

WUOLAH



David97

www.wuolah.com/student/David97



11665

SOModllesion4.pdf

Sesión 4 - Módulo I - SO



2º Sistemas Operativos



Grado en Ingeniería Informática



Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.





**KEEP
CALM
AND
ESTUDIA
UN POQUITO**

Sesión 4. Automatización de tareas

Apuntes

La automatización viene de la **planificación a largo plazo**, que era la realización de tareas cuando había memoria disponible para su ejecución.

1. **atd**. Automatizar una tarea. Hay diferentes colas para asignar prioridades.
2. **cron[d]**. Proceso para automatización periódica de tareas. Tiene una aplicación cliente, llamada **crontab**, que contiene la tabla de opciones.

1 Procesos demonio

Son procesos que se ejecutan en *background* y no está asociado a un terminal o proceso *login*. En ocasiones lanzan otros procesos para realizar la tarea encomendada.

2 Demonio atd

2.1 Orden at

Ordena la ejecución de órdenes a una determinada hora.

`at [-q queue] [-f <script>] [-mldbv] TIME`

`atrm` elimina trabajos por su número de trabajo, los cuales se pueden consultar con `atq`.

2.2 Salida estándar y salida de error estándar

Para una ejecución en el shell la entrada estándar es el teclado y la salida estándar la pantalla, pero al lanzar una ejecución asíncrona con `at` no se está creando un proceso hijo de la shell, por lo cual las entradas y salidas estándar no están asociadas con la consola de terminal.

Por defecto, la salida estándar y la salida de error estándar se envía al usuario que envió la orden como un correo electrónico usando la orden `/usr/bin/sendmail`. Si se lanza una ejecución que genera muchos mensajes de salida se podría redirigir a `/dev/null` para que ser “pierda” o a un archivo para consultarlo en caso de necesidad.

2.3 Orden batch

Es equivalente a `at`, excepto que no se especifica la hora de ejecución, sino que el trabajo se ejecutará cuando la **carga** de trabajos del sistema esté **bajo cierto valor umbral** (1,5 por defecto) o el valor especificado por el demonio `atd`.

2.4 Orden at: trabajando con colas

Las colas se designan con una única letra [a-z][A-Z]; la cola por omisión es la **a**; la cola **b** se usa para trabajos batch y, a partir de ahí, las distintas colas van teniendo menor prioridad al pasar de la **c** en adelante. La cola se especifica con la opción `[-q queue]` de `at`.

2.5 Aspectos de administración del demonio atd

Los archivos `/etc/at.deny` y `/etc/at.allow` determinan qué usuarios pueden usar la orden `at`.

3 Demonio cron

La especificación de las tareas que se desea que ejecute periódicamente se hace construyendo un archivo (llamado archivo “**crontab**”) que deberá tener un determinado formato, (llamado formato “**crontab**”). Será con la orden **crontab** con la que indicamos el archivo con formato “**crontab**” que deseamos comunicar al demonio cron.

3.1 Formato de los archivos crontab

Cada línea de código de un archivo **crontab** (excepto los comentarios) puede contener estos campos (que representan una orden):

minuto hora día-del-mes mes día-de-la-semana orden

Los campos de tiempo pueden contener:

***** → cualquier valor posible

n.º entero → activa dicho valor determinado

dos enteros separados por guión → indica un rango de valores

serie de enteros o rangos separados por coma → activa cualquier valor de la lista

Si se especifican valores concretos, por ejemplo, para el día del mes y día de la semana, se entiende como una condición **OR** (en lugar de **AND**).

3.2 La orden crontab

crontab <file>

3.3 Formato de los archivos crontab: especificando variables de entorno

Una línea de un archivo **crontab**, aparte de la línea de especificación de una orden para **cron**, puede tener una asignación de valores a variables de entorno, con la forma **<nombre>=<valor>**

Algunas variables son establecidas automáticamente por el demonio **cron**, como son:

- **SHELL** se establece a **/bin/sh**
- **LOGNAME** y **HOME** se toman del archivo **/etc/passwd**

3.4 Aspectos de administración del demonio crontab

Los archivos de configuración **/etc/cron.deny** y **/etc/cron.allow** determinan qué usuarios pueden ejecutar la orden **crontab**.

