Práctica 2

1. Trabajar con bash, órdenes y filtros
2. Ejercicios básicos con órdenes
3. Ejercicios con cut, sort y grep
4. Ejercicios con ps y find
5. Ejercicios sobre expresiones regulares
6. Ejercicios sobre filtros y tuberías
7. Ejercicios para practicar

1. Trabajar con bash, órdenes y filtros

En esta práctica practicaremos con el shell y resolveremos problemas de administración mediante órdenes básicas. Los objetivos de la práctica son los siguientes:

- Conocer los mecanismos básicos del shell y las órdenes.
- Utilizar redirecciones, filtros y tuberías.
- Utilizar órdenes básicas como find y grep.

Durante las sesiones prácticas veremos las características de la plataforma que nos ofrece la consola de Linux gracias al shell y al inventario de órdenes estándar. El profesorado realizará demostraciones en vivo de cómo funcionan las órdenes más comunes y pondrá ejemplos de cómo utilizar tuberías y filtros para realizar acciones avanzadas con la consola.

1.1 Entrenamiento básico

Al comenzar esta asignatura ya debemos tener nociones básicas del manejo de la consola de Linux. Empezaremos repasando estos conocimientos:

- Ver ayuda en línea con man o --help.
- Estructura del sistema de ficheros (rutas, directorio padre, directorio de usuario…)
- Órdenes básicas de manejo de ficheros: ls, cp, mv, rm, touch…
- Órdenes básicas de consola: date, stat, who, clear, exit...
- Uso de caracteres comodín para seleccionar conjuntos de archivos (*, ?, [], etc.)
- Cambiar permisos con chmod, chown y chgrp.

A continuación, debemos familiarizarnos con algunas utilidades básicas de la consola:

- find: busca ficheros recursivamente que cumplan ciertas características como, por ejemplo, nombre, longitud, fecha de modificación, tipo, usuario propietario...
- grep: busca una cadena de texto o un patrón.
- ps: explora la tabla de procesos del núcleo.

1.2 Flujos estándar y redirecciones

La entrada y la salida estándar de cualquier orden se puede redirigir a un fichero:

- orden < fichero_de_entrada
- orden > fichero_de_salida (se crea el fichero o se sobrescribe si ya existe)
- orden >> fichero_de_salida (se añade al fichero si ya existe o se crea si no)

El error estándar es un flujo de salida separado donde la mayoría de los programas emiten los mensajes de error. Se puede redirigir de esta manera:

- orden 2> fichero_de_errores (se crea el fichero o se sobrescribe si ya existe)
- orden 2>> fichero_de_errores (se añade al fichero si ya existe o se crea si no)

Un filtro es un programa que lee líneas de la entrada estándar, las procesa y devuelve las líneas procesadas por la salida estándar. Algunos filtros básicos son:

- cat: concatenar ficheros
- cut: seleccionar un grupo de caracteres o de campos de cada línea
- sort: ordenar las líneas, alfabética o numéricamente
- head: tomar las primeras N líneas de la entrada
- tail: tomar las últimas N líneas de la entrada
- tr: transformar caracteres de la entrada o suprimirlos
- join: hacer un "join" relacional de dos tablas de datos
- wc: contar caracteres, palabras o líneas
- nl: numerar las líneas

Generalmente, los filtros se usan combinados con redirecciones. Por ejemplo, si ejecutamos la orden nl sin asociarlo a ningún fichero empezará a numerar las líneas que introduzcamos por teclado. Para finalizad la entrada de datos desde la consola debemos usar la combinación de teclas Ctrl+d.

La salida de datos de una orden se puede conectar como entrada a una segunda orden usando una tubería. Por ejemplo, la ejecución de la orden:

- ls | wc -l

sirve para contar el número de ficheros en un directorio. Sería equivalente a ejecutar estas tres órdenes de forma consecutiva:

- ls > salida
- wc -l < salida
- rm salida

Claramente, las tuberías presentan varias ventajas: la sintaxis es más compacta, no hay ficheros intermedios y permite la ejecución concurrente o paralela de las órdenes a medida que se procesa el flujo de datos.

