

Comandos sudo y su Comando at Cron y crontab

Alfredo Abad

ISO-04-033-SUDO-SU-AT-CRON.pptx

14-sep-2023



1
Alfredo Abad



El comando “su”, “su –” y “su usuario –c”

- Sirve para asumir la identidad de otro usuario cuyo nombre se escribe como argumento
 - Si no se especifica usuario, se entiende que es root
 - La shell pedirá la contraseña del usuario (o root si se omitió el argumento)
 - **su** conserva el entorno del usuario que ejecuta **su**
- Ejecución de “**su –**”
 - Es como su, pero se inicia el entorno que le correspondería a root (abandonando el del usuario que ejecuta el comando su)
- Ejecución de “**su usuario –c comando**”
 - Ejecuta el comando desde la cuenta de usuario

2
Alfredo Abad

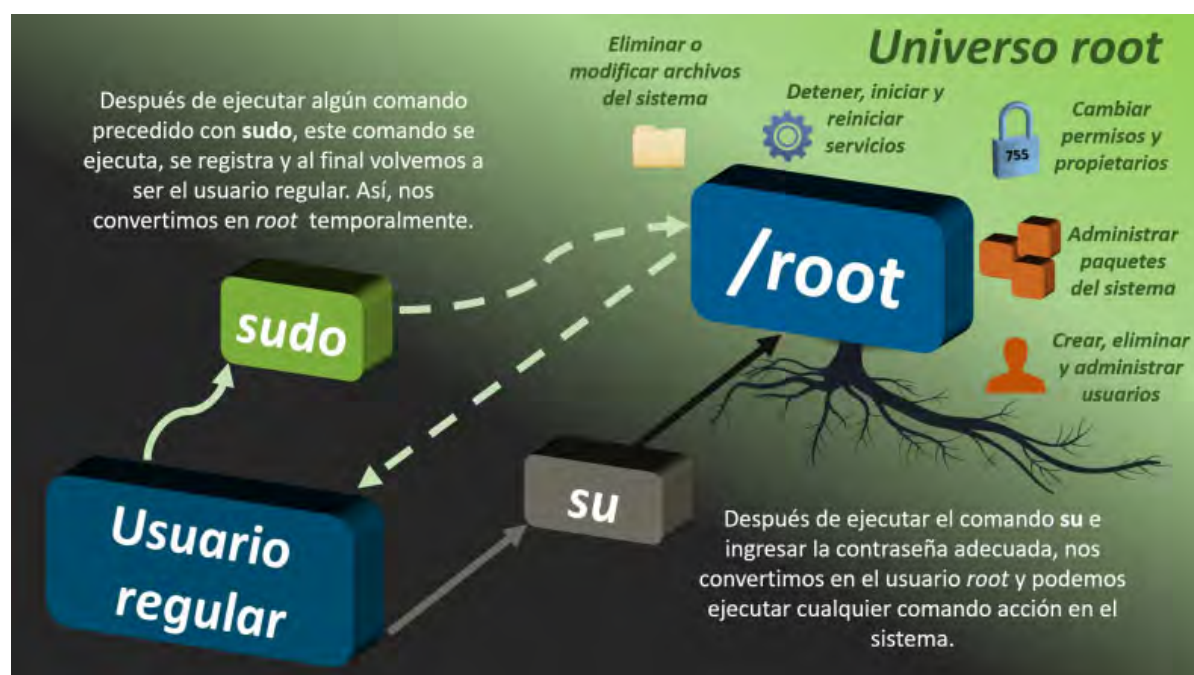


sudo frente a su

- **su** pide la contraseña de la cuenta de destino
- **sudo** pide la contraseña de la cuenta de origen
- Los permisos del usuario sudo se reflejan en el fichero **/etc/sudoers**, que configura el comportamiento de sudo
- Para ejecutar sudo no es necesario tener activada la cuenta de root
- Ejecución de “**sudo su**”
 - Crea una sesión de root, sin tener root activado, puesto que solo pide la contraseña de la cuenta de origen (no la de root)

3

Alfredo Abad



4

Alfredo Abad





Configurar sudo para administrar un sistema Linux

5

Alfredo Abad



Seguridad del comando sudo en la administración

- **sudo** proporciona mayor seguridad: nos permite administrar un sistema operativo sin la necesidad de iniciar una sesión con root
- Además el comando sudo registrará la totalidad de usos realizados en el siguiente log:
 - En Debian y distros derivadas: /var/log/auth.log
 - En Fedora y distros derivadas: /var/log/secure
 - Si usamos Arch y distros derivadas: /var/log/sudo-io
 - En CentOS: /var/log/secure
- Nota: Las ubicaciones son las predeterminadas por cada distro. Si lo creemos necesario podemos modificar la ruta en que se guardan los logs
- Finalmente, **sudo** nos permitirá restringir los comandos que puede ejecutar cada uno de los usuarios al que le permitiremos usar el comando sudo
- Si no está instalado en un sistema, se puede hacer ejecutando:
 - \$ su
 - # apt-get install sudo

6

Alfredo Abad



Configurar sudo para restringir su uso

Para restringir el uso del comando sudo abrimos una terminal y ejecutamos el siguiente comando:

```
su
```

A continuación escribimos la contraseña del usuario root y presionamos enter.

Una vez logueados como usuario root ejecutamos el siguiente comando en la terminal:

```
visudo
```

Nota: Para editar el fichero `/etc/sudoers` se recomienda usar visudo. Visudo previene que 2 usuarios puedan editar el fichero de forma simultánea y comprueba que la sintaxis que escribimos sea correcta.

Después de ejecutar el comando se abrirá el editor de textos en el que podremos configurar sudo según nuestras necesidades.

Para restringir el uso del comando sudo deberemos usar un comando del siguiente tipo:

```
nombre_usuario nombre_equipo = (usuario:grupo)
comando_restringir
```

El significado de cada uno de los términos de este comando es el siguiente:

7
Alfredo Abad



```
nombre_usuario nombre_equipo = (usuario:grupo)
comando_restringir
```

El significado de cada uno de los términos de este comando es el siguiente:

nombre_usuario: Es el nombre de usuario que puede usar el comando sudo. El nombre de usuario puede ser un usuario, un alias de usuario o un grupo.

nombre_equipo: Es el nombre del equipo o hostname en el que podemos aplicar el comando sudo. Sus valores pueden ser todos los equipos (ALL), un solo equipo, un alias de equipos, una dirección IP, etc.

