Engineer

cron

schemaläggaren i Unix som kör kommandon på bestämda tider

Vad det är

cron är tjänsten i Unix som startar kommandon vid bestämda tider, utan att någon behöver vara inloggad för att starta dem. Varje användares schema finns i en crontab: en tabell i ren text med ett jobb per rad. Samma ord används om kommandot `crontab`, som installerar, visar och tar bort tabellen.

POSIX standardiserar kommandot och tabellformatet, men inte programmet som läser tabellerna. Standardens motivering säger rakt ut att alla hänvisningar till en cron-demon och till cron-filer har tagits bort. Vilken demon som kör jobben är upp till varje system. Manualsidorna om cron som den här sidan bygger på kommer från cronie, vars crontab(5) anger Paul Vixie som upphovsman och beskriver formatet som ett V7-liknande format med ett antal utökningar som behåller kompatibiliteten.

En rad har sex fält åtskilda av blanksteg eller tabbar. De fem första anger när: minut (0 till 59), timme (0 till 23), dag i månaden (1 till 31), månad (1 till 12) och veckodag (0 till 6, där 0 är söndag). Varje fält är en asterisk för alla värden, ett tal, ett intervall med bindestreck som `1-5` eller en kommaseparerad lista av sådana. Det sjätte fältet är kommandot, som `sh` kör. Där slutar standarden. Steg som `*/15`, namn på tre bokstäver som `mon`, `7` för söndag, en kolumn med användarnamn i systemtabeller och förkortningar som `@daily` är utökningar i cronie; en tabell som använder dem är inte längre portabel.

`crontab -e` öppnar din tabell i en editor, `crontab -l` skriver ut den och `crontab -r` tar bort den, medan `crontab fil` ersätter den med innehållet i en fil. cronie har dessutom systemtabeller i `/etc/crontab` och `/etc/cron.d/`, där ett extra fält mellan tiderna och kommandot anger vilken användare jobbet körs som. Det hade POSIX förutsett: motiveringen noterar att system som härstammar från BSD sannolikt kommer att lägga till ett användarfält.

Använd begreppet i ett beslut

Använd ”Vad är cron?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Daglig skötsel av servrar, e-post, konton och uppdateringar, så att det som alla förlitar sig på förblir odramatiskt.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Drev 10 PostgreSQL- och MS SQL-servrar på Ubuntu — 10 databasservrar som driftsattes och hölls i gott skick

Drev 10 PostgreSQL- och MS SQL-servrar på Ubuntu Hör av er →

Varför det spelar roll

Mycket av underhållet av en server sker medan ingen tittar: gamla filer rensas bort, rapporter skickas med e-post, databaser dumpas och säkerhetskopior tas. cron är standardsättet att säga ”kör det här klockan tre på natten”, och verktyget har funnits i standarden sedan Issue 2, vilket ändringshistoriken på dess POSIX-sida visar. En tabell som håller sig till POSIX-fälten läses likadant på alla system som följer standarden.

Svagheten är inbyggd. Ett schemalagt jobb har ingen publik, så när det går sönder blir ingen skärm röd. POSIX säger att utdata och fel som ett kommando inte själv omdirigerar skickas till användaren med e-post, på ett sätt som varje implementation väljer själv. På en värd där lokal e-post inte når någonstans hamnar felet inte heller någonstans. Det mesta besväret med cron kan härledas till den tystnaden.

Två grannar delar nu på arbetet. På Linux-system som styrs av systemd kan en timerenhet starta samma jobb och lägger till det som cron saknar: den kan ta igen en körning som missades medan maskinen var avstängd, den kan sprida starttiderna slumpmässigt, och den startar aldrig en extra kopia av en tjänst som fortfarande kör. Inne i PostgreSQL läser tillägget `pg_cron` samma tidsscheman med fem fält, och en bakgrundsprocess i databasen håller reda på jobben. Vårt projekt med en jobbkö i River är ett exempel: i oktober 2026 räknar sidan upp 20 `pg_cron`-jobb under kön, som rensar partitioner, roterar salt och uppdaterar aggregat, alltså underhåll som databasen enligt sidan själv är bättre lämpad för.

Så ser det ut i praktiken

Ta två jobb: en städning på vardagar och en nattlig dump av en databas. Tabellen sätter först sin egen miljö, eftersom ett jobb inte ärver inställningarna från ditt inloggningsskal. Raderna med `NAMN=värde` och variabeln `MAILTO` är funktioner i

cronie, inte i POSIX.

```
SHELL=/bin/sh
PATH=/usr/local/bin:/usr/bin:/bin
MAILTO=ops@example.com
# minut timme dag månad veckodag kommando
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"
```

Det första jobbet läses som ”klockan 03.15, måndag till fredag”. Det andra körs varje natt klockan 02.30, skriver sina utdata och fel till en logg och är inlindat i `flock -n`, som ger upp direkt i stället för att vänta när låset redan är taget, så att en dump som drar ut på tiden aldrig får en tvilling. Det tredje visar det escapade procenttecknet: utan omvänt snedstreck skulle cron klippa av kommandot vid `%`.

Håll tabellen i en fil under versionshantering och installera den i sin helhet i stället för att redigera den direkt:

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

Det andra kommandot visar vad cron faktiskt kommer att läsa. Med flaggan `-T` i cronie kan du kontrollera syntaxen i en fil innan du installerar den, och filen måste sluta med en radbrytning.

