Engineer

cron

der Unix-Dienst für zeitgesteuerte Befehle aus einer crontab

Was es ist

cron ist der Unix-Dienst, der Befehle zu festgelegten Zeiten startet, ohne dass jemand dafür angemeldet sein muss. Der Zeitplan jedes Nutzers steht in einer crontab: einer Klartexttabelle mit einer Aufgabe pro Zeile. Dasselbe Wort bezeichnet den Befehl `crontab`, der diese Tabelle installiert, ausgibt und löscht.

POSIX standardisiert den Befehl und das Tabellenformat, aber nicht das Programm, das die Tabellen liest. Die Begründung sagt es ausdrücklich: „Alle Verweise auf einen cron-Daemon und auf cron-Dateien wurden weggelassen.“ Welcher Daemon die Aufgaben ausführt, bleibt dem jeweiligen System überlassen. Die hier verwendeten cron-Handbuchseiten stammen von cronie. Dessen crontab(5) nennt Paul Vixie als Autor und bezeichnet sein Format als V7-Stil mit kompatibel bleibenden Erweiterungen.

Eine Zeile hat sechs durch Leerraum getrennte Felder. Die ersten fünf bestimmen die Zeit: Minute (0 bis 59), Stunde (0 bis 23), Tag des Monats (1 bis 31), Monat (1 bis 12) und Wochentag (0 bis 6, wobei 0 für Sonntag steht). Jedes Feld enthält einen Stern für alle Werte, eine Zahl, einen Bereich mit Bindestrich wie `1-5` oder eine durch Kommas getrennte Liste daraus. Das sechste Feld ist der von `sh` ausgeführte Befehl. Hier endet der Standard. Schritte wie `*/15`, dreibuchstabige Namen wie `mon`, eine `7` für Sonntag, eine Nutzernamenspalte in Systemtabellen und Kurzformen wie `@daily` sind cronie-Erweiterungen. Eine Tabelle, die sie verwendet, ist nicht mehr portabel.

`crontab -e` öffnet Ihre Tabelle in einem Editor, `crontab -l` gibt sie aus und `crontab -r` entfernt sie. `crontab file` ersetzt sie durch den Inhalt einer Datei. cronie hält außerdem Systemtabellen in `/etc/crontab` und `/etc/cron.d/` vor.

Dort benennt ein zusätzliches Feld zwischen Zeitangaben und Befehl den Nutzer, unter dem er laufen soll. POSIX rechnete damit: Die Begründung merkt an, dass von BSD abgeleitete Systeme wahrscheinlich ein Nutzerfeld ergänzen.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist cron?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Server, E-Mail, Konten und Updates werden im Alltag betreut, damit die Dinge, auf die alle angewiesen sind, unauffällig funktionieren.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 10 PostgreSQL-/MS-SQL-Server auf Ubuntu — 10 bereitgestellte und funktionsfähig gehaltene Datenbankserver

10 PostgreSQL-/MS-SQL-Server auf Ubuntu Kontakt aufnehmen →

Warum es wichtig ist

Ein großer Teil der Serverpflege geschieht unbeobachtet: alte Dateien löschen, Berichte versenden, Datenbankdumps erstellen, Sicherungen anlegen. cron ist der übliche Weg zu sagen: „Führe das um drei Uhr morgens aus.“ Wie die Änderungshistorie seiner POSIX-Seite festhält, gehört das Werkzeug seit Issue 2 zum Standard. Eine Tabelle, die sich an die POSIX-Felder hält, wird auf jedem konformen System gleich gelesen.

Die Schwäche ist eingebaut. Eine geplante Aufgabe hat kein Publikum. Scheitert sie, färbt sich deshalb kein Bildschirm rot. POSIX sagt, dass nicht umgeleitete Ausgaben und Fehler eines Befehls auf einem von der Implementierung gewählten Weg per Mail an den Nutzer gehen. Auf einem Host, dessen lokale Mail nirgendwo ankommt, verschwindet dort auch der Fehler. Die meisten cron-Probleme gehen auf dieses Schweigen zurück.

Zwei benachbarte Werkzeuge teilen sich heute die Arbeit. Auf von systemd verwalteten Linux-Systemen kann eine Timer-Unit dieselbe Aufgabe starten und ergänzt, was cron fehlt: Sie kann einen während abgeschalteter Maschine verpassten Lauf nachholen, Startzeiten zufällig verteilen und startet nie eine zweite Kopie eines noch laufenden Dienstes. Innerhalb von PostgreSQL liest die Erweiterung `pg_cron` dieselben Zeitpläne mit fünf Feldern. Ein Hintergrundworker in der Datenbank verwaltet die Aufgaben. Das River-Jobqueue-Projekt des Unternehmens ist ein Beispiel: Im Oktober 2026 zählt

seine Seite 20 `pg_cron`-Aufgaben unterhalb der Queue. Sie bereinigen Partitionen, rotieren Salts und aktualisieren Aggregate, also Pflegearbeit, die laut Seite in die Datenbank gehört.