(usuario:grupo): Especificamos los usuarios y los grupos que podrá usar el usuario de sudo cuando ejecuta los comandos. Para poder ejecutar comandos con un usuario y grupo específico tendremos que usar los comandos `sudo -u` y `sudo -g`. Si no indicamos ningún usuario ni ningún grupo se usará el usuario root y el grupo root.

comando_restringir: Especificación/restricción de los comandos que pueden ejecutar los usuarios que pueden ejecutar el comando sudo.

8
Alfredo Abad



1- Dar permisos de usuario root a un usuario normal para ejecutar cualquier comando en nombre de cualquier usuario

La configuración que veremos en este apartado es la que usan el 99% de usuarios domésticos en sus equipos

Para conseguir este propósito tan solo tenemos que añadir la siguiente línea dentro del fichero `/etc/sudoers`.

```
joan ALL=(ALL:ALL) ALL
```

```

Terminal - joan@debian: ~
Archivo Editar Ver Terminal Pestañas Ayuda
nano 2.6.3 Fichero: /etc/sudoers

# Allow members of group sudo to execute any command
%sudo  ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:
#include /etc/sudoers.d

joan  ALL=(ALL:ALL) ALL
  
```

Muestra de mi fichero `/etc/sudoers`

Introduciendo esta línea damos permiso al usuario `joan` para que en cualquier equipo pueda ejecutar cualquier comando en nombre de cualquier usuario y grupo existente.

Una vez introducida la línea guardamos los cambios y cerramos el fichero.

9

Alfredo Abad



2- Dar permisos de usuario root a un usuario normal mediante sudo

Para cumplir con nuestro propósito tenemos que introducir la siguiente línea en el fichero `/etc/sudoers`:

```
joan debian=(root:root) ALL
```

Introduciendo esta línea damos permiso al usuario `joan` para que en el equipo `debian` pueda ejecutar cualquier comando en nombre del usuario `root`.

A diferencia del caso anterior en este caso el usuario `joan` únicamente podrá usar comando `sudo` en el equipo `debian` y únicamente lo podrá hacer en nombre del usuario `root`.

Una vez introducida la línea guardamos los cambios y cerramos el fichero.

3- Dar permisos para que todos los usuarios de un grupo tengan permisos de administrador

Para dar permisos de administración a todos los usuarios de un grupo tendríamos que introducir la siguiente línea en el fichero `/etc/sudoers`:

```
%joan ALL=(root:root) ALL
```

Introduciendo esta línea damos permisos a todos los usuarios del grupo `joan` para que en cualquier equipo puedan ejecutar cualquier comando en nombre del usuario `root`.

Una vez introducida la línea guardamos los cambios y el proceso ha finalizado.

10

Alfredo Abad



4- Dar permisos a un usuario para ejecutar comandos en nombre de varios usuarios

Para que un usuario pueda ejecutar comandos como si fuera otro usuario podemos añadir un comando similar al siguiente en el fichero **/etc/sudoers**:

```
joan debian=(martin,root:martin,root) ALL
```

Introduciendo esta línea damos permiso al usuario **joan** para que en el equipo **debian** pueda ejecutar cualquier comando en nombre del usuario **martin** y en nombre del usuario **root**.

Una vez guardados los cambios en el fichero **/etc/sudoers**, el usuario **joan** podrá por ejemplo crear un fichero en nombre del usuario **martin** ejecutando el siguiente comando en la terminal:

```
sudo -u martin -g martin touch prueba.txt
```

11
Alfredo Abad



5- Dar permisos a un usuario para que únicamente pueda usar los comandos que nosotros queramos

De forma fácil podemos limitar los permisos que damos a un usuario.

A modo de ejemplo podemos introducir el siguiente comando en el fichero **/etc/sudoers**:

```
joan ALL=(root:root) /usr/bin/passwd *, !/usr/bin/passwd  
root
```

Con este comando damos permiso al usuario **joan** para que en el cualquier equipo pueda cambiar la contraseña de cualquier usuario exceptuando la contraseña del usuario **root**.

Nota: Mediante el operador ***** indicamos que el usuario **joan** podrá cambiar la contraseña de todos los usuarios. Con el operador **!** Forzamos que el usuario **joan** no pueda cambiar la contraseña del usuario **root**.

6- Dar permisos para que únicamente los usuarios de un grupo puedan utilizar iptables

Si únicamente quiero que los usuarios del grupo **joan** puedan utilizar iptables introduciré la siguiente línea dentro del fichero **/etc/sudoers**:

```
%joan ALL=(root:root) /sbin/iptables
```

Una vez introducidos los cambios tan solo tenemos que guardarlos y cerrar el editor de textos.

Nota: Para facilitar el proceso de restricción y configuración de podemos usar alias.

12
Alfredo Abad



Configurar sudo para deshabilitar o limitar el período de gracia de sudo

En el momento de ejecutar un comando con sudo tenemos que introducir la contraseña de nuestro usuario.

Una vez introducida la contraseña nuestro sistema operativo la recordará durante un período de 5 minutos.

De este modo si a los 3 minutos de aplicar un comando con sudo aplicamos otro comando con sudo no tendremos que introducir la contraseña de nuevo.

Muchos usuarios consideran que este período de gracia es una brecha de seguridad.

Por lo tanto si queremos limitar o eliminar el tiempo que nuestro sistema almacena nuestra contraseña abrimos una terminal y ejecutamos el siguiente comando:

```
sudo visudo
```

Una vez se abra el editor de textos introduciremos la siguiente línea al final del archivo:

```
Defaults:ALL timestamp_timeout=0
```

Una vez introducida guardamos los cambios y cerramos el editor de textos.

En estos momentos cada vez que usemos sudo tendremos que introducir nuestra contraseña de usuario. Por lo tanto hemos eliminado el período de gracia para la totalidad de usuarios de nuestro equipo.

13
Alfredo Abad



En estos momentos cada vez que usemos sudo tendremos que introducir nuestra contraseña de usuario. Por lo tanto hemos eliminado el período de gracia para la totalidad de usuarios de nuestro equipo.

