Cron y Anacron

Universidad: Universidad Nacional de la Patagonia San Juan Bosco

Cátedra: Sistemas Operativos

Tema: Cron y Anacron

Alumnos

- Arcos Martin
- Maldonado Abel

Profesores

- Gómez Marcelo
- Marticorena Lucy
- Tosca Maximiliano

Año: 2022

ÍNDICE

Introducción

En el presente trabajo indagaremos y comentaremos acerca de las herramientas utilizadas para la ejecución programada de tareas en Linux, estas herramientas son denominadas Cron y Anacron.

Antes de introducirnos en el tema es necesario saber qué es crontab y en qué consiste.

Crontab

Crontab es el archivo de texto en donde se listan todas las tareas que deben ejecutarse y el momento en el que deben hacerlo. Esto lo que hace es verificar las fechas y horas en que se deben ejecutar las tareas, los permisos de ejecución y los realiza en background (segundo plano). Cada usuario del equipo posee su propio crontab, de hecho el */etc/crontab* se asume que es el archivo crontab del usuario root, cuando los usuarios normales (e incluso root) desean generar su propio archivo de crontab, entonces se utiliza el comando “crontab”.

Crontab facilita la tarea de gestionar las tareas y programas en un sistema multiusuario, como usuario root o un usuario simple del sistema.

Ejemplo

Vamos empezando con un ejemplo simple.

Vamos a automatizar la actualización de un sistema, para eliminar la molestia de “siempre tengo que estar actualizando y eso no me gusta!”.

Primero que nada haremos un script. Este script será llamado por cron y contendrá todas las instrucciones que queremos que haga, por lo tanto es necesario probarlo en varios casos y de varias formas antes de incluirlo a cron, un sencillo script de actualización como el siguiente:

```
!/bin/bash
script ejemplo de actualización
elija su distribución
debian-ubuntu
apt-get update & apt-get -y upgrade
fedora
yum -y update
Arch
pacman --noconfirm -Syu
```

Guardamos el script como actualizacion.sh (ej. directorio scripts el home). Cambiamos los permisos de ejecución del dichoso script con:

chmod a+x *~/scripts/actualizacion.sh*

Ejecutamos el script para verificar que todo ejecute sin problemas, modificamos lo necesario (no debe contener errores, si no cron solo repetirá un error una y otra vez). Ahora a agregar la tarea a nuestro crontab.

Como vemos crontab es una herramienta útil que se puede manejar desde la línea de comandos de Linux.

Administración de trabajos en crontab

- **crontab *archivo***: Reemplaza al existente archivo crontab con un archivo definido por el usuario
- **crontab -e**: Editar el archivo crontab del usuario, cada linea nueva será una nueva tarea de crontab.
- **crontab -l**: Crea una lista de todas las tareas de crontab del usuario
- **crontab -d**: Esto borra el crontab del usuario
- **crontab -c *directorio***: Define el directorio y crontab del usuario (este debe tener permisos de escritura y ejecución del usuario)
- **crontab -u *usuario***: prefijo para manejar el crontab de otro usuario, ejemplos:
    - $ sudo crontab -l -u root
    - $ sudo crontab -e usuario2

En un crontab, es importante especificar un nivel de seguridad mediante crontab -s o especificando el nivel requerido en la primera línea del crontab. Cada nivel se especifica en */etc/selinux/targeted/seusers*. Al usar crontab en el modo MLS, es especialmente importante:

- verificar/cambiar el rol actual,
- establecer el rol correcto para el directorio, que se utiliza para entrada/salida.

Cron

El nombre cron viene del griego chronos que significa “tiempo”. Sus primeras versiones fueron en Unix versión 7, escrita por Brian Kernighan.

Cron es un administrador regular de procesos (o tareas) en segundo plano que permite ejecutar comandos en un intervalo regular de tiempo, por ejemplo, cada minuto, cada hora, día, semana o mes. Si deseamos trabajar con cron, podemos hacerlo a través del comando crontab.

1.1. Funcionamiento de cron

Cron inicia en */etc/rc.d/* ó */etc/init.d* dependiendo de la distribución.

