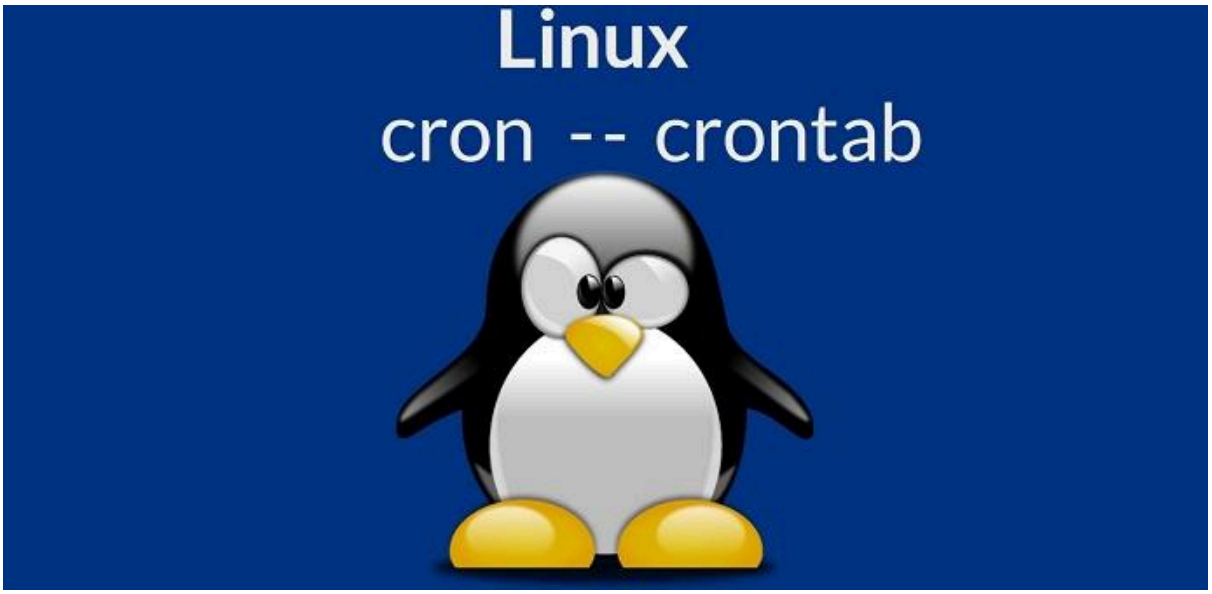
Cómo utilizar Cron y Crontab en Linux para programar tareas



¿Has intentado programar una tarea en Linux y no has sabido como? ¿A pesar de consultar el manual propio no has sido capaz? Cron y Crontab. Estos dos elementos son muy importantes a la hora de programar tareas periódicas y que se ejecuten sin que tengamos que estar pendientes de su estado.

Diferencias entre Cron y Crontab

En los sistemas operativos Windows existe algo similar denominado Tareas Programadas. En esta ocasión el proceso no se realiza de forma gráfica y se tiene que realizar a través de terminal. Por lo tanto, si tienes miedo a este elemento, este artículo es la mejor forma de perder ese temor y coger soltura.Hemos mencionado con anterioridad Cron y Crontab. En primer lugar, vamos a ver qué diferencias existen entre ambos elementos. Posteriormente aprenderemos a agregar tareas y para finalizar nos adentraremos en la gestión de los *jobs* (trabajos) de Cron.Aunque pueda parecer que estamos hablando de lo mismo, no es así, y se podrían considerar dos elementos dependientes uno del otro. Para ser más claros, son los encargados de que la programación de tareas en sistemas Linux sea posible.

¿Qué es Cron?