Si nuestro objetivo fuese simplemente reducir el período de gracia a 2 minutos, al final del archivo `/etc/sudoers` deberíamos introducir el siguiente comando:

```
Defaults:ALL timestamp_timeout=2
```

Después de introducir el comando guardamos los cambios y cerramos el editor de textos.

En estos momentos nuestro sistema operativo únicamente recordará nuestra contraseña de usuario durante 2 minutos. Si en vez de 2 minutos queremos que sea 1 minuto tendremos que reemplazar el 2 por el 1.

14
Alfredo Abad



Usar el comando sudo sin necesidad de usar contraseña

En contraposición al apartado anterior pueden existir usuarios a los que no les preocupe la seguridad y les molesta tener que introducir su contraseña.

Si este es vuestro caso tenéis que abrir una terminal y ejecutar el siguiente comando:

```
sudo visudo
```

Se abrirá el editor de textos con el contenido del archivo `/etc/sudoers`.

En el apartado donde restringimos el uso del comando sudo deberemos introducir la palabra **NOPASSWD**: justo antes de la parte del comando en que indicamos los comandos que se pueden ejecutar con sudo.

Por lo tanto en mi caso escribiría el siguiente comando:

```
joan ALL=(ALL:ALL) NOPASSWD: ALL
```

Finalmente guardaría los cambios y cerraría el fichero.

En estos momentos el usuario joan podrá ejecutar cualquier comando con sudo sin necesidad de introducir ninguna contraseña.

En el caso que quisiéramos que un usuario solo pudiera usar determinados comandos sin tener que utilizar contraseña podríamos reemplazar el comando anterior por el siguiente:

```
joan ALL=(ALL:ALL) NOPASSWD: /usr/bin/apt-get, /sbin/shutdown
```

De este modo el usuario joan podría gestionar los paquetes del equipo y cerrar el ordenador sin necesidad de introducir ninguna contraseña. Para el resto de tareas de administrador tendría que introducir la contraseña.



15

Alfredo Abad

Configurar sudo para limitar los intentos que un usuario tiene para adivinar la contraseña

De forma estándar disponemos de 3 intentos para adivinar la contraseña de nuestro usuario.

En el caso que fállemos 3 veces deberemos volver a introducir de nuevo el comando que queremos ejecutar.

Si queréis reducir el número de intentos a 2 tenemos que abrir una terminal y ejecutar el comando:

```
sudo visudo
```

A continuación se abrirá el editor de textos y al final del contenido del archivo `/etc/sudoers` introduciremos el siguiente comando:

```
Defaults:ALL passwd_tries=2
```

Una vez introducido el comando guardamos los cambios y cerramos el editor de texto.

En estos momentos únicamente tendremos 2 oportunidades para poner nuestra contraseña de usuario de forma correcta.



16

Alfredo Abad

Configurar sudo para definir el archivo y ubicación en que se guardarán los logs

En el inicio del artículo hemos visto la ubicación de logs de sudo para varias distribuciones.

Si queremos modificar esta ubicación lo podemos realizar fácilmente editando el archivo `/etc/sudoers`. Para ello abrimos una terminal y ejecutamos el siguiente comando:

```
sudo visudo
```

Seguidamente en la parte final del archivo añadimos la siguiente línea:

```
Defaults logfile=/var/log/sudo
```

Nota: La parte de color rojo del comando indica la ruta y el archivo en el que se guardaran los logs. En vuestro caso podéis colocar la ruta que queráis.

Una vez introducida la línea guardamos los cambios.

A partir de estos momentos la totalidad de logs referentes a sudo se guardaran en `/var/log/sudo`.

Si queremos consultar el log podemos ejecutar el siguiente comando en la terminal:

```
sudo less /var/log/sudo
```

17
Alfredo Abad



CONSULTAR LOS REGISTROS DE SUDO

Si queremos ver la totalidad de acciones realizadas con el comando sudo podemos abrir una terminal y ejecutar el siguiente comando:

```
sudo journalctl _COMM=sudo
```

Nota: Este comando funcionará para las distros que usen systemd. En la actualidad prácticamente la totalidad de distros usa Systemd.

Después de ejecutar el comando obtendremos el siguiente resultado:

```
Terminal - joan@debian: ~
sep 17 12:49:06 debian sudo[13459]: joan : TTY=pts/0 : PWD=/home/joan : USER=root : COMMAND=/usr/
sep 17 12:49:06 debian sudo[13459]: pam_unix(sudo:session): session opened for user root by joan(uid=
sep 17 12:49:31 debian sudo[13459]: pam_unix(sudo:session): session closed for user root
sep 17 12:49:38 debian sudo[14235]: pam_unix(sudo:auth): authentication failure; logname=joan uid=100
sep 17 12:49:40 debian sudo[14235]: joan : 1 incorrect password attempt : TTY=pts/1 : PWD=/home/j
sep 17 16:02:26 debian sudo[17991]: joan : TTY=pts/0 : PWD=/home/joan : USER=root : COMMAND=/usr/
sep 17 16:02:26 debian sudo[17991]: pam_unix(sudo:session): session opened for user root by joan(uid=
sep 17 16:05:36 debian sudo[17991]: pam_unix(sudo:session): session closed for user root
sep 17 16:05:46 debian sudo[19252]: joan : TTY=pts/0 : PWD=/home/joan : USER=root : COMMAND=/usr/
sep 17 16:05:46 debian sudo[19252]: pam_unix(sudo:session): session opened for user root by joan(uid=
sep 17 16:06:12 debian sudo[19252]: pam_unix(sudo:session): session closed for user root
sep 17 16:23:13 debian sudo[26116]: joan : TTY=pts/1 : PWD=/home/joan : USER=root : COMMAND=/bin/
sep 17 16:23:13 debian sudo[26116]: pam_unix(sudo:session): session opened for user root by joan(uid=
sep 17 16:23:13 debian sudo[26116]: pam_unix(sudo:session): session closed for user root
sep 17 16:23:23 debian sudo[26165]: joan : TTY=pts/1 : PWD=/home/joan : USER=root : COMMAND=/bin/
sep 17 16:23:23 debian sudo[26165]: pam_unix(sudo:session): session opened for user root by joan(uid=
sep 17 16:23:23 debian sudo[26165]: pam_unix(sudo:session): session closed for user root
sep 17 16:24:01 debian sudo[26384]: joan : TTY=pts/1 : PWD=/home/joan : USER=root : COMMAND=/bin/
sep 17 16:24:01 debian sudo[26384]: pam_unix(sudo:session): session opened for user root by joan(uid=
sep 17 16:34:58 debian sudo[26384]: pam_unix(sudo:session): session closed for user root
sep 17 16:35:01 debian sudo[30290]: joan : TTY=pts/1 : PWD=/home/joan : USER=root : COMMAND=/bin/
sep 17 16:35:01 debian sudo[30290]: pam_unix(sudo:session): session opened for user root by joan(uid=
lines 56-77/77 (END)
```

18
Alfredo Abad



DESACTIVAR O DESHABILITAR EL USUARIO ROOT

Una vez tenemos configurado sudo a nuestro gusto es posible que alguien se pueda plantear deshabilitar el usuario root.

