Engineer

cron

☑ 指定した時刻に、利用者のログインなしでコマンドを起動するUnixのサービス

どのようなものか

cronは、起動する人がログインしていなくても、決めた時刻にコマンドを開始するUnixのサービスです。利用者ごとの予定は、crontabに保存します。一行に一つのジョブを書く、プレーンテキストの表です。同じ名前の `crontab` コマンドで、その表の登録、表示、削除を行います。

POSIXが標準化しているのは、コマンドと表の形式であり、表を読むプログラムではありません。その根拠説明にも、「cronデーモンとcronファイルへの言及は、すべて省いた」と明記しています。どのデーモンがジョブを実行するかは、各システムに委ねています。このページが参照するcronのマニュアルはcronieのものです。そのcrontab(5)は、著者としてPaul Vixieを挙げ、互換性を保つ拡張を加えたV7形式と説明しています。

一行は、空白で区切る六つのフィールドからなります。最初の五つが時刻です。分（0から59）、時（0から23）、日（1から31）、月（1から12）、曜日（0から6で、0は日曜日）を表します。各フィールドには、すべての値を表すアスタリスク、数値、 `1-5` のようなハイフンで区切る範囲、または、それらをコンマで並べた一覧を書きます。六つ目は、 `sh` が実行するコマンドです。標準が定めるのは、ここまでです。 `*/15` のような間隔の指定、 `mon` のような三文字の名前、日曜日を表す `7` 、システム用の表の利用者名の列、 `@daily` のような省略形は、cronieの拡張です。それらを使う表は、ほかのシステムへそのまま移せるとは限りません。

`crontab -e` は表をエディターで開き、 `crontab -l` は表示し、 `crontab -r` は削除します。 `crontab file` は、ファイルの内容で表を置き換えます。cronieは、システム用の表を `/etc/crontab` と `/etc/cron.d/` にも保持します。そこでは、時刻とコマンドの間に、実行する利用者を指定するフィールドを追加します。POSIXも、そのような実装を想定していました。根拠説明は、BSD由来のシステムでは、利用者のフィールドを追加する可能性が高いと述べています。

概念を判断に生かす

「cronとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 サーバー、メール、アカウント、更新を日々管理し、誰もが頼る仕組みを平穏に動かし続けます。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 UbuntuでPostgreSQL・MS SQLサーバー10台を運用 — 10 導入し正常に保ったDBサーバー

UbuntuでPostgreSQL・MS SQLサーバー10台を運用 お問い合わせ →

なぜ重要なのか

サーバーの保守の多くは、誰も見ていない間に行います。古いファイルの削除、レポートのメール送信、データベースのダンプ、バックアップです。cronは、「これを午前三時に実行する」と指定する標準的な方法です。POSIXのページの変更履歴によると、このユーティリティーはIssue 2から標準に含まれています。POSIXのフィールドだけを使う表なら、準拠するどのシステムでも、同じ意味で読めます。

弱点も、仕組みそのものにあります。定期実行するジョブを見届ける人はいないため、不具合が起きても画面が赤くなるわけではありません。POSIXは、コマンドがリダイレクトしなかった出力とエラーを、各実装の選ぶ方法で利用者へメール送信すると定めています。ローカルメールがどこにも届かないホストでは、エラーの通知も届かないままです。cronの問題の多くは、この沈黙に行き着きます。

現在は、近い役割を持つ二つの仕組みも、この仕事を担います。systemdで動くLinuxシステムでは、タイマーユニットで同じジョブを開始でき、cronにない機能も加わります。電源が切れていた間の未実行分を後から実行でき、開始時刻をランダムに分散でき、実行中のサービスを二重に起動することもありません。PostgreSQL内では、`pg_cron` 拡張が、同じ五フィールドの予定を読み、データベース内のバックグラウンドワーカーがジョブを管理します。当社のRiverジョブキューのプロジェクトが一例です。そのページは、2026年10月時点で、キューの下に20個の `pg_cron` ジョブがあると記しています。パーティションの削除、ソルトの更新、集計値の再計算を行い、ページがデータベースの責務とする保守を担います。

実際の活用例

平日のファイル整理と、毎晩のデータベースのダンプという二つのジョブを考えます。ジョブはログインシェルの設定を引き継がないため、表の先頭で実行環境を設定します。`NAME=value` の行と `MAILTO` 変数は、POSIXではなくcronieの機能です。

```
SHELL=/bin/sh
PATH=/usr/local/bin:/usr/bin:/bin
MAILTO=ops@example.com
# 分 時 日 月 曜日 コマンド
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"
```

最初のジョブは、「月曜日から金曜日の03:15」と読みます。二つ目は毎晩02:30に実行し、出力とエラーをログへ書きます。全体を `flock -n` で囲んでいるため、ロックが既に取られていれば、待たずにすぐ終了します。ダンプに時間がかかっても、同じ処理が重複して動くことはありません。三つ目は、エスケープしたパーセント記号の例です。バックスラッシュがなければ、cronは `%` の位置でコマンドを区切ります。

現在登録されている表を直接編集するのではなく、バージョン管理するファイルに保存し、全体を登録します。

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

二つ目のコマンドは、cronが実際に読むものを表示します。cronieの `-T` オプションでは、登録前にファイルの構文を検査できます。ファイルの末尾には、改行が必要です。