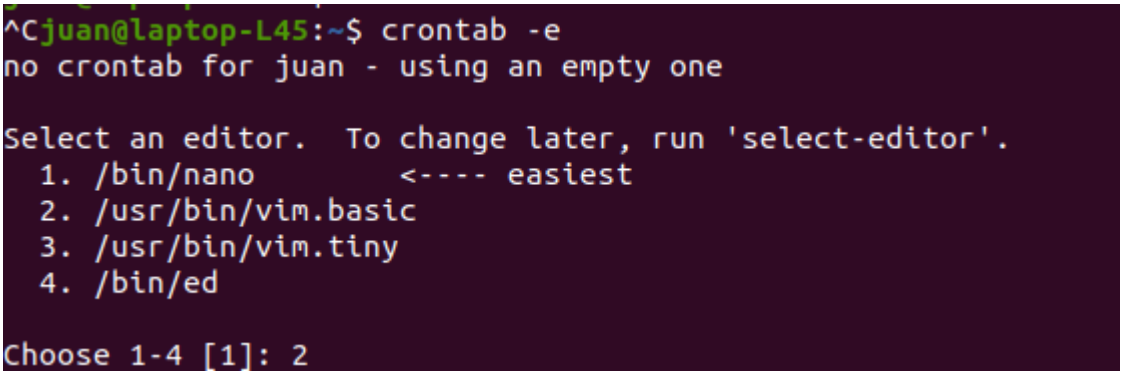
Es un demonio (proceso en segundo plano) que se ejecuta desde el mismo instante en el que arranca el sistema. Comprueba si existe alguna tarea (*job*) para ser ejecutada de acuerdo a la hora configurada en el sistema. Es muy importante que la zona horaria esté bien configurada ya que de lo contrario las ejecuciones puede que no coincidan con los ajustados.

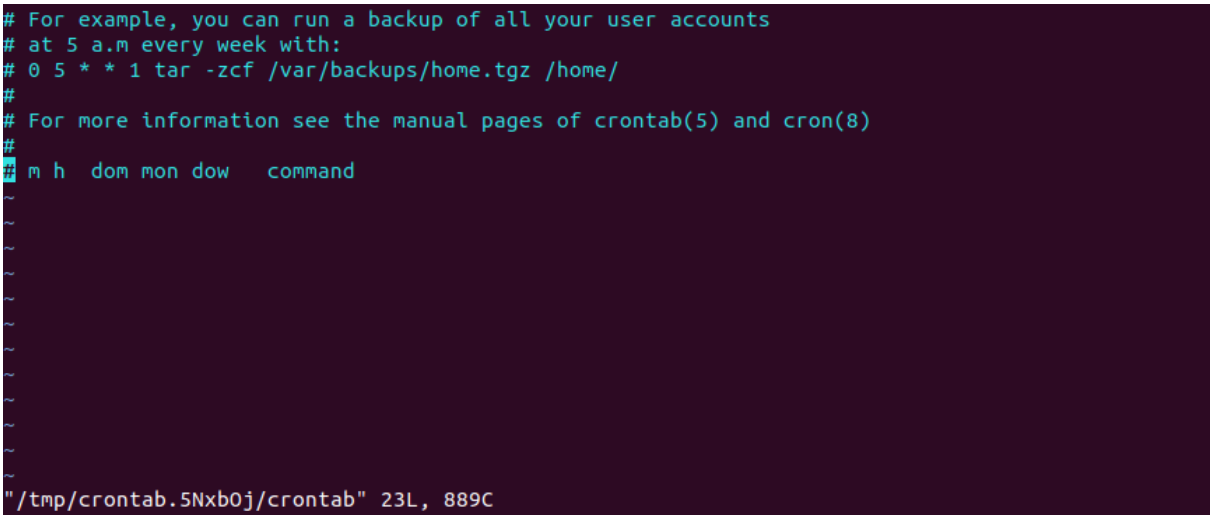
En función de la distribución, se inicia utilizando las carpetas ***/etc/rc.d/*** o ***/etc/init.d*** y cada minuto comprueba los ficheros ***/etc/crontab*** o ***/var/spool/cron*** es busca de posibles ejecuciones (ahora empieza todo a cuadrar).

¿Qué es Crontab?

Un archivo de texto. Por simple que parezca la definición es así. Aunque es verdad que se trata de un archivo con contenido especial. Posee una lista con todos los *scripts* a ejecutar. Generalmente cada usuario del sistema posee su propio fichero Crontab. Se considera que el ubicado en la carpeta **etc** pertenece al usuario **root**.

Para generar el archivo propio, cada usuario deberá hacer uso del comando crontab (sí, es el mismo nombre).