Cron se ejecuta en background (segundo plano), su funcionamiento consiste en buscar en */var/spool/cron* archivos crontab que tengan nombres de cuentas en */etc/passwd*; Los crontabs encontrados se cargan en la memoria. Cron también busca en el directorio */etc/anacrontab* y cualquier archivo en el directorio */etc/cron.d*, que tiene un formato distinto. Cron examina todos los crontabs almacenados y verifica cada trabajo para ver si necesita ejecutarse en el minuto actual.

Como usuario podemos agregar comandos o scripts con tareas a cron para automatizar algunos procesos. Esto es útil por ejemplo para automatizar la actualización de un sistema o un buen sistema de respaldos.

1.2. Configuración de tareas en cron

Se puede crear o editar tareas Cron de forma manual, desde consola, utilizando comandos específicos, o bien hacerlo de forma muy sencilla desde páginas web destinadas a la creación de tareas en cron.

1.2.1. Configuración manual

Para poder configurar una tarea cron de forma manual, tendríamos que ejecutar el siguiente comando: “crontab -e”.

La primera vez que abrimos este fichero, tendremos que seleccionar el editor que deseamos utilizar:

usuario@hl123456:~$ crontab -e

no crontab for dominio-ejemplo - using an empty one

Se selecciona un editor, y si el usuario lo desea después puede cambiar a un “editor mejor” o de su preferencia.

1. /bin/nano <---- “mejor”
2. /usr/bin/vim.basic
3. /usr/bin/vim.tiny

Una vez seleccionado el editor deseado, se abrirá el fichero para poder insertar el cron que deseamos programar.

1.2.1.1. Formato y sintaxis

El formato de configuración de cron es el siguiente:

```
*  *  *  *  *  user comando_a_ejecutar
|  |  |  |  |
|  |  |  |  |___día de la semana (1 - 6) (0 o 7: domingo)
|  |  |  |______mes (1 - 12)
|  |  |_________dia del mes (1 - 31)
|  |____________hora (0 - 23)
|_______________minutos (0 - 59)
```

Donde *user* define el usuario que va a ejecutar el comando, puede ser utilizado por el superusuario y especificar en qué usuario se ejecutará la tarea, este campo para el usuario "normal" puede omitirse.

Si queremos especificar todos los valores posibles de un parámetro se utiliza un asterisco (*), que es un comodín para cada parte de la expresión de programación cron.

* = siempre

Esto implica que si en lugar de un número utilizamos un asterisco, el comando indicado se ejecutará cada hora de cada día de cada mes y cada día de la semana, como en el siguiente ejemplo:

** 1 * * * /home/hosting/miscript.sh*

" * 1 * * * " significa que cron se ejecutará cada minuto que contenga la hora 1, es decir, 1:00, 1:01,… 1:59.

La última columna corresponde al comando que se quiere ejecutar. Recomendamos antes de configurar el comando en la tarea cron, comprobar manualmente que ese comando específico funciona correctamente.

1.2.1.2. Diferentes opciones del comando crontab

- **crontab -e**: edita el archivo crontab de un usuario. Cada línea que se configure será una tarea Cron.
- **crontab -l**: lista el archivo crontab de usuario, con todas sus tareas configuradas.
- **crontab -r**: elimina el archivo crontab de un usuario. El borrado no es recuperable.

