postdata9
www.wuolah.com/student/postdata9

32577

sesion4.pdf

Módulo I (actualizado)

2º Sistemas Operativos

Grado en Ingeniería Informática

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

KEEP CALM AND ESTUDIA UN POQUITO

WUOLAH + #QuédateEnCasa

Sesión 4:

Automatización de tareas

Índice:

Esta sesión nos dedicamos a trabajar con demonios.

Un **demonio** (daemon ó job) es un proceso que se ejecuta en **background** en un momento determinado o con una periocidad; es una tarea programada.

Vamos a ver las órdenes para crear demonios y administrarlos.

Las características de un proceso demonio:

- **Se ejecuta en background y no está asociado a un terminal o proceso login.**
- Muchos se inician durante el arranque del sistema y continúan ejecutándose mientras el sistema esté encendido; otros se inician y terminan cuando sea necesario.
- Cuando un demonio termina de forma imprevista, suele haber un mecanismo que lo reinicia; así como habrá un mecanismo que permitirá la comunicación entre el demonio y los procesos cliente, cuando este esté a la espera de un evento.
- Suele lanzar otros procesos para que realicen el trabajo, así no lo hace él de forma directa.
- Para conservar la filosofía de la modularidad de Unix, son programas que **no pertenecen al kernel**.
- Cuando se ejecutan con privilegio de superusuario (UID = 0), suelen tener por padre al proceso init (PID = 1).

El término demonio tiene dos equivalencias en español, demon y daemon. Daemon hace alusión a una entidad con vida propia, independiente, que se ejecuta en un plano invisible.

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 crond, en concreto: quién es su padre, qué terminal tienen asociado y cuál es su usuario.

$ ps -aux | grep cron Con lo que buscaríamos todos los procesos y filtraríamos por el nombre cron. La salida nos proporcionaría el PID del proceso, que en este caso es 1050.

$ ps -p 1050 -f Buscaríamos el proceso según su PID (-p) y mostraríamos la salida en formato largo (-f).

```
paula@postdata9:~$ ps -aux | grep cron
root        1050  0.0  0.0  33840  3212 ?        Ss   09:41   0:00 /usr/sbin/cron -f
paula       4992  0.0  0.0  16948  1108 pts/0    S+   10:30   0:00 grep --color=auto cron
paula@postdata9:~$ ps -p 1050 -f
UID          PID  PPID  C STIME TTY          TIME CMD
root        1050     1  0 09:41 ?        00:00:00 /usr/sbin/cron -f
```

Como vemos en la salida, el usuario de cron es root, su proceso padre es init (1) y no tienen asociado ningún terminal.

Ejecutamos las mismas órdenes para atd:

```
paula@postdata9:~$ ps -aux | grep atd
daemon      6500  0.0  0.0  28332  2392 ?        Ss   10:58   0:00 /usr/sbin/atd -f
paula       6526  0.0  0.0  16948  1004 pts/0    S+   11:00   0:00 grep --color=auto atd
paula@postdata9:~$ ps -p 6500 -f
UID          PID  PPID  C STIME TTY          TIME CMD
daemon      6500     1  0 10:58 ?        00:00:00 /usr/sbin/atd -f
```

El usuario de atd es **daemon**, su proceso padre es init (1) y no tiene asociado ningún terminal.

---

2. Ejecutar tareas a una determinada hora: demonio atd

Con el demonio atd, podemos realizar la ejecución de una orden en un momento determinado de tiempo. Para interaccionar con dicho servicio, tenemos las siguientes órdenes:

- **at**: realizar una ejecución a una determinada hora;
- **atq**: consultar las colas;
- **atrm**: eliminar los procesos;
- **batch**: para ejecutar órdenes cuando la carga del sistema sea baja.

2.1 Orden at

La sintaxis es: **$ at -q <cola> -f <script> <time>**

Lee órdenes de la entrada estándar o del script que le pasemos, al que podemos darle colas de prioridad, y lo ejecuta una sola vez a la hora especificada en time, usando /bin/sh. Los distintos formatos que puede admitir time se encuentran en /usr/share/doc/at/timespec.

