Engineer

cron

service Unix qui lance des commandes aux heures définies

Ce que c’est

cron est le service Unix qui lance des commandes à des heures définies, sans que personne ait besoin de se connecter pour les démarrer. Le calendrier des tâches de chaque utilisateur est conservé dans une crontab, une table en texte brut avec une tâche par ligne. Le même mot désigne la commande `crontab`, qui installe, affiche et supprime cette table.

POSIX normalise la commande et le format de la table, mais pas le programme qui lit les tables. Sa justification le dit clairement : « Toutes les références à un démon cron et aux fichiers cron ont été omises. » Le choix du démon qui exécute les tâches revient à chaque système. Les pages de manuel de cron utilisées ici proviennent de cronie, dont crontab(5) désigne Paul Vixie comme auteur et décrit un format de style V7, enrichi d’extensions qui restent compatibles.

Une ligne contient six champs séparés par des espaces. Les cinq premiers disent quand : minute (0 à 59), heure (0 à 23), jour du mois (1 à 31), mois (1 à 12) et jour de la semaine (0 à 6, où 0 est dimanche). Chaque champ peut être un astérisque pour toutes les valeurs, un nombre, une plage avec trait d’union comme `1-5` ou une liste de ces éléments séparés par des virgules. Le sixième est la commande, exécutée par `sh`. La norme s’arrête là. Les pas tels que `*/15`, les noms de trois lettres comme `mon`, le `7` pour dimanche, une colonne utilisateur dans les tables système et les raccourcis comme `@daily` sont des extensions de cronie ; une table qui les utilise n’est plus portable.

`crontab -e` ouvre votre table dans un éditeur, `crontab -l` l’affiche et `crontab -r` la supprime, tandis que `crontab file` la remplace par le contenu d’un fichier. cronie conserve aussi des tables système dans `/etc/crontab` et `/etc/cron.d/`, où

un champ supplémentaire entre les champs horaires et la commande indique l'utilisateur sous lequel l'exécuter. POSIX le prévoyait : sa justification note que les systèmes issus de BSD sont susceptibles d'ajouter un champ utilisateur.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que cron ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. L'entretien quotidien des serveurs, de la messagerie, des comptes et des mises à jour, pour que les outils indispensables restent sans histoire.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 10 serveurs PostgreSQL et MS SQL sur Ubuntu — 10 serveurs de bases de données déployés et maintenus

10 serveurs PostgreSQL et MS SQL sur Ubuntu Contactez-nous →

Pourquoi c'est important

Une grande partie de l'entretien d'un serveur se fait sans témoin : suppression d'anciens fichiers, envoi de rapports par courrier électronique, dumps de bases de données, sauvegardes. cron est le moyen standard de dire « exécutez ceci à trois heures du matin », et cet utilitaire figure dans la norme depuis l'Issue 2, comme le rappelle l'historique de sa page POSIX. Une table qui s'en tient aux champs POSIX se lit de la même façon sur tout système conforme.

La faiblesse est inhérente au dispositif. Une tâche planifiée n'a pas de public ; lorsqu'elle échoue, aucun écran ne vire au rouge. POSIX précise que la sortie et les erreurs non redirigées d'une commande sont envoyées par courrier électronique à l'utilisateur selon une méthode choisie par l'implémentation. Sur un hôte où personne ne reçoit le courrier électronique local, les erreurs envoyées ainsi disparaissent aussi. La plupart des problèmes de cron remontent à ce silence.

Deux mécanismes voisins se partagent désormais ce travail. Sur les systèmes Linux gérés par systemd, une unité timer peut lancer la même tâche et ajoute ce qui manque à cron : rattraper une exécution manquée lorsque la machine était éteinte, répartir aléatoirement les démarrages et ne jamais démarrer une seconde copie d'un service encore en cours. Dans PostgreSQL, l'extension `pg_cron` lit les mêmes expressions de planification à cinq champs, et un processus d'arrière-plan de la base assure le suivi des tâches. Notre projet de file de travaux River en est un exemple : en octobre 2026, sa page compte 20 tâches

`pg_cron` sous la file, qui purgent des partitions, renouvellent des sels et actualisent des agrégats, un entretien dont la page indique qu'il relève de la base de données.

À quoi cela ressemble en pratique

Prenez deux tâches : un nettoyage en semaine et un dump nocturne de base de données. La table définit d'abord son environnement, car une tâche n'hérite pas des réglages de votre shell de connexion. Les lignes `NAME=value` et la variable `MAILTO` sont des fonctions de cronie, pas de POSIX.

```
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"
```

La première tâche se lit « à 03:15, du lundi au vendredi ». La deuxième s'exécute chaque nuit à 02:30, écrit sa sortie et ses erreurs dans un journal et est exécutée via `flock -n`, qui abandonne immédiatement au lieu d'attendre si le verrou est déjà détenu ; un dump qui se prolonge n'est donc jamais exécuté en double. La troisième montre le signe de pourcentage échappé : sans la barre oblique inverse, cron couperait la commande à `%`.

Conservez la table dans un fichier versionné et installez-la en entier plutôt que de la modifier en direct :

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

La seconde commande affiche ce que cron lira réellement. L'option `-T` de cronie vérifie la syntaxe d'un fichier avant son installation, et le fichier doit se terminer par un saut de ligne.

