GNU/Linux utiliza cron para tareas repetitivas

Del idioma griego ?????? (romanizado o "latinizado" *Krónos,* castellanizado Cronos) era el dios de las cosechas en ese pueblo antiguo porque, supuestamente, era quien dictaba el tiempo de comenzar la siega. Por ello en GNU/Linux "heredamos" ese nombre para el *demonio* que se encarga de lanzar las tareas en el horario que los administradores de sistemas estamos compartiendo con nuestras familias (sábados, domingos, nochebuena, etc.), estamos marchando por nuestros derechos laborales (1° de mayo en Venezuela) o leyendo y asistiendo a reuniones en fechas patrias (19 de abril, 5 de julio, 24 de julio, etc.). Como es un programa tan útil y lo usamos demasiado, pues aquí va su homenaje -y si les resulta útil al colectivo programador, mejor-.

Introducción

El software originalmente desarrollado por Paul Vixie (**cron** y **crontab**), y que nosotros llamamos tareas programadas en GNU/Linux, pueden ser ejecutadas según lo que tengamos guardado en las siguientes ubicaciones:

- Donde los usuarios guardan sus tareas (bajo la forma de ficheros hechos con crontab) en **/var/spool/cron**
- Donde las aplicaciones y servicios del sistema *generalmente* guardan sus tareas en el directorio **/etc/cron.d**
- **/etc/anacrontab** es un directorio especial que se utiliza con los dos casos anteriores y que veremos luego con calma.

> Aunque bien pudiéramos editar **directamente** nuestros archivos con cualquier editor de texto, mejor es utilizar **crontab** porque nos ahorra el trabajo de reiniciar los servicios de tarea programada y evita en la medida de lo posible nuestros errores de tipeo o transcripción.

/var/spool/cron

crontab es una abreviatura de **cron table** y es el nombre de la aplicación que nos permitirá crear nuestras tareas programadas. Cada usuario tiene derecho a crear sus correspondientes, incluso el superusuario, con la orden **crontab -e**, la cual la primera vez que la ejecutemos nos preguntará cual editor de texto utilizar, veremos *algo parecido* a esto:

```
Select an editor.  To change later, run 'select-
editor'.    1. /bin/nano
```

En nuestro caso escogimos **nano**
y sin hacer ninguna modificación salimos con CTRL+X y nos muestra luego el siguiente mensaje:

```
crontab: installing new crontab
```

Al volver a repetir exactamente el mismo procedimiento anterior recibiremos el siguiente mensaje:

```
No modification made
```

Si a futuro queremos cambiar el editor de texto pues ejecutamos «**select-editor**». Ahora que estamos familiarizados con el ambiente explicamos un guion normal, común y corriente, como el siguiente (los colores los hemos agregado nosotros con propósitos didácticos):

```
# crontab -e  SHELL=/bin/bash
MAILTO=usuario@dominio.com
PATH=/
usr/local/bi
n:/usr/local/sbin:/bin:/
sbin:/usr/bin:/usr/sbin  HOME=/
# Para mayores detalles ejecute «man crontab»    # Ejemplo de horario par
a las tareas programadas:  # .---------------- minuto (0 - 59)  # |  .---
---------- hora (0 - 23)  # |  |  .---------- día del mes (1 - 31)  # |
|  |  .-------- número de mes (1 - 12) o sino, en inglés: jan,feb,mar,apr
...  # |  |  |  |  .---- día de la semana (0 - 6) (Domingo=0) o sino, en
inglés: sun,mon ...  # |  |  |  |  |  # *  *  *  *  * nombre-del-comando-
a-ser-ejecutado parámetro-del-comando
```

 # crontab -e

Esta primera línea es un comentario, como vemos debe comenzar con almohadilla o numeral y todo lo que esté a la derecha de este símbolo, hasta el final de la línea, no será ejecutado (*ergo* es un comentario).

SHELL=/bin/bash

Esta segunda línea especifica el entorno de lo que se va a ejecutar, generalmente **bash**. Si se omite el sistema tomará lo que está en especificado en el archivo **/etc/passwd** de cada usuario. Bien pudiera especificar otros lenguajes o métodos (pero todavía no lo hemos comprobado).

MAILTO=usuario@dominio.com

Con esta instrucción establecemos a cual correo electrónico enviaremos, si lo deseamos, el resultado de lo ejecutado.

PATH=/usr/local/bin:/usr/local/sbin:/bin:/sbin:/usr/bin:/usr/sbin

Si no se especifica la ruta del guion, el sistema lo buscará en todas esas ubicaciones, noten los dos puntos como separador de campos, si aun así no consigue el guion lo sabremos al recibir un correo electrónico (si lo hemos especificado). **Es una buena práctica colocar la ruta completa de cada uno de los guiones o ficheros,** osea, su ubicación absoluta.

HOME=/

Generalmente se omite este parámetro, que viene a ser el directorio predeterminado, si no está declarado pues toma valor de **/etc/passwd** de nuevo.