Ejercicios

Actividad 4.1 Consulta de información sobre procesos demonio

A partir de la información proporcionada por la orden `ps` encuentre los datos asociados a los demonios `atd` y `cron`, en concreto: quién es su padre, qué terminal tienen asociado y cuál es su usuario.

```

UID          PID  PPID  C  STIME TTY          TIME CMD
$ ps -ef | grep atd
daemon      9523      1  0 16:05 ?          00:00:00 /usr/sbin/atd -f
$ ps -ef | grep cron
root       1509      1  0 11:44 ?          00:00:00 /usr/sbin/cron -f
$ ps -p 1
  PID TTY          TIME CMD
   1 ?          00:00:03 systemd

```

| Demonio | Padre | Terminal asociado | Usuario |
|-------------|---------|---------------------------------------|---------|
| atd | systemd | ? → sin terminal asociado *Ej. 3.3 b) | daemon |
| cron | systemd | ? → sin terminal asociado | root |

Actividad 4.2 Ejecución postergada de órdenes con at (I)

Crea un archivo `genera-apunte` que escriba la lista de hijos del directorio `home` en un archivo de nombre `listahome-`date +%Y-%j-%T-$``, es decir, la yuxtaposición del literal “listahome” y el año, día dentro del año, la hora actual y pid (consulte la ayuda de `date`).

Creo el script `genera-apunte.sh` con el siguiente contenido:

```
#!/bin/sh
ls ~ > listahome-`date +%Y-%j-%T-$`
```

Lanzo la orden:

```

~$ at -f Universidad/S0/Prácticas/Modulo1/4-2_at/genera-apunte.sh 17:18
warning: commands will be executed using /bin/sh
job 7 at Wed Oct 9 17:18:00 2019

```

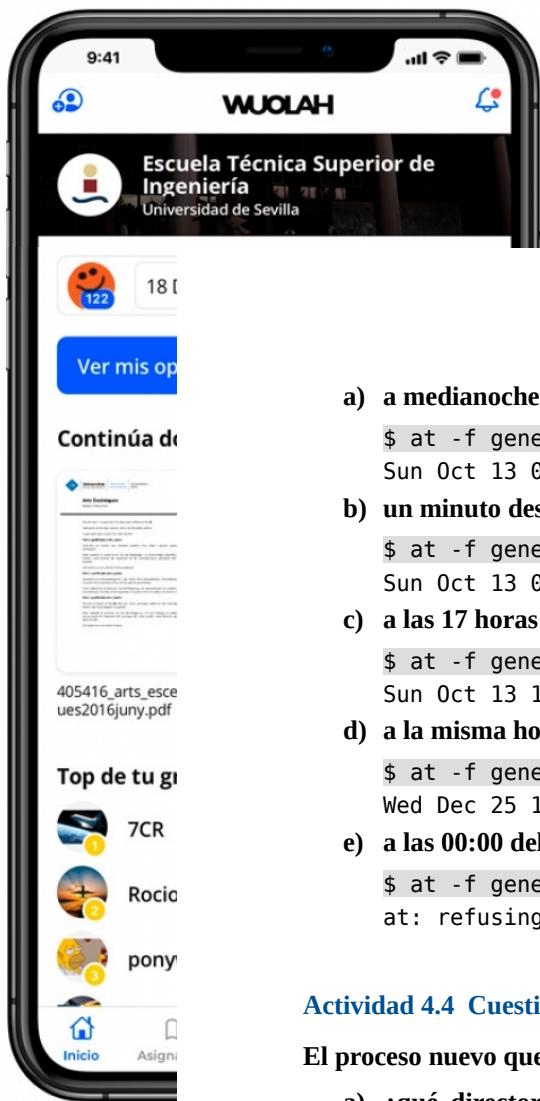
Lanza la ejecución del archivo `genera-apunte` un minuto más tarde de la hora actual. ¿En qué directorio se crea el archivo de salida?

El archivo de salida se crea en el mismo directorio desde el que se lanza la orden `at`.

Actividad 4.3 Ejecución postergada de órdenes con at (II)

Lanza varias órdenes `at` utilizando distintas formas de especificar el tiempo como las siguientes:

Estoy ejecutando estas órdenes en: **sábado 12 de Octubre de 2019, a las 10:59 A.M.**



Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.



