Prácticas de Sistemas Operativos

Módulo I. Administración del S.O. Linux

Sesión 4. Automatización de tareas

AVISO respecto al trabajo en las aulas de prácticas: Mientras no se diga lo contrario, trabajaremos sobre Ubuntu en nuestra cuenta de usuario personal; no tendremos privilegios de root ni tampoco el servicio de sendmail que pueda ver referenciados en las ayudas de las órdenes que describimos en este guión.

1. Introducción

Ya sea como usuario normal o como administrador del sistema habrá casos en que tendremos que lanzar órdenes cuya ejecución se realice en un determinado momento de tiempo o bien de forma periódica. El entrenamiento en esta materia es el propósito fundamental de esta práctica. Aparte del uso en sí de las órdenes propias para conseguir la ejecución postergada o periódica, hemos de comprender otros aspectos, como el tratamiento que se hace de las variables de entorno del proceso que ejecutará nuestra orden, o el tratamiento de la entrada estándar, salida estándar y salida de error estándar.

Además, el administrador del sistema será el responsable del buen funcionamiento de estos servicios del sistema operativo: el demonio **`atd`** (que proporciona el servicio de ejecución postergada de órdenes) y demonio **`cron`** (que proporciona el servicio de ejecución periódica de órdenes); el mantenimiento de estos demonios también tendrá que ser objeto de nuestro estudio y entrenamiento (parcialmente de momento dada la configuración del software que podemos utilizar en las aulas de prácticas).

2. Objetivos principales

- Conocer el concepto de proceso demonio y profundizar en los conocimientos adquiridos en la sesión anterior acerca de la información asociada a cada proceso (fundamentalmente a través de la orden **`ps`**).
- Conocer y saber utilizar la orden **`at`** y sus órdenes asociadas (**`atq`**, **`atrm`** y **`batch`**).
- Comprender qué valores tienen las variables de entorno del proceso que se lanzará para ejecutar una orden de ejecución postergada.
- Conocer el formato de archivo “crontab” y la orden **`crontab`** para lanzar órdenes de ejecución periódica.
- Comprender qué valores tienen las variables de entorno del proceso que se lanzará para ejecutar una orden de ejecución periódica.

- Respecto al servicio de ejecución periódica de órdenes, conocer la forma de habilitar y deshabilitar a usuarios para su uso.
- Respecto al servicio de ejecución periódica de órdenes, conocer la forma de iniciar y terminar este servicio.

3. Los procesos demonio

Características de un proceso demonio[1]:

- 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 sólo se ponen en marcha cuando son necesarios y se detendrán cuando dejen de serlo.
- En caso de que termine por algún imprevisto es muy común que exista un mecanismo que detecte la terminación y lo rearranque.
- En muchos casos está a la espera de un evento. En el caso frecuente de que el demonio sea un servidor, el evento por el que espera es que llegue una petición de realizar un determinado servicio; habrá siempre, por tanto, algún mecanismo que permita la comunicación entre el demonio en cuanto servidor y los procesos cliente.
- En otros casos, el demonio tiene encomendada una labor que hay que hacer de forma periódica, ya sea cada cierto tiempo o cuando se cumpla cierta condición.
- Es muy frecuente que no haga directamente el trabajo: lanza otros procesos para que lo realicen.
- Para conservar la filosofía de la modularidad propia de Unix, los demonios son programas y no parte del kernel.
- En muchos casos se ejecutan con privilegio de superusuario (UID=0) y tienen por padre al proceso **`init`** (PID=1)

El término castellano "*demonio*" tiene dos equivalencias en inglés con connotaciones bien diferentes: "*daemon*" y "*demon*". El uso del término "*demonio*" en el contexto informático no pretende hacer alusión a aspectos malvados que conlleva "demon", sino que pretende resaltar el aspecto de entidad con vida propia, independiente, que se ejecuta en un plano invisible, a la que alude "*daemon*".

1 Carretero, J. y otros; "Sistemas Operativos. Una visión Aplicada"; McGraw-Hill 2007. Stevens, W. R.; Rago, S. A.; "Advanced Programming in the Unix Environment", Addison-Wesley 2005

Actividad 3.1

A partir de la información proporcionada por la orden **ps** encuentre los datos asociados a los demonios **atd**[2] y **crond**, en concreto: quién es su padre, qué terminal tienen asociado y cuál es su usuario.