- la cola se especifica con una letra, que tomar los valores de [a-z] y [A-Z];
  - el valor a, especifica la cola para at;
  - el valor b, para batch.
- time se puede especificar en <hora> <momento> <añadido>:
  - **hora:**
    - hh:mm → para una hora y minutos determinados;
    - 9:00 AM → a las 9 de la mañana;
    - 2:30 PM → a las 14.30 de la tarde;
    - 1430 → a las 14.30 de la tarde;
    - 4 PM → a las 16.00;
    - 5 AM → a las 5.00;
    - now → ahora;
    - midnight → a las 00:00;
    - noon → 12:00 del medio día;
    - teatime → 16:00 de la tarde;
      - Junto a la hora podemos especificar la fecha:
        - hora mm/dd → 2:30 PM 12/23;
        - hora mm dd → 2:30 PM Dec 23;
        - hora mm/dd/yyyy → 2:30 PM 12/23/2020;
        - hora dd.mm.yyyy → 2:30 PM 23.12.2020;
  - **momento:**
    - tomorrow → el día siguiente a la misma hora a la que se ejecuta at, si no le especificamos antes la hora;
    - next week → la semana siguiente a la hora a la que se ejecuta at, si no le especificamos antes la hora;

- next monday → siguiente lunes a la hora a la que se ejecuta at, si no le especificamos antes la hora;
- next month → el siguiente mes a la hora a la que se ejecuta at, si no le especificamos antes la hora;

- **añadido:** se especificaría después de escribir la hora o el momento, añadiéndole <+> y los valores siguientes:
  - valor minutes;
  - valor hour;
  - valor days;
  - valor weeks;
  - valor months;
  - valor years;

    Ejemplos de esto:

    - now + 5 hours, ejecutaría la orden después de 5 horas de realizar la orden;
    - 4 PM + 3 weeks, ejecutaría la orden a las 16.00 después de 3 semanas;

La ejecución de at especificando el script en la orden, sin entrar en el intérprete, sería:

```
paula@postdata9:~$ at -f ./genera-apunte 11:22
warning: commands will be executed using /bin/sh
job 5 at Sun Dec 15 11:22:00 2019
```

En la salida, nos avisa que el proceso será ejecutado usando /bin/sh. Nos informa, además, de más de que el número del proceso es 5 y se ejecutará el domingo 15 de diciembre a las 11:22:00 de 2019.

Cuando ejecutamos at <hora> sin especificarle un script, se nos abre un intérprete para escribir el script que queremos ejecutar; para salir hacemos Ctrl + D, (Ctrl + C interrumpe la orden). Un ejemplo de esto sería el siguiente, en el que realizamos at <hora>, se nos abre el intérprete y escribimos la orden que queremos realizar; guardamos la salida en un fichero llamado ejemplo:

```
paula@postdata9:~$ at 11:11
warning: commands will be executed using /bin/sh
at> ls -l . > ejemplo
at> <EOT>
job 4 at Sun Dec 15 11:11:00 2019
paula@postdata9:~$ ls -l ejemplo
-rw-r--r-- 1 paula paula 1111 dic 15 11:11 ejemplo
paula@postdata9:~$ cat ejemplo
total 751700
drwxr-xr-x 9 paula paula      4096 dic 10 13:36 apache-jmeter-5.2.1
drwxr-xr-x 2 paula paula      4096 may 31  2019 database
drwxr-xr-x 5 paula paula      4096 dic 10 19:06 Descargas
drwxr-xr-x 2 paula paula      4096 oct 12 11:41 Documentos
-rw-r--r-- 1 paula paula         0 dic 15 11:11 ejemplo
drwxr-xr-x 9 paula paula      4096 dic 12 12:39 Escritorio
-rw-r--r-- 1 paula paula      8980 ene 14  2018 examples.desktop
```

Si al realizar at, nos dice “No atd running?”, ejecutamos la siguiente orden para activar el servicio de atd.

```
Can't open /var/run/atd.pid to signal atd. No atd running?
paula@postdata9:~$ systemctl start atd
```

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).