David Carrasco Chicharro

- a) a medianoche de hoy

```
$ at -f genera-apunte.sh -v midnight
```

Sun Oct 13 00:00:00 2019
- b) un minuto después de la medianoche de hoy

```
$ at -f genera-apunte.sh -v midnight +1 minute
```

Sun Oct 13 00:01:00 2019
- c) a las 17 horas y 30 minutos de mañana

```
$ at -f genera-apunte.sh -v 17:30 tomorrow
```

Sun Oct 13 17:30:00 2019
- d) a la misma hora en que estemos ahora pero del día 25 de diciembre del presente año

```
$ at -f genera-apunte.sh -v 25 DEC
```

Wed Dec 25 10:59:00 2019
- e) a las 00:00 del 1 de enero del presente año

```
$ at -f genera-apunte.sh -v 00:00 JAN 01 2019
```

at: refusing to create job destined in the past

Actividad 4.4 Cuestiones sobre at

El proceso nuevo que se lanza al cumplirse el tiempo que se especificó en la orden at ...

- a) ¿qué directorio de trabajo tiene inicialmente? ¿hereda el que tenía el proceso que invocó a at o bien es el home, directorio inicial por omisión?

El manual dice: “El directorio de trabajo, el entorno (excepto las variables BASH_VERSION, DISPLAY, EUID, GROUPS, SHELL_OPTS, TERM, UID, y _ y el umask se conservan desde el momento de la invocación”. Por tanto, hereda el que tenía el proceso que invocó a at.
- b) ¿qué máscara de creación de archivos umask tiene? ¿es la heredada del padre o la que se usa por omisión?

La máscara es heredada y se conserva, tal y como se indica en el apartado anterior; umask tiene el valor 0022
- c) ¿hereda las variables locales del proceso padre?

Tal y como se indica en el apartado a), el entorno se mantiene, excepto las variables señaladas.

Actividad 4.5 Relación padre-hijo con órdenes ejecutadas mediante at

El proceso nuevo que se lanza al cumplirse el tiempo que se especificó en la orden at ... ¿de quién es hijo? Investiga lanzando la ejecución retardada de un script que muestre la información completa sobre los procesos existentes y el pid del proceso actual; el script podría contener lo que sigue:

```
#!/bin/sh

nombrearchivo=`date +%Y-%j-%T-$$`
ps -ef > $nombrearchivo
echo Mi pid = $$ >> $nombrearchivo
```

```
$ at -f act4-5.sh now + 1 minute
job 13 at Sat Oct 12 11:53:00 2019
$ tail -n 4 2019-285-11\:53\:01-18210
daemon 18203 1262 0 11:53 ? 00:00:00 /usr/sbin/atd -f
david 18210 18203 0 11:53 ? 00:00:00 sh
david 18213 18210 0 11:53 ? 00:00:00 ps -ef
Mi pid = 18210
```

El proceso lanzado (PID=18210) es hijo del demonio atd que lo lanza (PID=18203)

Actividad 4.6 Script para orden at

Construye un script que utilice la orden find para generar en la salida estándar los archivos modificados en las últimas 24 horas (partiendo del directorio home y recorriéndolo en profundidad), la salida deberá escribirse el archivo de nombre “modificados_<año><día><hora>” (dentro del directorio home). Con la orden at provoque que se ejecute dentro de un día a partir de este momento.

```
#!/bin/sh
```

```
nombreakarchivo=modificados_`date +%Y-%j-%T`
find ~ -mtime 0 >> ~/${nombreakarchivo}
```

```
$ at -f act4-6.sh tomorrow
job 15 at Sun Oct 13 13:25:00 2019
```

Actividad 4.7 Trabajo con la orden batch

Lanza los procesos que sean necesarios para conseguir que exista una gran carga de trabajo para el sistema de modo que los trabajos lanzados con la orden batch no se estén ejecutando (puede simplemente construir un script que esté en un ciclo infinito y lanzarla varias veces en segundo plano). Utiliza las órdenes oportunas para manejar este conjunto de procesos (la orden jobs para ver los trabajos lanzados, kill para finalizar un trabajo,... y tal vez también las órdenes fg, bg para pasar de segundo a primer plano y viceversa, <Ctrl-Z> para suspender el proceso en primer plano actual, etc). Experimenta para comprobar cómo al ir disminuyendo la carga de trabajos habrá un momento en que se ejecuten los trabajos lanzados a la cola batch.

```
#!/bin/sh
nombreakarchivo=batch_`date +%Y-%j-%T`
echo "Prueba ej. 4.7" > ${nombreakarchivo}
```

Saturamos la CPU haciendo uso del script de la actividad 2 de la sesión 3 y entonces lanzamos la orden batch (siendo las 14:01:02).

```
$ batch
at> ./act4-7.sh
at> <EOT>
job 20 at Sat Oct 12 14:00:00 2019
```

```
$ ls -la batch*
-rw-rw-rw- 1 david david 15 oct 12 14:11 batch_2019-285-14:11:02
```

Finalmente, se ha ejecutado 10 minutos después desde su lanzamiento con batch.

