



Ejecutamos tareas programadas con Fcron

TRABAJO EN HORA

Orlando Rosu, 123RF

Fcron hace todo lo que Cron puede hacer y además ofrece un par de trucos adicionales para evitar el tiempo de inactividad. **POR MATIJA SUKLJE.**

Los administradores de Linux y muchos otros usuarios están bastante familiarizados con la utilidad Cron, un programador de trabajos basado en texto que automatiza la ejecución de tareas periódicas. Si usted no está familiarizado con Cron, encontrará varios debates online [1].

Cron lleva mucho tiempo por ahí, y ha pasado por muchas fases de evolución, pero la mayoría de las implementaciones modernas (Vixie Cron/ISC Cron, BCron, DCron, ... aún se basan en la idea de que su sistema está ejecutándose 24/7. Desafortunadamente, a menos que se sea un servidor dedicado, es probable que *no* esté encendido 24/7. Los usuarios suspenden y apagan los portátiles para llevarlos consigo, e incluso nuestros equipos de sobremesa y estaciones de trabajo normalmente pasan algún tiempo de inactividad para ahorrar energía. El resultado es que las tareas cron fallan regularmente, ya que el sistema no se encuentra en funcionamiento en el preciso momento en el que se supone que se va a ejecutar la tarea.

Para trabajar sobre este problema, algunas distribuciones ofrecen *Anacron* como una alternativa a Cron. Anacron le permite

crear una lista de tareas que tienen que ocurrir en intervalos predefinidos, y cuando arranca, la comprueba y realiza cualquier tarea que no se haya realizado. Anacron, sin embargo, tiene algunas limitaciones. Primero, que no es un servicio, debe ser ejecutado cada vez que se necesite – ya sea manualmente o mediante un script de arranque, o con Cron. Segundo, no está diseñado para manejar períodos de tiempo inferiores a un día. La combinación de estos problemas puede llevar a escenarios en los que ambos, Cron y Anacron, están actuando al mismo tiempo, y la misma tarea se ejecuta dos veces o ninguna.

Fcron [2], por otro lado, es una alternativa a Cron que hace lo que Vixie Cron y Anacron pueden hacer, además de algo más. Puede utilizar Fcron para planificar las tareas cron fijando el tiempo y la hora, intervalos de tiempo o incluso disponibilidad del sistema.

Cómo Empezar

Fcron está disponible en forma de paquete en la mayoría de las distribuciones GNU/Linux modernas – todo lo que tiene que hacer es encontrar el paquete (normalmente llamado “fcron”) e instalarlo con el gestor de paquetes de su distribución.

Para crear un fichero *fcrontab*, ejecute el siguiente comando en su terminal:

```
fcrontab -e
```

Para editar su fichero *fcrontab*, introduzca lo siguiente (como root):

```
fcrontab -e systab
```

Cuando su editor de texto por defecto lo abra, puede comenzar a añadir entradas al fichero *fcrontab*.

Tenga en cuenta que para permitir las características adicionales de Fcron, la sintaxis es un poco diferente de la sintaxis utilizada por Vixie Cron.

Tareas Cron Definidas con una Fecha y Hora Fijas

Para empezar, comenzaré con algo que cualquier Cron puede hacer – un simple trabajo que realiza una copia de seguridad de un documento cada día laborable a las 3am:

```
& 00 03 * * mon-fri cp 🚩
~/Documents/Importaint.odt 🚩
/mnt/backup/Importaint.odt~
```

Cuando guarde y salga de su editor, Fcron guardará automáticamente esta entrada. Pero antes de esto, comprobará la sintaxis de todas las tareas cron y le mostrará si hay que corregir algún error.

El símbolo ampersand (&) al comienzo de la línea le indica a Fcron que se trata de una tarea cron con una fecha y hora fijadas. Tras esto, encontrará dos campos de hora y tres campos de fecha: minutos, horas, día del mes, mes y día de la semana. El ejemplo anterior especifica la siguiente fecha: 00 minutos, 03 horas, cada día del mes, cada mes y de lunes a viernes. Al final de la línea se encuentra el comando que quiere ejecutar en la fecha especificada.

