Engineer

cron

planleggeren i Unix som kjører kommandoer til faste tider

Hva det er

cron er tjenesten i Unix som starter kommandoer til faste tider, uten at noen trenger å være logget inn for å starte dem. Hver brukers tidsplan ligger i en crontab: en tabell i ren tekst med én jobb per linje. Det samme ordet brukes om kommandoen `crontab`, som installerer, viser og sletter tabellen.

POSIX standardiserer kommandoen og formatet på tabellen, men ikke programmet som leser tabellene. Begrunnelsen i standarden sier rett ut at alle henvisninger til en cron-daemon og til cron-filer er tatt ut. Hvilken daemon som kjører jobbene, er opp til hvert enkelt system. Manualsidene om cron som denne siden bygger på, kommer fra cronie, der crontab(5) oppgir Paul Vixie som forfatter og beskriver formatet som et V7-lignende format med en rekke utvidelser som bevarer kompatibiliteten.

En linje har seks felt atskilt med mellomrom eller tabulatorer. De fem første sier når: minutt (0 til 59), time (0 til 23), dag i måneden (1 til 31), måned (1 til 12) og ukedag (0 til 6, der 0 er søndag). Hvert felt er en stjerne for alle verdier, et tall, et intervall med bindestrek som `1-5` eller en kommaseparert liste av slike. Det sjette feltet er kommandoen, som `sh` kjører. Der slutter standarden. Steg som `*/15`, navn på tre bokstaver som `mon`, `7` for søndag, en kolonne med brukernavn i systemtabeller og forkortelser som `@daily` er utvidelser i cronie; en tabell som bruker dem, er ikke lenger portabel.

`crontab -e` åpner tabellen din i en editor, `crontab -l` skriver den ut, og `crontab -r` fjerner den, mens `crontab fil` erstatter den med innholdet i en fil. cronie har i tillegg systemtabeller i `/etc/crontab` og `/etc/cron.d/`, der et ekstra felt mellom tidspunktene og kommandoen angir hvilken bruker jobben kjører som. Det hadde POSIX forutsett: begrunnelsen merker seg at systemer avledet av BSD trolig vil legge til et brukerfelt.

Bruk begrepet i en beslutning

Bruk «Hva er cron?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Daglig ettersyn av servere, e-post, kontoer og oppdateringer, slik at det alle er avhengige av, forblir kjedelig.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Drev 10 PostgreSQL- og MS SQL-servere på Ubuntu — 10 databaseservere produksjonssatt og holdt i god stand

Drev 10 PostgreSQL- og MS SQL-servere på Ubuntu Ta kontakt →

Hvorfor det er viktig

Mye av vedlikeholdet av en server skjer mens ingen ser på: gamle filer ryddes bort, rapporter sendes med e-post, databaser dumpes, og det tas sikkerhetskopier. cron er standardmåten å si «kjør dette klokken tre om natten» på, og verktøyet har vært med i standarden siden Issue 2, slik endringshistorikken på POSIX-siden viser. En tabell som holder seg til POSIX-feltene, leses likt på alle systemer som følger standarden.

Svakheten er innebygd. En planlagt jobb har ikke noe publikum, så når den går i stykker, blir ingen skjerm rød. POSIX sier at utdata og feil som en kommando ikke selv omdirigerer, sendes til brukeren med e-post på en måte hver implementasjon velger selv. På en vert der lokal e-post ikke havner noe sted, havner feilen heller ikke noe sted. Det meste av bryet med cron kan føres tilbake til denne tausheten.

To naboer deler nå arbeidet. På Linux-systemer som styres av systemd, kan en timer-enhet starte den samme jobben og legger til det cron mangler: den kan ta igjen en kjøring som ble hoppet over mens maskinen var slått av, den kan spre starttidene tilfeldig, og den starter aldri en ekstra kopi av en tjeneste som fortsatt kjører. Inne i PostgreSQL leser utvidelsen `pg_cron` de samme tidsplanene med fem felt, og en bakgrunnsprosess i databasen holder oversikt over jobbene. Vårt prosjekt med en River-jobbkø er et eksempel: i oktober 2026 teller siden 20 `pg_cron` -jobber under køen, som rydder i partisjoner, roterer salter og oppdaterer aggregater, altså vedlikehold som databasen ifølge siden gjør best selv.