La combinación de filtros y tuberías permite la creación de órdenes avanzadas:

- grep: ahora vemos que podemos usarlo como un filtro que se puede combinar con otras órdenes para resolver problemas complejos, especialmente cuando lo combinemos con expresiones regulares. Por ejemplo, podemos comprobar si un determinado usuario existe: grep -e ^usuario1: /etc/passwd | wc -l
- sed (stream editor): edita la entrada y la transforma usando las órdenes típicas de un editor como añadir o suprimir líneas y palabras, buscar y sustituir textos, etc. Por ejemplo, podemos utilizarlo para cambiar una palabra por otra en un fichero sin tener que abrirlo con un editor: sed 's/usuario1/usuario2/' /etc/passwd

También tenemos otros elementos que nos permiten generar órdenes complejas, como por ejemplo el parámetro -exec de la orden find, que permite pasar todo lo que encuentre la orden a otro comando para que ejecute alguna tarea. De forma más general, la orden xargs permite aplicar iterativamente una misma acción a un conjunto de líneas.

2. Ejercicios básicos con órdenes

1. Determinar en qué archivos se encuentra la cadena «PATH».
2. Obtener todos los procesos del sistema.
3. Obtener todos los procesos del usuario actual.
4. Finalizar un proceso con un identificador de proceso dado.
5. Finalizar todos los procesos que consisten en la ejecución de un determinado fichero ejecutable.
6. Lanzar el programa Firefox y matarlo desde una terminal
7. Verificar que el contenido de dos archivos coincide.
8. Hacer que todos los archivos existentes en mi directorio actual sólo puedan ser ejecutados por su propietario.
9. Crear un archivo vacío de nombre «prueba».
10. Cambiar la fecha de la última actualización al fichero «prueba» a 15/01/2002.
11. Cambiar el propietario del fichero «prueba».
12. Cambiar el grupo propietario del fichero «prueba».
13. Localizar a partir del directorio actual todos los archivos propiedad del usuario root.
14. Localizar todos los archivos en el sistema cuyo nombre contenga la cadena «rc».
15. Localizar todos los archivos del sistema que hayan sido modificados en el día de hoy.
16. Borrar todos los ficheros regulares con extensión «.txt» del directorio «/tmp» que tengan más de dos días de antigüedad.

3. Ejercicios con cut, sort y grep

1. Mostrar un listado en pantalla con el primer y el quinto campo de todas las líneas del fichero «/etc/passwd».
2. Mostrar un listado en pantalla que contenga desde el primer byte hasta el tercero y desde el quinto hasta el octavo del fichero «/etc/passwd».
3. Mostrar un listado en pantalla con el primer y el quinto campo de todas las líneas del fichero «/etc/passwd». El listado debe estar ordenado alfabéticamente por el segundo campo.
4. Mostrar un listado en pantalla con el primer, el tercer y el quinto campo de todas las líneas del fichero «/etc/passwd». El listado debe estar ordenado por el identificador de usuario.
5. Mostrar un listado en pantalla con el primer, el tercer y el quinto campo de todas las líneas del fichero «/etc/passwd». El listado debe estar ordenado según el contenido de los caracteres 2, 3 y 4 de cada línea.
6. Mostrar un listado con el primer y el quinto campo del fichero «/etc/passwd» que incluya solamente aquellas líneas que contengan la cadena «root» y esté ordenado alfabéticamente por el primer campo.
7. Mostrar un listado ordenado por el identificador de usuario de todas las líneas del fichero «/etc/passwd» que contengan la cadena «bash».
8. Mostrar un listado numerado de todas las líneas del fichero «/etc/passwd» que contengan la cadena «bash».
9. Mostrar el número de líneas del fichero «/etc/passwd» que contengan la cadena «bash».