Actividad 4.8 Utilización de las colas de trabajos de at

Construye tres script que deberás lanzar a las colas c, d y e especificando una hora concreta que esté unos pocos minutos más adelante (no muchos para ser operativos). Idea qué actuación deben tener dichos script de forma que se ponga de manifiesto que de esas colas la más prioritaria es la c y la menos es la e. Visualiza en algún momento los trabajos asignados a las distintas colas.

En primer lugar creo 3 scripts 4-8_[1-3].sh que imprimirán en un fichero su número de script y cuyo nombre contendrá la hora a la que se ejecuta.

Con otro script lanzo los 3 guiones anteriores un minuto después, cada uno en una cola:

```
#!/bin/sh
at -q e -f ./4-8_1.sh now + 1 minute
at -q d -f ./4-8_2.sh now + 1 minute
at -q c -f ./4-8_3.sh now + 1 minute
```

```
$ ./act4-8.sh
job 24 at Sat Oct 12 14:30:00 2019
job 25 at Sat Oct 12 14:30:00 2019
job 26 at Sat Oct 12 14:30:00 2019
```

```
$ atq -q c
26    Sat Oct 12 14:30:00 2019 c david
$ atq -q d
25    Sat Oct 12 14:30:00 2019 d david
$ atq -q e
24    Sat Oct 12 14:30:00 2019 e david
```

Deberían ejecutarse en el siguiente orden: 3, 2, 1.

Mostramos el resultado, ordenando por fecha de modificación (primero el más nuevo).

```
$ ls -lrc act_4-8*
-rw-rw-rw- 1 david david 9 oct 12 14:30 act_4-8_1_2019-285-14:30:00
-rw-rw-rw- 1 david david 9 oct 12 14:30 act_4-8_2_2019-285-14:30:00
-rw-rw-rw- 1 david david 9 oct 12 14:30 act_4-8_3_2019-285-14:30:00
```

Tal y como era de esperar se ha seguido el orden de ejecución especificado en las colas.

Actividad 4.9 Relación padre-hijo con órdenes ejecutadas mediante crontab

Al igual que se investigó en la Actividad 4.5 sobre quién es el proceso padre del nuestro, lanza el script construido en dicha actividad con una periodicidad de un minuto y analiza los resultados.

```
# tail -5 2019-285-10\:25\:02-1265
root    1167      1  0 10:04 ?        00:00:00 crond
root    1264    1167  0 10:25 ?        00:00:00 CROND
root    1265    1264  0 10:25 ?        00:00:00 /bin/sh /root/act4-5.sh
root    1267    1265  0 10:25 ?        00:00:00 ps -ef
Mi pid = 1265
```

```
# tail -5 2019-285-10\:26\:02-1272
root    1167      1  0 10:04 ?        00:00:00 crond
root    1271    1167  0 10:26 ?        00:00:00 CROND
root    1272    1271  0 10:26 ?        00:00:00 /bin/sh /root/act4-5.sh
root    1275    1272  0 10:26 ?        00:00:00 ps -ef
Mi pid = 1272
```

```
# tail -5 2019-285-10\:27\:01-1279
root    1167      1  0 10:04 ?        00:00:00 crond
root    1278    1167  0 10:27 ?        00:00:00 CROND
root    1279    1278  0 10:27 ?        00:00:00 /bin/sh /root/act4-5.sh
root    1281    1279  0 10:27 ?        00:00:00 ps -ef
Mi pid = 1279
```

```
# tail -5 2019-285-10\:28\:01-1284
root    1167      1  0 10:04 ?        00:00:00 crond
root    1283    1167  0 10:28 ?        00:00:00 CROND
root    1284    1283  0 10:28 ?        00:00:00 /bin/sh /root/act4-5.sh
root    1287    1284  0 10:28 ?        00:00:00 ps -ef
Mi pid = 1284
```