De este modo en caso de un ataque, el atacante tendrá una tarea adicional que será la de averiguar el nombre de usuario que puede utilizar sudo.

Nota: Antes de seguir adelante aseguraos que sudo lo tenéis configurado correctamente. Si sudo está mal configurado y deshabilitamos el usuario root podemos tener graves problemas.

Para bloquear al usuario root ejecutamos el siguiente comando en la terminal:

```
sudo usermod --lock --expiredate 1970-01-01 root
```

Después de ejecutar este comando únicamente podremos realizar tareas de administración del sistema con sudo.

Si un día quisiéramos reactivar el usuario root tan solo tendríamos que ejecutar el siguiente comando en la terminal:

```
sudo usermod --unlock --expiredate 99999 root
```

19
Alfredo Abad



CONSIDERACIONES DEL COMANDO SUDO

En este artículo hemos visto varias opciones para configurar sudo según nuestras necesidades. No obstante solo se recomienda usar sudo para ejecutar comandos en la terminal.

En el momento que tengamos que ejecutar aplicaciones gráficas con permisos de administrador es recomendable hacerlo con los comandos gksu, kdesu y beesu.

En función del entorno de escritorio que usemos deberemos usar un comando u otro. De este modo:

gksu: Será usado en entornos de escritorio Gnome y Xfce

kdesu: Lo usaremos en entornos de escritorio kde.

beesu: Es el comando que usaremos en Fedora.

A modo de ejemplo si en xfce quiero ejecutar thunar con permisos de administrador ejecutaré el siguiente comando en la terminal:

```
gksu thunar
```

Más sobre cómo abrir aplicaciones gráficas con permisos de administrador si utilizar sudo (mejor, usar gksu, kdesu, beesu):

<https://geekland.eu/abrir-aplicaciones-graficas-como-root/>

20
Alfredo Abad





How to Add a User to the sudoers File in Linux

<https://www.howtogeek.com/842739/how-to-add-a-user-to-the-sudoers-file-in-linux/>

21
Alfredo Abad



Ejemplo de “su” con y sin guion

A veces es necesario realizar tareas desde línea de comandos como si fuésemos otro usuario. Por ejemplo necesitamos correr algún comando a nombre del servidor Web Apache. Para ello es necesario pasar al usuario "www-data" en Debian y derivados.

En los sistemas operativos de la familia Unix, el comando su permite cambiar de usuario, o volverse superusuario (root). Veamos un ejemplo.

22
Alfredo Abad



Uso de su

Logueado no
Presentada
↑

- Actualmente me encuentro "logueado" como el usuario root (superusuario, un equivalente a "Administrator" en los sistemas operativos Windows):

```
root@debian:~# whoami
root
root@debian:~# groups
root
root@debian:~# id
uid=0(root) gid=0(root) groups=0(root)
```

Se observa además que root pertenece al grupo homónimo.

La sintaxis básica del comando su es la siguiente:

su USUARIO

Si no se especifica un nombre de usuario, su pasa a superusuario. Lógicamente no cualquiera puede pasar a superusuario, sino que es necesario conocer la contraseña de root.

root puede pasar a cualquier usuario sin necesidad de conocer la contraseña. De eso se trata ser superusuario, tener control total sobre el sistema. El resto de los usuarios necesitan conocer la contraseña del usuario al que desean cambiar. Si deseo cambiar del usuario "www-data" a "postgres", necesitaré indicar la contraseña de "postgres" de forma interactiva.

23
Alfredo Abad



- Para pasar al usuario "www-data", simplemente ejecutar su www-data:

```
root@debian:~# su www-data
$ whoami
www-data
$ groups
www-data
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Esto abre una nueva sesión (en una subshell) a nombre del usuario "www-data", y la sesión original como root queda en segundo plano.

Para finalizar la sesión como "www-data" y volver a la sesión original, simplemente ejecutar exit:

```
$ exit
root@debian:~#
```

24
Alfredo Abad



Cambiar de usuario cargando el perfil

- En general es deseable contar con el entorno (variables, aliases, etc.) al cambiar de usuario. Esto es simular un login completo. Utilicemos la variable de entorno PATH como ejemplo para entender qué sucede si no cargamos el entorno:

```
root@debian:~# su www-data
$ env | grep ^PATH
PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

Se observa que la variable PATH posee el valor por defecto de la distribución (Debian).

En su forma genérica, su abre una subshell sin cargar el entorno del perfil. Para cargar el perfil (entorno) es necesario agregar la opción - (o -l):

```
root@debian:~# su - www-data
$ env | grep ^PATH
PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/usr/local/go/bin:/var/www/gopath:/var/www/gopath/bin
```

Se observa que ahora la variable PATH tiene el contenido definido en el perfil de bash, llamado startup file (/etc/profile).

Además es posible preservar el entorno de la sesión actual (root) en la nueva sesión (www-data) utilizando la opción -m (tanto en GNU/Linux como en *BSD).

25
Alfredo Abad



Comando “at”

26
Alfredo Abad



El comando at

- El comando at es una utilidad versátil que permite a los usuarios programar un comando o script para que se ejecute en un momento específico en el futuro.
- Es especialmente útil para ejecutar tareas únicas, como tareas de mantenimiento, copias de seguridad o actualizaciones del sistema, sin necesidad de intervención manual.
- El comando at lee los comandos que se ejecutarán desde la entrada estándar o desde un archivo y los programa en consecuencia.

