CRON

Lucain publicó hace ya un tiempo un excelente tutorial sobre cron y crontab que me parece vale la pena compartir. Cron es una suerte de equivalente a Tareas Programadas en Windows, sólo que se maneja desde el terminal. Aquellos que prefieran una interfaz visual para lograr el mismo objetivo, pueden ver este otro artículo.

¿Qué es cron?

El nombre cron viene del griego chronos que significa “tiempo”. En el sistema operativo Unix, cron es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab.

Cómo funciona

El demonio cron inicia de /etc/rc.d/ o /etc/init.d dependiendo de la distribucion. Cron se ejecuta en el background, revisa cada minuto la tabla de tareas crontab /etc/crontab o en /var/spool/cron en búsqueda de tareas que se deban cumplir. Como usuario podemos agregar comandos o scripts con tareas a cron para automatizar algunos procesos. Esto es util por ejemplo para automatizar la actualizacion de un sistema o un buen sistema de respaldos.

¿Qué es Crontab?

Crontab es un simple archivo de texto que guarda una lista de comandos a ejecutar en un tiempo especificado por el usuario. Crontab verificará la fecha y hora en que se debe ejecutar el script o el comando, los permisos de ejecución y lo realizará en el background. Cada usuario puede tener su propio archivo 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 utilizaremos el comando crontab.

Crontab es la manera mas sencilla de administrar tareas de cron en sistemas multiusuario, ya sea como simple usuario de sistema o usuario root.

Utilizando crontab

Vamos empezando con un ejemplo simple.

Vamos a automatizar la actualización de un sistema, para eliminar la molesta de “siempre tengo que andar 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 actualizacion como este:

```
#!/bin/bash
#script ejemplo de actualizacion
#elija su distribucion
```

#debian-ubuntu

#apt-get update & apt-get -y upgrade

#fedora

#yum -y update

#Arch

#pacman --noconfirm -Syu

Quitale el # a la línea de tu distro. En caso de que sea Ubuntu/Debian, a la que empieza con apt-get.

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

chmod a+x ~/scripts/actualizacion.sh

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

Agregar tareas a crontab

Ejecutamos la edición del crontab con crontab -e, en algunas distros (como ubuntu) nos da la opcion de elegir el editor de textos que deseemos, los demás nos quedamos con vi. El archivo crontab lucirá algo asi.

m h dom mon dow user command

donde:

m corresponde al minuto en que se va a ejecutar el script, el valor va de 0 a 59

h la hora exacta, se maneja el formato de 24 horas, los valores van de 0 a 23, siendo 0 las 12:00 de la medianoche.

dom hace referencia al día del mes, por ejemplo se puede especificar 15 si se quiere ejecutar cada dia 15

dow significa el día de la semana, puede ser numérico (0 a 7, donde 0 y 7 son domingo) o las 3 primeras letras del día en inglés: mon, tue, wed, thu, fri, sat, sun.

user 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.

command refiere al comando o a la ruta absoluta del script a ejecutar, ejemplo: /home/usuario/scripts/actualizar.sh, si acaso llama a un script este debe ser ejecutable

Para que quedara claro unos cuantos ejemplos de tareas de cron explicados:

15 10 * * * usuario /home/usuario/scripts/actualizar.sh

Ejecutará el script actualizar.sh a las 10:15 a.m. todos los días

15 22 * * * usuario /home/usuario/scripts/actualizar.sh

Ejecutará el script actualizar.sh a las 10:15 p.m. todos los días

00 10 * * 0 root apt-get -y update Usuario root

Ejecutará una actualización todos los domingos a las 10:00 a.m

45 10 * * sun root apt-get -y update

Usuario root ejecutará una actualización todos los domingos (sun) a las 10:45 a.m

30 7 20 11 * usuario /home/usuario/scripts/actualizar.sh

El día 20 de noviembre a las 7:30 el usuario correra el script

30 7 11 11 sun usuario /home/usuario/scripts/pastel_con_velitas.sh

El día 11 de noviembre a las 7:30 a.m. y que sea domingo, el usuario festejará su sysadmin (o sea a mí)

01 * * * * usuario /home/usuario/scripts/molestorecordatorio.sh

Un molesto recordatorio cada minuto de cada hora todos los días (NO recomendable).

Igual se pueden manejar rangos especiales:

30 17 * * 1,2,3,4,5

A las 5:30 de la tarde todos los días de lunes a viernes.

00 12 1,15,28 * *

A las 12 del día todos los días primero, quince y 28 de cada mes (ideal para nóminas)

Si esto resulta confuso, crontab maneja cadenas especiales para definir estos rangos.

@reboot Ejecuta una vez, al inicio

@yearly ejecuta sólo una vez al año: 0 0 1 1 *

@annually igual que @yearly

@monthly ejecuta una vez al mes, el día primero: 0 0 1 * *

@weekly Semanal el primer minuto de la primer hora de la semana. 0 0 * * 0″.

@daily diario, a las 12:00A.M. 0 0 * * *

@midnight igual que @daily

@hourly al primer minuto de cada hora: 0 * * * *

Su uso es muy sencillo.

@hourly usuario /home/usuario/scripts/molestorecordatorio.sh

@monthly usuario /home/usuario/scripts/respaldo.sh

@daily root apt-get update && apt-get -y upgrade

Por último y no menos importante:

Administracion de trabajos en cron

crontab archivo

Remplaza el existente archivo crontab con un archivo definido por el usuario

crontab -e

Editar el archivo crontab del usuario, cada linea nueva sera una nueva tarea de crontab.

crontab -l

Lista todas las tareas de crontab del usuario

crontab -d

Borra el crontab del usuario

crontab -c dir

Define el directoriod e crontab del usuario (este debe tener permisos de escritura y ejecucion del usuario)

crontab -u usuario

prefijo para manejar el crontab de otro usuario, ejemplos:

$ sudo crontab -l -u root
$ sudo crontab -e usuario2
#crontab -d -u usuario

Esta herramienta, como muchas otras se pueden ver a mas profundidad y con mas detalle en: