Engineer

cron

planlæggeren i Unix, der kører kommandoer til faste tider

Hvad det er

cron er den tjeneste i Unix, der starter kommandoer på faste tidspunkter, uden at nogen er logget ind for at starte dem. Hver brugers tidsplan ligger i en crontab: en tabel i ren tekst med ét job pr. linje. Samme ord bruges om kommandoen `crontab`, der installerer, viser og sletter tabellen.

POSIX standardiserer kommandoen og tabellens format, men ikke det program, der læser tabellerne. Standardens begrundelse siger ligeud, at alle henvisninger til en cron-dæmon og til cron-filer er taget ud. Hvilken dæmon der kører jobbene, er op til det enkelte system. De manualsider om cron, som siden bygger på, kommer fra cronie, hvis crontab(5) nævner Paul Vixie som forfatter og beskriver formatet som et V7-lignende format med en række udvidelser, der bevarer kompatibiliteten.

En linje har seks felter adskilt af mellemrum eller tabulatortegn. De fem første siger hvornår: minut (0 til 59), time (0 til 23), dag i måneden (1 til 31), måned (1 til 12) og ugedag (0 til 6, hvor 0 er søndag). Hvert felt er en stjerne for alle værdier, et tal, et interval med bindestreg som `1-5` eller en kommasepareret liste af den slags. Det sjette felt er kommandoen, som `sh` udfører. Dér slutter standarden. Trin som `*/15`, navne på tre bogstaver som `mon`, `7` for søndag, en kolonne med brugernavn i systemtabeller og forkortelser som `@daily` er udvidelser i cronie; en tabel, der bruger dem, er ikke længere portabel.

`crontab -e` åbner din tabel i en editor, `crontab -l` viser den, og `crontab -r` fjerner den, mens `crontab fil` erstatter den med indholdet af en fil. cronie har desuden systemtabeller i `/etc/crontab` og `/etc/cron.d/`, hvor et ekstra felt mellem tidspunkterne og kommandoen angiver, hvilken bruger jobbet kører som. Det havde POSIX forudset: begrundelsen bemærker, at systemer afledt af BSD sandsynligvis vil tilføje et brugerfelt.

Brug begrebet i en beslutning

Brug »Hvad er cron?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Daglig drift af servere, mail, konti og opdateringer, så det, alle er afhængige af, forbliver kedeligt.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har drevet 10 PostgreSQL- og MS SQL-servere — 10 databaseservere udrullet og holdt i god drift

Har drevet 10 PostgreSQL- og MS SQL-servere Kontakt os →

Hvorfor det betyder noget

Meget af vedligeholdelsen af en server foregår, mens ingen kigger: gamle filer ryddes op, rapporter sendes med mail, databaser dumpes, og der tages backup. cron er standardmåden at sige »kør det her klokken tre om natten«, og værktøjet har været med i standarden siden Issue 2, som ændringshistorikken på dens POSIX-side viser. En tabel, der holder sig til POSIX-felterne, læses ens på alle systemer, der overholder standarden.

Svagheden er indbygget. Et planlagt job har intet publikum, så når det går i stykker, bliver ingen skærm rød. POSIX siger, at output og fejl, som en kommando ikke selv omdirigerer, sendes til brugeren med mail på en måde, som hver implementering selv vælger. På en vært, hvor lokal mail ikke ender nogen steder, ender fejlen heller ikke nogen steder. Det meste af besværet med cron kan føres tilbage til den tavshed.

To naboer deler nu arbejdet. På Linux-systemer, der styres af systemd, kan en timer-unit starte det samme job og tilføjer det, cron mangler: den kan indhente en kørsel, der blev sprunget over, mens maskinen var slukket, den kan sprede starttidspunkter tilfældigt, og den starter aldrig en ekstra kopi af en tjeneste, der stadig kører. Inde i PostgreSQL læser udvidelsen `pg_cron` de samme tidsplaner med fem felter, og en baggrundsproces i databasen holder styr på jobbene. Vores projekt med en River-jobkø er et eksempel: i oktober 2026 tæller siden 20 `pg_cron`-jobs under køen, som rydder partitioner op, roterer salts og opdaterer aggregater, altså vedligeholdelse, som siden siger hører hjemme i databasen.

Sådan ser det ud i praksis

Tag to jobs: en oprydning på hverdage og et natligt dump af en database. Tabellen sætter først sit eget miljø, fordi et job ikke arver indstillingerne fra din login-skal. Linjerne med `NAVN=værdi` og variablen `MAILTO` er funktioner i cronie,

ikke i POSIX.

```
SHELL=/bin/sh
PATH=/usr/local/bin:/usr/bin:/bin
MAILTO=ops@example.com
# minut time dag måned ugedag 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 første job læses som »klokken 03.15, mandag til fredag«. Det andet kører hver nat klokken 02.30, skriver sit output og sine fejl til en log og er pakket ind i `flock -n`, der giver op med det samme i stedet for at vente, når låsen allerede er taget, så et dump, der trækker ud, aldrig får en tvilling. Det tredje viser det escapede procenttegn: uden backslash ville cron skære kommandoen over ved `%`.

Hold tabellen i en fil under versionsstyring, og installer den samlet i stedet for at rette i den direkte:

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

Den anden kommando viser det, cron faktisk kommer til at læse. Med cronies tilvalg `-T` kan du tjekke en fils syntaks, før du installerer den, og filen skal slutte med et linjeskift.

Det samme natlige dump som systemd-timer kræver to units. Servicen siger, hvad der skal køres, og timeren siger hvornår:

```
# /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` gemmer tidspunktet for den seneste kørsel på disken og starter en kørsel, der blev sprunget over, så snart timeren er aktiv igen. `RandomizedDelaySec=` lægger en tilfældig forsinkelse på op til ti minutter oveni, så en flåde af maskiner med den samme fil ikke rammer databaseserveren i samme sekund. `systemd-analyze calendar '*-*-* 02:30:00'` viser udtrykket i normaliseret form og tidspunktet for næste udløsning, og det er den hurtigste måde at afprøve en tidsplan på, før man stoler på den.