27
Alfredo Abad



Instalación del comando at

- La mayoría de las distribuciones de Linux vienen con el comando at preinstalado.
 - Sin embargo, si no está presente en su sistema, puede instalarlo usando el administrador de paquetes para su distribución.
- Para distribuciones basadas en Debian, use el siguiente comando:
 - **sudo apt-get install at**
- Para distribuciones basadas en Red Hat, use este comando:
 - **sudo yum instalar at**

28
Alfredo Abad



Sintaxis y opciones

- La sintaxis básica del comando at es la siguiente:
 - **at [OPTIONS] TIME**
- Las opciones disponibles para el comando at incluyen:
 - **-f**: especifica un archivo que contiene los comandos que se van a ejecutar.
 - **-t**: especifica la hora a la que ejecutar los comandos utilizando una marca de tiempo de Unix.
 - **-m**: envía un correo electrónico al usuario cuando se completa el trabajo.
 - **-q**: especifica una cola en la que colocar el trabajo.

29

Alfredo Abad



Programación de un trabajo para que se ejecute una única vez

- Para programar un trabajo de una sola vez, simplemente proporcione el tiempo deseado para la ejecución. El comando at admite varios formatos de hora, como:
 - **Tiempo relativo**: "ahora + 1 hora" o "ahora + 30 minutos"
 - **Hora absoluta**: "2:30 PM" o "15:30"
 - **Fecha y hora**: "10:00 AM mañana" o "2023-04-01 18:00"
- Por ejemplo, para programar un trabajo de una sola vez para crear un archivo que contenga "Hello, World!" en el directorio /tmp después de una hora, use el siguiente comando:
 - **echo "echo 'Hello, World!' > /tmp/hello_world.txt" | at now + 1 hour**
- Alternativamente, puede programar el comando de la siguiente manera:
 - **at now + 1 hour echo 'Hello, World!' > /tmp/hello_world.txt**
- Presiona **CTRL + D** para salir del terminal del comando

30

Alfredo Abad



Listado y administración de trabajos programados

- Para enumerar todos los trabajos programados para el usuario actual, use el comando "atq":
 - **atq**
- Para eliminar un trabajo programado, use el comando "atrm" seguido del ID del trabajo:
 - **atrm**

31
Alfredo Abad



Mejores prácticas para el comando at

- Siempre verifique que el comando at esté instalado y habilitado en su sistema.
- Usa comentarios descriptivos en tus trabajos at para que sea más fácil entender su propósito.
- Pruebe sus comandos o scripts antes de programarlos con el comando at.
- Recuerde que el comando at está diseñado para trabajos de una sola vez.
 - Use el comando cron para tareas recurrentes.

32
Alfredo Abad



Ejemplos de programaciones que podrían resolverse con comandos at

- Programar una tarea a las 10:00: a las 10:00
- Programe una tarea a las 10:00 a. m. del próximo 25 de julio: a las 10:00 a. m. del 25 de julio
- Programe una tarea a las 10:00 a. m. del 22 de junio de 2023: a las 10:00 a. m. 22/06/2023
- Programe una tarea a las 10:00 a. m. en la misma fecha que el próximo mes: a las 10:00 a. m. el próximo mes
- Programar una tarea mañana a las 10:00 a. m.: mañana a las 10:00 a. m.
- Programe una tarea para ejecutarla justo después de 1 hora: ahora + 1 hora
- Programe una tarea para ejecutarla justo después de 30 minutos: ahora + 30 minutos
- Programe tareas para ejecutarlas justo después de 1 y 2 semanas: ahora + 1 semana ahora + 2 semanas
- Programe tareas para ejecutar justo después de 1 y 2 años: ahora + 1 año ahora + 2 años
- Programar tareas para ejecutar a la medianoche (12:00 AM): a la medianoche

33
Alfredo Abad



Programación de varios comandos en un único at

- Si necesita programar varios comandos para que se ejecuten secuencialmente en un solo trabajo, puede hacerlo ingresando cada comando en una línea separada en el terminal de comandos at.
- Después de ingresar todos los comandos, presione **CTRL + D** para salir y guardar el trabajo.

```
at now + 10 minutes
echo 'First command' > /tmp/output.txt
echo 'Second command' >> /tmp/output.txt
echo 'Third command' >> /tmp/output.txt
```

34
Alfredo Abad



Uso de scripts

- En lugar de ingresar varios comandos en el terminal at de comandos, puede crear una secuencia de comandos de shell que contenga todos los comandos que desea ejecutar y usar la opción **-f** para especificar el archivo de secuencia de comandos.
- Asegúrese de que el script sea ejecutable.

```
# Create a script called myscript.sh
echo '#!/bin/bash' > myscript.sh
echo 'echo "Task 1" > /tmp/script_output.txt' >> myscript.sh
echo 'echo "Task 2" >> /tmp/script_output.txt' >> myscript.sh
chmod +x myscript.sh

# Schedule the script to run in 15 minutes
at -f myscript.sh now + 15 minutes
```

35
Alfredo Abad



Recibir notificaciones por correo electrónico

- Use la opción **-m** si desea recibir una notificación por correo electrónico cuando se complete el trabajo.
- Esto puede ser útil para monitorear el estado de sus tareas programadas.

```
at -m now + 5 minutes
echo 'Sending email notification' > /tmp/email.txt
```

36
Alfredo Abad



Especificación de colas

- El comando `at` le permite especificar una cola en la que colocar el trabajo utilizando la opción `-q`, seguida de una sola letra.
- Los trabajos en diferentes colas son independientes entre sí.

```
at -q a now + 1 hour
echo 'Task in queue A' > /tmp/queue_A.txt

at -q b now + 2 hours
echo 'Task in queue B' > /tmp/queue_B.txt
```

37
Alfredo Abad



Conclusión

- El comando `at` es una herramienta esencial para los usuarios de Linux que necesitan programar trabajos únicos.
- Al comprender su sintaxis y uso, puede automatizar tareas de manera efectiva y mejorar la eficiencia de su flujo de trabajo.
- Recuerde utilizar las mejores prácticas al programar trabajos para garantizar que su sistema funcione sin problemas y que sus tareas se completen a tiempo.
- Ya sea que esté programando tareas de mantenimiento, copias de seguridad o actualizaciones del sistema, el comando `at` ofrece una solución sencilla y poderosa para la programación de trabajos únicos.
- Recuerde, el comando **`at`** es una herramienta versátil y poderosa, pero no es la única opción para programar tareas en Linux.
 - Para tareas recurrentes o requisitos de programación más complejos, considere usar el sistema **`cron`**, que brinda flexibilidad y control adicionales sobre la programación de tareas.

