Problemas introductorios sobre la shell Bash (Ejemplos resueltos-Segunda Parte)

September 27, 2024

Problema 1. Los archivos /etc/passwd y /etc/shadow son la base de uno de los métodos de autenticación más comunes en los sistemas Linux. El archivo /etc/passwd contiene información es una tabla en formato texto con información básica de cada usuario, mientras que el fichero /etc/shadow contiene un password encriptado para cada usuario. Estos archivos son utilizados por los procesos "login" para autenticar un usuario y lanzar una shell u otro programa como un proceso del usuario autenticado.

El archivo /etc/passwd es una tabla de texto. Tiene este aspecto:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
```

Cada línea es la información del usuario, que se descompone en campos separados por ":" El significado de los campos es el siguiente:

1. El primer campo es el nombre del usuario. Este nombre es el que tenemos que utilizar para autenticarnos con login, y es también el que obtenemos cuando escribimos el comando whoami en nuestra shell o bien consultamos la variable USER.

2. El segundo campo en los sistemas linux modernos, siempre será el carácter "x". Antiguamente se utilizaba para guardar el password del usuario encriptado. Actualmente el password encriptado se guarda en el archivo /etc/shadow.

3. El tercer campo es el identificador numérico del usuario. Se denomina UID y es único para cada usuario.

4. El cuarto campo es el identificador numérico del grupo primario del usuario. Se denomina GID. El resto de grupos a los que pertenece el usuario se puede consultar en el archivo /etc/group

5. El quinto campo es información personal del usario. Puede ser un único valor, o más de uno, en cuyo caso tendríamos una lista separada por comas. El primer valor es el nombre completo del usuario.

6. El sexto campo nos dice la ruta hasta el directorio home del usuario. En realidad esto lo debemos interpretar como el directorio de trabajo del proceso que se lanza después de ejecutar el programa especificado en el séptimo campo. En el caso particular de que se lance una shell sí sería el directorio home del usuario.

7. El séptimo campo establece la shell o el programa que se ejecutará para el usuario una vez se autentique con login.

En este problema vamos a utilizar el filtro grep para extraer información del archivo /etc/passwd.

El filtro grep sirve para determinar en un grupo de archivos cuáles contienen líneas que coinciden con cierto patrón y volcar esas líneas en la salida estándar. El formato básico de grep es:

