Automatización

1. Con cron

a. Presentación

El servicio **cron** permite la programación de eventos repetidos. Funciona con ayuda de una tabla, llamada una **crontab**. Es un archivo de texto que se puede editar con un simple editor, como por ejemplo vi. Para modificar su crontab personal, utilice el comando **crontab** para editar la tabla con el parámetro `-e`.

Se guardan los archivos contrabs en `/var/spool/cron` .

El servicio **cron** debe estar ejecutándose para que las crontabs estén activas.

```
$ ps -ef|grep cron
root     3634     1  0 18:28 ?        00:00:00 /usr/sbin/cron
```

b. Formatos

El formato de un registro de crontab es el siguiente:

Minutos	Horas	Día del mes	Mes	Día semana	Comando
1	2	3	4	5	6

Utilice el formato siguiente para los valores periódicos:

- Un valor para indicar cuándo se debe ejecutar el comando. P. ej.: el valor 15 en el campo minuto significa el decimoquinto minuto.
- Una lista de valores separados por comas. P. ej.: 1,4,7,10 en el campo mes para enero, abril, julio, octubre.

- Un intervalo de valores. P. ej.: 1-5 en el campo día de la semana indica de lunes (1) a viernes (5). El 0 es el domingo y el 6, el sábado.
- El carácter * para todos los valores posibles. P. ej.: * en el campo día del mes indica todos los días del mes o de los meses.
- Una / indica un intervalo. Pe: */5 en el campo minutos indica cada 5 minutos.

c. Ejemplos

Ejecución de df todos los días, todo el año, cada cuarto de hora:

```
0,15,30,45 * * * * df > /tmp/libre
```

O

```
*/15 * * * * df > /tmp/libre
```

Arranque de un comando cada 5 minutos a partir de 2 (2, 7, 12, etc.) a las 18 horas los días 1 y 15 de cada mes:

```
2-57/5 18, 1, 15 * * comando
```

Ejecución de un comando todos los días laborables a las 17 horas:

```
0 17 * * 1-5 fin_trabajo.sh
```

Listar las crontabs activas:

```
$ crontab -l
```

Suprimir la crontab activa:

```
$ crontab -r
```

Editar la crontab de un usuario particular:

```
# crontab -u user
```

d. crontab sistema

La configuración crontab general para el sistema se encuentra en **`/etc/crontab`** . Su sintaxis difiere un poco. Hay un campo adicional. Permite especificar el usuario con el que se ejecutará el comando. Observe que puede especificar las variables de entorno en sus crontabs.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

Aquí todos los días a las 4:02 h de la mañana se ejecuta **`run-parts /etc/cron.daily`** . El script **run-parts** acepta como parámetro un directorio y ejecuta todos los programas presentes en éste.

```
$ ls cron.daily/
00-logwatch  0anacron   makewhatis.cron  slocate.cron
00webalizer  logrotate  rpm              tmpwatch
```

Entre los programas ejecutados, fíjese en **logrotate**, que permite efectuar copias de seguridad y renombrar archivos logs y archivos de diario del sistema para que éstos no se

vuelvan inabarcables a consecuencia de su tamaño. El programa **tmpwatch** se encarga de limpiar el sistema de los archivos ociosos (en /tmp por ejemplo).

Para terminar, el directorio `/etc/cron.d` contiene crontabs adicionales siempre en formato crontab sistema. Por lo tanto, una aplicación que debe ejecutar tareas recurrentes deberá crearse su propio archivo en vez de modificar una tabla existente.

e. Control de acceso

Se puede controlar el acceso con el comando **crontab** por usuario con los archivos `/etc/cron.allow` y `/etc/cron.deny` .

- Si cron.allow está creado, sólo los usuarios explícitamente indicados pueden utilizar **cron** (ver Automatización - Con at, en este capítulo).
- Si cron.allow no está creado, cron comprueba la presencia de un archivo cron.deny. Todos los usuarios que no estén en él están autorizados a utilizar cron. Si está vacío, todo el mundo está autorizado a utilizar **cron**.
- Si no existe ninguno de los dos archivos, sólo root puede utilizar cron.

2. Con at

a. Presentación

El comando **at** y los comandos asociados permiten la gestión de los batchs. A diferencia de la contrab, las modificaciones son volátiles: se pierden cuando se termina la sesión. Le corresponde a usted colocar la lista de los comandos en un posible archivo y cargarlo si es necesario mediante los scripts de su perfil.

Para que funcione at, el servicio **atd** (*at daemon*) debe estar en marcha.

```
$ ps -ef | grep atd
at      7988    1  0 21:05 ?       00:00:00 /usr/sbin/atd
```

b. Formatos

Para simplificar, hay dos maneras de utilizar at:

- pasándole una línea de comandos de manera interactiva,
- pasándole un archivo ejecutable que contiene los comandos que se deben ejecutar.