4. Ejecutar tareas a una determinada hora: demonio atd

Con el demonio ***atd*** podemos provocar la ejecución de una orden en un momento de tiempo especificado. Como usuarios de este servicio interaccionamos con atd con las siguientes órdenes:

at : ordenar la ejecución de órdenes a una determinada hora

atq: consultar la lista de órdenes

atrm: eliminar órdenes

batch: ordenar la ejecución de órdenes que se ejecutarán cuando la carga del sistema sea baja.

4.1. Orden at

Su sintaxis completa es:

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

La orden ***at*** lee órdenes de la entrada estándar o del archivo <script> y provoca su ejecución a la hora especificada en TIME (una sola vez), usando **/bin/sh**. TIME admite muchos formatos, por ejemplo HH:MM. Para una descripción de las posibilidades para expresar la hora vea la ayuda de at y el contenido del archivo **/usr/share/doc/at/timespec.**

Con el ejemplo siguiente, a una determinada hora (17:10) se generará la lista de archivos del directorio home en un archivo de nombre ***listahome***

```
at 17:10
at> ls ~ > listahome
at> <Ctrl-D>
```

2 Si el demonio atd no está instalado en su sistema puede descargarlo del repositorio: http://rpm.pbone.net con nombre at-3.1.12-5.fc14.i686.rpm. Hay que instalarlo y arrancar el servicio con la orden "`service atd start`".

Actividad 4.1

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?

Actividad 4.2

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

b) un minuto después de la medianoche de hoy

c) a las 17 horas y 30 minutos de mañana

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

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

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

4.2. Sobre el entorno de ejecución de las órdenes

Cuando sea el momento que se especificó en la orden **`at`**, se lanzará una shell **`/bin/sh`** para ejecutar el archivo <script> que se indicó (o si no se especificó, se lanza el conjunto de ordenes que se tomaron de la entrada estándar); podemos preguntarnos sobre cuál es el entorno de este nuevo proceso..... proponemos que investigue usted mismo para dar respuesta a las siguientes preguntas.

Actividad 4.3

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

1. ¿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?
2. ¿qué máscara de creación de archivos umask tiene? ¿es la heredada del padre o la que se usa por omisión?
3. ¿hereda las variables locales 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 4.4

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
```

4.3. Salida estándar y salida de error estándar

Si ejecutando el intérprete de órdenes de forma interactiva lanzamos la ejecución de un script, la nueva ejecución del shell que se lanza tiene como entrada estándar, salida estándar y salida de error estándar al teclado (para la entrada) y pantalla (para salida y salida de error) del terminal desde el que estamos trabajando. Sin embargo, al lanzar la ejecución asíncrona de una tarea con la orden **`at`** NO estamos creando un proceso hijo de nuestra shell, este nuevo proceso NO tendrá como entrada estándar, salida estándar y salida de error estándar los asociados a nuestra consola de terminal.

Cuando la orden lanzada de forma retardada sea ejecutada, lo que se genere en 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`** (la ubicación concreta puede depender de la instalación), si el servicio está disponible. Hemos de tener esto en cuenta y considerar sus consecuencias. Por ejemplo, hemos de vigilar que no se generen un número excesivamente elevado de salidas que pudieran crear problemas de espacio.

En el caso de lanzar la ejecución de órdenes que generan muchos mensajes en la salida estándar o salida de error estándar podríamos redirigirlas a /dev/null para que se “pierda” y no inunde el buzón de correo:

```
at now + 1 minute
at> tar cvf /backups/backup.tar . 1>> ~/salidas 2> /dev/null
at> <ctrl-D>
```

También puede ser conveniente, como se ve arriba, redirigir la salida estándar del programa a un archivo para que pueda consultarse en caso de necesidad.

En el ejemplo anterior estamos seguros del buen funcionamiento de la línea en que está la orden **`tar`**, de modo que sabemos que sus mensajes de error no serán de nuestro interés. Pero situémonos ahora en el caso de que **hemos lanzado una orden con at sin haber redirigido la salida de error.** Entonces nos pasará totalmente desapercibido el error, no tendremos ningún resultado de la ejecución de nuestro trabajo y nos parecerá que no ha funcionado el mecanismo de ejecución postergada. Entre los errores más frecuentes está el de crear un script sin darle inicialmente permiso de ejecución, de modo que cuando llegue su momento la ejecución no será posible, no tendremos a la vista el mensaje de error “Permiso denegado” y parecerá que el mecanismo de la orden at no funciona. Otro error frecuente es no tener incluido en la lista de

búsqueda al directorio actual, de modo que invocar a un script que cuelga de donde estamos simplemente por su nombre sin anteponerle **./** dará el error "orden no encontrada" que, en el caso de una ejecución retardada, se perderá. Si el mecanismo **sendmail** está funcionando entonces recibiremos un correo con los mensajes de error. Si no es el caso, hemos de ser nosotros mismos quienes redirijamos las salidas de error para poder rastrear la ejecución de nuestra orden.

Actividad 4.5

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.

4.4 Orden batch

La orden **batch** es equivalente a **at** excepto que no especificamos la hora de ejecución, sino que el trabajo especificado se ejecutará cuando la carga de trabajos del sistema esté bajo cierto valor umbral que se especifica a la hora de lanzar el demonio **atd**.

Actividad 4.6

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). Utilice 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). Experimente para comprobar cómo al ir disminuyendo la carga de trabajos habrá un momento en que se ejecuten los trabajos lanzados a la cola batch.

4.5. Orden at: trabajando con colas

La orden **at** gestiona distintas colas de trabajos que esperan ser ejecutados; se designan con una única letra de la a ***a*** la ***z*** y de **A** a **Z**; la cola por omisión es **a**, la cola **b** se usa para los trabajos batch, y a partir de ahí las distintas colas van teniendo menor prioridad al ir pasando de **c** en adelante. La cola a la que deseamos enviar un trabajo es especificada con la opción **[-q queue]** en la orden **at** (consulte la ayuda de **at** para más información).

Actividad 4.7

Construye tres script que deberá 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**. Visualice en algún momento los trabajos asignados a las distintas colas.

4.6. Aspectos de administración del demonio atd

Los dos archivos de configuración **`/etc/at.deny`** y **`/etc/at.allow`** determinan qué usuarios pueden usar la orden at. El archivo **`at.allow`**, si existe, contiene una lista con el nombre de todos los usuarios habilitados para ello (uno por línea). Si no existe el archivo allow, se comprobará el contenido del archivo deny y se entenderá que un usuario podrá usar at a no ser que esté presente en deny. Si ninguno de estos dos archivos existe entonces qué usuarios están autorizados depende del sistema linux concreto y de los parámetros de configuración.

5. Ejecuciones periódicas: demonio cron

Muchas de las tareas de administración se tienen que llevar a cabo de forma periódica, a continuación veremos cómo el sistema operativo nos brinda facilidades para ello.

Uno de los demonios básicos es **`cron`**, responsable de ejecutar órdenes con una periodicidad determinada. La especificación de las tareas que se desea que ejecute 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`**.

5.1 Formato de los archivos crontab: especificando órdenes

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

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

Los campos minuto, hora, día-del-mes, mes y día-de-la-semana se encargan de indicar la periodicidad con que deseamos que se ejecute **orden**. Si se trata de una orden shell o un script se lanzará **`/bin/sh`** para su ejecución, y si es la ruta de un archivo ejecutable entonces se provocará su ejecución.

Cada uno de los campos de determinación del tiempo (minuto, hora, día-del-mes, mes, día-de-la-semana) puede contener:

un asterisco, que indica cualquier valor posible

un número entero, que activa ese valor determinado

dos enteros separados por un guión, que indican un rango de valores

una serie de enteros o rangos separados por una coma, activando cualquier valor de los que aparecen en la lista.

Ejemplo de especificación de una hora:

```
1 20 * * 1-5
```

esto significa "a las 20 horas y 1 minuto de lunes a viernes".

Si se han indicado valores concretos para día-del-mes y día-de-la-semana, se entenderá que la condición es que se cumpla ***día-del-mes OR día-de-la-semana*** (en lugar de AND como podría pensarse); por ejemplo