Wie es in der Praxis aussieht

Nehmen Sie zwei Aufgaben: eine Bereinigung von Montag bis Freitag und einen nächtlichen Datenbankdump. Die Tabelle setzt zuerst ihre eigene Umgebung, weil eine Aufgabe die Einstellungen Ihrer Anmeldeshell nicht erbt. Die Zeilen `NAME=value` und die Variable `MAILTO` sind cronie-Funktionen, keine POSIX-Funktionen.

```
SHELL=/bin/sh
PATH=/usr/local/bin:/usr/bin:/bin
MAILTO=ops@example.com
# min hour dom mon dow command
15 3 * * 1-5 find "$HOME" -name core -exec rm -f {} +
30 2 * * * flock -n /var/lock/dump-db dump-db >> "$HOME/dump.log" 2>&1
0 * * * * date +\%s >> "$HOME/hourly.log"
```

Lesen Sie die erste Aufgabe als „um 03:15, Montag bis Freitag“. Die zweite läuft jede Nacht um 02:30, schreibt Ausgaben und Fehler in ein Log und ist mit `flock -n` umschlossen. Ist die Sperre schon belegt, bricht das Werkzeug sofort ab, statt zu warten. Ein lange laufender Dump bekommt deshalb nie einen Zwilling. Die dritte Zeile zeigt das maskierte Prozentzeichen: Ohne den Rückstrich würde cron den Befehl bei `%` abschneiden.

Halten Sie die Tabelle in einer versionierten Datei und installieren Sie sie vollständig, statt sie direkt im laufenden System zu bearbeiten:

```
crontab jobs.cron
crontab -l
```

Der zweite Befehl gibt aus, was cron tatsächlich lesen wird. cronies Option `-T` prüft die Syntax einer Datei vor der Installation. Die Datei muss mit einem Zeilenumbruch enden.

Derselbe nächtliche Dump als systemd-Timer braucht zwei Units. Der Dienst sagt, was ausgeführt wird, der Timer, wann:

```
# /etc/systemd/system/dump-db.service
[Service]
Type=oneshot
ExecStart=/usr/local/bin/dump-db

# /etc/systemd/system/dump-db.timer
[Timer]
OnCalendar=*-*-* 02:30:00
Persistent=true
RandomizedDelaySec=10min

[Install]
WantedBy=timers.target
```

`Persistent=true` speichert die Zeit des letzten Laufs auf der Festplatte und startet einen verpassten Lauf, sobald der Timer wieder aktiv ist. `RandomizedDelaySec=` ergänzt eine zufällige Verzögerung von bis zu zehn Minuten. Eine Maschinenflotte mit derselben Datei trifft den Datenbankserver so nicht in derselben Sekunde. `systemd-analyze calendar '*-*-* 02:30:00'` gibt den normalisierten Ausdruck und den nächsten Ausführungszeitpunkt aus. Das ist der schnellste Weg, einen Zeitplan zu testen, bevor man sich darauf verlässt.

Ein Zeitplan sagt, wann eine Aufgabe startet, nicht, ob sie ihre Arbeit erledigt hat. Auf einem Host des Unternehmens mit 464 MB kehrt der Totmannschalter das um: Ein Timer startet mit zufälliger Streuung viermal pro Stunde ein Skript. Es sendet nur dann ein Lebenszeichen an einen externen Dienst, wenn weder verfügbarer Arbeitsspeicher noch freier Festplattenplatz unter den jeweiligen Grenzwert gefallen sind. Bleiben die Lebenszeichen aus, wird Alarm ausgelöst.

Wo es schiefgeht

Der Befehl funktioniert in Ihrer Shell, aber nicht in cron. POSIX sichert einer Aufgabe nur `HOME`, `LOGNAME`, `PATH` und `SHELL` zu. `PATH` muss nur die Standardwerkzeuge finden können. Aus Ihrem Profil wird nichts geladen. Die Lösung sind absolute Pfade oder ein oben in der Tabelle gesetzter `PATH` sowie ein Test mit derselben spärlichen Umgebung.

Fehler verschwinden. Nicht umgeleitete Ausgaben werden per Mail versendet. cronie schickt sie an den Eigentümer der Tabelle, an die Adresse in `MAILTO` oder bei `MAILTO=""` nirgendwohin. Leiten Sie Ausgaben in ein Log um, das jemand liest. Wichtige Aufgaben sollten ihren Erfolg an etwas melden, das sich bei ausbleibenden Meldungen beschwert, wie der oben beschriebene Totmannschalter.