En los dos casos, debe facilitar a at una hora de ejecución. El formato de esta hora es bastante flexible. Para programar la ejecución de una línea de comandos a las 21:20 h de manera interactiva:

```
$ at 21:20
warning: commands will be executed using /bin/sh
at> echo hola
at> <EOT>
job 4 at 2008-05-08 21:20
```

Después de haber introducido el comando o los comandos que se deben ejecutar a las 21:20 h, pulse [Entrar], y en una línea vacía pulse [Ctrl] **D** (final de entrada). El comando **at** confirma la programación del comando.

Para programar la ejecución de un comando (script o binario) a las 21:25 h:

```
$ at -f /home/alejandro/test.sh 21:25
warning: commands will be executed using /bin/sh
job 6 at 2008-05-08 21:25
```

Hora

Se puede formatear la hora de la manera siguiente:

- HHMM o HH:MM.
- La hora puede tener el formato de 12 o 24 h. Con el formato de 12 horas, puede especificar AM (mañana) o PM (tarde).
- Midnight (medianoche), noon (mediodía), teatime (16:00 h, típicamente inglés).

- MMJJAA, MM/JJ/AA o JJ.MM.AA para una fecha particular.
- Now: ahora.
- + n minutes/hours/days/weeks: la hora actual a la que se añaden n minutos/horas /días/semanas.

Si la hora especificada es inferior a la hora actual, se ejecuta el comando al día siguiente.

```
$ at 21:30 09.05.2008
warning: commands will be executed using /bin/sh
at> echo ¡hola!
at> <EOT>
job 9 at 2008-05-09 21:30
$ at now + 2 days
warning: commands will be executed using /bin/sh
at> echo en dos días
at> <EOT>
job 10 at 2008-05-10 21:29
```

También existe el comando **batch**, que no tiene en cuenta la hora. Ejecuta el comando en cuanto el nivel de trabajo de la máquina lo permite. Se puede especificar la hora, pero en este caso se considera como «a partir de esta hora en cuanto sea posible».

c. Control de las tareas

El comando **atq** (*at queue*) permite listar las tareas programadas:

```
$ atq
10      2008-05-10 21:29 a alejandro
9       2008-05-09 21:30 a alejandro
```

Se colocan los jobs (tareas) en el directorio `/var/spool/atjobs` , a razón de un ejecutable por tarea.

```
# ls -l /var/spool/atjobs/
-rwx------ 1 alejandro users 5620 may  8 21:29 a000090133cf92
-rwx------ 1 alejandro users 5628 may  8 21:30 a0000a0133d531
```

Si mira el contenido del ejecutable, verá que su comando no está solo. Está situado al final, pero el script ubica todo el entorno durante la creación de la entrada at.

```
#cat a0000a0133d531
#!/bin/sh
# atrun uid=1000 gid=100
# mail     alejandro 0
umask 22
LESSKEY=/etc/lesskey.bin; export LESSKEY
NNTPSERVER=news; export NNTPSERVER
INFODIR=/usr/local/info:/usr/share/info:/usr/info; export INFODIR
MANPATH=/usr/local/man:/usr/share/man:/opt/gnome/share/man; export
MANPATH
KDE_MULTIHEAD=false; export KDE_MULTIHEAD
... (unas 80 líneas) ...
cd /home/alejandro || {
      echo 'Execution directory inaccessible' >&2
      exit 1
}
echo ¡hola!
```

El comando **atrm** permite suprimir una tarea:

```
$ atrm 10
$ atrm 9
$ atq
```

d. Control de acceso

Es posible controlar el acceso al comando **at** por usuario con los archivos `/etc/at.allow` y `/etc/at.deny` .

- Si at.allow está creado, sólo los usuarios explícitamente indicados pueden utilizar **at**.
- Si at.allow no está creado, at comprueba la presencia de un archivo at.deny. Todos los usuarios que no están en él están autorizados a utilizar at. Si está vacío, todo el mundo está autorizado a utilizar el comando at.
- Si no existe ninguno de los dos archivos, sólo root puede utilizar **at**.

3. Con anacron

El comando **anacron** ejecuta periódicamente los comandos con una frecuencia especificada en días. A diferencia de cron, el equipo no debe funcionar las 24h. Anacron puede remplazar a cron en equipos que se apagan con frecuencia (estaciones de trabajo, equipo casero, etc.): los comandos se ejecutan cuando el tiempo es alcanzado o sobrepasado.