```
0,30 * 13 * 5
```

significa "cada media hora el viernes y cada media hora el día 13 del mes" y no "cada media hora de todos los viernes 13".

Consulte la ayuda del formato de archivo **`crontab`** (en la sección 5 del manual) para una descripción completa de todas las posibilidades sobre la especificación del tiempo.

5.2. La orden crontab

Se encarga de instalar, desinstalar o listar los trabajos que procesará el demonio cron. La sintaxis para especificar el archivo <file> como archivo con formato "crontab" que contiene la lista de trabajos para cron es:

crontab <file>

Actividad 5.1

Al igual que se investigó en la Actividad 4.4 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.

Actividad 5.2

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). Utilice 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`**.

Pruebe la orden **`crontab -l`** para ver la lista actual de trabajos (consulte la ayuda para ver las restantes posibilidades de esta orden para gestionar la lista actual de trabajos).

Actividad 5.3

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.

Actividad 5.4

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?

5.3 Formato de los archivos crontab: especificando variables de entorno

Una línea de un archivo crontab puede ser o bien una línea de especificación de una orden para cron como hemos explicado hasta ahora, o bien una línea de asignación de valores a variables de entorno, con la forma

<nombre>=<valor>

Algunas variables de entorno son establecidas automáticamente por el demonio cron, como las siguientes:

- SHELL se establece a /bin/sh

- LOGNAME y HOME se toman del archivo /etc/passwd

Un detalle importante es que sobre <valor> no se realizan sustituciones de variables, de modo que una línea como la siguiente no funcionaría para utilizar los elementos de la lista de búsqueda actual:

PATH=$HOME/SO:$PATH

Actividad 5.5

Practicamos ahora lo que acabamos de explicar situándonos en lo que hemos realizado en la actividad 5.3. 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** láncelo con la orden **`crontab`**. Compruebe 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 5.3 en que el directorio $HOME/SO no estaba en la lista de búsqueda).

Actividad 5.6

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 5.7

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 4.5; ahora queremos que se copien los archivos encontrados por find utilizando la orden **cpio:**

<orden find de la actividad 4.5> | cpio -pmduv /tmp/salvado$HOME

Una característica interesante de esta orden (y que no tiene **cp**) es que puede tomar de la entrada estándar los archivos origen a copiar; con las opciones que se presentan en el ejemplo anterior replica la estructura original manteniendo metadatos de especial interés como usuario propietario y grupo propietario (consulte la ayuda de cpio para obtener información sobre cada una de las opciones).

Esto puede ser una labor interesante de programar para un administrador de sistemas, con objeto de tener una copia de los archivos que se han modificado; esto tendrá sentido si previamente se ha hecho un salvado global. Por ejemplo una vez al mes se puede hacer un salvado global, y diariamente salvar únicamente los archivos modificados en ese día.

Una posibilidad interesante es que la copia se haga en un dispositivo que acabamos de montar, y justo al terminar lo desmontamos; esto aumenta la seguridad de esa copia ante posibles ataques:

mount /dev/loop0 /directoriosalvados

<orden find> | <orden cpio>

umount /dev/loop0

Como última observación, si el dispositivo y punto de montaje usados en esa orden mount no están en el fstab serán más difíciles de detectar por un intruso que acceda a nuestro sistema.

5.4 Aspectos de administración del demonio crontab

Para realizar este apartado hemos de trabajar como usuario root; desde las aulas de prácticas la única forma de conseguirlo es lanzando User Mode Linux (UML) como se indicaba en el primer guión.

Los dos archivos de configuración **/etc/cron.deny** y **/etc/cron.allow** determinan qué usuarios pueden ejecutar la orden crontab. Su significado es equivalente a los archivos **/etc/at.deny** y **/etc/at.allow** explicados anteriormente.

Actividad 5.8

Como usuario root, deshabilite/habilite a un determinado usuario para utilizar el servicio cron; compruebe que efectivamente funciona.

Actividad 5.9

Iniciar y terminar el servicio cron. Pruebe 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`**