Ein Prozentzeichen teilt den Befehl. Im Befehlsfeld wird `%` zu einem Zeilenumbruch. Nur die erste Zeile wird ausgeführt, der Rest wird ihr als Standardeingabe übergeben. Deshalb scheitert `date +%s`. Maskieren Sie das Zeichen als `\%` oder verschieben Sie den Befehl in ein Skript und rufen Sie dieses auf.

Die beiden Tagesfelder sind mit Oder verknüpft, nicht mit Und. Sind sowohl Monatstag als auch Wochentag eingeschränkt, läuft die Aufgabe an jedem Tag, der eines davon erfüllt. Das POSIX-Beispiel `0 0 1,15 * 1` läuft am 1., am 15. und an jedem Montag. Beschränken Sie nur eines der beiden Felder oder verwenden Sie einen systemd-Kalender. Dort passt `Mon *-*-01..07 09:00` nur, wenn Wochentag und Datum beide stimmen: am ersten Montag des Monats.

Schritte beginnen jede Stunde neu. cronie wertet einen Schritt nur innerhalb seines Felds aus. `*/35` im Minutenfeld läuft deshalb um :00 und :35, nicht alle 35 Minuten. `*/23` im Stundenfeld läuft um Mitternacht und um 23:00. Verwenden Sie einen Schritt, der den Bereich ohne Rest teilt, oder einen systemd-Timer mit `OnUnitActiveSec=`, der ab dem letzten Start des Dienstes zählt.

Die Uhr wird umgestellt. cronie dokumentiert, dass durch eine Sommerzeitumstellung übersprungene Zeiten nie passen und die entsprechenden Aufgaben daher nicht laufen. Doppelt auftretende Zeiten lassen Aufgaben zweimal laufen. Planen Sie außerhalb der Umstellungsstunden Ihrer Zeitzone oder setzen Sie `CRON_TZ` auf eine Zone ohne Sommerzeit.

Der letzten Zeile fehlt der Zeilenumbruch. cronie verlangt für jeden Eintrag einen abschließenden Zeilenumbruch. Fehlt er beim letzten, behandelt es die Tabelle als zumindest teilweise beschädigt und schreibt eine Warnung ins Syslog. Die Lösung ist ein abschließender Zeilenumbruch, den eine Editoreinstellung oder eine Prüfung im Review sicherstellen kann.

Läufe überlappen sich. Das crontab-Format hat kein Feld dafür, was mit einem noch laufenden Vorgänger geschehen soll. Ein systemd-Timer lässt einen laufenden Dienst in Ruhe. Verstrich der nächste geplante Zeitpunkt, während der Dienst noch lief, startet der Timer ihn unmittelbar nach seinem Ende erneut, sofern `DeferReactivation=` nicht gesetzt ist. `pg_cron` lässt eine zweite Instanz warten, bis die erste abgeschlossen ist. Bei schlichtem cron hilft eine Sperre wie `flock -n` im Beispiel oben.

Die Maschine war ausgeschaltet. Eine crontab-Zeile benennt einen Zeitpunkt. Das Format hat keine Möglichkeit zu sagen: „Führe es später aus, wenn dieser Zeitpunkt verpasst wurde." Läuft die Maschine unter systemd, hilft `Persistent=true` an einem `OnCalendar=`-Timer.

Alle Maschinen zugleich. Eine auf hundert Hosts kopierte Tabelle startet hundert Aufgaben in derselben Minute. Helfen können `RandomizedDelaySec=` an einem systemd-Timer oder in cronie die Variable `RANDOM_DELAY` beziehungsweise ein Tilde-Bereich wie `6~15`. Er wählt beim Lesen der Tabelle eine zufällige Minute.

Ein bloßes `crontab` löscht die Tabelle. Ohne Argument wartet der Befehl auf eine neue Tabelle über die Standardeingabe. Ein Dateiende, gewöhnlich durch Ctrl-D, ersetzt die aktuelle Tabelle durch nichts. Drücken Sie stattdessen die Unterbrechungstaste und halten Sie die Tabelle in einer Datei, die Sie mit `crontab file` installieren.

Weiterführende Lektüre

- crontab, POSIX.1-2024 — die sechs Felder des Standards, Standardumgebung, Mailregel und Begründung.
- crontab(5), cronie — Schritte, `MAILTO`, `CRON_TZ`, `RANDOM_DELAY`, Sommerzeit und Fallstricke.
- systemd.timer(5) — `OnCalendar=`, `Persistent=`, `RandomizedDelaySec=` und laufende Dienste.
- systemd.time(7) — Syntax von Kalenderereignissen, Kurzformen und ihre Validierung.
- flock(1) — einen Befehl mit einer Sperre umschließen, mit `-n` zum sofortigen Abbruch bei belegter Sperre.
- pg_cron — cron-Zeitpläne für SQL-Aufgaben, als Erweiterung in PostgreSQL gehalten.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist cron?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/cron/