Lanza la ejecución del archivo genera-apunte un minuto más tarde de la hora actual.

¿En qué directorio se crea el archivo de salida?

Creamos el archivo genera-apunte:

$ vi genera-apunte

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

```
paula@postdata9:~$ vi genera-apunte
#!/bin/bash
ls /home > listahome-`date +%Y-%j-%T-$$`
```

Lanzamos el script un minuto más tarde:

$ at -f ./genera-apunte 11:22

Al esperar un minuto, vemos que se nos ha creado el fichero listahome, y comprobamos su contenido:

$ ls -l listahome-2019-349-11\:22\:00-6976

$ cat listahome-2019-349-11\:22\:00-6976

```
paula@postdata9:~$ at -f ./genera-apunte 11:22
warning: commands will be executed using /bin/sh
job 5 at Sun Dec 15 11:22:00 2019
paula@postdata9:~$ ls -l listahome-2019-349-11\:22\:00-6976
-rw-r--r-- 1 paula paula 6 dic 15 11:22 listahome-2019-349-11:22:00-6976
paula@postdata9:~$ cat listahome-2019-349-11\:22\:00-6976
paula
```

El archivo listahome se crea en el directorio en el que hemos ejecutado 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 (será de utilidad la opción -v):

a) **a medianoche de hoy**

$ at -f ./genera-apunte midnight

$ atq vemos el número del proceso

$ atrm 7 para eliminar el proceso lanzado

```
paula@postdata9:~$ at -f ./genera-apunte midnight
warning: commands will be executed using /bin/sh
job 7 at Mon Dec 16 00:00:00 2019
paula@postdata9:~$ atq
7       Mon Dec 16 00:00:00 2019 a paula
paula@postdata9:~$ atrm 7
paula@postdata9:~$ atq
```

b) **un minuto después de la medianoche de hoy**

$ at -q a -f ./genera-apunte midnight+1 minute (lo ejecutamos con la cola a)

$ atq

$ atrm 9

```
paula@postdata9:~$ at -q a -f ./genera-apunte midnight+1 minute
warning: commands will be executed using /bin/sh
job 9 at Mon Dec 16 00:01:00 2019
paula@postdata9:~$ atq
9       Mon Dec 16 00:01:00 2019 a paula
paula@postdata9:~$ atrm 9
paula@postdata9:~$ atq
paula@postdata9:~$
```

c) **a las 17 horas y 30 minutos de mañana**

$ at -f ./genera-apunte 17:30 tomorrow

$ atq el número del proceso es 11

```
paula@postdata9:~$ at -f ./genera-apunte 17:30 tomorrow
warning: commands will be executed using /bin/sh
job 11 at Mon Dec 16 17:30:00 2019
paula@postdata9:~$ atq
11      Mon Dec 16 17:30:00 2019 a paula
```

d) **a la misma hora en que estemos ahora pero del día 25 de diciembre del presente año**

$ at -f ./genera-apunte Dec 25 2019

$ at -d 12 eliminamos el proceso, es igual que atrm

```
paula@postdata9:~$ at -f ./genera-apunte Dec 25 2019
warning: commands will be executed using /bin/sh
job 12 at Wed Dec 25 11:51:00 2019
```

e) **a las 00:00 del 1 de enero del presente año**

$ at -f ./genera-apunte 00:00 Jan 01 2020

```
paula@postdata9:~$ at -f ./genera-apunte 00:00 Jan 01 2020
warning: commands will be executed using /bin/sh
job 14 at Wed Jan  1 00:00:00 2020
```

Utiliza las órdenes atq y atrm para familiarizarte con su funcionamiento (consulta la ayuda de estas órdenes).

2.2 Sobre el entorno de ejecución de las órdenes

Cuando haya que ejecutar el script en el momento que se especificó en la orden at, se lanzará un shell /bin/bash para ejecutarlo.