Slik ser det ut i praksis

Ta to jobber: en opprydding på hverdager og et nattlig dump av en database. Tabellen setter først sitt eget miljø, fordi en jobb ikke arver innstillingene fra innloggingsskallet ditt. Linjene med `NAVN=verdi` og variabelen `MAILTO` er funksjoner i cronie, ikke i POSIX.

```
SHELL=/bin/sh
PATH=/usr/local/bin:/usr/bin:/bin
MAILTO=ops@example.com
# minutt time dag måned ukedag 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"
```

Den første jobben leses som «klokken 03.15, mandag til fredag». Den andre kjører hver natt klokken 02.30, skriver utdata og feil til en logg og er pakket inn i `flock -n`, som gir opp med en gang i stedet for å vente når låsen allerede er tatt, så et dump som drar ut, aldri får en tvilling. Den tredje viser det escapede prosenttegnet: uten omvendt skråstrek ville cron kuttet kommandoen ved `%`.

Hold tabellen i en fil under versjonskontroll, og installer den samlet i stedet for å redigere den direkte:

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

Den andre kommandoen viser det cron faktisk kommer til å lese. Med valget `-T` i cronie kan du sjekke syntaksen i en fil før du installerer den, og filen må slutte med et linjeskift.

Det samme nattlige dumpet som systemd-timer krever to enheter. Tjenesten sier hva som skal kjøres, og timeren sier når:

```
# /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` lagrer tidspunktet for siste kjøring på disk og starter en kjøring som ble hoppet over, så snart timeren er aktiv igjen. `RandomizedDelaySec=` legger på en tilfeldig forsinkelse på opptil ti minutter, så en flåte av maskiner med den samme filen ikke treffer databaseserveren i samme sekund. `systemd-analyze calendar '*-*-* 02:30:00'` viser uttrykket i normalisert form og når det slår til neste gang, og det er den raskeste måten å prøve en tidsplan på før man stoler på den.

En tidsplan sier når en jobb starter, ikke om den gjorde jobben sin. På en av vertene våre med 464 MB snur dødmannsknappen dette på hodet: en timer starter et skript fire ganger i timen med tilfeldig forskyvning, og skriptet pinger en tjeneste utenfor maskinen bare når verken minne eller disk har falt under grensen sin. Når pingene uteblir, går alarmen.

Hvor det går galt

Kommandoen virker i skallet ditt, men ikke i cron. POSIX lover en jobb bare `HOME`, `LOGNAME`, `PATH` og `SHELL`, og `PATH` trenger bare å finne standardverktøyene. Ingenting fra profilen din blir lastet inn. Løsningen er å bruke absolutte stier eller sette `PATH` øverst i tabellen, og å teste jobben i det samme sparsomme miljøet.

Feil forsvinner. Utdata som ikke omdirigeres, sendes med e-post. cronie sender det til eieren av tabellen, til adressen i `MAILTO` eller ingen steder i det hele tatt når `MAILTO=""`. Løsningen er å omdirigere utdata til en logg som noen leser, og for jobber som betyr noe, å la jobben melde fra om suksess til noe som klager når meldingene uteblir, slik dødmannsknappen ovenfor gjør.

Et prosenttegn deler kommandoen. I kommandofeltet blir `%` til et linjeskift; bare den første linjen kjøres, og resten sendes til den som standard inndata, så `date +%s` går i stykker. Løsningen er å escape tegnet som `\%` eller å flytte kommandoen inn i et skript og kalle det.