```
grep opciones patrones archivos
```

Si no se introducen archivos o uno de ellos es "-" se utiliza la entrada estándar. Los patrones se especifican como expresiones regulares. Se admiten tres variantes de las expresiones regulares: básica, extendida y Perl. En cuanto a las opciones podemos distinguir cuatro clases:

- Tipo de expresión regular: básicas, extendidas, perl o cadena.
- Control del encaje de la expresión regular: obtener las expresiones de un archivo, ignorar la distinción entre mayúsculas y minúsculas, obtener las líneas que no encajen en lugar de las que encajen, y otras.
- Formato de la salida: mostrar solo el número de líneas que encajan, mostrar solo los nombres de los arhivos con encajes o no encajes, mostrar delante de la línea que encaja el nombre del archivo o el número de línea, mostrar los alrededores de la línea, y otras.
- Control de los archivos que se procesan: hacer b'usqueda recursiva, archivos y directorios a excluir de la búsqueda, tratamiento de los archivos binarios, etcćetera.

1. Utiliza el comando grep para contar el número de usuarios en /etc/passwd, cuyo directorio de trabajo se establece en /bin
2. Utiliza el comando grep para obtener una lista de usuarios ordenado alfabéticamente, que al autenticarse lanzan el programa /usr/sbin/nologin. El programa nologin muestra un mensaje acerca de que el sistema no permite el login de un usuario y termina con código de salida diferente de cero.
3. El archivo /etc/group define los grupos de usuarios del sistema. Cada línea se refiere a un grupo. El primer campo es el nombre del grupo, y el cuarto y último campo es una lista separada por comas de los usuarios que pertenecen a un grupo. Obtén una lista de los grupos a los que pertenece el usuario que tengan más de un usuario.

Solución. En la solución de este ejercicio, para cada apartado se muestra el comando, la salida que realmente produce, y la explicación de la misma.

1.
   - (a) `grep -c '/bin:[^:]*$' /etc/passwd`
     (b) `cut -d ':' -f 6 /etc/passwd | grep -c -F -x '/bin'`
   - 3
     user1@host:~$
   - Se muestran dos soluciones. La primera hace uso de expresiones regulares básicas, donde nos aprovechamos de que el campo buscado es el penúltimo y establecemos que hay que encontrar /bin en ese campo. La expresi'on regular utiliza una expresión de corchetes para indicar todos los caracteres excepto ":", y un operador de repetición que indica la repetición de 0 o más veces del carácter. La opción -c hace que grep muestre el número de líneas que encajan en lugar de los encajes.

     La segunda solución utiliza el filtro cut par tomar una columna concreta de la table y luego se aprovecha de grep mediante una tubería para seleccionar por líneas a partir de la b'usqueda de una cadena. La opción -F establece que no se interprete el patr'on como una expresión regular sino como una cadena, mientras que la opción -x exige el encaje completo de la línea en el patrón.
2.
   - (a) `grep '/usr/sbin/nologin$' /etc/passwd | cut -d ':' -f 1 | sort`
   - daemon
     bin
     sys
     ...
     user1@host:~$

- La solución comienza con un grep que usa una expresión reugular básica para encajar el último campo con la ruta buscada. Luego usa el filtro cut para quedarnos solo con la primera de las columnas, que es la que contiene el nombre del usuario. Finalmente se utiliza el filtro sort para realizar el ordenamiento alfabético.

3.
   - (a) `grep ":${USER},\|,${USER},\|,${USER}$" /etc/group | cut -d ':' -f 1,4`
     (b) `grep ":${USER},[^:]*$\|,${USER}$\|:[^:]*,${USER},[^:]*$" /etc/group | cut -d ':' -f 1,4`
     (c) `grep -E ":${USER},[^:]*$|,${USER}$|:[^:]*,${USER},[^:]*$" /etc/group | cut -d ':' -f 1,4`
   -
     ```
     daemon
     bin
     sys
     ...
     user1@host:~$
     ```
   - Se muestran tres soluciones, algunas más robustas que otras. La primera asume que la separación por comas solo se puede dar en el último campo, el que corresponder al listado de usuarios de cada grupo. Por contra, las dos últimas fuerzan la búsqueda en el último campo.

     En la primera solución se asume que no hay comas junto a la cadena del usuario, salvo en el último campo. Se utilizan el operador OR de las expresiones regulares básicas "\|", para examinar diferentes opciones.

□

Problema 2. El comando ls sirve para mostrar la información de los archivos en un directorio. Su comportamiento por defecto es utilizar los archivos del directorio de trabajo de bash y ordenar la información por el nombre del archivo.

Si se pasa a ls un argumento, este puede ser un directorio o un archivo. Además ls admite varios archivos o directorios, actuando en cada uno de ellos.

Con las opciones de ls podemos conseguir entre otras cosas:

- Hacerlo trabajar recursivamente, de modo que entre en todos los directorios a partir del especificado (-R)
- Cambiar el ordenamiento por defecto: tamaño, tiempo, versión, extensión, o simplemente en el orden en el que están registrados en el directorio (–sort= y opciones cortas equivalentes).
- Mostrar más información, además del nombre (-l)
- Visualizar archivos ocultos (-a, -A)
- Dar el formato adecuado a aspectos como la fecha, el tamaño, etcétera

En este problema tenemos que usar el comando ls combinado con otros filtros para mostrar en una tabla los tamaños y el nombre de los 10 archivos de mayor tamaño en el directorio /usr/bin, utilizando como medida del tamaño bloques de 1KB.