4. Ejercicios con ps y find

1. Mostrar un listado que contenga el PID de todos los procesos lanzados en el sistema que en el campo «command» tengan la cadena «bash».
2. Obtener el número de procesos que hay lanzados en el sistema que en el campo «command» tengan la cadena «bash».
3. Mostrar un listado con todos los procesos que se estén ejecutando en el sistema. Cada línea contendrá el identificador y la orden asociada al proceso correspondiente. El listado deberá estar ordenado por el identificador de proceso.
4. Mostrar el tiempo de sistema consumido, el identificador del proceso y la orden ejecutada para todos los procesos que se estén ejecutando en el sistema. El listado debe estar ordenado por el tiempo del sistema consumido.
5. Mostrar todos los procesos pertenecientes al usuario «usuario1» que se estén ejecutando en el sistema, pero solo aquellos que se han lanzado en una terminal (tty) y ordenados por el nombre de la terminal.
6. Mostrar todos los procesos que se están ejecutando en el sistema, que no pertenezcan al usuario «root», con los siguientes atributos: su tamaño en memoria virtual, su identificador de proceso, el nombre del usuario y la orden ejecutada.
7. Mostrar un listado de todos los archivos contenidos en el directorio «/home/usuario1» cuyo propietario sea el usuario «usuario1» y que sean de tipo regular o directorio.
8. Realizar una búsqueda en el directorio «/home/usuario1» de todos los ficheros regulares del usuario «usuario1» y mostrar los permisos para cada uno de ellos.
9. Obtener los archivos del sistema que no sean propiedad del usuario «root» ni del grupo «root» y cuyo tamaño supere los 1024 KiB.
10. Obtener los archivos del sistema que sean propiedad del usuario «root» que sean ejecutables y tengan un tamaño que supere los 1024 KiB.
11. Listar los archivos de los directorios «/bin», «/sbin» y «/usr/bin» cuyos permisos sean 777.

5. Ejercicios sobre expresiones regulares

En estos ejercicios vamos a trabajar con el fichero de texto «/usr/share/dict/words». Este es un fichero que encontramos en muchos sistemas Unix y que contiene una lista enorme de palabras del idioma inglés. Si el fichero no está en el sistema debemos instalarlo ejecutando la orden **sudo dnf install words**.

Los ejercicios consisten en usar grep y expresiones regulares para encontrar:

1. Palabras que empiezan por vocal.
2. Palabras que tienen exactamente cinco letras.
3. Palabras que tienen menos de cinco letras.
4. Palabras que empiezan por vocal y terminan en «tion».
5. Palabras que empiezan por vocal, terminan en «tion» y tienen hasta ocho letras.
6. Palabras que no contienen vocales.
7. Palabras en las que «sh» aparece al menos dos veces.

6. Ejercicios con sobre filtros y tuberías

1. Obtener los nombres de las cuentas de usuario del sistema, ordenados alfabéticamente.
   (pista: cut)
2. Obtener los nombres de cuentas de usuario del sistema que contengan alguna letra mayúscula.
   (pista: grep)
3. Encontrar los GID que se utilizan más de una vez en el fichero «/etc/passwd». El GID es el campo número 4. Hay que devolver la lista de GID que se repiten.
   (pista: uniq)
4. Quitar los comentarios de un fichero de texto. Un comentario empieza por el carácter «#» y llega hasta el final de la línea. La salida de la orden debe ser el contenido del fichero original, con todos los comentarios eliminados.
   (pista: cut)
5. Contar el número de apariciones de una palabra dentro de un fichero de texto. La palabra se pasa como argumento en la orden. Se considera que una palabra es un grupo de caracteres sin espacios ni tabuladores: el espacio o el tabulador actúan como separadores de palabras.
   (pista: usar la opción de grep que muestra por pantalla solo el patrón buscado)
6. Determinar cuántos shells distintos se están usando en fichero «/etc/passwd».
   (pista: sort, uniq, wc)
7. Descubrir si existen varios usuarios con el mismo UID.
   (pista: sort, uniq)
8. ¿Cuántos dispositivos de tipo carácter o bloque hay definidos en el sistema?
   (pista: los dispositivos que reconoce el sistema están en «/dev»)
9. Obtener el número de procesos que hay actualmente ejecutándose en el sistema resolviendo funciones del escritorio Gnome
   (nota: todos los programas del entorno Gnome comienzan por «gnome-»)
10. De los procesos propiedad del usuario «usuario1», obtener sus identificadores y el tiempo que llevan ejecutándose. La relación debe estar en orden inverso por el tiempo de ejecución.
    (pista: uso combinado ps y sort)
11. Listar los archivos del usuario «usuario1» a los que se ha accedido en los últimos siete días.
    (pista: find y expresiones numéricas).
12. Encontrar los ficheros del usuario «usuario1» que terminan con extensión «.change» y en la misma orden cambiar su propietario a «root».
    (pista: find con ejecución de orden en cada hallazgo).
13. Encontrar los ficheros del usuario «usuario1» que terminan con extensión «.sh» y concatenar el contenido de todos ellos en un único fichero llamado «mis_scripts.txt».
    (pista: find con ejecución de orden en cada hallazgo).