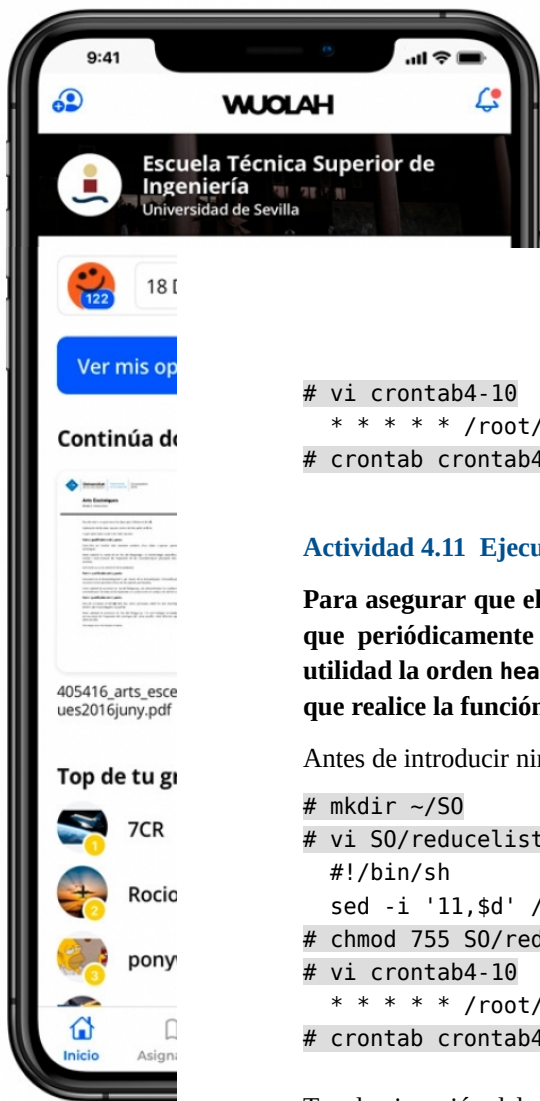
El padre del proceso siempre es el proceso CROND, que a su vez es hijo de crond.

Actividad 4.10 Ejecución de scripts con crontab (I)

Construye un script que sea lanzado con una periodicidad de un minuto y que borre los nombres de los archivos que cuelguen del directorio /tmp/varios y que comiencen por “core” (cree ese directorio y algunos archivos para poder realizar esta actividad). Utiliza la opción -v de la orden rm para generar como salida una frase de confirmación de los archivos borrados; queremos que el conjunto de estas salidas se añadan al archivo /tmp/listacores.

```
# mkdir /tmp/varios
# touch /tmp/varios/core{1..10}.txt
# vi act4-10.sh
#!/bin/sh
rm -vf /tmp/varios/core* >> /tmp/listacores

# chmod 755 act4-10.sh
```



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.



David Carrasco Chicharro

```
# vi crontab4-10
* * * * * /root/act4-10.sh
# crontab crontab4-10
```

Actividad 4.11 Ejecución de scripts con crontab (II)

Para asegurar que el contenido del archivo `/tmp/listacores` no crezca demasiado, queremos que periódicamente se deje dicho archivo solo con sus 10 primeras líneas (puede ser de utilidad la orden `head`). Construye un script llamado `reducelista` (dentro del directorio `~/S0`) que realice la función anterior y lance su ejecución con periodicidad de un minuto.

Antes de introducir ningún comando del ejercicio, el archivo `/tmp/listacores` tiene 310 líneas.

```
# mkdir ~/S0
# vi S0/reducelista
#!/bin/sh
sed -i '11,$d' /tmp/listacores
# chmod 755 S0/reducelista
# vi crontab4-10
* * * * * /root/S0/reducelista
# crontab crontab4-10
```

Tras la ejecución del cron el archivo contiene únicamente las 10 primeras líneas.

Actividad 4.12 Ejecución de scripts con crontab (III)

Construye un sencillo script que escriba en el archivo `~/S0/listabusqueda` una nueva línea con la fecha y hora actual y después el valor de la lista de búsqueda, por ejemplo:

```
. . .
2011-297-12:39:10 - /usr/local/bin:/usr/local/bin:/usr/bin...
. . .
```

Ejecuta este script desde el lenguaje de órdenes y también lánzalo como trabajo `crontab` y compara los resultados, ¿se tiene en ambos casos la misma lista de búsqueda?

```
# vi ~/S0/listabusqueda
#!/bin/sh
echo `date +%Y-%j-%T` - $PATH >> /root/S0/listabusqueda
# chmod 755 act4-12.sh
# ./act4-12.sh
# cat S0/listabusqueda
2019-285-11:34:19 - /usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:...
# vi /root/crontab4-12
* * * * * /root/act4-12.sh
# crontab crontab4-12
# cat S0/listabusqueda
2019-285-11:34:19 - /usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:...
2019-285-11:39:01 - /usr/bin:/bin
2019-285-11:40:01 - /usr/bin:/bin
```

El resultado es distinto cuando se ejecuta desde terminal respecto al resultado que proporciona crontab. El \$PATH cambia en ambos, siendo en crontab por defecto /usr/bin:/bin.