To dagfelt betyr enten–eller, ikke begge. Når både dag i måneden og ukedag er begrenset, kjører jobben på enhver dag som passer med ett av dem. POSIX' eget eksempel, `0 0 1,15 * 1`, kjører den første og den femtende i måneden og i tillegg hver mandag. Løsningen er å begrense bare ett av de to feltene eller å bruke en kalender i systemd, der `Mon *-*-01..07 09:00` bare passer når både ukedagen og datoen stemmer: den første mandagen i måneden.

Steg begynner på nytt hver time. cronie evaluerer et steg bare innenfor sitt eget felt, så `*/35` i minuttfeltet kjører hver time på minutt 0 og minutt 35, ikke med 35 minutters mellomrom, og `*/23` i timefeltet kjører klokken 23 og ved midnatt. Løsningen er et steg som går opp i intervallet, eller en systemd-timer med `OnUnitActiveSec=`, som teller fra siste gang tjenesten ble startet.

Klokken stilles om. cronie dokumenterer at tidspunkter som forsvinner når klokken stilles frem til sommertid, aldri passer, så de jobbene kjører ikke, og at tidspunkter som kommer to ganger når klokken stilles tilbake, får jobber til å kjøre to ganger. Løsningen er å legge jobber utenfor timene der tidssonen din skifter, eller å sette `CRON_TZ` til en sone uten sommertid.

Den siste linjen mangler linjeskift. cronie forventer at hver oppføring slutter med et linjeskift; gjør ikke den siste det, regner cronie tabellen som i det minste delvis ødelagt og skriver en advarsel til syslog. Løsningen er et avsluttende linjeskift, som en innstilling i editoren eller en sjekk under gjennomgang kan sikre.

Kjøringer overlapper. crontab-formatet har ikke noe felt for hva som skal skje når den forrige kjøringen fortsatt pågår. En systemd-timer lar en tjeneste som fortsatt kjører, være i fred. Har timerens neste tidspunkt passert mens tjenesten kjørte, starter timeren tjenesten igjen så snart den er ferdig, med mindre `DeferReactivation=` er satt; `pg_cron` lar en ny kjøring av samme jobb vente til den første er ferdig. Løsningen for vanlig cron er en lås, som `flock -n` i eksempelet ovenfor.

Maskinen var slått av. En crontab-linje angir et øyeblikk, og formatet kan ikke uttrykke «kjør det senere hvis øyeblikket er passert». Løsningen er, der systemd styrer maskinen, `Persistent=true` på en timer med `OnCalendar=`.

Alle maskiner på én gang. En tabell som kopieres til hundre verter, starter hundre jobber i samme minutt. Løsningen er `RandomizedDelaySec=` på en systemd-timer, eller i cronie variabelen `RANDOM_DELAY` eller et intervall med tilde som `6~15`, som velger ett tilfeldig minutt når tabellen leses.

En `crontab` uten argument tømmer tabellen. Skrevet alene venter kommandoen på en ny tabell på standard inndata, og en filslutt (som regel Ctrl-D) erstatter den nåværende med ingenting. Løsningen er å trykke avbruddstasten i stedet og å holde tabellen i en fil som du installerer med `crontab fil`.

Les mer

- crontab, POSIX.1-2024 — standardens seks felt, standardmiljøet, regelen for e-post og begrunnelsen.
- crontab(5), cronie — steg, `MAILTO`, `CRON_TZ`, `RANDOM_DELAY`, sommertid og forbehold.
- systemd.timer(5) — `OnCalendar=`, `Persistent=`, `RandomizedDelaySec=` og tjenester som kjører.
- systemd.time(7) — syntaksen for kalenderhendelser, forkortelsene og hvordan de valideres.
- flock(1) — en lås rundt en kommando, med `-n` for å gi opp når låsen er tatt.
- pg_cron — cron-tidsplaner for SQL-jobber, holdt inne i PostgreSQL som en utvidelse.

Ordliste

Velg ut fra deres faktiske rammer

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er cron?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/cron/