7. Ejercicios para practicar

Los ejercicios de esta sección consisten en una serie de cuestiones de dificultad creciente sobre órdenes y filtros del shell que proponemos a los estudiantes para que los resuelvan por su cuenta. Todos los ejercicios planteados se pueden resolver en una sola línea, empleando una combinación de órdenes comunes y tuberías. No hace falta escribir scripts para resolverlos.

Recomendamos que primero trabajen los ejercicios del bloque A, luego los del bloque B y por último los ejercicios avanzados, estos últimos en el orden que cada uno desee. Tengan en cuenta que los ejercicios avanzados están por encima de la capacidad media que se exige para aprobar la asignatura. Serían «de notable» o «de sobresaliente». Están pensados para que los estudiantes trabajen en profundidad con órdenes y sepan interpretar situaciones de la vida real relacionadas con la administración de sistemas.

BLOQUE A: varios ejercicios de filtros, tuberías y redirecciones

Ejercicio A1: ¿Qué hace la siguiente orden?

```
date > /tmp/foo.txt ; who >> /tmp/foo.txt
```

¿Cuál es la diferencia con esta otra orden?

```
date > /tmp/foo.txt ; who > /tmp/foo.txt
```

Ejercicio A2: ¿Qué hace la siguiente orden?

```
ls -lR /boot 2> /tmp/fichA > /tmp/fichB
```

Ejercicio A3: Combinando las órdenes head y tail, muestre la sexta línea del fichero «/etc/passwd».

Ejercicio A4: Contar el número de ficheros del directorio «/boot» (y sólo «/boot», sin recursividad) cuyo nombre contenga una o más letras mayúsculas (pista: grep, wc).

Ejercicio A5: Cada usuario del sistema tiene asignado un shell por defecto (último campo del fichero «/etc/passwd»). Escriba una lista ordenada alfabéticamente de los nombres de usuario (login name) que no usan «/bin/bash» como shell por defecto (pista: cut, grep, sort).

Ejercicio A6: ¿Qué hace la siguiente orden? Modifíquela para que sea más eficiente:

```
cat /etc/group | sort -t: -k1 | cut -d: -f1,3
```

Ejercicio A7: Enumere los shells que usan los usuarios de su sistema, sin que haya repeticiones (pista: sort y uniq).

BLOQUE B: ejercicios con find, grep y ps

Ejercicio B1: ¿Qué hace la siguiente orden?

```
find /etc /boot -type f -newer /etc/passwd
```

Ejercicio B2: Localice todos los archivos que no pertenezcan al usuario «root» cuyo tamaño sea mayor que 10 KiB y menor que 50 KiB.

Ejercicio B3: Calcule la suma MD5 de cada uno de los ficheros bajo «/bin» y «/sbin» que sean de tamaño inferior a 10KiB o bien superior a 1MiB. El cálculo de la suma MD5 se realiza con la orden «md5sum».

Ejercicio B4: Localice todos los archivos del directorio «/root» que tengan permiso de lectura para el grupo propietario, independientemente del resto de permisos.

Ejercicio B5: ¿Qué hace la siguiente orden?

```
cd ; find . -maxdepth 1 \( -empty -o -newer /etc/passwd \)
```

Ejercicio B6: Construye una expresión regular que reconozca direcciones de correo electrónico (por ejemplo hola-mundo@gmail.com).

Utiliza el comando grep para que lea líneas de la entrada estándar (o un fichero de texto) y devuelva solamente aquellas cadenas de texto que son direcciones de correo.

En caso de que la necesites, la especificación completa del formato de las direcciones de correo electrónico está en el estándar RFC822 de Internet (apartado 6.1).

Ejercicio B7: Muestre el tiempo transcurrido desde su lanzamiento, el PID y la orden de todos los procesos que se están ejecutando en el sistema.

Ejercicio B8: Muestre el PID y el propietario de todos los procesos que estén ejecutando el programa bash. La lista debe estar ordenada por el nombre del propietario.

Ejercicio B9: ¿Qué hace la siguiente orden?

```
ps -e --forest
```

Ejercicio B10: ¿Qué hace la siguiente orden?

```
ps -e -opid,ppid,user,pcpu,cputime,cmd --sort=cputime
```

Ejercicio B11: Localice los 5 procesos que han consumido más CPU hasta el momento.