El proceso se ejecuta en el mismo directorio en el que fue ejecutada la orden at, hereda del padre la máscara de creación de archivos, así como las variables locales, excepto BASH_VERSINFO, DISPLAY, EUID, GROUPS, SHELLOPTS, TERM, UID y _.

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?**

$ man at

```
For  both  at  and  batch, commands are read from standard input or the
file specified with the -f option and executed.  The working directory,
the environment (except for the variables BASH_VERSINFO, DISPLAY, EUID,
GROUPS, SHELLOPTS, TERM, UID, and _) and the umask  are  retained  from
the time of invocation.
```

En el man de at, especifica que el directorio de trabajo, el entorno y umask son los del momento de la invocación; por tanto, el directorio de trabajo es el mismo que cuando se ejecutó la orden at, lo hereda del proceso padre.

b) **¿Qué máscara de creación de archivos umask tiene? ¿es la heredada del padre o la que se usa por omisión?**

La información viene dada en la misma captura de antes, la máscara de creación de archivos (umask) la hereda del padre.

c) **¿Hereda las variables locales del proceso padre?**

Las variables locales, excepto BASH_VERSINFO, DISPLAY, EUID, GROUPS, SHELLOPTS, TERM, UID y _, las hereda del proceso padre.

Experimenta con la orden at lanzando las órdenes adecuadas para encontrar las respuestas. (Puede encontrar información en la ayuda de at)

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:

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

- Creamos el script:

$ vi script

```
#!/bin/bash
nombrearchivo=`date +%Y-%j-%T`
ps -ef > $nombrearchivo
echo Mi pid = $$ >> $nombrearchivo
```

- Lo lanzamos con at para que se ejecute en el minuto siguiente:

  $ at -f ./script now+1 minute

- Vemos el archivo que ha creado:

  $ cat 2019-349-14\:19\:00

```
daemon   12010  6500  0 14:19 ?        00:00:00 /usr/sbin/atd -f
paula    12011 12010  0 14:19 ?        00:00:00 sh
paula    12013 12011  0 14:19 ?        00:00:00 ps -ef
Mi pid = 12011
```

Nuestro PID es 12011 y nuestro padre es el proceso con PID 12010, que se corresponde con el daemon. Por tanto, es hijo del proceso atd.

2.3 Orden at: trabajando con colas

La orden at gestiona distintas colas de procesos para ejecutarse, se especifica con una única letra, que puede tomar su valor en los intervalos [a-z] y [A-Z]. La cola con letra a es la utilizada por omisión; la b se usa para los trabajos batch; a partir de aquí, las colas van teniendo menor prioridad.

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.

- Creamos un script que ejecute los 3 scripts con at:

  $ vi script4

  #!/bin/bash

  at -q c -f ./script now+1 minute

  at -q d -f ./script now+1 minute

  at -q e -f ./script now+1 minute

- Lo lanzamos:

  $ at -f ./script4 now + 1 minute

- Vemos las prioridades de los procesos:

  $ atq -q c

  $ atq -q d

  $ atq -q e

Podemos comprobar que se ha ejecutado antes c, después d y por último e.

2.4 Aspectos de administración del demonio atd

Hay dos archivos de configuración, los cuales son:

- **/etc/at.allow**: contiene una lista de nombres de usuarios que pueden usar el demonio at. Si este fichero no existe, se revisaría el fichero /etc/at.deny;
- **/etc/at.deny**: contiene una lista de nombres de los usuarios que no pueden usar el demonio at; si este archivo se encuentra vacío, supone que cualquier usuario puede usar at.

Si ninguno de estos dos archivos existen, sólo el superusuario puede ejecutar at.

3. Ejecuciones periódicas: demonio cron

Uno de los demonios básicos es cron, que nos permite ejecutar órdenes con una periocidad determinada. Utilizaremos la orden crontab para poder trabajar con el demonio cron. A dicha orden, le pasamos un archivo que deberá tener un determinado formato.