La configuración de anacron está ubicada en `/etc/anacrontab` y tiene el formato siguiente:

```
Periodo    espera    identificador    comando
```

Por ejemplo:

```
1 5 cron.daily nice run-parts --report /etc/cron.daily
7 10 cron.weekly nice run-parts --report /etc/cron.weekly
@monthly 15 cron.monthly nice run-parts --report /etc/cron.monthly
```

La segunda línea implica que cada 7 días **cron.weekly** debe ser ejecutado. Al arrancar, anacron verifica la fecha de la última ejecución. Si no han transcurrido más de siete días, **cron.weekly** se arranca 10 minutos más tarde (espera en minutos para evitar cualquier arranque simultáneo).

4. Con systemd

systemd es omnipresente y tiene tendencia a reemplazarlo todo, también a cron. Una unidad llamada "timer" permite simular y por lo tanto reemplazar a cron. Verá que este ya es el caso. El comando siguiente le dará una lista de los timers activos:

```
# systemctl list-timers
NEXT                         LEFT          LAST                         PASSED
UNIT                         ACTIVATES
Wed 2021-05-05 23:55:46 UTC 3h 49min left Mon 2021-04-26 21:04:22 UTC 1 weeks 1 days ago
fwupd-refresh.timer          fwupd-refresh.service
Thu 2021-05-06 00:00:00 UTC 3h 54min left Wed 2021-05-05 19:47:44 UTC 18min ago
logrotate.timer              logrotate.service
Thu 2021-05-06 00:00:00 UTC 3h 54min left Wed 2021-05-05 19:47:44 UTC 18min ago
man-db.timer                 man-db.service
Thu 2021-05-06 03:49:26 UTC 7h left       Tue 2021-04-27 00:04:22 UTC 1 weeks 1 days ago
motd-news.timer              motd-news.service
Thu 2021-05-06 05:39:46 UTC 9h left       Mon 2021-04-26 17:51:39 UTC 1 weeks 2 days ago
apt-daily.timer              apt-daily.service

...
```

La última línea es la relativa a la actualización de la base de datos local de apt. Esta indica que:

- el servicio apt-daily se ejecutará el 6/05 a las 05h39 de la mañana,
- todavía quedan 8 horas,
- la última ejecución fue el 26/04 a las 17h51,
- esto fue hace dos días
- la unidad timer asociada es **apt-daily-upgrade.timer**.

Los detalles de la unidad arrojan una información interesante (la salida está limitada a los datos que nos interesan), una línea TimersCalendar nos informa del uso de un calendario para obtener una ejecución regular de la tarea descrita en Unit :

```
# systemctl show apt-daily-upgrade.timer
Unit=apt-daily-upgrade.service
TimersCalendar={ OnCalendar=*-*-* 06:00:00 ; next_elapse=Thu 2021-05-06 06:00:00 UTC }
FragmentPath=/lib/systemd/system/apt-daily-upgrade.timer
```

La unidad timer en cuestión contiene una sección Timer que le permite controlar la ejecución regular de cualquier tipo de tarea, todo lo que systemd pueda arrancar o parar como servicio o como objetivo:

- **onCalendar** indica que la tarea será arrancada todos los días a partir de las 6:00 de la mañana
- **RandomizedDelaySec** indica que se añadirá que un periodo de tiempo aleatorio entre 0 et 60 minutos en el momento previsto, con el objetivo de evitar una ejecución simultánea de tareas pesadas :

```
[Timer]
OnCalendar=*-*-* 6:00
RandomizedDelaySec=60m
```

Se puede crear fácilmente un servicio que se ejecutará a intervalos regulares. He aquí dos unidades: un servicio y su timer, que arrancará el servicio cada hora. Los archivos están ubicados en **`/etc/systemd/system`** :

```
# cat /etc/systemd/system/mycronjob.service
[Unit]
Description=Mi tarea horaria

[Service]
Type=oneshot
ExecStart=/usr/local/bin/mycronjob.sh

[Install]
WantedBy=multi-user.target

# cat /etc/systemd/system/mycronjob.timer
```

```
[Unit]
Description= Mi tarea horaria

[Timer]
OnCalendar= *-*-* *:00:00
Unit=mycronjob.service

[Install]
WantedBy=multi-user.target
```

Active y arranque el timer como cualquier servicio systemd:

```
# systemctl enable mycronjob.timer
# systemctl start mycronjob.timer
```

He aquí algunos valores prácticos para onCalendar:

Plazo	onCalendar
Cada minuto	`*-*-* *:*:00`
Cada hora	`*-*-* *:00:00`
Cada día	`*-*-* 00:00:00`
Cada mes	`*-*-01 00:00:00`
Cada semana	`Mon *-*-* 00:00:00`
Cada año	`*-01-01 00:00:00`
Cada trimestre	`*-01,04,07,10-01 00:00:00`
Cada semestre	`*-01,07-01 00:00:00`