同じ毎晩のダンプをsystemdタイマーで行うには、二つのユニットを使います。サービスが実行する内容を、タイマーが時刻を定めます。

```
# /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` は、最後の実行時刻をディスクに保存し、タイマーが再開するとすぐに未実行分を開始します。 `RandomizedDelaySec=` は、最大で10分のランダムな遅延を加えます。同じファイルを配った多数のマシンが、同じ秒にデータベースサーバーへアクセスしないようにするためです。 `systemd-analyze calendar '*-*-* 02:30:00'` は、正規化した式と次の発火時刻を表示します。予定を信頼する前に確かめる、最も手早い方法です。

予定が示すのは、ジョブがいつ始まるかであり、仕事を果たしたかではありません。当社の464 MBのホストでは、デッドマンズスイッチで、これを逆に捉えています。タイマーは、一時間に四回、時刻に揺らぎを加えてスクリプトを開始します。スクリプトが外部サービスへ生存通知を送るのは、メモリーもディスクも、残量がしきい値を下回っていない場合だけです。通知が途絶えると、警報が鳴ります。

問題が起きるところ

シェルでは動くコマンドが、cronでは動きません。 POSIXがジョブに保証する環境変数は、 `HOME` 、 `LOGNAME` 、 `PATH` 、 `SHELL` だけです。 `PATH` も、標準のユーティリティーを見つけられれば十分です。プロファイルの設定は、一切読み込みません。絶対パスを使うか、表の先頭で `PATH` を設定し、同じ最小限の環境でジョブを試します。

エラーが消えます。 リダイレクトしない出力は、メールで送ります。cronieでは、表の所有者か、 `MAILTO` で指定したアドレスへ送信し、 `MAILTO=""` なら、どこにも送りません。誰かが読むログへ出力をリダイレクトします。重要なジョブでは、先ほどのデッドマンズスイッチのように、成功を報告し、その報告が止まると異常を知らせる仕組みも使います。

パーセント記号がコマンドを分断します。 コマンドのフィールドでは、 `%` は改行となります。最初の行だけを実行し、残りは標準入力としてそのコマンドへ渡すため、 `date +%s` は壊れます。 `\%` とエスケープするか、コマンドをスクリプ

トへ移して呼び出します。

日と曜日は、両方に一致する必要はなく、どちらかに一致すれば実行します。 日と曜日の両方を制限すると、どちらかに一致する日にジョブを実行します。POSIX自身の例である `0 0 1,15 * 1` は、1日、15日、毎週月曜日に動きます。制限するのを片方だけにするか、systemdのカレンダーを使います。`Mon *-*-01..07 09:00` なら、曜日と日付の両方が合う場合、つまり、その月の最初の月曜日だけに一致します。

間隔の指定は、毎時、先頭から始まります。 cronieは、間隔を、そのフィールドの範囲内だけで評価します。そのため、分の `*/35` は、35分おきではなく、:00と:35に動きます。時の `*/23` は、午前零時と23:00に動きます。範囲を等分できる間隔にするか、最後にサービスを開始した時点から数える、systemdタイマーの `OnUnitActiveSec=` を使います。

時計が切り替わります。 cronieの説明では、夏時間への切り替えで飛ばされた時刻は、一致しないため、ジョブを実行しません。同じ時刻が繰り返されれば、二回実行します。使うタイムゾーンで時刻が切り替わる時間帯を避けるか、`CRON_TZ` を夏時間のない地域に設定します。

最後の行に改行がありません。 cronieは、すべての項目の末尾に改行を求めます。最後の行にないと、表の少なくとも一部が壊れていると扱い、syslogに警告を書きます。対策は、末尾の改行です。エディターの設定や、レビュー時の検査で保証できます。

実行が重複します。 crontabの形式には、前の実行が続いている場合の振る舞いを指定するフィールドがありません。systemdタイマーは、実行中のサービスには手を加えません。サービスの実行中に次の予定時刻が過ぎていれば、`DeferReactivation=` を設定しない限り、完了した瞬間に再びサービスを開始します。`pg_cron` では、最初の実行が終わるまで、二つ目の実行を待たせます。通常のcronでは、先ほどの例の `flock -n` のように、ロックを使います。

マシンの電源が切れていました。 crontabの行が指定するのは、一つの時点です。「その時点を過ぎていたら、後で実行する」と指定する方法はありません。systemdで動くマシンなら、`OnCalendar=` タイマーに `Persistent=true` を設定します。

すべてのマシンが、一斉に動きます。 百台のホストに同じ表を配ると、同じ分に百個のジョブが始まります。systemdタイマーでは `RandomizedDelaySec=` を、cronieでは `RANDOM_DELAY` 変数か、 `6~15` のようなチルダによる範囲を使います。後者は、表を読んだ時点で、範囲内から一つの分をランダムに選びます。

引数のない `crontab` が、表を消します。 引数を付けずに実行すると、コマンドは標準入力から新しい表を待ちます。入力の終わり、通常はCtrl-Dを送ると、現在の表を空の内容で置き換えます。代わりに割り込みキーを押します。表は、`crontab file` で登録するファイルに保持します。

参考資料

- crontab、POSIX.1-2024 — 標準の六フィールド、既定の環境、メールの規則、根拠説明を示します。
- crontab(5)、cronie — 間隔の指定、 `MAILTO` 、 `CRON_TZ` 、 `RANDOM_DELAY` 、夏時間、注意点を説明します。
- systemd.timer(5) — `OnCalendar=` 、 `Persistent=` 、 `RandomizedDelaySec=` 、実行中のサービスを説明します。
- systemd.time(7) — カレンダーイベントの構文、省略形、検証方法を説明します。
- flock(1) — コマンド全体にロックを掛け、取得済みなら `-n` で終了する方法を説明します。
- pg_cron — PostgreSQLの拡張としてデータベース内に保持する、SQLジョブ用のcron形式の予定です。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「cronとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/cron/