Nombre: ……

Fecha: /12/2011 Grupo: 1 □ 2 □ 3 □ 4 □

PRÁCTICA 19
TRABAJANDO CON TUBERÍAS. REDIRECCIÓN DE LA ENTRADA, SALIDA Y ERRORES

Hasta ahora hemos visto en las prácticas anteriores cómo ejecutar mandatos en Linux de forma secuencial (ejecutar un mandato, esperar a que el mismo produzca su salida, ejecutar otro a continuación y así sucesivamente). En la práctica de hoy recuperaremos la idea de redireccionamiento de la salida o de la entrada de un mandato. Muchos de los mandatos en Unix tienen un canal o flujo de entrada (stdin), a partir del cual leen la información que debe ser procesada. Algunos mandatos no requieren de una entrada explícita de información (por ejemplo, ls se ejecuta sin entrada adicional). Sin embargo, otros como por ejemplo "grep" (mandato usado para búsqueda de expresiones regulares) sí que requieren de una entrada estándar en donde buscar las expresiones regulares.

Por defecto, la entrada estándar está definida como el propio intérprete de mandatos en que se ejecuta el mandato correspondiente. Realiza el siguiente ejercicio a modo de comprobación:

1. Visita la página del manual del mandato "grep". Vamos a ejecutar ahora un mandato que nos permita filtrar todas las líneas de un fichero que contengan tu nombre de cuasi:

$grep "cuasi"

¿Qué ha sucedido con el prompt? ¿Qué espera el intérprete de mandatos?

Escribe en el mismo las siguientes líneas:

Hola, soy mi_cuasi y estamos en SI (pulsa "Enter")

¿Qué ha sucedido? Escribe ahora una nueva línea:

Esto es todo por hoy

¿Qué sucede?

El mandato grep nos muestra todas las líneas que contengan el patrón ("mi_cuasi") que le hemos dado entrecomillado. Como no le hemos dicho cuál era su entrada estándar (stdin) de información (por ejemplo, un fichero) ha tomado por defecto como entrada estándar el propio intérprete. Detén el mandato grep mandándole un señal de fin de fichero (EOF, Ctrl + D).

2. Al igual que un flujo de entrada por defecto (stdin), los mandatos también tienen un flujo de salida (stdout). Por ejemplo, con el mandato anterior, cada vez que encontrábamos la cadena "mi_cuasi" en una línea, el mandato mostraba de nuevo esa línea en la misma shell. Por tanto, como

has podido comprobar, el flujo de salida por defecto de este mandato (y en general de los mandatos en Linux) es la propia shell.

En la misma terminal, ejecuta el mandato "ls". ¿Dónde aparece la salida producida por ese mandato? Más adelante en la práctica veremos cómo modificar ese comportamiento (lo cierto es que ya lo hemos usado con anterioridad, como cuando hemos hecho "history >> mandatos" o "yes hola > /dev/null").

3. Aparte de un flujo estándar de entrada (por defecto la shell) y de un flujo estándar de salida (de nuevo, por defecto la shell), cualquier mandato puede producir también una salida de errores (conocida como stderr). En general estos errores nos mostrarán información sobre operaciones que no han podido ser completadas con éxito (por ejemplo, por no disponer de permisos, por falta de recursos del ordenador…).

Realiza las siguientes acciones; muévete al Escritorio (cd) de tu máquina. Crea un directorio de nombre "prohibido" (mkdir). Quítale todos los permisos (por ejemplo, con "chmod 000 prohibido", o su equivalente "chmod a-rwx prohibido"). Trata de listar su contenido. ¿Cuál es el resultado? La salida que has obtenido es lo que se conoce como salida de error. Por defecto, también es volcada a la propia shell donde nos encontramos ejecutando el programa, pero existen formas de redirigirla que veremos a lo largo de la práctica.

Cada uno de los tres flujos anteriores de información (entrada estándar o stdin, salida estándar o stdout y salida de error o stderr) reciben un valor numérico que nos permite referirnos a ellos: stdin es el flujo 0, stdout es el flujo 1 y stderr es el flujo 2. Es importante que retengas la idea de que cada mandato en Linux puede contar con un flujo de entrada, uno de salida y uno de error.

4. Los dos operadores que nos permiten redirigir la salida de un mandato (y por salida entendemos tanto la salida de error como la salida estándar) son ">" y ">>". La diferencia entre ">" y ">>" es que el primero (">") siempre crea un nuevo fichero conteniendo la información volcada (y si existía un fichero con ese nombre lo "pisa"), mientras que el segundo (">>") crea un fichero con la salida, o si el fichero ya existía simplemente concatena su salida con la información que ya hubiese en el fichero.

5. Vuelve a repetir la operación que hicimos con anterioridad sobre la carpeta "prohibido", pero redirigiendo ahora la salida de errores a un fichero de nombre "resultado":

$ls –l prohibido 2>resultado

Comprueba el contenido del fichero "resultado" (less resultado).