Agregar tareas a Crontab

Partiendo de que podemos ejecutar tareas, en primer lugar, vamos a configurar un *script* muy sencillo que llamaremos consulta.sh:

```
#!/bin/bash
#script de ejemplo

sudo ls -l / > archivoResultado
```

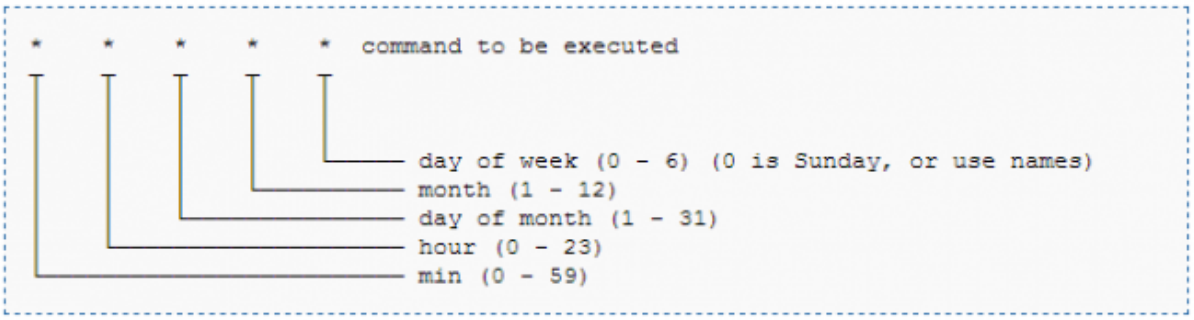
Con la ayuda del manual que hemos citado con anterioridad cambiamos los permisos:

```
chmod ugo+x consulta.sh
```

Este aspecto es muy importante, ya que de lo contrario el comando nunca será ejecutado.

Estructura del crontab

Ha llegado el momento de editar el fichero que posee las tareas. Para ello nos vamos a ayudar del comando crontab -e. Nos encontramos la siguiente estructura:



5 asteriscos y el comando a ejecutar. Cada uno de los 5 asteriscos significa:

- m: minuto
- h: hora
- dom: día de la semana
- mon: mes
- **dow: día del mes**

Los valores que pueden adoptar cada una de estas variables se encuentran en la imagen. Por ejemplo, en el caso del último, se permite la utilización del nombre de los días escritos en inglés.

Aunque en la imagen anterior no aparezca, es necesario indicar entre el comando y el último asterisco (el día) el propietario del archivo.

Para que quede todo claro, vamos a utilizar algunos ejemplos:

Ejecutar todos los días a las 7 de la tarde

```
00 19 * * * usuario /ubicacion/del/script/consulta.sh
```

Ejecutar todos los domingos a las 7 de la tarde:

```
00 19 * * 0 usuario /ubicacion/del/script/consulta.sh
```

Ejecutar el script todos los 4 de febrero a las 7 de la tarde:

*00 19 4 2 * usuario /ubicacion/del/script/consulta.sh*

Hay que decir que en Linux existen algunas cadenas de texto reservadas para ejecutar procesos durante determinados periodos:

- @reboot: Ejecuta una vez y nada más iniciarse el equipo.
- @yearly: ejecuta sólo una vez al año: 0 0 1 1 *
- @monthly: ejecuta una vez al mes y el primer día: 0 0 1 * *
- @weekly: Todas las semanas, el primer minuto de la primer hora de la semana: 0 0 * * 0.
- @daily: todos los días a las 12 de la noche: 0 0 * * *
- @midnight: Tiene el mismo efecto que el anterior.
- @hourly: todas las horas durante su primer minuto: 0 * * * *

Aunque os hemos puesto cuál sería el formato, al hacer uso de crontab -e se pueden utilizar estos términos para definir el periodo.

Administración de los *jobs* del Cron

Tal y como hemos podido ver el funcionamiento es muy sencillo. Por último, falta conocer algunos comandos básicos para controlar el cron de nuestro sistema Linux:

- Para reemplazar el archivo existente por otro que defina el usuario se debe utilizar el siguiente comando:

  ```
  crontab archivo
  ```

- Para editar el archivo existente en la actualidad se utiliza el comando que ya hemos visto a lo largo de este artículo:

  ```
  crontab -e
  ```

- Listar todas las tareas existentes en el crontab del usuario:

  ```
  crontab -l
  ```

- Borrar el crontab que está configurado:

  ```
  crontab -d
  ```

- Definir el directorio en el que se almacenará el archivo de crontab. Para realizar esta operación se deben tener permisos de ejecución en dicho directorio:

  ```
  crontab -c dir
  ```

- Instrucción para manejar el crontab de otros usuarios existentes en el sistema:

  ```
  crontab -u usuario
  ```

Cómo podéis ver, programar la ejecución de tareas no es nada complicado y se puede realizar de forma rápida si se tiene claro todo lo mencionado en este artículo.