1.2.1.3. Algunos ejemplos

Acción a realizar	Línea de ejecución en el crontab
Programamos un script que realiza una copia de seguridad todos los días a las 02:00 a.m	*00 02 * * * /home/hosting/backup.sh*
Ahora programamos el mismo script que realiza un backup, pero limitado al sábado y al domingo a las 03:00 de la mañana	*00 03 * * 6,7 /home/hosting/backup.sh*
Ejecutamos el script cron.sh todos los miércoles a las 12:20 y guardar la salida en el fichero cron.log:	*20 12 * * 3 /home/usuario/cron.sh >> /home/usuario/cron.log*
Si queremos indicar dos o más valores en cada parámetro, es necesario separarlos por comas. Por ejemplo, para ejecutar el script cron.sh todos los miércoles a las 12:10 h y a las 12:20 h:	*10,20 12 * * 3 /home/usuario/cron.sh*
Si queremos indicar que se ejecute el fin de semana cada 15 minutos podemos hacerlo de dos formas diferentes. Como hemos hecho en el ejemplo anterior, con comas O bien usando la sintaxis */15	*0,15,30,45 * * * 6,7 /home/usuario/cron.sh* **/15 * * * 6,7 /home/usuario/cron.sh*
Este es un ejemplo de ejecución de un fichero PHP cada 15 minutos:	**/15 * * * * /home/usuario/.bin/php /home/usuario/fichero.php*
Si queremos que se ejecuten dos comandos, de forma consecutiva, separaremos los comandos con punto y coma	**/15 * * * 6,7 /home/usuario/cron.sh; /home/usuario/backup.sh*
En el ejemplo que figura a continuación programamos la descarga de un fichero, todos los días, el día 8 de mayo a las 07:15 h	*15 7 8 5 * wget -N http://dominio-ejemplo.com/documentacion.tar.gz*
La descarga de un fichero solo se realizará el día 8 de mayo a las 07:15 h si ese día coincide que es lunes	*15 7 8 5 1 wget -N http://dominio-ejemplo.com/documentacion.tar.gz*

Existen varios valores predefinidos que podemos utilizar para sustituir algunas expresiones, evitando tener que configurarlas:

VALOR	TIEMPO DE EJECUCIÓN	EQUIVALENCIA
@yearly	Una vez al año	0 0 1 1 *
@annually	Una vez al año	0 0 1 1 *
@monthly	Una vez al mes	0 0 1 * *
@weekly	Una vez a la semana	0 0 * * 0
@daily	Una vez al día	0 0 * * *
@midnight	Una vez al día	0 0 * * *
@hourly	Una vez, cada hora	0 * * * *

1.2.2. Herramientas para Cron

Con el objetivo de automatizar tareas en Linux y evitar errores, se han diseñado ciertas herramientas en línea de formato cron, como por ejemplo:

1. Easycron
2. Cron Job Generator[1]
3. Contrab Guru[2]

1.2.2.1. Easycron

Programador de tareas para llamar URL específicas por intervalos de tiempo o momentos determinados. Esta herramienta te permite administrar trabajos cron en tu panel de usuario, programar las labores de acuerdo con la fecha y hora de tu sistema, ver registros en ejecución de cada tarea, e incluso recibir avisos vía mail acerca de la ejecución de trabajos cron.

1.2.2.2. Cron Job Generator

Existen herramientas en páginas webs donde se puede generar entradas para el crontab. Esto le permite crear trabajos cron para tareas controladas por tiempo de manera más fácil y rápida sin tener que preocuparse por la sintaxis de crontab.permite programar secuencias de comandos para que se ejecuten en una hora determinada.

1.2.2.3. Contrab Guru

Editor rápido y sencillo para expresiones de programación cron. Brinda alertas instantáneas acerca de posibles errores o inconvenientes que se presenten.

[1] https://www.generateit.net/cron-job/
[2] https://crontab.guru/

1.3. Archivos de log

Para consultar la **bitácora** de las tareas ejecutadas por cron se consulta la lista de tareas definidas en las ubicaciones:

- */etc/anacrontab*
- */etc/cron.daily*
- */etc/cron.hourly*
- */etc/cron.monthly*
- */etc/cron.weekly*

Hay dos formas de comprobar los cambios en crontables. El primer método es comprobar el tiempo de modificación de un archivo. El segundo método es usar el soporte de inotify. El uso de inotify se registra en el registro */var/log/cron* después de iniciar el daemon. El soporte de inotify verifica los cambios en todos los cronables y accede al disco duro solo cuando se detecta un cambio.

Esto es posible con la instrucción "*cat /var/log/syslog* | *grep cron*", esto mostrará un listado con todas las interacciones del usuario con cron, crontab y anacron, mostrando el nombre del archivo, fecha, hora y qué acción se realizó, así sea una modificación, la ejecución o una falla en la tarea.

1.4. ¿Cómo saber si cron o un trabajo cron se está ejecutando?