La sintaxis del mandato anterior debería entenderse como "realiza la operación "ls –l prohibido", y su segundo flujo de información generado, es decir stderr, redirígelo al fichero resultado".

6. Veamos ahora también una nueva forma de generar un fichero de texto siguiendo las mismas ideas. El mandato "cat" se utiliza para concatenar uno o más ficheros (concatenar un único fichero es equivalente a crear un fichero igual a ese). Su sintaxis es "cat fichero1 fichero2 fichero3". Por defecto, el resultado de concatenar los ficheros, se muestra en el mismo intérprete. Ejecuta el mandato:

$cat > ciudades (equivalente a cat 1> ciudades);

El intérprete espera que le demos la entrada para el mandato "cat". La salida del mandato se hará al fichero "ciudades". Como no le hemos dado ningún fichero al mandato cat para concatenar, por defecto considera la propia shell como "fichero" de entrada. Escribe el nombre de 10 ciudades, pulsando "Enter" después de cada una de ellas (pulsa "Ctrl + D" para terminar la operación).

7. Comprueba con "less ciudades" que el fichero se ha creado correctamente y que contiene la información que esperabas.

8. Ejecuta el mandato

$cat >> ciudades

Introduce el nombre de otras dos ciudades (pulsando "Enter" entre medio). ¿Cuál es el contenido del fichero ahora? ¿Se ha sobreescrito el mismo?

9. Otro mandato que nos permite mandar mensajes es "echo". Ejecuta el siguiente mandato:

$echo "El usuario activo es $USER"

¿Cuál es el resultado obtenido? De nuevo, la salida estándar para "echo" es la propia shell, pero esto puede ser cambiado por medio de la redirección de su salida:

$echo "Murcia" >> ciudades

Comprueba el contenido del fichero "ciudades" de nuevo. ¿Cuál es ahora el mismo?

Como hemos ido viendo en las prácticas anteriores de Linux, en Linux la mayor parte de la información del sistema se gestiona por medio de ficheros de texto; es por este motivo que existen múltiples mandatos que nos permiten tratar este tipo de ficheros:

10. Ejecuta el mandato:

$sort ciudades

¿Cuál ha sido el resultado obtenido?

11. Ejecuta el mandato:

$sort –r ciudades

¿Cuál ha sido el resultado obtenido?

12. Ejecuta el mandato:

$sort ciudades > ciudades.ordenadas

Comprueba que el fichero "ciudades.ordenadas" contiene la misma información que el fichero ciudades.

13. Concatena el fichero de usuarios de tu ordenador (/etc/passwd) con el fichero de grupos (/etc/group) y redirige la salida a un fichero en el Escritorio de nombre "usuarios_y_grupos".

Otro mandato útil para trabajar con textos es el mandato "grep". El mismo busca dentro de un texto todas las líneas que coincidan con la expresión o patrón que nosotros le indiquemos. Por ejemplo, supón que en el fichero "usuarios_y_grupos" queremos conocer la información referente a nuestro usuario (alumno). Ejecuta el siguiente mandato:

$grep "alumno" usuarios_y_grupos

¿Qué información has obtenido como respuesta?

14. Comprueba el contenido del fichero "/etc/protocols" por medio de less. Supón que sólo estás interesado en los protocolos relacionados con "ip". Ejecuta el mandato:

$grep "ip" /etc/protocols

15. Busca tu nombre de usuario en todos los ficheros de la carpeta "/etc":

$grep "alumno" /etc/*

Observa que has recibido varios mensajes de error ("Permiso denegado").

16. Vuelve a ejecutar el mandato anterior redireccionando la salida estándar a un fichero de nombre "apariciones_alumno" y los errores obtenidos a un fichero "errores". Repasa los ejemplos que hicimos al principio de la práctica con la redirección de "1" y "2". Comprueba el contenido de ambos ficheros.

17. Comprueba la utilidad del mandato wc (por ejemplo, por medio de man). Comprueba el número de caracteres del fichero "usuarios_y_grupos". Comprueba su número de palabras. Comprueba su número de líneas.

Aparte de las utilidades anteriores, existe otro operador de control en la shell de Linux que nos permite redirigir la salida de un mandato (del modo como hemos hecho con ">" y ">>") para que se convierta en la entrada de un nuevo mandato. Hasta ahora hemos visto varios ejemplos de cómo hacer lo mismo pasando la información por un fichero intermedio:

$cat /etc/passwd /etc/group > usuarios_y_grupos

$grep “alumno” usuarios_y_grupos

Por medio de lo que se conoce como interconexiones o tuberías (del inglés pipe) podemos hacer que la salida de un mandato se convierta directamente en la entrada de otro. El carácter que se utiliza para crear una tubería que redirija la salida de un mandato a la entrada de otro es “|”.

18. Prueba la salida del siguiente mandato:

$cat /etc/passwd /etc/group | grep “alumno”

¿Qué resultado has obtenido? ¿Cuál ha sido en este caso el flujo de entrada para el mandato “grep”?

