Laboratorio 3

Comandos básicos para redirigir la E/S y gestionar la seguridad de los ficheros y directorios.

Bucle `while` en JavaScript

Entrega obligatoria al final de la sesión presencial de laboratorio

Objetivos:

- Conocer los comandos básicos de Unix para redirigir la entrada y salida y encadenar órdenes
- Conocer los comandos que permiten gestionar los permisos de los ficheros y directorios
- Entender la estructura de los bucles y escribir programas que utilicen bucles en JavaScript

Entorno:

Ordenadores del laboratorio. Pero las actividades que se describen pueden realizarse en cualquier equipo con un sistema de tipo Unix.

Actividades previas:

Antes de la sesión de laboratorio que corresponda a su grupo deberá usted leer este documento y realizar las actividades guiadas que en él se describen, bien en un ordenador del mismo laboratorio (en horario libre) o en cualquier otro ordenador. De no hacerlo, es posible que no le dé tiempo a realizar la entrega. De este modo, la sesión de laboratorio tiene la función de una clase de tutoría para resolver las dudas que se le hayan podido presentar.

Resultados: **Cuatro ficheros de texto** según se indica en los enunciados, que subirá al Moodle al final de la sesión de laboratorio. No se podrán subir dichos ficheros después de la hora de finalización del laboratorio de su grupo. En concreto para esta práctica:

a) Uno de los ficheros de texto contendrá todas las órdenes Unix que ha utilizado en las actividades guiadas descritas en los siguientes apartados, como se indica al final de este documento. El fichero de texto debería llevarlo al laboratorio ya generado si ha realizado todas las actividades previamente.

b) El segundo fichero será un programa JavaScript que responda a las especificaciones que se describen en el correspondiente apartado al final de este documento. El programa debe hacerlo previamente para llevar a la sesión de laboratorio el programa ya escrito.

c) El profesor proporcionará al comienzo del laboratorio un enunciado con las actividades a realizar durante la sesión presencial. El resultado serán dos ficheros adicionales que subirá al Moodle.

Inicio

a) Abra un terminal de texto desde la interfaz gráfica y cree un directorio nuevo denominado `lab3` que cuelgue de su directorio personal y muévase a él.

b) Borre la historia de las órdenes que ha tecleado previamente y escriba su nombre y apellidos,

para que quede registrado en la historia

```
$ history -c
$ Nombre Apell1 Apell2
```

Si en algún momento se equivoca al teclear o en las órdenes que debe ejecutar, no se preocupe y repita las órdenes. No es necesario que borre todo el historial y repita de nuevo el ejercicio, se consideraran válidas las últimas órdenes ejecutadas de cada apartado.

En lo sucesivo, el símbolo **$** significa que debe usted escribir la orden indicada.

1. Redirección

Muchos de los procesos iniciados al ejecutar una orden UNIX leen los datos que necesitan de la entrada estándar, escriben en la salida estándar y envían los mensajes de error a la salida de errores. Por defecto, la entrada estándar es el teclado y la pantalla es tanto la salida estándar como la salida de errores.

En el primer laboratorio usó la orden `cat prueba` para ver el contenido del fichero `prueba` en la pantalla. `cat` por tanto leía los datos del fichero y los sacaba por la salida estándar, es decir la pantalla. Si únicamente pulsa `cat` sin indicar un nombre de fichero, la orden `cat` lee los datos de la entrada estándar que es el teclado y los sigue sacando por la pantalla. Pruebe a teclear:

```
$ cat
```

A continuación, teclee palabras en el teclado, por ejemplo nombres de frutas, todas ellas separadas por `Intro`, es decir cada una en una línea. Finalmente teclee `Ctrl D` (`^D`) (fin de fichero) para indicar que se termina la entrada de texto por teclado. Observe que según introduzca una palabra seguida de `Intro`, dicha palabra se sacará inmediatamente por la pantalla, viéndolas por tanto duplicadas.

1.1. Redirección de la salida

En Unix se puede redirigir tanto la entrada como la salida de las órdenes. Se utiliza el símbolo “>” para redirigir la salida de una orden. En el siguiente ejemplo se redirige la salida estándar a un fichero:

```
$ ls > lista-ficheros.txt
```

En este ejemplo se ejecuta el comando `ls` y el resultado se escribe en el fichero `lista-ficheros.txt`. Ya que la salida de `ls` fue redirigida a ese fichero, no apareció ningún resultado por pantalla. Compruebe cuál es el contenido de dicho fichero.