Para comprobar si cron se está ejecutando, se busca en los procesos en ejecución con el comando ps. El comando de cron se mostrará en la salida como "crond". La entrada en esta salida para grep cron se puede ignorar, pero la otra entrada para crond se puede ver ejecutándose como root.

Y para comprobar que un trabajo cron se está ejecutando al finalizar el trabajo, el archivo /ruta /cron.end tendrá la marca de tiempo cuando finalice el cron. Entonces, con el comando ls -lrt /path/cron.{start, end} dirá cuándo comenzó el trabajo y si aún se está ejecutando (la orden dirá si aún se está ejecutando).

1.5. Mecanismo de seguridad

En los archivos crontab tienen que ser archivos regulares o enlaces simbólicos a archivos regulares, no deben ser ejecutables o escribibles para nadie más que el propietario.

Cron requiere que cada entrada en un crontab termine en un carácter de nueva línea. Si a la última entrada en un crontab le falta una nueva línea (es decir, terminada por EOF), cron considerará que el crontab (parcialmente) está roto. Por consiguiente se escribirá una advertencia en syslog (protocolo estándar utilizado para enviar mensajes de registro o eventos del sistema a un servidor específico). Además cron admite el control de acceso con PAM si el sistema tiene PAM instalado.

Existe un mecanismo por default que se aplica para que algunos usuarios puedan o no crear tareas en crontab.

Los archivos usados son: *cron.deny* (ubicado en */etc/cron.deny*) y *cron.allow* (ubicado en */etc/cron.allow*). El archivo allow no viene por defecto instalado por lo que hay que crearlo (touch /etc/cron.allow), solo se cuenta con el archivo deny, el cual es usado para denegar/bloquear los permisos de los usuarios, mientras que el archivo allow es donde se reflejan los usuarios con permisos.

Para denegar el acceso a un usuario primero es necesario convertirse en super-usuarios, luego se debe editar el archivo *deny* colocando dentro el nombre de los usuarios a bloquear (en el caso de querer bloquear a todos los usuarios se debe colocar la palabra "ALL"). Mientras que en el archivo *allow* se debe reflejar el nombre de los usuarios que sí van a poder crear archivos en crontab.

Para deshacer el bloqueo se puede borrar el nombre del usuario bloqueado del archivo deny asi como tambien limpiar el mismo, otra opción es colocando el nombre del usuario en el archivo allow para que este obtenga el permiso (esto solo sirve si el archivo allow se configuró de forma que todos los usuarios estén bloqueados).

1.6. Ventajas y Desventajas

Una de las ventajas que nos ofrece cron se encuentran las que puede ser creado o modificado tanto por los usuarios como por el superusuario root, otra ventaja que tenemos es que se está ejecutando continuamente en segundo plano y además sirve para ejecutar procesos periódicos a "deshora" o en una hora programada que no afecte al rendimiento de los usuarios.

La principal desventaja es que el equipo necesariamente tiene que estar encendido para que pueda realizar las tareas.

Anacron

Cron es interesante en la administración de sistemas, pero lo es solo en equipos de alta disponibilidad, es decir, equipos que estén en constante funcionamiento. Para solucionar esto en equipos “normales” que pasan periodos de tiempo apagados, existe la herramienta llamada **anacron,** que no es más que un programa que complementa a Cron y se ejecuta de forma periódica a través de:

- Tareas programadas de Cron.
- Scripts que se ejecutan en el momento de arrancar nuestro equipo.

El servicio anacron puede ser utilizado, **sólo por el usuario root**, para ejecutar comandos periódicamente, con una frecuencia de días. Cuando anacron se ejecuta (al iniciarse el sistema o bien manualmente), lee las tareas programadas únicamente del fichero */etc/anacrontab*. Este fichero es parecido a un fichero crontab, en su interior puede haber igualmente comentarios y variables (PATH, SHELL, MAILTO, HOME, etc.), pero las líneas que especifican la tarea y el momento de ejecución, varían en su sintaxis.

Cuando los comandos terminan de ejecutarse, anacron registra la fecha de finalización en un fichero dentro del directorio */var/spool/anacron* para chequear el momento de la próxima ejecución. Estos ficheros tienen por nombre el identificador del comando.