3.1 Formato de los archivos crontab: especificando órdenes

Cada línea del archivo que se le pasa a la orden crontab, puede tener los siguientes campos:

minuto *hora* *dia_mes* *mes* *dia_semana* *orden*

Los 5 primeros campos indican la periocidad con que se desea ejecutar la orden especificada en el último campo. Cada uno de estos campos puede contener:

- un número entero, que indica el valor del campo;
- dos enteros separados por un guión, especificando un rango de valores;
- valores o rangos, separados por una coma, que activa cualquier valor que aparece en la lista;
- un asterisco, que indica cualquier valor posible.

Varios ejemplos para comprenderlo:

- 1 20 * * 1-5

  esto significa de lunes a viernes todos los meses a las 20:01.

- 0,30 * 13 * 5

  esto significa cada media hora el viernes y cada media hora el día 13 del mes.

El último campo especifica lo que queremos ejecutar. Si en el último campo indicamos una orden o un script, se lanzará /bin/sh para ejecutarlo; si en cambio, es la ruta de un archivo ejecutable, se provoca su ejecución.

Si no especificamos el shell, utilizará el que tiene asignado en /etc/passwd

Sintaxis: **$ crontab <archivo>**

Esta orden instala, desinstala o lista los trabajos que realiza el demonio cron. El archivo que le pasamos como argumento debe tener el formato previamente explicado, el formato crontab.

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.

- Creamos el fichero con el formato cron:

```
$ vi cron_4.9
    SHELL=/bin/sh
    PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
    #min  h  dm  m  ds  o
    * * * * * /home/paula/script
```

- Lo ejecutamos:

```
$ crontab cron_4.9
```

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.

- Creamos el script:

```
$ vi script_4.10
    #!/bin/bash
    rm -v /tmp/varios/core* >> /tmp/listacores
```

- Creamos el archivo cron:

```
$ vi cron_4.10
    SHELL=/bin/sh
    PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
    #min  h  dm  m  ds  o
    * * * * * /home/paula/script
```

- Lo ejecutamos:

```
$ crontab cron_4.10
```

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 ~/SO) que realice la función anterior y lance su ejecución con periodicidad de un minuto.

- Creamos el script:

```
$ vi reducelista
    #!/bin/bash
    head -n 10 /tmp/listacores > /tmp/aux
    rm /tmp/listacores
    mv /tmp/aux /tmp/listacores
```

  Nos quedamos con las 10 primeras líneas y las guardamos en un fichero auxiliar. Después eliminamos el fichero de listacores y le cambiamos el nombre a aux por listacores.

- Creamos el archivo cron:

```
$ vi cron_4.11
    SHELL=/bin/sh
    PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
    #min  h  dm  m  ds  o
    * * * * * /home/paula/reducelista
```

- Lo ejecutamos:

```
$ crontab cron_4.11
```

Actividad 4.12. Ejecución de scripts con crontab (III)

Construye un sencillo script que escriba en el archivo ~/SO/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?

- Creamos el script:

```
$ vi script_4.12
    #!/bin/bash
    fecha=`date +%Y-%j-%T`
    echo $fecha - $PATH >> /home/paula/listabusqueda
```

- Creamos el archivo cron:

  $ vi cron_4.12

  SHELL=/bin/sh

  PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

  #min h dm m ds o

  *** * * * * /home/paula/script_4.12**

- Lo ejecutamos en la terminal:

  $ ./script_4.12

  - Comprobamos el resultado:

    $ cat listabusqueda

    2019-303-18:08:15 - /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

- Lo ejecutamos con cron:

  $ crontab cron_4.12

  - Comprobamos el resultado:

    $ cat listabusqueda

    2019-303-18:10:02-/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/bin

Como podemos ver, el PATH no es el mismo, ya que en el archivo cron hemos especificado que fuera dicha ruta.

3.3 Formato de los archivos crontab: especificando variables de entorno

En el archivo cron, además de órdenes, podemos especificar una asignación de valores a variables de entorno con la forma <nombre>=<valor>. Sobre <valor> no se realizarían susticiones de variables, con lo que al realizar PATH=$HOME/SO:$PATH no funcionaría.