38
Alfredo Abad



Cron y crontab: El modo batch de un sistema Linux

39
Alfredo Abad



¿Para qué sirven cron y crontab?

- Cron es un demonio, es decir un programa que se ejecuta en segundo plano sin que el usuario tenga intervención.
- La función de cron es ejecutar, en un momento especificado previamente, una determinada tarea. La mayoría de las veces es por necesidades del sistema, aunque los usuarios podemos indicarle otras a partir de la edición de un archivo de texto conocido como crontab.
- En el post anterior habíamos dicho que los comandos para crear crontab son:
 - **crontab -e** para el usuario por defecto
- **O**
 - **crontab -u nombre_de_usuario** para cualquiera de los otros.
- Crontab es un archivo de texto que le brinda a Cron las instrucciones sobre qué debe hacer y cuando hacerlo.

40
Alfredo Abad



Acerca del uso de cron mediante crontab

- Para crear nuestro crontab debemos tener en cuenta lo siguiente:
 - Se utiliza una línea para cada tarea.
 - Se debe indicar la fecha y hora de ejecución de la tarea. En caso de que se trate de una tarea que requiera periodicidad. Por ejemplo, todos los miércoles a las 5 de la mañana, el resto de los parámetros se reemplazan por asteriscos (*).
 - En caso de que se quiera asignar más de un valor para un determinado parámetro cada valor debe separarse por una coma.
 - Los parámetros se separan con un espacio.
 - Se debe conocer el directorio donde está el lanzador del comando
- Por ejemplo, si queremos que el ordenador de nuestros hijos se apague todos los días a las 20, la instrucción sería:
 - **0 20 * * * /sbin/shutdown**
- En caso de que busquemos que el apagado sea solo los domingos cambiamos la instrucción a:
 - **0 20 * * 0 /sbin/shutdown**

41
Alfredo Abad



Existen algunos atajos que nos ahorran tener que escribir todos los parámetros

- **@hourly**: Ejecuta un comando a la hora en punto. Run once an hour (0 * * * *)
- **@daily**: Ejecuta el comando al comienzo de cada día. Run once a day (0 0 * * *)
- **@weekly**: Ejecuta el comando al comienzo del primer día de la semana. Run once a week (0 0 * * 0)
- **@monthly**: Ejecuta el comando al comenzar el primer día de cada mes. Run once a month (0 0 1 * *)
- **@yearly**: Ejecuta el comando en el primer minuto del año.
- **@reboot**: Run once after reboot
- Algunos ejemplos del uso de este comando son:
 - **@daily /bin/sh /ruta_al_script/nombre_del_script.sh** ejecuta un script en Bash.
 - **@hourly /bin/python3 /ruta_al_script/nombre_del_script.py** ejecuta un script en python cada hora.
- En todos los casos los scripts deberán tener permisos de ejecución.
- También deben especificarse la ruta completa a los ficheros ejecutables.

42
Alfredo Abad



cron y anacron: analogías y diferencias

- **cron** es lo que en los sistemas y derivados se conoce como un *daemon* es decir un programa que se ejecuta en segundo plano y sin intervención del usuario.
 - Está pensado para servidores, es decir equipos que funcionan en forma casi permanente sin necesidad de que nadie esté prestándole atención, pero requiere frecuentes tareas de mantenimiento.
- **anacron** es un programa normal **más apto para computadoras de escritorio** que no están continuamente encendidas.
 - Es por eso que, a diferencia de cron, donde la menor unidad de tiempo es el minuto, se trabaja con una frecuencia mínima de un día.
- Por el mismo motivo, **cron no incluye alternativas para el caso de que el equipo no esté encendido** mientras que anacron repasa las tareas pendientes al iniciar sesión.
 - Cuando encuentra un trabajo no iniciado en el momento establecido, ejecutará el comando especificado en el campo de comando después de esperar el número de minutos indicado en el campo de retraso.
 - Luego registrará la fecha en un archivo de marca de tiempo.
- Más información en: <https://geekland.eu/planificar-tareas-con-cron-y-anacron-en-linux/>

43
Alfredo Abad



Cron: How to Schedule Shutdown in Ubuntu Linux

<https://geekrewind.com/how-to-schedule-shutdown-in-ubuntu-linux/>



44
Alfredo Abad



Configuración de tareas en cron

Resolución de los errores más frecuentes

45
Alfredo Abad



Cron: de sistema y de usuarios

- En GNU/Linux, el programa cron es un demonio (o "servicio", como prefieran llamarle) para ejecutar tareas programadas, llamadas "**cronjobs**".
- En ocasiones puede sucedernos que creamos un cronjob, esperamos su ejecución programada, y en el log (/var/log/cron si se trata de Red Hat/Fedora/CentOS o /var/log/syslog si se trata de una distribución basada en Debian) nos encontramos con diferente tipo de errores
 - Por ejemplo: "(CRON) bad command", "/bin/sh: Desktop: command not found", "Error: bad username; while reading" o "/bin/sh: root: command not found".
 - Todos estos mensajes de error se producen cuando tenemos un problema en la sintaxis del cronjob.
- El demonio cron levanta tareas programadas de diferentes fuentes.
 - Primero busca los crontabs de los usuarios, los cuales se almacenan en el directorio /var/spool/cron en Red Hat/Fedora/CentOS, o el en directorio /var/spool/cron/crontabs en Debian y derivados, cuyos nombres de archivo coinciden con los nombres de login de los usuarios.
 - Luego levanta los crontabs de sistema, los cuales se almacenan en el archivo /etc/crontab y en el directorio /etc/cron.d.