En el fichero */etc/anacrontab* definimos que una tarea se tiene que realizar cada X días. En el caso que haya alguna tarea que no se haya realizado se ejecutará de forma completamente automática. Una vez realizada la tarea se guardará un registro con la fecha de ejecución en la ubicación */var/spool/anacron*.

2.1. Planificación de tareas

Las tareas ejecutadas por Anacron se definen en el fichero */etc/anacrontab*. Para editar este fichero y añadir una tarea se ejecuta el siguiente comando en la terminal:

sudo nano /etc/anacrontab

Una vez abierto el editor de texto se tendrá que programar la tarea completando ciertos campos.

2.1.1. Formato y sintaxis

El formato de configuración de cron es el siguiente:

dias_de_retraso minutos_de_retraso identificador comando

Cada uno de los parámetros que se acaban de citar se deberán sustituir por los siguientes valores:

Parámetro	Explicación del parámetro
días_de_retraso	Indicamos la frecuencia en días con que queremos que se ejecute la tarea. Si la ejecución de la tarea tiene que ser diaria se escribe 1 o @daily. Si es semanal se escribiremos 7 o @weekly, si es mensual se escribe 30 o @monthly. Si se quiere que se ejecute cada dos días escribiremos 2.
minutos_de_retraso	Es un número entero que indica los minutos que transcurren desde que Anacron detecta que hay que ejecutar el comando o script hasta que se ejecuta. El retraso entre la detección de ejecución y la ejecución puede ser útil para evitar sobrecargas de CPU y memoria si se tienen que realizar muchas tareas de forma simultánea.
identificador	Un nombre cualquiera que describe e identifica la tarea que se ejecutará. Este nombre nos servirá para, por ejemplo, buscar información en los logs.
comando	Detalla el comando o script que queremos ejecutar.

Por lo tanto, para realizar una copia de seguridad semanal mediante un script se tendrá que usar el siguiente código:

```
.------------------------------------------------- Dias_de_retraso
| .------------------------------------------------- Minutos_de_retraso
| |  .------------------------------------------------- Identificador
| |  |                          .----------------------------- Comando
| |  |                          |
7 10 scriptBackup.weekly        /bin/bash /home/user/desktop/scriptBackup.sh
```

Una vez introducido el texto se guardan los cambios y se cierra el fichero. A partir de entonces: A los 7 días, o en el momento que se abra el ordenador y se detecte que llevo más de 7 días sin realizar la copia, Cron llamará a Anacron y se iniciará el proceso para realizar la copia de seguridad. Una vez toque iniciar la copia se realizará una pausa de 10 minutos y acto seguido se realizará la copia.

Podemos añadir parámetros adicionales en el fichero de configuración:

- Si debajo de LOGNAME= root añadimos START_HOURS_RANGE=3-22, entonces los trabajos únicamente se ejecutarán entre las 3 de la madrugada y las 10 de la noche. Si queremos conseguir lo mismo sin añadir ningún parámetro también podríamos modificar la configuración de Cron.
- Otra opción es añadir el parámetro RANDOM_DELAY=10. Lo que hará este parámetro es añadir un retardo aleatorio adicional de 0 a 10 minutos a cada una

de las tareas que se van a ejecutar. De esta forma se repartirá mejor la carga del sistema.

2.1.2. Diferentes opciones del comando anacrontab

- -f → Fuerza la ejecución de todos los trabajos, ignorando las marcas de tiempo.
- -u →Actualiza las marcas de tiempo de todos los trabajos a la fecha actual, pero no ejecuta ninguno.
- -s → Serializa la ejecución de trabajos. Anacron no inicia un nuevo trabajo antes de que finalice el anterior.
- -n → Ejecuta trabajos inmediatamente e ignora los retrasos especificados en el archivo /etc/anacrontab. Esta opción implica -s.
- -d → No bifurca a Anacron en segundo plano. En este modo, Anacron generará mensajes informativos a error estándar, así como a syslog. Anacron envía por correo el resultado de cualquier trabajo.
- -q → Suprime cualquier mensaje de error estándar. Solo aplicable con -d.
- -t some_anacrontab → Utiliza el anacrontab especificado, en lugar del predeterminado de /etc/anacrontab.
- -T → Prueba Anacrontab. Comprueba la validez del archivo de configuración /etc/anacrontab. Si hay un error en el archivo, se muestra en la salida estándar y Anacron devuelve el valor de 1. Los anacrontabs válidos devuelven el valor de 0.
- -S spooldir → Utiliza el spooldir especificado para almacenar las marcas de tiempo. Esta opción es necesaria para los usuarios que deseen ejecutar anacron ellos mismos.
- -V → Imprime la información de la versión y sale.
- -h → Imprime un mensaje de uso corto y sale.