EJERCICIOS AVANZADOS

Ejercicio avanzado 1: buscar ficheros homónimos en varios directorios (dificultad baja). En las carpetas «/usr/bin» y «/etc» existen varios ficheros homónimos (que tienen el mismo nombre), por ejemplo «/usr/bin/passwd» y «/etc/passwd». Obtenga la lista de todos esos ficheros homónimos.

Ejercicio avanzado 2: filtrar procesos (dificultad media). En dos órdenes, muestre:

- Los 10 procesos que llevan más tiempo en el sistema.
- Los 10 procesos que han consumido más CPU hasta el momento.

En ambos casos, muestre la siguiente información de cada proceso: PID, PPID, orden que se lanzó, tiempo en el sistema y consumo acumulado de CPU. Ordene la lista por el criterio de filtrado que se solicita en cada caso (tiempo en el sistema o consumo de CPU).

Ejercicio avanzado 3: propietarios en un directorio (dificultad media). Obtenga la lista de usuarios que son propietarios de algún fichero en el directorio «una_carpeta». Cumpla con estos requisitos:

- La búsqueda debe considerar cualquier tipo de fichero: regulares, dispositivos, directorios...
- No haga recorridos recursivos, sólo rastree los elementos que están directamente accesibles desde el directorio «una_carpeta».
- El listado no debe mostrar duplicados (un usuario sólo puede aparecer una vez en la lista).

Ejercicio avanzado 4: propietarios de múltiples ficheros (dificultad media). Obtenga la lista de usuarios que son propietarios de más de un fichero en el directorio «/tmp» cumpliendo con estos requisitos:

1. La búsqueda debe considerar cualquier tipo de fichero: regulares, dispositivos, directorios...
2. No haga recorridos recursivos, sólo rastree los elementos que están directamente accesibles desde el directorio «/tmp».
3. El listado no debe mostrar duplicados (un usuario sólo puede aparecer una vez en la lista).

Ejercicio avanzado 5: comprimir ficheros grandes (dificultad media). Comprima con gzip cada uno de los ficheros mayores de 10 KiB que existan bajo el directorio actual, de manera que cada fichero mayor de 10 KiB sea reemplazado por su versión comprimida.

Ejercicio avanzado 6: localizar documentos HTML (dificultad media-alta). Obtenga la lista de todos los directorios del sistema que contienen documentos HTML. Los documentos HTML son aquellos que tienen extensión «.htm» o «.html», sin distinguir mayúsculas.

Ejercicio avanzado 7: trasladar archivos grandes (dificultad alta). Este ejercicio es difícil, así que lo plantearemos por iteraciones de complejidad creciente. Todas las versiones deben poder resolverse con una única orden. No se preocupe si no llega a realizar la iteración 4: en la asignatura lo máximo que le podríamos pedir es algo parecido a la versión 3 y normalmente será algo similar a la versión 2.

Iteración 1. Muestre las rutas todos los ficheros del sistema que midan más de 100MiB y que no hayan sido accedidos en el último mes.

Iteración 2. Elimine todos los ficheros que cumplan con los criterios de la versión 1.

Iteración 3. Traslade cada uno de esos ficheros a la carpeta «/grandes-sin-usar», cambiando su nombre por «fichero.YYYYMMDD», donde «fichero» es el nombre original y «YYYYMMDD» es la fecha actual (hoy) en formato año-mes-día.

Ejemplo: si ejecutamos el script el día 12/10/2015, el fichero «penicula.mkv» se traslada a «/grandes-sin-usar/penicula.mkv.20151012».

Iteración 4 (final). Lo mismo que la versión 3, pero el nombre del fichero trasladado debe ser «nombre.YYYYMMDD.extensión», donde «nombre» es el nombre original del fichero sin su extensión, «YYYYMMDD» es la fecha de último acceso el fichero y «extensión» son los últimos caracteres del nombre del fichero, después del carácter «.», si existe.

Ejemplo: el fichero «penicula.mkv», accedido por última vez el día 7/1/2014, se convertirá en «penicula.20140107.mkv».

Contribuciones

García Rodríguez, Carmelo Rubén

Rodríguez Barrera, Eduardo

Santana Jaria, Oliverio Jesús

Santos Espino, José Miguel