19. Por supuesto, varias tuberías pueden ser enlazadas de forma sucesiva. Crea un mandato que cumpla la siguiente función: concatenar los ficheros /etc/passwd y /etc/group (cat), filtrar todas las líneas que contengan la palabra alumno (grep) y contarlas (wc).

20. Crea un nuevo mandato que de nuevo concatene los ficheros /etc/passwd y /etc/group, filtre todas las líneas que contengan la cadena “root” y las muestre ordenadas en orden inverso (sort).

21. Repite el mandato del ejercicio anterior, pero redireccionando la salida además a un fichero de nombre “informacion_root”.

22. Crea un mandato que, a partir del listado largo de los contenidos del directorio /usr/bin, muestre todas aquellas entradas que contienen “mk”.

23. Crea un mandato que, a partir del listado largo de los contenidos del directorio /sbin, muestre todas aquellas entradas que contienen “mk”.

24. Trata de crear un mandato que, a partir del contenido de los directorios /sbin y /usr/bin (ls), recupere todas las líneas que contienen la cadena “mk”. ¿Has sido capaz?

Aparte de las tuberías, el intérprete de mandatos nos ofrece otros operadores de control que nos permiten realizar ciertas operaciones adicionales de mayor complejidad. Los operadores de control más usados son:

“mandato1; mandato2”: permite ejecutar mandatos de forma secuencial, primero mandato1 y después mandato2. Observa que esto no tiene nada que ver con las tuberías, que redireccionan la salida del primer mandato para que sirvan como entrada del segundo.

“mandato1 & mandato2”: ejecuta de forma simultánea mandato1 y mandato2.

“mandato1 && mandato2”: ejecuta mandato2 si se ha ejecutado con éxito mandato1.

“mandato1 || mandato2”: ejecuta mandato2 sólo si no ha ejecutado con éxito mandato1.

25. Con los operadores de control anteriores, trata de repetir la operación solicitada en el ejercicio 24. Ayúdate de un fichero auxiliar.

26. En un solo mandato crea un directorio de nombre “datos” (mkdir); dentro del mismo (cd) crea un fichero de nombre “personal” (touch); en dicho fichero escribe tus datos personales (echo) y la titulación a que perteneces (echo); ten cuidado de concatenar la información, no “pisarla”. Comprueba el contenido del fichero creado.

27. En un solo mandato recupera la lista de mandatos usados en la práctica (history); filtra todas las ocurrencias de grep que hay en la misma (grep); cuenta el número de líneas (wc).

28. Lista los contenidos del directorio /home/alumno con sus permisos (ls) y filtra (grep) todos los ficheros para los que algún usuario (o grupo) tiene permisos de lectura, escritura y ejecución (rwx).

29. Crea un fichero de nombre colores (cat > colores) en el que puedas introducir desde teclado diversos nombres de colores; en el mismo mandato ordénalo de forma alfabética (sort) y muestra el resultado por el intérprete de mandatos.

30. Captura la página web de la Universidad (wget); comprueba con qué nombre ha sido guardada en el directorio que te encuentras; en un mandato posterior filtra todas las líneas que contengan la cadena de caracteres “style”.

31. Haciendo uso de los mandatos para gestión de procesos y tareas de la práctica anterior, genera un listado de todos los procesos activos en tu sistema (ps) y filtra aquellos que pertenezcan al usuario “root”, redirigiendo los mismos a un fichero “procesos_root”. Comprueba el resultado.

32. Repite el mandato anterior pero para los procesos del usuario alumno, creando un fichero “procesos_alumno”.

33. Ejecuta las siguientes órdenes y observa el resultado. Apunta cuáles se han completado de forma satisfactoria, cuáles no, y qué mandatos han completado su tarea y cuáles no.

```
echo $PATH ; echo $SHELL
ecHo $PATH ; echo $SHELL
echo $PATH  && echo $SHELL
ecHo $PATH  && echo $SHELL
echo $PATH  && ecHo $SHELL
echo $PATH1  && echo $SHELL
echo $PATH  || echo $SHELL
ecHo $PATH  || echo $SHELL
echo $PATH1 || echo $SHELL1
ecHo $PATH1 || echo $SHELL1
```

34. La concatenación de mandatos y el uso de tuberías se pueden convertir en armas muy potentes de programación. En Linux, esta posibilidad de crear mandatos que permitan cumplir múltiples tareas, ha dado en llamarse "one liners". En la página http://www.linuxtotal.com.mx/index.php?cont=info_shell_007 puedes encontrar algunos de estos mandatos. En general, si buscas "Linux one liners", podrás encontrar mandatos complicados que permitiren completar tareas complejas en muy poco espacio.

35. Redirige todos los mandatos (history) de la práctica a un fichero de nombre mandatos_practica_19 y cuelga la misma en tu página de inicio en belenus junto al informe de la práctica.