**Engineer**

# cron

ejecutar comandos Unix según una programación horaria

## Qué es

cron es el servicio de Unix que inicia comandos a horas determinadas, sin que nadie tenga que estar conectado para ejecutarlos. La programación de cada usuario se guarda en un crontab: una tabla de texto plano con una tarea por línea. La misma palabra da nombre al comando `crontab`, que instala, muestra y elimina esa tabla.

POSIX estandariza el comando y el formato de la tabla, pero no el programa que la lee. Su justificación lo dice claramente: «Se han omitido todas las referencias a un demonio cron y a archivos cron». Cada sistema decide qué demonio ejecuta las tareas. Las páginas del manual de cron utilizadas aquí proceden de cronie, cuyo crontab(5) identifica a Paul Vixie como autor y describe el formato como uno de estilo V7 con extensiones que mantienen la compatibilidad.

Una línea tiene seis campos separados por espacios en blanco. Los cinco primeros indican cuándo: minuto (0 a 59), hora (0 a 23), día del mes (1 a 31), mes (1 a 12) y día de la semana (0 a 6, donde 0 es domingo). Cada campo puede ser un asterisco para todos los valores, un número, un intervalo con guion como `1-5` o una lista de estos separados por comas. El sexto campo es el comando, que ejecuta `sh`. Ahí termina el estándar. Los pasos como `*/15`, los nombres de tres letras como `mon`, el `7` para el domingo, la columna de nombre de usuario en las tablas del sistema y las abreviaturas como `@daily` son extensiones de cronie; una tabla que las utilice deja de ser portable.

`crontab -e` abre la tabla en un editor, `crontab -l` la muestra y `crontab -r` la elimina, mientras que `crontab file` la sustituye por el contenido de un archivo. cronie también guarda tablas del sistema en `/etc/crontab` y `/etc/cron.d/`,

donde un campo adicional entre los tiempos y el comando indica con qué usuario se ejecuta. POSIX ya lo contemplaba: su justificación señala que es probable que los sistemas derivados de BSD añadan un campo de usuario.

Utilice el concepto en una decisión

Aplique «¿Qué es cron?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. El cuidado diario de servidores, correo, cuentas y actualizaciones para que los servicios de los que depende todo el mundo sigan funcionando sin sorpresas.

**Explore nuestros servicios** Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. 10 servidores PostgreSQL y MS SQL en Ubuntu — 10 servidores de bases de datos desplegados y mantenidos

**10 servidores PostgreSQL y MS SQL en Ubuntu** Póngase en contacto →

## Por qué importa

Gran parte del mantenimiento de un servidor ocurre sin que nadie lo observe: se borran archivos antiguos, se envían informes por correo, se vuelcan bases de datos y se hacen copias de seguridad. cron es la forma estándar de decir «ejecute esto a las tres de la mañana», y la utilidad forma parte del estándar desde la edición 2, como recoge el historial de cambios de su página en POSIX. Una tabla que se limite a los campos de POSIX se interpreta igual en cualquier sistema conforme al estándar.

La debilidad es inherente. Una tarea programada no tiene público, así que, cuando falla, ninguna pantalla se pone roja. POSIX establece que la salida y los errores que un comando no redirija se envían por correo al usuario mediante un método que elige cada implementación. En un host cuyo correo local no llega a ninguna parte, el error tampoco. La mayoría de los problemas con cron se remontan a ese silencio.

Dos alternativas cercanas comparten ahora ese trabajo. En sistemas Linux administrados por systemd, una unidad de temporizador puede iniciar la misma tarea y añadir lo que le falta a cron: recuperar una ejecución perdida mientras la máquina estaba apagada, distribuir aleatoriamente las horas de inicio y no iniciar nunca una segunda copia de un servicio que sigue ejecutándose. Dentro de PostgreSQL, la extensión `pg_cron` interpreta los mismos horarios de cinco campos, y un proceso en segundo plano de la base de datos lleva el control de las tareas. El proyecto de cola de tareas con River ofrece un ejemplo: en octubre de

2026, su página cuenta 20 tareas de `pg_cron` por debajo de la cola, que eliminan particiones antiguas, rotan sales y actualizan agregados, labores de mantenimiento que, según la página, corresponden a la base de datos.