46
Alfredo Abad



Directorios derivados de crontab

- La definición de tareas programadas es propensa a errores ya que los crontabs de usuario tienen una sintaxis diferente a la de los crontabs de sistema.
 - Los **crontabs de sistema** incluyen el campo "usuario" para determinar el UID (user ID del proceso) con el que se ejecuta cada tarea.
 - En los **crontabs de usuario** este campo no es necesario, porque cada usuario sólo puede ejecutar tareas programadas a su nombre.
- Por otro lado, desde el crontab de sistema se derivan los directorios **/etc/cron.hourly/**, **/etc/cron.daily/**, **/etc/cron.weekly/** y **/etc/cron.monthly/**
 - Los cuales se utilizan para ejecutar tareas automáticamente cada hora, día, semana o mes.
 - Pero estos directorios no guardan cronjobs, sino directamente los scripts o binarios que se desean ejecutar.

47
Alfredo Abad



Enlaces simbólicos en los directorios derivados de crontab

- Dentro del crontab de sistema se encuentran diferentes tareas programadas donde se definen el minuto en que se ejecutan los scripts cada hora (**cron.hourly**); la hora y minuto en que se ejecutan los script diarios (**cron.daily**); y la hora, minuto y día en que se ejecutan los scripts semanales y mensuales (**cron.weekly** y **cron.monthly**).
- Por ejemplo, si queremos ejecutar un script diariamente sólo es necesario almacenarlo (o poner un link simbólico) dentro del directorio **/etc/cron.daily**. Pero si deseamos ejecutarlo una vez al día, y a una hora en particular, necesitaremos definir nuestra propia tarea programa en nuestro crontab, o definir un cronjob de sistema en el archivo **/etc/crontab** o dentro del directorio **/etc/cron.d/**
- Esta cantidad de archivos y directorios pueden provocar confusión y llevarnos a cometer un error, como por ejemplo definir una tarea de sistema sin campo usuario, o una tarea de usuario con campo usuario (ambas incorrectas).

48
Alfredo Abad



Crontab de usuario

- Las tareas programadas de los usuarios se definen en el área spool, tal como se ha mencionado anteriormente. La forma más sencilla de editar el crontab del usuario es utilizando la herramienta crontab, la cual se utiliza para tal propósito.

Podemos listar las tareas programadas mediante crontab -l;

```
pepe@servidor07:~$ crontab -l
no crontab for pepe
```

Para editar el crontab debemos utilizar crontab -e, por ejemplo en Ubuntu Server:

```
pepe@servidor07:~$ crontab -e
no crontab for pepe - using an empty one
```

Select an editor. To change later, run 'select-editor'.

1. /bin/ed
2. /bin/nano <---- easiest
3. /usr/bin/vim.basic
4. /usr/bin/vim.tiny

Choose 1-4 [2]:

49
Alfredo Abad



- La primera vez que se edita pregunta qué editor deseamos utilizar (nano por defecto) y guarda nuestra preferencia en el archivo ~/.selected_editor.

Por ejemplo, selecciono 4 para utilizar el editor vi:

```
# m h dom mon dow  command
~
~
~
-- INSERT --
```

Suponiendo que el usuario "pepe" desea ejecutar cada 5 minutos el script "tarea01.sh", el cual se encuentra en el directorio /home/pepe/tareas_programadas/ y desea enviar tanto la salida estándar como la standard error al archivo de log /var/log/tarea01.log, la sintaxis es la siguiente:

```
# m h dom mon dow  command
*/5 * * * * ~/tareas_programadas/tarea01.sh 2>&1 >> /var/log/tarea01.log
~
~
~
-- INSERT --
```

50
Alfredo Abad



- Las líneas que comienzan con el caracter '#' son comentarios. La primera columna indica el minuto (de 0 a 59, "*" / 5" para que se ejecute cada 5 minutos), el resto de las columnas indican la hora (de 0 a 23), el día del mes (de 1 a 28 ó 31 dependiendo del mes), el mes (de 1 a 12) y el día de la semana (de 0 a 7, donde 0 y 7 equivalen a domingo).

En los campos día de la semana y mes es posible utilizar los tres primeros caracteres del nombre (sin importar el case). Por ejemplo "mon" o "Mon" para el lunes, "jun" para junio, etc.

También se permite utilizar listas y rangos, por ejemplo en el campo "día del mes" se podría especificar "2-4,8-10" para que la tarea se ejecute los días 2, 3, 4, 8, 9 y 10.

Para mayor información referirse al manual de crontab:

man 5 crontab

Luego de guardar y cerrar el editor podemos ver el listado de tareas nuevamente, para verificar que se haya actualizado correctamente:

```
pepe@servidor07:~$ crontab -l
# m h dom mon dow   command
*/5 * * * * /home/pepe/tareas_programadas/tarea01.sh 2>&1 >> /var/log/tarea01.log
```

51

Alfredo Abad



Crontab del sistema

- El crontab de sistema se utiliza generalmente cuando se requiere que una tarea se ejecute con privilegios de administrador (root).
 - La definición es igual a la de un crontab de usuario con la excepción de que se debe incluir el campo "usuario".
- Es posible editar directamente el archivo **/etc/crontab** aunque es más prolijo agregar una definición dentro del directorio **/etc/cron.d/** el cual sirve para almacenar cronjobs de sistema para diferentes usuarios.
- Por ejemplo podemos crear el archivo **/etc/cron.d/tarea01** con el siguiente contenido:
 - */5 * * * * root /home/pepe/tareas_programadas/tarea01.sh 2>&1 >> /var/log/tarea01.log**
 - Notar que tras la definición del día de la semana se ha agregado el nombre de usuario con el que se ejecutará el proceso que lleve a cabo al tarea programada, en este caso "root".
- Cada vez que definimos una nueva tarea programada dentro del directorio **/etc/cron.d/** el servicio cron la levanta automáticamente.
 - No es necesario reiniciar el servicio.

