MANUAL 5

SHELL SCRIPTS

PROGRAMAR TAREAS

1. Programación de tareas

A continuación, se describe cómo programar tareas rutinarias o únicas (una sola vez) del sistema mediante los comandos `crontab` y `at`.

Comando	Descripción
`crontab`	Programa varias tareas del sistema en intervalos regulares
`at`	Programa una tarea del sistema una sola vez

1.1. crontab

El comando `crontab` permite programar tareas rutinarias para que se ejecuten diariamente, semanalmente o mensualmente.

Algunos ejemplos de tareas que se podrían programar con el comando `crontab` son:

- Eliminar archivos de pocos días de antigüedad de directorios temporales.
- Realizar supervisiones de seguridad diaria.
- Ejecutar copias de seguridad del sistema.
- Etc.

1.1.1. Términos relacionados con crontab

- **demonio (daemon)**: Un demonio (daemon en inglés) es, generalmente, un programa que se ejecuta en segundo plano y que ofrece algún tipo de servicio.
- **cron**: el proceso demonio `cron` es un proceso que se ejecuta todo el tiempo en los sistemas UNIX o Linux. `cron` responde a eventos temporales y examina los archivos de configuración de los directorios `/var/spool/cron/crontabs` y `/etc/crontab` en busca de tareas que ejecutar.
- **/etc/crontab**: archivo de configuración cron que controla las tareas cron del sistema. Cada tarea cron se corresponde con una entrada crontab.
- **/var/spool/cron/crontabs**: directorio donde se almacenan los archivos de configuración `crontab` de cada usuario. Al crear un archivo `crontab`, éste se colocará automáticamente en el directorio `/var/spool/cron/crontabs` y recibirá su nombre de usuario.
- **Tarea cron**: comando o script que ejecuta `cron` a intervalos regulares. Se pueden dividir en tareas cron del sistema o del usuario:
    - **Tarea cron del sistema**: tarea cron que se ejecuta como root y suele realizar tareas de mantenimiento en todo el sistema.
    - **Tarea cron del usuario**: tarea cron ejecutada bajo la cuenta de un usuario y que realiza tareas designadas por el mismo, como ejecutar un programa a intervalos periódicos.

- **Entrada crontab**: Una entrada contrab es una línea de un archivo crontab que posee un formato especial y que especifica en qué minuto, hora, día de la semana y mes se debe ejecutar una tarea cron en particular.
- **crontab**: comando que permite crear, enumerar, modificar y eliminar archivos de configuración de tareas cron para usuarios y para el sistema.

NOTA:

`crontab` puede referirse a tres cosas diferentes pero relacionadas. Puede hacer referencia al comando `crontab`, al archivo de configuración `/etc/crontab` o al archivo de configuración que contiene las tareas cron del usuario (ubicado en `/var/spool/cron/crontabs`)

1.1.2. Estructura del archivo crontab

Los archivos `crontab` contienen una tarea por línea (las líneas de comentario que comiencen por # se ignoran). Cada una de estas líneas contiene siete campos separados por espacio en blanco. Los cinco primeros campos se utilizan para especificar la fecha y hora en que se ejecutará la tarea. El sexto campo es el nombre de la cuenta de usuario que se va a utilizar cuando se ejecute el programa. Y el séptimo campo es el comando a ejecutar.

La estructura de cada línea (entrada crontab) es la siguiente:

```
minuto hora diames mes diasemana usuario comando
```

donde:

- **minuto** se corresponde con el minuto en que se va a ejecutar la tarea (0-59)
- **hora** se corresponde con la hora en que se va a ejecutar la tarea (0-23)
- **diames** se corresponde con el día del mes en que se va a ejecutar la tarea (1-31)
- **mes**: se corresponde con el mes en que se va a ejecutar la tarea, puede ser numérico (1-12) o las tres primeras letras del mes en inglés: jan, feb, …, dec.
- **diasemana** se corresponde con el día de la semana en que se va a ejecutar la tarea, puede ser numérico (0-7, donde 0 y 7 son domingo) o las tres primeras letras del día en inglés: mon, tue, wed, thu, fri, sat, sun.
- **usuario** define el usuario que va a ejecutar el comando, puede ser root, u otro usuario diferente siempre y cuando tenga permisos de ejecución del script. (En una tarea cron del usuario, se puede no especificar el nombre del usuario utilizado para ejecutarla, como se hace con las tareas cron del sistema).
- **comando** se refiere al comando o a la ruta absoluta del script a ejecutar. Ejemplo: */home/usuario/miscript.sh*

NOTA:

La diferencia de formato entre `/etc/crontab` y los crontabs de usuario (ubicados en `/var/spool/cron/crontabs`) es que el `/etc/crontab` agrega el campo **usuario**, donde se especifica bajo que usuario se ejecutarán las tareas.

A la hora de añadir una entrada crontab hay que tener en cuenta lo siguiente:

- Se debe utilizar un espacio para separar cada campo.
- Se debe utilizar una coma para separar varios valores. Ejemplo: 0,6,12,18 coincidirá con cualquiera de los valores especificados.
- Se debe utilizar un guión para designar un rango de valores. Ejemplo: 9-17 en el campo de la hora especificará una hora que va desde las 9:00 a.m. a las 5 p.m.
- Se debe utilizar un asterisco como comodín para incluir todos los valores posibles.
- Se debe utilizar una barra (/) junto a otras opciones de múltiples valores para especificar valores escalonados, es decir, la distancia que hay entre los valores indicados. Ejemplo: un */10 en los minutos indica una tarea que se ejecuta cada 10 minutos.
- Se debe utilizar una marca de comentario (#) al principio de una línea para indicar un comentario o una línea en blanco.

Ejemplos de fecha y hora en una entrada cron:

`* * * * *`	Cada minuto
`*/20 * * * *`	Cada 20 minutos
`40 * * * *`	40 minutos después de cada hora (1:40, 2:40, etc.)
`40 8 * * *`	Cada día a las 8:40 am
`40 8 5 * *`	El quinto día de cada mes a las 8:40 am
`40 8 5 12 *`	Todos los 5 de diciembre a las 8:40 am
`40 8 5 dec *`	Todos los 5 de diciembre a las 8:40 am
`40 8 * * 6`	Todos los sábados a las 8:40 am
`40 8 * * sat`	Todos los sábados a las 8:40 am
`40 8 * 10-12 6`	Todos los sábados de octubre, noviembre y diciembre a las 8:40 am
`40 8 5 12 6`	Todos los sábados de diciembre y el 5 de diciembre a las 8:40 am

Ejemplo: Ejecuta el script `miscript.sh` ubicado en `/home/usuario/` a las 16:15 todos los martes

```
15 16 * * 2 /home/usuario/miscript.sh
```

1.1.3. Sintaxis del comando crontab

El comando `crontab` permite crear, enumerar, modificar y eliminar archivos de configuración de tareas cron para usuarios y para el sistema.

Su sintaxis es la siguiente:

`crontab -e`	Edita su archivo crontab
`crontab -l`	Imprime su archivo crontab
`crontab -r`	Elimina su archivo crontab

Es decir, para añadir una tarea en el fichero `crontab`, se ejecuta el comando `crontab -e`, que permite a un usuario particular mantener las entradas de su fichero `crontab` personal. Con `crontab -l` lista en la salida estándar todas las tareas cron del usuario actual (funciona como si utilizásemos un `cat`). Finalmente, si lo que se desea es eliminar un archivo `crontab` hay que ejecutar el comando `crontab -r`.

El usuario `root` puede añadir la opción `-u usuario` para trabajar con los archivos crontab de otros usuarios.

1.2. at

El comando `at` permite programar una tarea (comando o secuencia de comandos) para que se ejecute una sola vez a la hora especificada.