## Cómo se aplica en la práctica

Considere dos tareas: una limpieza entre semana y un volcado nocturno de la base de datos. La tabla define primero su propio entorno, porque una tarea no hereda la configuración del intérprete de comandos de su sesión. Las líneas `NAME=value` y la variable `MAILTO` son funciones de cronie, no 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 primera tarea se lee «a las 03:15, de lunes a viernes». La segunda se ejecuta cada noche a las 02:30, escribe la salida y los errores en un registro y está envuelta en `flock -n`, que desiste de inmediato en vez de esperar si el bloqueo ya está ocupado, de modo que un volcado largo nunca tiene un duplicado simultáneo. La tercera muestra el signo de porcentaje escapado: sin la barra inversa, cron cortaría el comando en `%`.

Guarde la tabla en un archivo bajo control de versiones e instálela completa, en lugar de editar directamente la tabla instalada:

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

El segundo comando muestra lo que cron va a leer realmente. La opción `-T` de cronie comprueba la sintaxis de un archivo antes de instalarlo, y el archivo debe terminar con un salto de línea.

El mismo volcado nocturno como temporizador de systemd requiere dos unidades. El servicio indica qué ejecutar; el temporizador, cuándo:

```
# /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` guarda en disco la hora de la última ejecución e inicia una ejecución perdida en cuanto vuelve el temporizador. `RandomizedDelaySec=` añade un retraso aleatorio de hasta diez minutos, para que una flota de máquinas con el mismo archivo no llegue al servidor de base de datos en el mismo segundo. `systemd-analyze calendar '*-*-* 02:30:00'` muestra la expresión normalizada y cuándo se activará la próxima vez, la forma más rápida de probar una programación antes de confiar en ella.

Una programación dice cuándo empieza una tarea, no si ha hecho su trabajo. En un host de la empresa con 464 MB, el mecanismo de alerta por ausencia de señales invierte ese planteamiento: un temporizador inicia un script cuatro veces por hora, con una variación aleatoria, y el script solo envía una señal a un servicio externo cuando ni la memoria disponible ni el espacio libre en disco han caído por debajo de su umbral. Cuando cesan las señales, se activa la alarma.

## Dónde surgen los problemas

**El comando funciona en su intérprete de comandos, pero no en cron.** POSIX solo garantiza a una tarea `HOME`, `LOGNAME`, `PATH` y `SHELL`, y basta con que `PATH` permita encontrar las utilidades estándar. No se carga nada de su perfil. La solución es utilizar rutas absolutas o definir `PATH` al principio de la tabla y probar la tarea con ese mismo entorno mínimo.

**Los errores desaparecen.** La salida no redirigida se envía por correo. cronie la manda al propietario de la tabla, a la dirección de `MAILTO` o a ninguna parte cuando `MAILTO=""`. La solución es redirigir la salida a un registro que alguien lea

y, para las tareas importantes, hacer que notifiquen el éxito a un sistema que avise cuando dejen de llegar las notificaciones, como el mecanismo de alerta por ausencia de señales descrito antes.

**Un signo de porcentaje divide el comando.** En el campo del comando, `%` se convierte en un salto de línea; solo se ejecuta la primera línea y el resto se le pasa como entrada estándar, por lo que `date +%s` falla. La solución es escaparlo como `\%` o mover el comando a un script y llamarlo.

**Los dos campos de día se combinan con «o», no con «y».** Cuando están restringidos tanto el día del mes como el día de la semana, la tarea se ejecuta si coincide cualquiera de los dos. El propio ejemplo de POSIX, `0 0 1,15 * 1`, se activa el día 1, el 15 y todos los lunes. La solución es restringir solo uno de los dos o usar un calendario de systemd, donde `Mon *-*-01..07 09:00` solo coincide cuando encajan tanto el día de la semana como la fecha: el primer lunes del mes.