Ayuda: Posibles filtros que suelen emplearse para completar ls: tr, cut, sort, head, tail, uniq. No es necesario emplearlos todos, es una lista de filtros que debes conocer, en este problema se emplean algunos de ellos.

Solución. Para resolver este problema hay que tener en cuenta lo siguiente:

- La salida típica del comando ls -l es algo así:

```
-rwxr-xr-x 1 root   root         5164 Oct 19  2020  xsubpp
-rwxr-xr-x 1 root   root        40360 Mar 18  2018  xvidtune
-rwxr-xr-x 1 root   root        18744 Feb 29  2020  xvinfo
-rwxr-xr-x 1 root   root        51592 Feb 29  2020  xwininfo
```

```
-rwxr-xr-x 1 root   root        18712 Apr 15  2020  xxd
-rwxr-xr-x 1 root   root        80384 Apr 20  2020  xz
lrwxrwxrwx 1 root   root            2 Apr 20  2020  xzcat -> xz
lrwxrwxrwx 1 root   root            6 Apr 20  2020  xzcmp -> xzdiff
-rwxr-xr-x 1 root   root         6632 Apr 20  2020  xzdiff
lrwxrwxrwx 1 root   root            6 Apr 20  2020  xzegrep -> xzgrep
lrwxrwxrwx 1 root   root            6 Apr 20  2020  xzfgrep -> xzgrep
-rwxr-xr-x 1 root   root         5628 Apr 20  2020  xzgrep
-rwxr-xr-x 1 root   root         1802 Apr 20  2020  xzless
-rwxr-xr-x 1 root   root         2161 Apr 20  2020  xzmore
-rwxr-xr-x 1 root   root        63720 Apr  6  2021  yelp
-rwxr-xr-x 1 root   root        39256 Sep  5  2019  yes
lrwxrwxrwx 1 root   root            8 Nov  7  2019  ypdomainname -> hostname
```

El problema es que no es una tabla "bien definida". Podemos apreciar en cada fila columnas separadas por espacios, pero el número de espacios entre cada columna es variable. Además, hay campos problemáticos. Uno de ellos es la fecha, ls muestra la fecha en diferentes formatos dependiendo de la fecha que quiera indicar. Por ejemplo, podemos encontrarnos con:

```
-rw-r--r-- 1 nacho nacho        831 Dec 10  2021 informe.txt
-rw-r--r-- 1 nacho nacho         35 Oct  5 18:03 not
-rw-r--r-- 1 nacho nacho        653 Dec  3  2021 p.rxrt
-rw-r--r-- 1 nacho nacho        400 Nov  4  2021 p.sh
-rw-r--r-- 1 nacho nacho        646 Dec 15  2021 p.txt
-rw-r--r-- 1 nacho nacho        956 Dec 15  2021 p1.txt
-rw-r--r-- 1 nacho nacho        520 Nov  4  2021 p2.sh
drwxr-xr-x 3 nacho nacho       4096 Oct  8  2021 software
-rw-r--r-- 1 nacho nacho        646 Dec 15  2021 sysinfo.txt
drwxr-xr-x 4 nacho nacho       4096 Oct  5 13:58 temp
```

donde vemos que en algunos casos el año se sustituye por la hora y los minutos.

Otro campo problemático, es el del nombre del archivo. En este nombre pueden aparecer espacios, ya que se admiten, o bien porque esté mostr'andose un enlace como por ejemplo:

```
lrwxrwxrwx 1 root   root            8 Nov  7  2019  ypdomainname -> hostname
```

Finalmente, el tamaño del archivo se da por defecto en Bytes. Esta forma de representación se puede cambiar para que den el resultado en bloques del tamaño deseado.