El demonio cron, por defecto, establece el SHELL a /bin/sh, y el LOGNAME y HOME lo obtiene del archivo /etc/passwd.

Actividad 4.13. Ejecución de scripts con crontab (II)

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 linea 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**

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).

- Creamos el script que creará el archivo cron:

```
$ vi script_4.13
    #!/bin/bash
    echo "SHELL=/bin/sh" > crontab-reducelista
    echo "PATH=" `pwd`"/:$HOME/SO:"$PATH >> crontab-reducelista
    echo "* * * * * reducelista" >> crontab-reducelista
```

- Ejecutamos el script para que cree el archivo cron:

```
$ ./script_4.13
```

- Ejecutamos el archivo generado con crontab:

```
$ crontab cron-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

- Creamos el script:

  $ vi script_4.15

  find ~ -mtime 1 | cpio -pmduv /tmp/salvado$HOME

- Creamos el archivo cron:

  $ vi cron_4.13

  SHELL=/bin/sh

  PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

  #min h dm m ds o

  0 0 * * * /home/paula/reducelista

- Lo ejecutamos:

  $ crontab cron_4.11

3.4 Aspectos de administración del demonio crontab

Tenemos dos archivos, al igual que para el demonio atd, de administración de cron:

- **/etc/cron.allow**: contiene una lista de nombres de usuarios que pueden usar el demonio cron. Si este fichero no existe, se revisaría el fichero /etc/cron.deny;
- **/etc/cron.deny**: contiene una lista de nombres de los usuarios que no pueden usar el demonio cron; si este archivo se encuentra vacío, supone que cualquier usuario puede usar cron.

Si ninguno de estos dos archivos existen, sólo el superusuario puede ejecutar cron.

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.

- Añadimos el usuario en el fichero /etc/cron.deny

  $ vi /etc/cron.deny

  paula (añado en la última línea)

- Ejecutamos cron con dicho usuario

  $ crontab cron_4.15

  You (paula) are not allowed to use this program (crontab)

  See crontab(1) for more information

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

Podemos hacerlo de dos formas distintas:

$ service crond start

Starting crond: [OK]

$ /sbin/service crond start

Starting crond: [OK]

Terminar el servicio cron: /sbin/service crond stop

Podemos hacerlo de dos formas distintas:

$ service crond stop

Stopping crond: [OK]

$ /sbin/service crond stop

Stopping crond: [OK]

---

4. Preguntas de repaso

1. Responda Verdadero o Falso. Contamos con tres scripts para obtener los menús del comedor disponibles en diferentes facultades: ETSIIT, PTS, Farmacia y Ciencias. Y estas son las entradas del crontab:

0,5,10,15,20,25,30,35,40,45,50,55 * * * * /menú_comedor_etsiit.sh

***/5 * * * * /menú_comedor_pts.sh**

5 * * * * /menú_comedor_farmacia.sh

0 * 8 13 2 /menú_comedor_ciencias.sh

1. **El menú de la ETSIIT se obtiene cada 5 minutos:**

   Verdadero

2. **El menú del PTS se obtiene cada 5 minutos:**

Verdadero

3. **El menú de Farmacia se obtiene cada 5 minutos:**

   Falso

4. **El menú del PTS se obtiene cada medio minuto:**

   Falso

5. **El menú de Ciencias se obtiene todos los martes 13 a las 08:00:**

   Falso

2. El archivo /etc/cron.deny:

Si existe, contiene una lista con el nombre de todos los usuarios inhabilitados para usar cron

3. Si queremos ejecutar un demonio periódicamente:

Podemos usar los demonios con la orden crontab <arhivo formato crontab>, y dicho archivo sigue un formato especial para indicar la periodicidad y la orden a ejecutar.

4. Al realizar crontab -l tenemos el siguiente demonio programado: 2 * * * * ls - li > resultado

¿Cada cuánto tiempo se ejecutará?

Todos los días de cualquier mes, a cualquier hora, en el minuto 2.