

Sistemas Operativos

Sesión 4. Automatización de tareas



Sistemas Operativos Practica 4

Actividad 3.1

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.

Introducimos la siguiente orden en el terminal: `# ps -lA`

Donde: `ps`: orden de búsqueda de procesos.

`l`: opción para que nos los muestre con detalles.

`A`: opción que muestra todos los procesos.

Este comando se rige por la siguiente regla: `# ps [opción]`

```

1 S      0 1166      1 0 80    0 - 3419 pause ?          00:00:00 sendmail
1 S     51 1174      1 0 80    0 - 2984 pause ?          00:00:00 sendmail
1 S      0 1185      1 0 80    0 - 949  hrttime ?         00:00:00 crond
5 S      0 1198      1 0 80    0 - 718  hrttime ?         00:00:00 atd
4 S      0 1211      1 0 80    0 - 926  wait  ?            00:00:00 login
4 S      0 1212 1211 1 80    0 - 794  wait  tty0          00:00:00 bash
4 R      0 1224 1212 0 80    0 - 656  -      tty0          00:00:00 ps
[root@localhost ~]#

```

.kernel32 : kernel32-3.0.4

Viendo la imagen podemos afirmar que:

	padre	terminal	usuario
atd	1	? por lo tanto no tiene una terminal asociada	0 el usuario es root
crond	1	? por lo tanto no tiene una terminal asociada	0 el usuario es root

Actividad 4.1

[man date](#), [man at](#)

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?



```
.kernel32 : kernel32-3.0.4 - Konsole
Archivo Editar Ver Marcadores Preferencias Ayuda
[root@localhost HDD]# date +%H:%m
14:10
```

El script que generamos simplemente tiene que contener lo siguiente:

```
#!/bin/bash

ls -ARl ~> listahome-$(date +%Y-%j-%T-$$).txt
```

En primer lugar miramos la hora que es con la siguiente sentencia:

```
# date +%H:%m
```

Donde: date es el comando para mostrar fechas.

 + para iniciar la entrada de ordenes e formato.

 %H muestra la hora (24h).

 %m muestra los minutos.

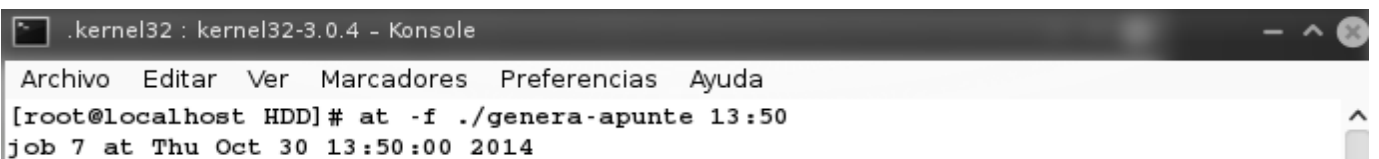
La orden que introducimos para que se ejecute el fichero dentro de un minuto es:

```
# at -f ./genera-apunte 14:51
```

Donde: at es el comando que realizara el guardado de la tarea

 -f es la opcion para ejecutar ficheros.

 ./genera-apunte el script que vamos a ejecutar



```
.kernel32 : kernel32-3.0.4 - Konsole
Archivo Editar Ver Marcadores Preferencias Ayuda
[root@localhost HDD]# at -f ./genera-apunte 13:50
job 7 at Thu Oct 30 13:50:00 2014
```

Comprobamos que todo se creo adecuadamente con un simple ls.

```
.kernel32 : kernel32-3.0.4 - Konsole
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
[root@localhost HDD]# ls
at.txt
genera-apunte
listahome-2014-302-13:50:00-1238.txt
man-pages-es-1.55-1.noarch.rpm
man-pages-es-extra-0.8a-1.noarch.rpm
```

Otra solución es directamente el siguiente comando y comprobar con atq:

```
# at -f ./genera-apunte now + 1 minutes
```

Actividad 4.2

Lanza varias órdenes at utilizando distintas formas de especificar el tiempo como las siguientes: (señal de utilidad la opción -v):

a) a medianoche de hoy

```
# at -f ./genera-apunte midnight
```

b) un minuto después de la medianoche de hoy

```
# at -f ./genera-apunte midnight+1minutes
```

c) a las 17 horas y 30 minutos de mañana

```
# at -f ./genera-apunte 17:30 tomorrow
```

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

```
# at -f ./genera-apunte midnight DEC 25
```

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

```
# at -f ./genera-apunte JAN 1
```

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

```
.kernel32 : kernel32-3.0.4 - Konsole
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
[root@localhost HDD]# atq
18      Fri Oct 31 00:00:00 2014 a root
20      Fri Oct 31 17:30:00 2014 a root
1       Wed Oct 29 06:58:00 2014 a root
22      Thu Jan  1 12:07:00 2015 a root
21      Thu Dec 25 00:00:00 2014 a root
19      Fri Oct 31 00:01:00 2014 a root
```

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?