- Los problemas anteriores podemos ir solventándolos con filtros. Un filtro que se usa comunmente es "tr". Este filtro sirve para transformar caracteres individuales. En este caso, lo vamos a usar para una operación que se llama squeeze (estrujar), que básicamente elimina repeticiones consecutivas de espacios. Pueden probar por ejemplo en la línea de comandos:

  ```
  tr -s ' '
  ```

  Verán que si entre dos palabras escriben más de un espacio en la salida tendrán uno solo. Esto ayuda a "regularizar" salidas como la de ls.

- El problema de la fecha se puede solucionar desde el propio comando. El comando ls admite la opción –time-style. Esta opción permite diferentes formatos estándar o incluso definir nuestro propio formato. La forma de introducir el formato libre está descrita en el man del comando date. Así si usan la opción de este modo:

`--time-style=+%s`

forzarán a ls a mostrar el tiempo de la última modificación del archivo como el número de segundos transcurridos desde el 1 de Enero de 1970 a las 0h UTC. Así "regularizarán" también este campo.

- En cuanto al nombre del archivo, aquí no hay muchas soluciones, pero fíjense que es el último campo, y esto verán que ayuda mucho y realmente no será un problema para solucionar este ejercicio.
- Entonces, algunas soluciones posibles pueden ser:

1. ```
   ls -l --block-size=1K --time-style=+%s /usr/bin | tail -n +2 | \
   tr -s ' ' | cut -d ' ' -f 5,7- | sort -n -r -t ' ' -k 1 | head -n 10
   ```
2. ```
   ls -l --block-size=1K --time-style=+%s --sort=size /usr/bin | tail -n +2 | \
   tr -s ' ' | cut -d ' ' -f 5,7- |  head -n 10
   ```

La diferencia entre la primera y la segunda solución es que la segunda aprovecha la opción –sort del comando, para ordenar por tamaño, que por defecto colocará primero los archivos de mayor tamaño. Las opción –block-size, hace que se muestren los tamaño en bloques, cuyo tamaño es especificado en la propia opción, en este caso 1KB. Además se ha establecido el estilo de la información temporal para que se muestre en segundos.

El primer filtro que se aplica es tail. En este caso se busca eliminar la primera línea de la salida de ls, que muestra el total y que no nos interesa en este caso.

A continuación se elimina los espacios repetidos con el filtro tr. La idea es que todas las columnas queden separadas por un espacios para que el filtro cut pueda trabajar adecuadamente.

El filtro cut define el separador con la opción -d (observen el quoting alrededor del espacio). Luego se establecen los campos a mostrar con la opción -f. Aquí es donde nos ayuda bastante que el nombre del archivo esté en la última columna porque al especificar 7- le decimos: todos los campos desde el 7 en adelante.

El filtro sort, en la primera de las soluciones utiliza varias opciones. La opción -n indica que se ordene numéricamente y no por orden alfab'etico. La opción -t ' ', establece el separador de campos para sort. La opción -k 1 le indica que ordene por el primer campo. Para terminar, la opci'on -r establece que sort hará el ordenamiento invertido, de mayor a menor.

Finalmente, el último filtro, head, nos permite quedarnos con un número de líneas comenzando desde el principio del archivo.

□

Problema 3. Obtener los nombres de usuario de los tres procesos que más memoria consumen, mostrar sus nombres y el total de la memoria consumido. Utiliza el comando awk en la solución.

Solución.

- Una posible solución es:

```
ps ax -o user,%mem --sort %mem --noheader| \
tail -3 | awk 'BEGIN {printf "Los usuarios:\n"}\
{printf "%s\n",$1 ; sum += $2} \
END {printf "Consumen un %.2f%% de la memoria\n",sum}'
```

- En la solución propuesta es importante el uso de la opción –sort de ps, que simplifica notablemente la línea de comando. Este sort ordena de menor a mayor por el criterio que se especifique. El filtro tail se encarga de obtener los tres últimos procesos, que estarán en las tres últimas líneas. Observen que la salida de ps no va a tener cabecera debido al uso de la opción –noheader.