Samma nattliga dump som systemd-timer kräver två enheter. Tjänsten anger vad som ska köras och timern anger när:

```
# /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` sparar tidpunkten för den senaste körningen på disk och startar en missad körning så snart timern är aktiv igen. `RandomizedDelaySec=` lägger till en slumpmässig fördröjning på upp till tio minuter, så att en flotta av maskiner med samma fil inte träffar databasservern i samma sekund. `systemd-analyze calendar '*-*-* 02:30:00'` visar uttrycket i normaliserad form och när det utlöses nästa gång, och det är det snabbaste sättet att pröva ett schema innan man litar på det.

Ett schema anger när ett jobb startar, inte om det gjorde sitt arbete. På en av våra värdar med 464 MB vänder dödmansgreppet på det: en timer startar ett skript fyra gånger i timmen, varje gång lite förskjutet på måfå, och skriptet pingar en tjänst utanför maskinen bara när varken minnet eller disken har sjunkit under sin gräns. När pingarna uteblir går larmet.

Var det går fel

Kommandot fungerar i ditt skal men inte i cron. POSIX lovar ett jobb bara `HOME`, `LOGNAME`, `PATH` och `SHELL`, och `PATH` behöver bara hitta standardverktygen. Ingenting från din profil läses in. Lösningen är att använda absoluta sökvägar eller sätta `PATH` överst i tabellen, och att testa jobbet i samma sparsamma miljö.

Fel försvinner. Utdata som inte omdirigeras skickas med e-post. cronie skickar dem till tabellens ägare, till adressen i `MAILTO` eller ingenstans alls när `MAILTO=""`. Lösningen är att omdirigera utdata till en logg som någon läser och, för jobb som spelar roll, att låta jobbet rapportera att det lyckats till något som klagar när rapporterna uteblir, så som dödmansgreppet ovan gör.

Ett procenttecken delar kommandot. I kommandofältet blir `%` en radbrytning; bara den första raden körs och resten skickas till den som standardindata, så `date +%s` går sönder. Lösningen är att escapa tecknet som `\%` eller att flytta kommandot till ett skript och anropa det.

Två dagfält betyder antingen–eller, inte båda. När både dag i månaden och veckodag är begränsade körs jobbet varje dag som passar något av dem. POSIX:s eget exempel, `0 0 1,15 * 1`, körs den första och den femtonde i månaden och

dessutom varje måndag. Lösningen är att begränsa bara ett av de två fälten eller att använda en kalender i systemd, där `Mon *-*-01..07 09:00` bara passar när både veckodagen och datumet stämmer: den första måndagen i månaden.

Steg börjar om varje timme. cronie beräknar ett steg bara inom sitt eget fält, så `*/35` i minutfältet körs varje timme vid minut 0 och minut 35, inte med 35 minuters mellanrum, och `*/23` i timfältet körs vid midnatt och klockan 23. Lösningen är att välja ett steg som går jämnt upp i intervallet, eller att använda en systemd-timer med `OnUnitActiveSec=`, som räknar från den senaste gången tjänsten startades.

Klockan ställs om. cronie dokumenterar att tider som försvinner när klockan ställs fram till sommartid aldrig matchar, så de jobben körs inte, och att tider som inträffar två gånger när klockan ställs tillbaka får jobb att köras två gånger. Lösningen är att lägga jobben utanför de timmar då din tidszon ställs om, eller att sätta `CRON_TZ` till en zon utan sommartid.

Den sista raden saknar radbrytning. cronie förväntar sig att varje post slutar med en radbrytning; gör den sista inte det betraktar cronie tabellen som åtminstone delvis trasig och skriver en varning till syslog. Lösningen är att alltid avsluta filen med en radbrytning, något som en inställning i editorn eller en kontroll vid granskningen kan garantera.

Körningar överlappar. crontab-formatet har inget fält för vad som ska hända när den förra körningen fortfarande pågår. En systemd-timer låter en tjänst som fortfarande kör vara i fred. Har timerns nästa tidpunkt passerat medan tjänsten körde, startar timern tjänsten igen så snart den är klar, om inte `DeferReactivation=` är satt; `pg_cron` låter en ny körning av samma jobb vänta tills den första är klar. Lösningen är att i vanlig cron använda ett lås, som `flock -n` i exemplet ovan.

Maskinen var avstängd. En crontab-rad anger ett ögonblick, och formatet kan inte uttrycka ”kör det senare om ögonblicket har passerat”. Lösningen är att, där systemd styr maskinen, sätta `Persistent=true` på en timer med `OnCalendar=`.

Alla maskiner på en gång. En tabell som kopieras till hundra värdar startar hundra jobb i samma minut. Lösningen är att använda `RandomizedDelaySec=` på en systemd-timer, eller i cronie variabeln `RANDOM_DELAY` eller ett intervall med tilde som `6~15`, som väljer en slumpmässig minut när tabellen läses.

Ett `crontab` utan argument tömmer tabellen. Skrivet ensamt väntar kommandot på en ny tabell på standardindata, och ett filslut (oftast Ctrl-D) ersätter den nuvarande med ingenting. Lösningen är att trycka på avbrottstangenten i stället och att hålla tabellen i en fil som du installerar med `crontab fil`.

Läs mer

- crontab, POSIX.1-2024 — standardens sex fält, standardmiljön, regeln för e-post och motiveringen.
- crontab(5), cronie — steg, `MAILTO`, `CRON_TZ`, `RANDOM_DELAY`, sommartid och förbehåll.
- systemd.timer(5) — `OnCalendar=`, `Persistent=`, `RandomizedDelaySec=` och tjänster som kör.
- systemd.time(7) — syntaxen för kalenderhändelser, förkortningarna och hur de valideras.
- flock(1) — ett lås kring ett kommando, med `-n` för att ge upp när låset är taget.
- pg_cron — cron-tidsscheman för SQL-jobb, i PostgreSQL som ett tillägg.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är cron?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/cron/