El directorio inicial es el `home`, y hereda la ubicación de ejecución de un script.

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:

```
nombreadarchivo=`date +%Y-%j-%T`

ps -ef > $nombreadarchivo

echo Mi pid = $$ >> $nombreadarchivo
```

Fichero que obtenemos: (parte final)

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	12:01	?	00:00:00	/sbin/init
root	2	0	0	12:01	?	00:00:00	[kthreadd]
root	3	2	0	12:01	?	00:00:00	[ksoftirqd/0]
root	1185	1	0	12:01	?	00:00:00	crond
root	1198	1	0	12:01	?	00:00:00	/usr/sbin/atd
root	1210	1	0	12:01	?	00:00:00	login -- root
root	1212	1210	0	12:06	tty0	00:00:00	-bash
root	1245	2	0	12:20	?	00:00:00	[flush-98:0]
root	1249	1198	0	12:21	?	00:00:00	/usr/sbin/atd
root	1250	1249	0	12:21	?	00:00:00	sh

root	1252	1250	0	12:21	?	00:00:00	/bin/bash
root	1254	1252	0	12:21	?	00:00:00	ps -ef

Mi pid = 1252

Como podemos ver el padre de mi ejecución es el proceso 1250

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.

El script generado para esta función es el siguiente:

```
#!/bin/bash

find /root -mtime 0 > modificados_$(date +"<%Y"><%j"><%h:%m">")
```

Lista que obtenemos:

```
/root/HDD
/root/HDD/Script
/root/HDD/listahome-2014-302-13:50:00-1238.txt
/root/HDD/find
/root/HDD/listahome-2014-303-11:37:00-1329.txt
/root/HDD/modificados_<2014><303><Oct:10>
/root/HDD/.directory
/root/HDD/2014-303-12:22:00
/root/HDD/modificados_<2014><30><12:10>
/root/HDD/modificados_<2014><303><Oct:10>
/root/HDD/2014-303-12:19:52
/root/HDD/listahome-2014-303-12:00:00-1372.txt
```

Actividad 4.6

[man kill](#), [man jobs](#), [man bg](#), [man fg](#)

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.

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.

Actividad 5.1

[man crond](#)

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.

Editamos con la funcion crontab -e:

```
# * * * * * /root/HDD/Script
```

Y obtenemos el siguiente fichero pasado 1 min:

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	19:34	?	00:00:00	/sbin/init
root	2	0	0	19:34	?	00:00:00	[kthreadd]
root	1185	1	0	19:34	?	00:00:00	crond
root	1198	1	0	19:34	?	00:00:00	/usr/sbin/atd
root	1210	1	0	19:34	?	00:00:00	login -- root
root	1214	1210	0	19:35	tty0	00:00:00	-bash

root	1265	1185	0	19:43	?	00:00:00	CROND
root	1266	1265	0	19:43	?	00:00:00	/bin/bash /root/HDD/script
root	1269	1266	0	19:43	?	00:00:00	ps -ef

Mi pid = 1266

Como podemos ver el padre es 1265.

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).

Introducimos en crontab -e:

```
* * * * * /root/remove
```

Script remove:

```
#!/bin/bash

rm -fv /tmp/varios/core* >> /tmp/listacores
```

Actividad 5.3

[man head](#), [man tail](#)

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.

```
tail /tmp/listacores > /tmp/listacores2 && mv -f /tmp/listacores2 /tmp/listacores
```

Actividad 5.4

Construye un sencillo script que escriba en el archivo ~/S0/listabusqueda una nueva línea con 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?

✓ 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 línea la asignación y además el directorio \$HOME/SO a la variable PATH de la lista de búsqueda actual

.... 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. 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 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

[man cpio](#)

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, 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, 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.

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`