Horarios

```
# Para mayores detalles ejecute «man crontab»  # ===  # Ejemplo de horario para las tareas programadas:  # .---------------- minuto (0 - 59)  # |  .------------- hora (0 - 23)  # |  |  .---------- día del mes (1 - 31)  # |  |  |  .------- número de mes (1 - 12) o sino en inglés: jan,feb,mar, apr ...  # |  |  |  |  .---- día de la semana (0 - 6) (Domingo=0) o sino en inglés: sun,mon ...  # |  |  |  |  |  # *  *  *  *  * nombre-del-comando-a-ser-ejecutado parámetro-del-comando
```

Esta es la parte que más confunde a la mayoría de los programadores y usuarios. Son cinco campos que si los colocamos como en el ejemplo (solamente asteriscos) ejecutará nuestro guion cada minuto. Obviamente esa no es la idea, pero ahora ya sabemos para qué sirve cada asterisco. De izquierda a derecha comenzamos con minutos, horas, días de los meses, meses, años y días de las semanas. Haremos ejemplos prácticos.

Todos los días sábados

```
* * * * 7 ls > ~/lista_archivos.txt
```

Con esta instrucción haremos que **todos los días sábados, cada minuto de esos días** liste los archivos (comando **ls**) y los guarde en la carpeta de usuario (comodín «~» para indicar dicha carpeta) en el archivo **lista_archivos.txt**, ahora bien *¿cuáles nombres obtendremos?* Como dijimos, si se omite el parámetro **HOME=/** (donde lo que está a la derecha del signo de igualdad será la ruta deseada), entonces el sistema consultará al fichero **/etc/hosts** y allí "ubicará" al comando **ls**, *pero como muy bien explicamos, esa no es una buena práctica de programación, ya que lo correcto, lo explícito, sería lo siguiente:*

```
* * * * 7 ls ~/ > ~/lista_archivos.txt
```

¿Para qué serviría este ejemplo? Bien pudiéramos revisar los días sábado la lista de los archivos presentes *al finalizar el día sábado o al apagar el equipo (lo que suceda primero), y sin mayor detalle* (podemos usar otros parámetros del comando **ls** para obtener mayor información y en vez de *sobrescribir* - comando ">" - el fichero de reporte, escojamos la opción de *adicionar* al final - comando ">>" -).

Todos los primeros de mes a las 9 y 7 minutos de la noche

```
7 21 1 * * ~/miguion1.sh -parametro1 ; ~/miguion2.sh -parametro1
```

En este caso ejecutaremos dos guiones almacenados, desde luego, en dos ficheros distintos, ubicados en nuestra carpeta de usuario. Notemos el uso de punto y coma para separar los comandos y la hora en formato militar o 24 horas.

Si queremos ver para confirmar el contenido de la lista de tareas programadas solo tenemos que ejecutar **crontab -l**

Cada hora los primeros de mes de marzo, junio y diciembre

```
0 * 1 3,6,12 * * ~/miguion1.sh -parametro1 ; ~/miguion2.sh -parametro1
```

En este caso usamos las comas para separar los números de mes. **Nota:** si necesitamos ejecutar una sola vez una tarea deberemos usar el comando **at** o el comando **batch**; repetimos: **crontab** es para tareas repetitivas.

Cada hora desde las 9:07 a. m. hasta las 4:07 p. m. de lunes a viernes

A veces necesitamos un recordatorio "en horas de oficina", pues se hace de esta manera:

```
7 9-5 * * mon-fri ~/tareas/recordatorio.sh
```

Aunque no lo habíamos explicado, al guion debemos darle permiso de ejecución: en este caso está ubicado en la carpeta del usuario dentro de una carpeta llamada "tareas". Fíjense entonces que el signo de guion "9-5" significa que se ejecutará a las 9:07 a. m., 10:07 a. m., 11:07 a. m., 12:07 a. m., 1:07 p. m., 2:07 p. m., 3:07 p. m. y 4:07 p. m. (5:07 p. m. está fuera del rango).

Además, si en vez de cada hora lo necesitamos *cada dos horas*, deberemos escribir "7-9/2": 7:07 a. m., 9:07 a. m., 11:07 a. m., 1:07 p. m., y 3:07 p. m. (5:07 p. m. está fuera del rango).

Permisos de usuario

Usuarios permitidos y usuarios prohibidos

Con un gran poder viene una gran responsabilidad: a veces necesitamos que ciertos usuarios, a pesar de estar en el grupo de los administradores, **no puedan programar tareas.** Para llevar control de ello, debemos escribir en el fichero **/etc/cron.d/cron.deny** cada uno de los usuarios prohibidos, uno solo por cada línea. Como podrán imaginar, los usuarios explícitamente permitidos estarán en la lista **/etc/cron.d/cron.allow**, pero ¿cómo funciona esto exactamente? *Pongan atención:*

- Si **cron.allow** existe, solamente los usuarios que estén en la lista podrán crear, editar, visualizar y remover sus tareas programadas.
- Si **cron.allow** *no existe* todos los usuarios están permitidos *excepto los que se encuentren en la lista de los prohibidos* **(cron.deny).**
- Si ninguna de las dos listas existe, se necesitarán privilegios de superusuario para ejecutar el comando **crontab**, así mismo necesitará de privilegios de superusuario para editar **cron.deny** y **cron.allow**