2.3. Ventajas y desventajas de Anacron

Las ventajas que ofrece anacron es que el equipo no necesita estar encendido todo el tiempo para realizar las tareas, acumula las tareas no ejecutadas y las ejecuta cuando el equipo se enciende.

Las desventajas que presenta son que solo puede ser accedido por el usuario root; ya que no es un “demonio”, no está continuamente activo en segundo plano, solo se activa a través de llamadas de Cron o por un script en el arranque del sistema operativo

Diferencias entre cron y anacron

Una vez descrito qué es Cron y Anacron se detalla en la siguiente tabla las principales diferencias.

Cron	Anacron
Permite ejecutar tareas en una fecha y minuto determinado. Se tiene un control total de la hora y fecha en que se ejecutará la tarea.	Permite planificar tareas en periodos de tiempo diarios, semanales o mensuales. Ejecuta tareas que hace más de un determinado número de días que no se ha realizado.
El equipo tiene que estar encendido en el momento que toca realizar la tarea. En caso contrario la tarea nunca será realizada.	Si en el momento de realizar la tarea el equipo está apagado, la tarea se realizará cuando el equipo se encienda y Cron llame a Anacron.
Está constantemente activo en segundo plano.	Solo se activa a través de llamadas de Cron o por un script en el arranque del sistema operativo.
Todo usuario puede planificar tareas con Cron.	Únicamente los usuarios con privilegios de administrador pueden planificar tareas con Anacron.
Prácticamente la totalidad de distribuciones Linux traen Cron instalado de serie.	Es posible que Anacron no se instale de forma predeterminada. En caso que no esté instalado y lo quieran usar tan solo tienen que instalar el paquete anacron mediante el comando “*sudo apt install anacron.”*

Conclusión

A través de este informe se presentó brevemente las herramientas utilizadas para la ejecución programada de tareas en Linux. La planificación de tareas en Cron es especialmente útil para equipos que se mantienen en constante funcionamiento las 24 horas de los 365 días del año. Como contrapartida, planificar tareas con Cron en combinación con Anacron es especialmente útil en equipos que se apagan con frecuencia. Por lo tanto se recomendaría:

- Usar cron para programar tareas en equipos como lo son servidores.
- Usar anacron en combinación con cron para programar tareas en equipos personales o de oficina.

Bibliografía

- https://www.sudo.ws/
- https://www.sudo.ws/man/1.8.27/sudoers.man.html
- https://www.poftut.com/what-is-sudo-command-in-linux/
- https://platzi.com/clases/2292-terminal/37619-procesos-en-foreground-y-background/
- https://www.generateit.net/cron-job/
- https://www.elmundoenbits.com/2012/03/control-de-seguridad-en-crontab.hhttps://dinahosting.com/ayuda/como-configurar-tareas-cron-de-forma-manual/
- https://dinahosting.com/ayuda/tareas-cron-desde-panel-control/
- https://blog.desdelinux.net/cron-crontab-explicados/
- https://ayudalinux.com/comandos-cron-crontab-para-que-sirven/amp/#Que_es_Crontab
- https://www.compuhoy.com/como-se-si-se-esta-ejecutando-un-trabajo-cron-en-linux/#%C2%BFComo_puedo_saber_si_cron_se_esta_ejecutando_en_Linux
- https://dinahosting.com/ayuda/como-configurar-tareas-cron-de-forma-manual/
- https://www.fpgenred.es/cron/anacron.html
- https://geekland.eu/planificar-tareas-con-cron-y-anacron-en-linux/
- https://www.youtube.com/watch?v=1ert9_l4MpY&ab_channel=ElConocimientoTotalmenteLibre