**Los pasos vuelven a empezar cada hora.** cronie evalúa un paso únicamente dentro de su propio campo, de modo que `*/35` en los minutos se ejecuta a los minutos :00 y :35, no cada 35 minutos, y `*/23` en las horas se ejecuta a medianoche y a las 23:00. La solución es usar un paso que divida el intervalo sin dejar resto o un temporizador de systemd con `OnUnitActiveSec=`, que cuenta desde la última vez que se inició el servicio.

**Los relojes cambian.** cronie documenta que las horas omitidas por el cambio al horario de verano nunca coinciden, por lo que esas tareas no se ejecutan, y que las horas repetidas hacen que se ejecuten dos veces. La solución es programarlas fuera de las horas en que cambia su zona horaria o definir `CRON_TZ` con una zona sin horario de verano.

**La última línea no tiene salto de línea final.** cronie requiere que cada entrada termine con un salto de línea; cuando la última no lo tiene, considera la tabla al menos parcialmente defectuosa y escribe una advertencia en syslog. La solución es un salto de línea final, que puede garantizarse con una opción del editor o una comprobación en la revisión.

**Las ejecuciones se solapan.** El formato crontab no tiene ningún campo que indique qué debe ocurrir si la ejecución anterior sigue en curso. Un temporizador de systemd no inicia otra instancia de un servicio que ya se está ejecutando. Si pasa la siguiente hora programada mientras el servicio aún está en marcha, el

temporizador vuelve a iniciarlo en cuanto termina, salvo que se defina `DeferReactivation=` ; `pg_cron` hace que una segunda instancia espere a que termine la primera. La solución para cron sin añadidos es un bloqueo, como `flock -n` en el ejemplo anterior.

**La máquina estaba apagada.** Una línea de crontab señala un instante, y el formato no ofrece una forma de decir «ejecútelo después si ese instante ya pasó». La solución, cuando systemd administra la máquina, es `Persistent=true` en un temporizador `OnCalendar=` .

**Todas las máquinas a la vez.** Una tabla copiada a cien hosts inicia cien tareas en el mismo minuto. La solución es `RandomizedDelaySec=` en un temporizador de systemd o, en cronie, la variable `RANDOM_DELAY` o un intervalo con virgulilla como `6~15` , que elige un minuto aleatorio al leer la tabla.

**Un `crontab` sin argumentos borra la tabla.** Si se escribe sin argumentos, el comando espera una tabla nueva en la entrada estándar, y un fin de archivo (normalmente Ctrl-D) sustituye la actual por una vacía. La solución es pulsar la tecla de interrupción en su lugar y guardar la tabla en un archivo que se instala con `crontab file` .

## Lecturas adicionales

- crontab, POSIX.1-2024 — los seis campos del estándar, el entorno predeterminado, la regla del correo y la justificación.
- crontab(5), cronie — pasos, `MAILTO` , `CRON_TZ` , `RANDOM_DELAY` , horario de verano y advertencias.
- systemd.timer(5) — `OnCalendar=` , `Persistent=` , `RandomizedDelaySec=` y servicios en ejecución.
- systemd.time(7) — la sintaxis de eventos de calendario, sus abreviaturas y cómo validarla.
- flock(1) — ejecutar un comando bajo un bloqueo, con `-n` para desistir si ya está ocupado.
- pg_cron — programaciones cron para tareas SQL, conservadas dentro de PostgreSQL como una extensión.

Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

**Vea cuánto cuesta** Póngase en contacto →

## Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es cron?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

**Explore nuestros servicios**

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

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por **D.**

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

**Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.**

**Impreso desde https://engineer.company/es/glossary/cron/**