52

Alfredo Abad



Directorios cron.hourly, cron.daily, cron.weekly y cron.monthly

- Cada versión de toda distribución GNU/Linux utiliza diferentes horarios para ejecutar los scripts dentro de los directorios `/etc/cron.hourly/`, `/etc/cron.daily/`, `/etc/cron.weekly/` y `/etc/cron.monthly/`. Por ejemplo el sistema que utilizo como ejemplo (Ubuntu Server) ejecuta `cron.hourly` en el minuto 17, `cron.daily` a las 6:25, `cron.weekly` los domingos a las 6:47 y `cron.monthly` el primer día del mes a las 6:52:

```
root@servidor07:/home/pepe# cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

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

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

Recuerden que estos directorios no poseen definiciones de tareas programadas (cronjobs) sino que directamente contienen los scripts o binarios a ejecutar.

53
Alfredo Abad



El demonio cron lleva un log de actividad en el archivo `/var/log/cron` o directamente en el syslog.

Errores comunes: Asteriscos de más

- `* / 5 * * * * root /root/tareas_programadas/tarea01.sh 2>&1 >> /var/log/tarea01.sh`

Un error común es agregar un asterisco de más en la definición de timing de la tarea programada. En algunas distribuciones, por ejemplo en Slackware 13.1 veremos el error:

```
/bin/sh: Desktop: command not found
```

En otras versiones de cron veremos el error:

```
(CRON) bad command
```

54
Alfredo Abad



Errores más comunes: Campo usuario de más

- `*/5 * * * * root /root/tareas_programadas/tarea01.sh 2>&1 >> /var/log/tarea01.sh`

Este es el caso en el que agregamos el campo usuario en el crontab del usuario. Por supuesto, si se tratase de un cronjob de sistema la sintaxis sería la correcta. El error en el log suele ser:

`/bin/sh: root: command not found`

En algunas versiones de cron, por ejemplo la que se incluye en Slackware 13.1, no utiliza el campo "usuario" en los cronjobs de sistema, ya que por defecto se ejecutan como root. Por lo tanto veremos el mismo error y debemos remover el campo usuario para solucionar el problema.

55
Alfredo Abad



Errores más comunes: Falta un campo de usuario

- El caso inverso al anterior en el que nos falta agregar el campo usuario en un cronjob de sistema. Por supuesto, si se tratase del crontab del usuario la sintaxis sería la correcta. El error en el log suele ser:

`Error: bad username; while reading`

56
Alfredo Abad



Más información

- Para mayor información acerca de cron es recomendable ir a la fuente más precisa, que son los manuales:

```
$ man -a cron
$ man -a crontab
```




Cron cheatsheet for sysadmins

Format			Examples	
Min	Hour	Day Mon Weekday		
* * * * * script/command to be executed				
		Day of Week (0=Sun .. 6=Sat)		
		Month (1..12)		
		Day of Month (1..31)		
		Hour (0..23)		
		Minute (0..59)		
Minutes	0..59	the command/script would be executed at the specified minute.	0 * * * *	/opt/backup.sh perform a system backup every hour.
Hours	0..23	the command/script would be executed at the specified hour.	* /7 * * * *	/opt/ping.sh check if the remote server is online every 7 minutes.
Days	1..31	the days of the months in which the script or command would be executed.	0 */6 * * *	/opt/emptytrash.sh empty trash every 6 hours.
Months	1..12	the month in which the script would be executed.	20 14 * * *	/opt/upgrade upgrade the system at 14:20 PM of every day.
Weekdays	0..6	the days of the week in which the script gets executed. 0 is Sunday.	5 9 * 4 *	/opt/upgrade upgrade the system at 09:05 AM in April.
			20 14 * * ?	/opt/update.sh update system At 14:20 PM of every day.
			6 11 * * 3	/opt/upgrade.sh upgrade the system at 11:06 AM of every Wednesday.
			0 22 * * 1-5	/opt/upgrade.sh upgrade the system at 22:00 PM on every day-of-week from Monday through Friday.
			0 0 * * 2	/opt/upgrade.sh upgrade the system at midnight (00:00) of every Tuesday.
			10 8 * * 4L	/opt/monitor.sh monitor the system at 08:10 AM on the last Thursday of every month.
			15 0 * * 4#2	/opt/upgrade upgrade the system at at 00:15 AM on the second Thursday of every month.
			0 0 0 1 * *	/opt/backup.sh perform a sys backup every 1st of month (monthly).
			0 0 0 1 1 *	/opt/backup.sh perform a sys backup every 1st of January (yearly).
			5 12 * * 6	/opt/emptytrash.sh clears the trash at 12:05 PM on Sunday.
			@reboot	/opt/backup.sh perform a system backup at reboot.



Special strings

@reboot	command will be executed once at system startup (non-standard).
@hourly	command will be executed once an hour, same as ("0 * * * *") but non-standard.
@daily	command will be executed once each day, same as ("0 0 * * *") but non-standard.
@midnight	same as @daily but also non-standard.
@weekly	command will be executed once every week, same as ("0 0 * * 0") but non-standard.
@monthly	command will be executed once every month, same as ("0 0 1 * *") but non-standard.
@yearly	command will be executed once every year, same as ("0 0 1 1 *") but non-standard.
@annually	same as @yearly but also non-standard.



Crontab

crontab -e	Edit or create a crontab file if doesn't already exist.
crontab -l	Display the crontab file.
crontab -r	Remove the crontab file.
crontab -u username -l	D isplay another user's crontab file.
crontab -u username -e	Edit another user's crontab file.
crontab -v	Display the last time you edited your crontab file.


Asterik (*)	this operator is used to represent all potential values in a field. Write an asterisk in the Minute column, for example, if you want your cron job to execute every minute.
Hyphen (-)	to determine a range of values, use this operator. For example, if you want to set up a cron job from Monday through Friday, simply write 1-5 in the weekday column.
Slash (/)	to split a value, use this operator. For instance, if you want a script to run every 6 hours, enter */6 in the Hour field.
Comma (,)	to list numerous values, use this operator. Writing 1,5 in the Day of the week field, for example, will schedule the task to be executed every Monday and Friday.
Last (L)	this operator can be used in the month and weekday fields. Writing 6L in the day-of-week field, for example, signifies the last Saturday of the month.
Weekday (W)	this operator is used to get the closest weekday from a given time. If the 1st of the month is a Saturday, for example, entering 1W in the day-of-month field will execute the command on the following Monday (the 3rd).
Hash (#)	It is only permitted for the Day Of Week field, which must be followed by a number between 1 and 5. For instance, 5#2 denotes "the second Friday" of a given month.
Question mark (?)	can be used instead of '*' in the Day of Month and Day of Week fields. Use this operator to enter "no specified value" for the "day of the month" and "day of the week" fields.


Special characters

@linuxopsys

59

Alfredo Abad





How (and Why) to Replace cron Jobs With systemd Timers

<https://www.howtogeek.com/replace-cron-jobs-with-systemd-timers/>

60

Alfredo Abad

