdateien und erinnern im Format stark an .ini-Dateien unter Windows. Selbstgeschriebene Units gehören in /etc/systemd/system/ etwa als simplebackup.service. Vom System mitgelieferte Units finden sich in /lib/systemd/system/. Eine simple Service-Unit sieht folgendermaßen aus:

```
[Unit]
Description=Ein simples Backup
[Service]
ExecStart=/usr/local/bin/backup.sh
```

Die erste Sektion unter dem Titel `[Unit]` beschreibt den Dienst. `Description` legt die Beschreibung fest, die dann im Journal von Systemd erscheint. Unter `[Service]` finden sich die Einstellungen für das eigentlich auszuführende Programm. `ExecStart` definiert das zu startende Programm oder Skript. In unserem Beispiel ist das Backup-Skript /usr/local/bin/backup.sh. Man sollte hier immer den absoluten Pfad angeben, das schützt vor Problemen mit nicht gesetzten Pfadvariablen. Man kann die Unit mit `sudo systemctl start simplebackup` testen. Das führt das Skript sofort einmal aus. Falls Sie einen bereits vorhandenen Service ändern wollen, müssen Sie diesen von /lib/systemd/system/ nach /etc/systemd/system kopieren und dort bearbeiten.

... und Timer

Um den erstellten Service regelmäßig zu starten, bedarf es eines Timers. Timer-Units ähneln Service-Units im Aufbau. Systemd unterscheidet zwischen ereignisbezogenen (engl. „montonous") und kalendarischen (engl. „realtime") Timern. Ereignisbezogene Timer lösen beispielsweise zehn Minuten nach dem Systemstart eine Service-Unit aus. Kalendarische starten ihre Unit zu bestimmten Zeitpunkten, also etwa um 13 Uhr. Unser Beispiel verwendet einen kalendarischen Timer. Tragen Sie in die Datei /etc/systemd/system/backuptimer.timer folgendes ein:

```
[Unit]
Description=Das tägliche Backup
[Timer]
OnCalendar=13:00
OnUnitActiveSec=24h
Persistent=true
Unit=simplebackup.service
[Install]
WantedBy=timers.target
```

Statt des Abschnitts `Service` gibt es hier den Abschnitt `Timer`. Da Timer nur andere Units starten können, jedoch keine Programme, wird hier mit `Unit` der gewünschte Dienst angegeben. Wenn Sie `Unit` leerlassen, startet Systemd einen Service, der so heißt wie der Timer. Das Schlüsselwort

`OnCalendar` legt den Zeitpunkt der Ausführung fest. Systemd akzeptiert hier nicht nur Zeitangaben, sondern auch Schlüsselwörter wie `daily` oder `hourly`. Wer sich nicht von der gewohnten Cron-Syntax lösen will, darf diese hier in abgewandelter Form verwenden. `*-*-* *:30:00` (oder verkürzt `*:30`) würde das Backup jede halbe Stunde auslösen.

Ist ein ereignisbezogener Timer gewünscht, gibt es Alternativen zu `OnCalendar`. Zum Beispiel `OnBootSec`, das für die Ausführung des Timers nach dem Booten des Systems sorgt, oder `OnUnitActiveSec`, welches einen Zeitpunkt nach dem letzten Start des Services repräsentiert.

Im Beispiel stellt `OnUnitActiveSec` sicher, dass der Timer auf jeden Fall alle 24 Stunden auslöst. Der Parameter `Persistent` weist Systemd an, den letzten Startzeitpunkt des Timers auf der Festplatte zu speichern, damit die Aufgabe auch ausgeführt wird, wenn der PC um 13 Uhr mal aus war. Eine Liste der möglichen Timertypen finden Sie in der Tabelle.

Auf die Plätze, fertig, los!

Zum Abschluss müssen Sie den Timer aktivieren und starten. Er kümmert sich dann in Zukunft um den Start des Services. Die erste Zeile aktiviert den Timer, sodass er bei jedem Systemstart auf das ihm zugewiesene Ergebnis wartet. Die zweite startet den Timer, sodass dieser sofort auf seinen Auslösezeitpunkt wartet. So muss man nicht erst neu starten, damit der Timer aktiv ist.

```
sudo systemctl enable backuptimer.timer
sudo systemctl start backuptimer.timer
```

Zur Kontrolle können Sie mit `systemctl list-timers` nachsehen, ob der Timer aktiv ist. Falls ja, stößt er das Backup jeden Tag um 13 Uhr an. Systemctl verrät auch gleich, wann der nächste Start geplant ist und wie lang der letzte her ist. Aufgaben laufen standardmäßig innerhalb der Minute, für die sie geplant sind. Für Aufgaben, die mit häufiger als einmal pro Minute ausgeführt werden sollen, muss man die Option `AccuracySec` setzen.