  El filtro awk permite construir una salida estándar mediante la iteración línea por línea de la entrada estándar. Los comandos a aplicar a cada línea se pueden especificar mediante un archivo o en la propia línea de comandos como es el caso aquí. Cada comando a aplicar tiene dos partes: condición, acción.

La condición, establece cuando aplicar la acción, y si se omite, la acción se aplicará. Las condiciones BEGIN y END son especiales. La primera se cumple antes de empezar la iteración sobre las líneas y la segunda, cuando todas las líneas han sido procesadas. Las variables de awk $1, $2, etcétera hacen referencia a los campos en los que se divide la línea. Por defecto awk asume que el separador de campos es un blank, aunque esto se puede configurar. La parte correspondiente a la acción se pone entre llaves y puede incluir una o varias órdenes separadas por ";". En el ejemplo se está usando printf para formatear el resultado. El programa awk tiene sus propias variables, que pueden ser operadas aritméticamente como se aprecia en el ejemplo con la variable sum que sirve para ir acumulando el porcentaje de memoria consumida.

□

Problema 4. Este problema es continuación del problema 2. Utiliza el filtro awk para añadir un encabezado a la tabla resultante y para mostrar el espacio total ocupado por los archivos.

Solución. • La solución puede ser:

1. ```
   ls -l --sort=size --time-style=+%s /usr/bin  | tail -n +2 | tr -s ' ' | head -n 10 | \
   awk 'BEGIN {total=0;printf "%-10s %-10s\n","Size","File"} \
   {size=$5/1024;total+=size; printf "%-10.0f",size;\
   for(i=7;i<=NF;i++)  printf "%s%s",$i,(i<NF ? OFS : ORS);} \
   END {printf "Total= %.0fKB\n",total}'
   ```
2. ```
   ls -l --sort=size --time-style=+%s /usr/bin  | tail -n +2 | \
   awk 'BEGIN {total=0;printf "%-10s %-10s\n","Size","File"} \
   NR<=10 {size=$5/1024;total+=size;\
   printf "%-10.0f",size; \
   for(i=7;i<=NF;i++)  printf "%s%s",$i,(i<NF ? OFS : ORS);}\
   END {printf "Total= %.0fKB\n",total}'
   ```

- La salida producida es la siguiente:

```
Size       File
150944     mongosh
23276      snap
15131      qemu-system-mips64el
15097      qemu-system-mips64
14956      qemu-system-mipsel
14956      qemu-system-mips
14837      gdb-multiarch
8242       gdb
6899       ctest
6644       qemu-mipsn32-static
Total= 270981KB
```

- Podemos observar que hay una variciación en la parte del comando que depende de ls. Ahora no estamos pidiendo los tamaños en bloques de 1KB porque si calcularámos el total a partir de esto, los redondeos podrían afectar demasiado al resultado del total. En lugar de ello mantenemos los tamaños de los archivos en bytes, y en el filtro awk hacemos el cálculo para obtener el resultado en kilobytes.

  En el comando awk estamos haciendo uso del comando printf para controlar el formato de la salida, con justificación a la izquierda y un ancho fijo para la columna de 10 caracteres. La diferencia entre las dos soluciones es que en la segunda estamos prescindiendo de tr porque awk interpreta varios espacios seguidos como una separación de campos y prescindimos de head porque el propio awk puede usarse para seleccionar las líneas, estableciendo la condición correspondiente.

  La complejidad de las soluciones mostradas con awk viene del campo del nombre del archivo. Este campo puede tener espacios, y en awk a diferencia de cut no se pueden crear "grupos" de campos.

Entonces hay que utilizar un bucle que parte del primer campo donde sabemos que empieza el nombre del archivo y continúa hasta el último campo del registro, cuyo número está en la variable interna de awk NF. El formato de las líneas se consigue usando las variables OFS que es el separador entre campos configurado (normalmente el espacio) y ORS que es el carácter configurado como fin de registro (normalmente newline).

□