Actividad 4.13 Variables de entorno en archivos crontab

Practicamos ahora lo que acabamos de explicar situándonos en lo que hemos realizado en la Actividad 4.11. Construye un script que generará un archivo crontab llamado crontab-reducelista que deberá contener:

- como primera línea la asignación a la variable PATH de la lista de búsqueda actual y además el directorio \$HOME/SO
- después la indicación a cron de la ejecución con periodicidad de 1 minuto del script reducelista

```
# vi act4-13.sh
#!/bin/sh
echo "SHELL=/bin/sh" > crontab-reducelista
echo "PATH=\"`pwd`\"/:$HOME/SO:$PATH" >> crontab-reducelista
echo "* * * * * reducelista" >> crontab-reducelista
```

Una vez construido crontab-reducelista lánzalo con la orden crontab. Comprueba que con esta nueva lista de búsqueda podremos hacer alusión a reducelista especificando únicamente su nombre independientemente del directorio de trabajo en que nos situemos (no como ocurría en la Actividad 4.11 en que el directorio \$HOME/SO no estaba en la lista de búsqueda).

```
# ./act4-13.sh
# crontab crontab-reducelista
```

Actividad 4.14 Archivos crontab de diferentes usuarios

Vamos a lanzar un archivo crontab cuyo propietario es otro usuario. Visualiza el contenido del archivo /fenix/depar/lsi/so/ver-entorno y /fenix/depar/lsi/so/crontabver. Comprueba con ls -l que el propietario es el usuario lsi. Sin copiarlos, úsalos para lanzar la ejecución cada minuto del script /fenix/depar/lsi/so/ver-entorno. Analiza el archivo de salida: ¿de qué línea del archivo /etc/passwd se toman LOGNAME y HOME, de la línea del propietario del archivo crontab o de la línea del usuario que lanza el archivo crontab?

Actividad 4.15 Ejecución de scripts con crontab (IV)

El objetivo es ejecutar todos los días a las 0 horas 0 minutos una copia de los archivos que cuelguen de \$HOME que se hayan modificado en las últimas 24 horas. Vamos a programar este salvado incremental utilizando la orden find que usábamos en la Actividad 4.6; ahora queremos que se copien los archivos encontrados por find utilizando la orden cpio:

<orden find de la Actividad 4.6> | cpio -pmduv /tmp/salvado\$HOME

```
# vi act4-15.sh
#!/bin/sh
find ~ -mtime 0 | cpio -pmduv /tmp/salvado$HOME
# chmod 755 act4-15.sh
# vi crontab4-15
0 0 * * * /root/act4-15.sh
# crontab crontab4-15
```

Actividad 4.16. Gestión del servicio crond como usuario root

Prueba las siguientes operaciones sobre el demonio crond:

1. Como usuario root, deshabilita/habilita a un determinado usuario para que pueda utilizar el servicio cron; comprueba que efectivamente funciona.

Como root:

```
# vi /etc/cron.deny
Usuario_A
# su Usuario_A → como user:
```

```
$ vi script.sh
#!/bin/sh
echo "Usuario A" > ~/arch_Usuario_A.txt
$ chmod 755 script.sh
$ vi crontab_Usuario_A
* * * * * ./script.sh
$ crontab crontab_Usuario_A
You (usuario_A) are not allowed to use this program (crontab)
See crontab(1) for more information
$ exit → Como root:
```

```
# vi /etc/cron.deny → elimino la línea escrita con el nombre del Usuario A
# su Usuario_A
$ crontab crontab_Usuario_A
$ crontab -l
* * * * * ./script.sh
```

2. Iniciar y terminar el servicio cron. Prueba las siguientes órdenes para iniciar y terminar este servicio:

Iniciar el servicio cron: `/sbin/service crond start`

Terminar el servicio cron: `/sbin/service crond stop`