Cuando se especifican tareas cron con tiempos fijados, debe rellenar los cinco campos de fecha y hora; en el caso de que no quiera poner restricciones en algún campo, debería utilizar el asterisco (*, significa "todo"). En el ejemplo anterior, no importaba qué mes o día del mes fuera, siempre y cuando se tratara de un día de lunes a viernes.

Por comodidad, puede utilizar tres letras abreviadas del nombre del mes y del día de la semana en vez de números. También puede utilizar listas (separadas por comas), rangos (conectados con un guión), saltarse rangos (precedidos por una barra) y excepciones en rangos (precedidos por una tilde).

Considérese la siguiente tarea cron, la cual le molesta con una notificación cada 20 minutos desde las 9 de la mañana hasta las 4 de la tarde cada día, desde el día 10 hasta el 25 de Marzo con un aviso, con la excepción del día 15 (porque ese día lo tiene libre).

```
& 00,20,40 9-16 10-25~15 mar * >
echo "¿Ha terminado ya? >
¿Ha terminado ya?"
```

Otro ejemplo de la vida real sería si desea ejecutar un script en su directorio home cualquier otro día a las 4 de la tarde y la salida tiene que enviarse por correo electrónico a su jefe automáticamente:

```
&mailto(boss@office.com) >
* * */2 * * ~/script.sh
```

En este ejemplo también puede ver una de las opciones que puede configurar para las tareas cron con Fcron. Al final de este artículo se describen más detalles sobre estas opciones.

Hasta aquí, probablemente no está lo bastante impresionado como para cambiar el Cron que viene por defecto en su distribución. Pero Fcron ofrece algunas características adicionales que lo hacen mejor que un Cron ordinario.

Tareas Cron Definidas por Intervalos

Lo más probable es que, a menos que esté ejecutando un servidor, su sistema no esté 24/7 funcionando, de modo que los trabajos programados en una fecha y hora fijadas darán lugar a muchas tareas no realizadas.

Afortunadamente, Fcron tiene una característica que le permite que las entradas se ejecuten en intervalos de tiempo predefinidos. Si el sistema se está ejecutando en cualquier momento en ese intervalo, la tarea se realiza. Creo que esta característica es la parte más útil de Fcron. En mi portátil, la mayoría de las tareas del sistema las tengo configuradas de esta forma. Considere las siguientes entradas:

```
%daily,nice(7) * 10-14 >
makewhatis -u
%daily,nice(7) 30 16-18 >
updatedb
```

Estas entradas son de mi *systab* (el fichero de sistema *fcrontab*). El % al principio de la línea le indica a Fcron que lo que sigue es una tarea cron definida en un intervalo. Lo siguiente es una palabra reservada que indica el tipo de intervalo que se está intentando configurar. La palabra clave podría ser un término como *hourly* (cada hora), *daily* (diariamente), *weekly* (semanalmente) o *monthly* (mensualmente), o puede representar una medida de tiempo absoluta: *mins* (minutos), *hours* (horas), *days* (días), *dow* (día de la semana) o *mons* (meses). Cada uno de los dos tipos de palabras reservadas actúa de forma diferente. La terminación **ly* es autoexplicativa: *daily* ejecuta la tarea cron diariamente, y *weekly* la ejecuta una vez en algún momento entre el lunes y el viernes y luego espera a la semana siguiente para repetir la operación.

Después de la coma, hay otra opción, *nice*, que ejecuta la tarea cron con el nivel indicado entre paréntesis. (La utilidad *nice* cambia la prioridad de un proceso en el planificador del kernel). Lo siguiente son los campos de fecha y hora necesarios para definir el intervalo de tiempo, a continua-

ción el comando. Así que las entradas anteriores al cron actualizan las páginas man (*makewhatis -u*) cada día, si el sistema está encendido alguna vez entre las 10:00 am y las 2:00 pm (10-14), y actualiza la base de datos *slocate* (*updatedb*) si el sistema está encendido desde las 4:30 am hasta las 6:30 pm (30 16-18).