Le même dump nocturne, sous forme de timer systemd, demande deux unités. Le service dit quoi exécuter, le timer dit quand :

```
# /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` stocke sur disque l'horodatage de la dernière exécution et lance une exécution manquée dès que le timer est de nouveau actif. `RandomizedDelaySec=` ajoute un retard aléatoire allant jusqu'à dix minutes, pour qu'un parc de machines ayant le même fichier ne sollicite pas le serveur de base de données à la même seconde. `systemd-analyze calendar '*-*-* 02:30:00'` affiche l'expression normalisée et la prochaine échéance, le moyen le plus rapide de tester une planification avant de s'y fier.

Une planification indique quand une tâche démarre, pas si elle a fait son travail. Sur l'un de nos hôtes de 464 MB, le mécanisme de l'homme mort inverse cette logique : un timer lance un script quatre fois par heure, avec une variation aléatoire, et le script ne contacte un service externe que si ni la mémoire disponible ni l'espace disque libre ne sont passés sous leur seuil. Lorsque les signaux s'arrêtent, l'alarme se déclenche.

Où les problèmes surviennent

La commande fonctionne dans votre shell, mais pas dans cron. POSIX ne promet à une tâche que `HOME`, `LOGNAME`, `PATH` et `SHELL`, et ce `PATH` doit seulement permettre de trouver les utilitaires standard. Rien de votre profil n'est chargé. Utilisez des chemins absolus ou définissez `PATH` en tête de table, puis testez la tâche avec ce même environnement minimal.

Les erreurs disparaissent. La sortie non redirigée est envoyée par courrier électronique. cronie l'adresse au propriétaire de la table, à l'adresse dans `MAILTO`, ou nulle part lorsque `MAILTO=""`. Redirigez la sortie vers un journal que

quelqu'un lit et, pour les tâches importantes, faites signaler leur réussite à un système qui se plaint lorsque les signaux cessent, comme le mécanisme de l'homme mort ci-dessus.

Un signe de pourcentage coupe la commande. Dans le champ de commande, `%` devient un saut de ligne ; seule la première ligne s'exécute, et le reste lui est fourni comme entrée standard, ce qui casse `date +%s` . Échappez-le sous la forme `\%` , ou déplacez la commande dans un script que vous appelez.

Les deux champs de jour signifient « ou », pas « et ». Quand le jour du mois et celui de la semaine sont tous deux restreints, toute journée correspondant à au moins l'un des deux déclenche la tâche. L'exemple de POSIX, `0 0 1,15 * 1` , se déclenche le 1er, le 15 et tous les lundis. Ne restreignez qu'un seul des deux champs ou utilisez un calendrier systemd, où `Mon *-*-01..07 09:00` ne correspond que si le jour de semaine et la date conviennent tous deux : le premier lundi du mois.

Les pas recommencent à chaque heure. cronie évalue un pas uniquement dans son propre champ ; `*/35` dans les minutes s'exécute donc à :00 et :35, pas toutes les 35 minutes, et `*/23` dans les heures s'exécute à minuit et à 23:00. Choisissez un pas qui divise la plage sans reste, ou un timer systemd avec `OnUnitActiveSec=` , qui compte depuis le dernier démarrage du service.

Les horloges changent. cronie précise que les heures sautées lors d'un changement d'heure ne correspondent jamais et que ces tâches ne s'exécutent donc pas ; les heures répétées les font s'exécuter deux fois. Planifiez en dehors des heures de changement de votre fuseau, ou définissez `CRON_TZ` sur un fuseau sans heure d'été.

La dernière ligne n'a pas de saut de ligne final. cronie exige que chaque entrée se termine par un saut de ligne ; sinon, il considère la table comme au moins partiellement défectueuse et écrit un avertissement dans syslog. Ajoutez le saut final, ce qu'un réglage d'éditeur ou une vérification en revue peut garantir.

Les exécutions se chevauchent. Le format crontab ne prévoit aucun champ indiquant quoi faire si l'exécution précédente se poursuit. Un timer systemd ne touche pas à un service encore en cours. Si sa prochaine échéance est passée pendant l'exécution, il redémarre le service dès que celui-ci se termine, sauf si

`DeferReactivation=` est défini ; `pg_cron` fait attendre la seconde instance jusqu'à la fin de la première. Pour cron seul, utilisez un verrou, comme le `flock -n` de l'exemple.

La machine était éteinte. Une ligne de crontab désigne un instant ; le format n'a aucun moyen de dire « exécutez plus tard si cet instant est passé ». Lorsque systemd gère la machine, utilisez `Persistent=true` sur un timer `OnCalendar=` .

Toutes les machines démarrent à la fois. Une table copiée sur cent hôtes démarre cent tâches dans la même minute. Utilisez `RandomizedDelaySec=` sur un timer systemd ou, dans cronie, la variable `RANDOM_DELAY` ou une plage avec tilde telle que `6~15` , qui choisit une minute aléatoire à la lecture de la table.

Un `crontab` sans argument efface la table. Sans argument, la commande attend une nouvelle table sur l'entrée standard, et une fin de fichier, généralement Ctrl-D, remplace la table actuelle par une table vide. Appuyez plutôt sur la touche d'interruption et gardez la table dans un fichier installé avec `crontab file` .

Pour aller plus loin

- crontab, POSIX.1-2024 — les six champs de la norme, l'environnement initial de la tâche, la règle d'envoi par courrier électronique et la justification.
- crontab(5), cronie — pas, `MAILTO` , `CRON_TZ` , `RANDOM_DELAY` , heure d'été et réserves d'utilisation.
- systemd.timer(5) — `OnCalendar=` , `Persistent=` , `RandomizedDelaySec=` et services en cours d'exécution.
- systemd.time(7) — syntaxe des événements calendaires, raccourcis et validation.
- flock(1) — exécuter une commande sous verrou, avec `-n` pour abandonner lorsqu'il est détenu.
- pg_cron — calendriers cron pour les tâches SQL, conservés dans PostgreSQL sous forme d'extension.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que cron ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/cron/