Durch die Angabe von `1s` erhöht sich die Präzision auf eine Sekunde. Die systemweite Timer-Genauigkeit kann in der Datei /etc/systemd/systemd.conf mit der Option `DefaultTimerAccuracySec` konfiguriert werden. Aus Gründen der Effizienz fasst Systemd Timer zusammen, die innerhalb einer Zeiteinheit geplant sind. So muss die Hardware nicht unnötig aufwachen. Die maximale Genauigkeit für die Timer-Ausführung beträgt 1 µs, sprich eine Mikrosekunde. Diese Einstellung zu verwenden ist aber nicht unbedingt sinnvoll. Denn so wird das Zusammenfassen von Timern verhindert und Energie verschwendet, weil Systemd die CPU weckt. Wenn sichergestellt sein soll, dass bestimmte Timer nicht gleichzeitig laufen, kann die Option `RandomizedDelaySec` den Startzeitpunkt um einen zufälligen Zeitraum verschieben.

Timer für jedermann

Die Systemd-Timer funktionieren auch für einzelne User, sofern die eigene Distribution Systemd auch für einzelne User startet, was nicht überall der Fall ist. Ob Ihre Distribution einen Systemd auf Benutzerebene startet, sehen Sie daran, ob der Prozess `/usr/lib/systemd/systemd --user` unter Ihrem Benutzernamen läuft.

Die Konfiguration ähnelt denen der systemweiten Timer, jedoch müssen Sie die Unit-Files im Ordner ~/.config/systemd/user/ erstellen. User sind in der Lage, sowohl vorgegebene System-Units als auch selbstgeschriebene Units auszuführen. Systemweite, also von einem Administrator vorgegebene User-Units gehören in /etc/systemd/user/. Diese kann jeder User für sich aktivieren und sie stehen allen zur Verfügung. Systemd führt die Timer-Unit des Nutzers dann mit dessen Rechten aus, jedoch nicht mit den Umgebungsvariablen des Benutzers, daher ist auch hier eine absolute Pfadangabe notwendig. Darüber hinaus laufen die Timer normalerweise nur, wenn der User auch eingeloggt ist. Dies können Sie mit dem Befehl `loginctl enable-linger BENUTZERNAME` ändern. Die Unit-Dateien brauchen Sie für die Benutzer-Timer nicht zu ändern. Einzig die systemctl-Kommandos müssen Sie um den Parameter `--user` ergänzen:

```
systemctl --user daemon-reload
systemctl --user enable␣
↳backuptimer.timer
systemctl --user start␣
↳backuptimer.timer
```

Der erste Befehl weist Systemd an, die Unit-Dateien neu einzulesen, der zweite aktiviert den Timer, der dritte startet ihn. Danach laufen Timer und Unit unter dem Konto des Benutzers und daher auch mit seinen Rechten.

Timer-Typen für Systemd

Parameter	Bedeutung
OnActiveSec	ab letzter Aktivierung des Timers
OnBootSec	ab Rechnerstart
OnStartupSec	ab Start von Systemd
OnUnitActiveSec	ab letzter Aktivierung des Services
OnUnitInactiveSec	ab letztem Beenden des Services
OnCalendar	kalendarischer Timer

Bugs und debuggen

Wer sichergehen will, dass der Service wie gewünscht arbeitet, kann ihn mit `systemctl start simplebackup` direkt ausführen. Treten dabei keine Fehler auf und das Backup läuft durch, dann sollte dem automatischen Backup nichts im Wege stehen. Den Output des Services zeigt, wie von Systemd gewohnt, der Befehl `journalctl -u simplebackup` an. *(mls@ct.de)* **ct**

Literatur

[1] Lennart Poettering, Kay Sievers, Thorsten Leemhuis, Das Init-System Systemd, Teil 1: http://heise.de/-1563259
[2] Thorsten Leemhuis, Sammelstelle, Log-Informationen beim Journal von Systemd abrufen, c't 13/14, S. 168

```
[redacted] systemctl list-timers
NEXT                         LEFT          LAST                         PASSED       UNIT                         ACTIVATES
Mi 2016-07-20 18:27:30 CEST  4s left       Mi 2016-07-20 18:27:17 CEST  8s ago       backuptimer.timer            simplebackup.service
Mi 2016-07-20 19:00:00 CEST  32min left    Mi 2016-07-20 18:00:06 CEST  27min ago    rsnapshot-hourly.timer       rsnapshot@alpha.service
Do 2016-07-21 00:00:00 CEST  5h 32min left  Mi 2016-07-20 00:00:06 CEST  18h ago      logrotate.timer              logrotate.service

11 timers listed.
Pass --all to see loaded but inactive timers, too.
```

Eine schnelle Übersicht über alle Timer schafft Systemctl.