El ejemplo anterior utiliza la palabra clave *daily*, que tiene que tener el intervalo configurado en minutos y horas. Lo mismo ocurre para *weekly*. Por analogía, *hourly* necesita sólo el campo minutos y *monthly* necesita los minutos, horas y días.

Si se especifica *mid* con una de las palabras clave **ly* (*middaily*, *midweekly*), el intervalo comienza y finaliza en la mitad

Glosario

Una tarea cron es una tarea que el sistema Cron ha de ejecutar periódicamente. El fichero *crontab* es un fichero de configuración para un usuario que define tareas para ejecutar bajo la cuenta del usuario. El fichero *systab* es un archivo que especifica tareas cron para el sistema. El sistema de *crontab* propio de Fcron utiliza el fichero *fcrontab* para la configuración. La sintaxis de *fcrontab* es parecida, pero difiere ligeramente de la notación clásica del *crontab* de Visie/ISC Cron.

Configuración de Fcron

Los ficheros de configuración de Fcron son parecidos a la implementación de Vixie Cron (que es el que viene por defecto en la mayoría de las distribuciones). En mi sistema Gentoo, estos ficheros se encuentran en */etc/fcron/*:

- *fcron.conf* – Le permite especificar su editor de texto preferido, programa de correo y alguna otra configuración.
- *fcron.allow* – Especifica a qué usuarios se les permite utilizar Fcron (all significa que cada usuario puede tener un fichero *fcrontab*).
- *fcron.deny* – Especifica los usuarios que no quieren utilizar Fcron (reemplaza la configuración de *fcron.allow*).

Por defecto sería OK, a menos que necesite restricciones extra.

Además de las tareas programadas, el fichero *fcrontab* puede incluir configuración del entorno (por ejemplo, las variables HOME y SHELL). Si, por ejemplo, está preocupado por la seguridad y quiere reemplazar la configuración de la shell del */etc/passwd*, puede escribir lo siguiente en su fichero *fcrontab*: SHELL=/bin/sh

del intervalo dado. Por ejemplo, *midweekly* ejecuta el comando entre el jueves de esta semana y el miércoles de la semana que viene. La tarea se cron

```
%hours 20,40 9-10 ➤
* * * finger boss
```

ejecuta cada día sólo una vez a las 9:20, 9:40, 10:20 ó 10:40, ya que cuando se utiliza *hours* (o cualquiera de las otras palabras claves **s*), Fcron ignora todos los campos de tiempo que son menores que la palabra clave. Pero si se configura un campo (distinto con un asterisco), Fcron tendrá en cuenta el momento en el que la tarea cron se pondrá en marcha.

Si el ejemplo anterior utiliza la palabra clave *mins*, el comando (comprobando el último registro y otra información del usuario *boss*) se ejecutará a las 9:20, 9:40, 10:20 y 10:40 cada día, ya que los minutos forman parte del intervalo.

Como ya ha adivinado por la sintaxis, la clave **s* ofrece una funcionalidad que es una mezcla entre un trabajo fijado por una hora-fecha y la clave **ly*.

Tareas Cron Definidas por Disponibilidad del Sistema

¿Qué ocurre si usted desea ejecutar una tarea que depende del tiempo de funcionamiento de su sistema? Claro, podría escribir un complejo script con un bucle y una función *sleep*, pero ¿por qué no hacerlo con más elegancia? Fcron puede incluso funcionar con tareas cron específicas, a veces a partir de la disponibilidad del sistema (o, concretando, sobre la base del tiempo de funcionamiento desde el inicio del servicio Fcron).

Si desea crear un registro donde almacenar la media de carga y una lista de usuarios que se han registrado cada 15 minutos de tiempo de actividad (por ejemplo, para fines estadísticos), podría utilizar la siguiente entrada:

```
@ 15 w >> /root/user_stats
```

La *@* al comienzo de la línea le indica a Fcron que se trata de una tarea cron que representa el tiempo de actividad del sistema. Tras esto viene el tiempo de frecuencia establecido en minutos. Y por último, pero no menos importante, viene su comando.