Para programar una tarea con el comando `at` los pasos son los siguientes:

1. Ejecutar el comando `at` y especificar la hora en que se desea ejecutar el trabajo.

```
$ at time [date]
```

`time`

Especifica la hora en que se desea programar el trabajo. Si no se especifica la hora según el reloj de 24 horas entonces hay que añadir `am` o `pm`. Se aceptan horas en diferentes formatos: `8pm`, `8 pm`, `8:00pm`, `8:00 pm`, `20:00`.

`date`

Especifica las primeras tres (o más) letras de un mes o un día de la semana, o las palabras clave `today` o `tomorrow`. Si solo se especifica hora y no se especifica fecha se asume la siguiente instancia, ya sea hoy o mañana. Se aceptan fechas en diferentes formatos: `december 25 2019`, `25 december 2019`, `december 25`, `25 december`, `12/25/2019`, `25.12.2019`, `thursday`, etc.

2. Escribir los comandos o las secuencias de comandos que se desean ejecutar (uno por línea) en el indicador `at`, que se muestra con los símbolos `at>`

3. Salir del comando `at` y guardar el trabajo programado presionando `Control+D`. Al trabajo `at` se le asigna un número de cola. Este número se muestra al salir del comando `at`.

Ejemplo: El siguiente trabajo `at` elimina archivos con extensión `.cpp` de la cuenta de usuario `juan` a las 23:45 del último día del año.

```
$ at 11:45pm December 31
at> rm /home/juan/*.cpp
at> Press Control-d
commands will be executed using /bin/sh
job 5 at Tue Dec 31 23:45:00 2019
```

Ejemplo: Se programa para las 4:00 a.m del sábado la ordenación del archivo `/etc/passwd` y el resultado se redirige a un archivo llamado `ordenacion.txt` ubicado en `/home/juan`

```
$ at 4 am Saturday
at> sort /etc/passwd > /home/juan/ordenacion.txt
```

Para enumerar los trabajos programados pendientes de ejecutarse hay que utilizar el comando `atq`

Ejemplo:

```
$ atq
9 Fri Nov 20 09:50:00 2019 a usuario
10 Fri Nov 20 09:51:00 2019 a usuario
```

Para eliminar un trabajo se puede utilizar el comando `atrm` o `at -r`, seguido del número de tarea

Ejemplo: Elimina el trabajo con número de cola 10

```
$ atrm 10
```

2. Otros comandos

2.1. sleep

El comando `sleep` espera una cantidad de tiempo concreta. Es muy útil para retrasar un comando durante un tiempo determinado. El tiempo se especifica mediante un entero (segundos), o mediante un entero seguido de la letra s (también para expresar segundos), m (minutos), h (horas) o d (días).

Ejemplo: Provoca una espera de 5 segundos

```
$ sleep 5
```

Ejemplo: Provoca una espera de 5 segundos

```
$ sleep 5s
```

Ejemplo: Provoca una espera de 3 minutos

```
$ sleep 3m
```

Ejemplo: Provoca una espera de 1 hora

```
$ sleep 1h
```

Ejemplo: Retrasa el comando `echo` durante 5 segundos

```
$ sleep 5s && echo "Han pasado 5 segundos"
```

2.2. watch

El comando `watch` permite ejecutar un comando especificado como parámetro a intervalos regulares. Por defecto, el intervalo predeterminado es cada dos segundos.

Opciones más importantes:

`-n segundos`	Define en segundos el tiempo entre ejecuciones
`-d`	Resalta las diferencias en el resultado mostrado por pantalla para enfatizar los cambios que se han producido entre las ejecuciones.

Ejemplo: Establece un tiempo de 5 segundos entre cada ejecución del comando date

```
$ watch -n 5 date
```

Ejemplo: Establece un tiempo de 5 segundos entre cada ejecución del comando date y resalta las diferencias en la salida entre las ejecuciones.

```
$ watch -d -n 5 date
```