Correr como otro usuario

El superusuario podrá editar tareas programadas que se ejecuten con la clave de un usuario en particular... **¿pero acaso por algún problema en los privilegios, osea, se busca algún privilegio en especial?** Pues no, ¿recuerdan que el símbolo "~" funciona como comodín? Pues bien, podremos guardar datos en la carpeta de cada usuario:

```
15 07 * * * maria /etc/bin/mi-guion-diario.sh > ~/resultado.txt  30 12 * * * jose /etc/bin/mi-guion-diario.sh > ~/resultado.txt  00 15 * * * jesus /etc/bin/mi-guion-diario.sh > ~/resultado.txt
```

De esta manera solo tenemos que mantener un solo guion en la carpeta "**etc/bin**" (al cual le tenemos que dar derecho de ejecución) y crear un grupo llamado "**empleados**" al cual ese archivo pertenecerá -y los tres usuarios de ejemplo también al mismo grupo-. Así las cosas a la 7 y cuarto de la mañana María recibirá su informe, José a las 12 y media de la tarde y Jesús a las 3 de la tarde, cada uno en su propia carpeta de usuario ***(ejemplo didáctico).***

Ejemplo avanzado

Gustamos de instalar en las máquinas que usamos un proyecto del señor Atareao (Lorenzo Carbonell Cerezo) cuyo código fuente reposa en GitHub.

Dicho programa permite cambiar el *papel tapiz* en nuestro escritorio para los ambientes GNOME y MATE (la historia del desarrollo está escrita aquí). Entonces esto es lo que instala en el **crontab**, todo va en una sola línea pero lo hemos dividido para facilitar su lectura, noten que llaman a un guion hecho en lenguaje Python:

```
* */12 * * * export DISPLAY=:0;export DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus;  export GSETTINGS_BACKEND=dconf;/usr/bin/python3 /usr/share/national-geographic-wallpaper/ngdownloader.py   >/dev/null 2>&1 &&  gsettings set org.mate.background picture-filename   "/home/jimmy/.config/national-geographic-
wallpaper/potd.jpg" >/dev/null 2>&1 # NGW_EVERY_TWELVE
```

/etc/cron.d

En este directorio las aplicaciones guardan sus tareas programadas porque poco sentido tiene el crear una cuenta de usuario para cada programa. Conserva el mismo formato que para los usuarios pero combina funciones avanzadas; mostramos acá la nuestra (Ubuntu 18.04):

```
# The first element of the path is a directory where the debian-sa1  # script is located  PATH=/usr/lib/sysstat:/usr/sbin:/usr/sbin:/usr/bin:/sbin:/bin    # Activity reports every 10 minutes everyday  5-55/10 * * * * root command -v debian-sa1 > /dev/null && debian-sa1 1 1    # Additional run at 23:59 to rotate the statistics file  59 23 * * * root command -v debian-sa1 > /dev/null && debian-sa1 60 2
```

Todos lo que hemos hecho hasta ahora nos basamos en la suposición de que el ordenador está encendido las 24 horas, todos los días... claro si el equipo está apagado, pues no se ejecuta la tarea programada **¿qué podemos hacer en este caso?**

anacron

El programa **anacron** se encarga de revisar cuándo fue la última vez que se ejecutó una tarea programada. Pongamos por caso que respaldamos diariamente pero el equipo duró tres días apagado porque sabotearon el sistema eléctrico: al encender **anacron** ejecutará el respaldo... ***una sola***

vez (no importa cuántas veces se hayan dejado de realizar).

Tengamos muy en cuenta que **anacron** no se ejecuta en tiempos específicos y exactos, sino a lo largo de períodos clasificados en los siguiente sdirectorios:

- **/etc/cron.hourly** a lo largo de cada hora.
- **/etc/cron.daily** a lo largo de cada día.
- **/etc/cron.weekly** a lo largo de cada semana.
- **/etc/cron.monthly** a lo largo de cada mes.

Para colocar nuestros propios ficheros o "pseudo tareas programadas" la mejor técnica es ubicarlos en **/usr/local/bin** lo cual permitirá que lo probemos bien y cuando lo tengamos listo le hacemos enlaces simbólicos a la carpeta según el período de tiempo que necesitemos.

Fuentes consultadas

En idioma castellano

En idioma inglés

- «How to use cron in Linux» by David Both (backed by archive.today).
- «crontab (cron table)». (backed by archive.org).
- «Newbie: Intro to cron» (backed by archive.org).
-
- «How can I redirect and append both stdout and stderr to a file with Bash?» (backed by archive.org).
- «Redirection» by BASH Hackers (backed by archive.org).
-
- «Linux crontab command» (backed by archive.org).
- «Cron» (backed by archive.org)
- «ARCHIVED: What are cron and crontab, and how do I use them?» (backed by archive.org).
- «Controlling Access to the crontab Command» (backed by archive.org).