Por ejemplo, digamos que quiere recibir un agradable correo de su sistema en su cumpleaños, esto es, cuando el tiempo de

actividad del sistema se incrementa en un año (un día de orgullo para cualquier administrador).

```
@mail 365d echo "*POP*" ➤
¡Hurra! ¡Tú bebé ya tiene otro ➤
año! :D"
```

La opción *mail* le envía un correo al usuario propietario de *fcrontab/cronjob* (en contraposición a la opción *mailto*, que le permite especificar un destinatario). El campo de frecuencia está ocupado por *365d* para 365 días. Fcron ofrece la opción de definir la frecuencia con el uso de multiplicadores.

Un número representa el número de unidades de tiempo precedida por una *s* para los segundos, *h* para las horas, *d* para los días, *w* para las semanas o *m* para los meses. (Nótese que las unidades de minuto no tienen multiplicadores). Además, puede combinar los multiplicadores, así si quiere ejecutar un script cada dos semanas, tres días, 10 horas y 15 minutos de tiempo de actividad, podría introducir:

```
@ 2w3d10h15 sh ➤
/root/script.sh
```

Opciones

Las opciones pueden definirse globalmente o en la línea de *crontab*. Una opción que se encuentra en una línea por sí misma debe estar precedida por un signo de exclamación (*!*, sin espacios intermedios). En una línea normal de tarea cron, las opciones tienen que seguir directamente tras el signo *@* o *&... o*, en el caso de una línea *%*, después de la palabra clave y separados por una coma.

Si configura varias opciones, las separaría con una coma (sin espacios). Si una opción tiene un argumento, debería adjuntar el argumento entre paréntesis tras la opción. Las opciones en las líneas de tareas cron siempre invalidan las opciones globales, y todas las opciones no pueden ser utilizadas con todos los tipos de tareas cron.

La línea

```
&bootrun(true),nice(8) ➤
30 3 * * * python /root/➤
admin_report.py
```

ejecuta su script de python como root cada día a las 3:30am (30 3). La opción *nice(8)* ejecuta el proceso en el nivel 8 de nice. (El rango de niveles de nice va desde -20, la

prioridad más alta, al 19, la prioridad más baja). Aunque el *&* especifica un tiempo fijo para el script, la opción *bootrun(true)* asegura que el trabajo se ejecutará cuando arranque el servicio Fcron. Las siguientes series de entradas:

```
@first(5) 1h sh one_script.sh
@first(10) 1h sh ➤
another_script.sh
@first(20) 1h sh ➤
yet_another_script.sh
```

configuran tres scripts para ejecutarse cada hora en tiempo de actividad.

Sin la opción *first()*, los tres scripts intentarían ejecutarse a la vez. Los valores especificados en *first()* aseguran que los trabajos comenzarán en tiempos diferentes. El primer script siempre se ejecuta cinco minutos después de una hora de actividad, el segundo cada 10 minutos tras otra hora de actividad y el tercero cada 20 minutos tras otra hora de actividad. En otras palabras, *one_script.sh* se ejecutará a las 1:05, 2:05, 3:05,... de actividad; *another_script.sh* se ejecutará a las 1:10, 2:10, 3:10,...; y *yet_another_script.sh* se ejecutará a las 1:20, 2:20, 3:30,...

Ahora utilizaré la opción *mailto* comentada anteriormente con la opción *random* para coger un tiempo aleatorio en el intervalo de tiempo, en vez de ejecutar el trabajo lo más pronto posible sin el intervalo:

```
%daily,random,mailto(tux) ➤
* 8-20 ping example.com
```

La tarea cron anterior hará ping al servidor *example.com* cada día en un momento diferente entre las 8:00am y las 8:00pm y luego envía un correo de salida al usuario *tux*.

Conclusión

Fcron puede hacer todo lo que Vixie Cron y Anacron pueden hacer, pero mejor y con más control y menor número de advertencias. Para más información de Fcron, vea la documentación en la web del proyecto [3].

RECURSOS

- [1] Cron: <http://unixgeeks.org/security/newbie/unix/cron-1.html>
- [2] Web de FCron: <http://fcron.free.fr>
- [3] Documentación de Fcron: <http://fcron.free.fr/doc/en/index.html>