En tidsplan siger, hvornår et job starter, ikke om det gjorde sit arbejde. På en af vores værter med 464 MB vender vores dead man's switch det om: en timer starter et script fire gange i timen med en tilfældig forskydning, og scriptet pinger kun en tjeneste udenfor, når hverken hukommelse eller disk er faldet under sin grænse. Når pingene udebliver, går alarmen.

Hvor det går galt

Kommandoen virker i din skal, men ikke i cron. POSIX lover kun et job `HOME`, `LOGNAME`, `PATH` og `SHELL`, og `PATH` behøver kun at finde standardværktøjerne. Intet fra din profil bliver indlæst. Løsningen er at bruge absolutte stier eller sætte `PATH` øverst i tabellen og at teste jobbet i det samme sparsomme miljø.

Fejl forsvinder. Output, der ikke omdirigeres, sendes med mail. cronie sender det til tabellens ejer, til adressen i `MAILTO` eller slet ingen steder, når `MAILTO=""`. Løsningen er at omdirigere output til en log, som nogen læser, og for de jobs, der betyder noget, at lade jobbet melde succes til noget, der klager, når meldingerne udebliver, sådan som vores dead man's switch ovenfor gør.

Et procenttegn deler kommandoen. I kommandofeltet bliver `%` til et linjeskift; kun den første linje køres, og resten sendes til den som standardinput, så `date +%s` går i stykker. Løsningen er at escape tegnet som `\%` eller at flytte kommandoen ind i et script og kalde det.

To dagfelter betyder enten-eller, ikke begge. Når både dag i måneden og ugedag er begrænset, kører jobbet på enhver dag, der passer med et af dem. POSIX' eget eksempel, `0 0 1,15 * 1`, kører på månedens første og femtende dag og desuden hver mandag. Løsningen er kun at begrænse det ene af de to felter eller at bruge en kalender i systemd, hvor `Mon *-*-01..07 09:00` kun passer, når både ugedagen og datoen stemmer: den første mandag i måneden.

Trin starter forfra hver time. cronie evaluerer kun et trin inden for sit eget felt, så `*/35` i minutfeltet kører hver time på minut 0 og minut 35, ikke med 35 minutters mellemrum, og `*/23` i timefeltet kører klokken 23 og ved midnat. Løsningen er et trin, der går lige op i intervallet, eller en systemd-timer med `OnUnitActiveSec=`, som tæller fra seneste gang, tjenesten blev startet.

Uret skifter. cronie dokumenterer, at tidspunkter, der forsvinder, når uret stilles om til sommertid, aldrig passer, så de jobs kører ikke, og at tidspunkter, der kommer to gange, når uret stilles tilbage, får jobs til at køre to gange. Løsningen er at lægge jobs uden for de timer, hvor din tidszone skifter, eller at sætte `CRON_TZ` til en zone uden sommertid.

Den sidste linje mangler linjeskift. cronie forventer, at hver post slutter med et linjeskift; gør den sidste ikke det, betragter cronie tabellen som i det mindste delvis ødelagt og skriver en advarsel til syslog. Løsningen er et afsluttende linjeskift, som en indstilling i editoren eller et tjek under review kan sikre.

Kørsler overlapper. crontab-formatet har intet felt for, hvad der skal ske, når den forrige kørsel stadig er i gang. En systemd-timer lader en tjeneste, der stadig kører, være i fred. Er timerens næste tidspunkt passeret, mens tjenesten kørte, starter timeren tjenesten igen, så snart den er færdig, medmindre `DeferReactivation=` er sat; `pg_cron` lader en ny kørsel af samme job vente, til den første er afsluttet. Løsningen for almindelig cron er en lås, som `flock -n` i eksemplet ovenfor.

Maskinen var slukket. En crontab-linje angiver et øjeblik, og formatet kan ikke udtrykke »kør det senere, hvis øjeblikket er gået«. Løsningen er, hvor systemd styrer maskinen, `Persistent=true` på en timer med `OnCalendar=`.

Alle maskiner på én gang. En tabel, der kopieres til hundrede værter, starter hundrede jobs i samme minut. Løsningen er `RandomizedDelaySec=` på en systemd-timer eller i cronie variablen `RANDOM_DELAY` eller et interval med tilde som `6~15`,

der vælger ét tilfældigt minut, når tabellen læses.

En `crontab` uden argument tømmer tabellen. Skrevet alene venter kommandoen på en ny tabel på standardinput, og et filslut (som regel Ctrl-D) erstatter den nuværende med ingenting. Løsningen er at trykke på afbrydelsestasten i stedet og at holde tabellen i en fil, som du installerer med `crontab fil` .

Læs mere

- crontab, POSIX.1-2024 — standardens seks felter, standardmiljøet, reglen for mail og begrundelsen.
- crontab(5), cronie — trin, `MAILTO` , `CRON_TZ` , `RANDOM_DELAY` , sommertid og forbehold.
- systemd.timer(5) — `OnCalendar=` , `Persistent=` , `RandomizedDelaySec=` og tjenester, der kører.
- systemd.time(7) — syntaksen for kalenderhændelser, forkortelserne, og hvordan de valideres.
- flock(1) — en lås om en kommando, med `-n` for at give op, når låsen er taget.
- pg_cron — cron-tidsplaner for SQL-jobs, holdt inde i PostgreSQL som en udvidelse.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er cron?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

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

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/cron/