En el siguiente ejemplo, se crea un fichero (de nombre `frutas.txt`) con una lista de nombres de fruta que va introduciendo por el teclado. Teclee:

```
$ cat > frutas.txt
Pera
Manzana
Plátano
Melocotón
^D
```

Dicha orden lee los datos de la entrada estándar, el teclado, y como la salida está redirigida (“>”) al fichero `frutas.txt`, en lugar de sacarlos por la salida estándar, es decir la pantalla, los redirige al fichero `frutas.txt`. Si `frutas.txt` no existe, previamente crea dicho fichero. Si

`frutas.txt` existe, antes de redirigir la salida a dicho fichero, borra su contenido.

Para comprobar el contenido del fichero `frutas.txt` teclee:

```
$ cat frutas.txt
```

La orden **`sort`** ordena alfabéticamente o numéricamente una lista. Compruebe, por ejemplo, el resultado de la siguiente orden:

```
$ sort frutas.txt
```

La opción `-o` seguida del nombre del fichero hace que el resultado de `sort` se guarde en otro fichero:

```
$ sort frutas.txt -o frutas2.txt
```

Para ver otro ejemplo de redirección de la salida, teclee:

```
$ sort > verduras.txt
Puerro
Coliflor
Lechuga
^D
```

Dicha orden lee los datos de la entrada estándar, el teclado, una vez ordenados los nombres de las verduras, como la salida está redirigida (“>”) al fichero `verduras.txt`, los guarda ya ordenados en el fichero.

Si ahora queremos añadir más frutas al fichero `frutas.txt` ejecutaremos la orden `cat > frutas.txt`, las nuevas frutas que escribiera en el teclado se escribirán en el fichero `frutas.txt`, pero perderíamos las que habíamos introducido con la primera orden. Para poder añadir información a un fichero sin perder los datos que ya tiene dicho fichero, utilizamos “>>”.

```
$ cat  >>  frutas.txt
Cereza
Melón
Sandía
^D
```

Con esta orden, hemos añadido los nuevos nombres de frutas al fichero `frutas`. Es decir, se han seguido leyendo los datos de la entrada estándar, el teclado, pero la salida estándar se ha redirigido al fichero sin borrar su contenido previo. Compruébelo tecleando:

```
$ cat frutas.txt
```

Ahora ya tenemos dos ficheros, `frutas.txt` con 7 frutas y `verduras.txt` con 3 verduras ordenadas alfabéticamente.

Podemos utilizar la redirección para unir (concatenar) los dos ficheros y crear uno nuevo con el contenido de ambos ficheros:

```
$ cat  frutas.txt  verduras.txt  >  comida.txt
```

Lo que estamos haciendo es leer el contenido de los ficheros `frutas.txt` y `verduras.txt`

y redirigir la salida al fichero `comida.txt`. Compruébelo:

```
$ cat  comida.txt
```

Si queremos que el fichero `comida` tenga su contenido ordenado alfabéticamente podemos hacer:

```
$ sort  frutas.txt  verduras.txt  > comida.txt
```

Lo que estamos haciendo es introducir a la orden `sort` el contenido de dos ficheros, `frutas.txt` y `verduras.txt`. Dicha orden ordena los datos y la salida es redireccionada con ">" al fichero `comida.txt`. Compruébelo:

```
$ cat  comida.txt
```

1.2 Redirección de la entrada

Para contar el número de ficheros y directorios que hay en su directorio de trabajo actual, puede teclear:

```
$ ls   >  nombres.txt
$ wc  -w   <  nombres.txt
```

Con la primera orden, usando redirección de la salida, se crea el fichero `nombres.txt` con los nombres de los ficheros y directorios que cuelgan de su directorio de trabajo actual.

En la segunda orden (con el símbolo "<" se hace redirección de la entrada estándar) se cuenta el número de palabras del fichero `nombres.txt`.

El resultado de ejecutar ambas órdenes permite obtener por tanto el número de ficheros y directorios que hay en su directorio de trabajo actual. Tenga en cuenta que la primera orden creará primero el fichero `nombres.txt` y luego ejecuta la orden `ls`, con lo que cuando ejecute la orden `wc`, contará también dicho fichero `nombres.txt`.

2. Tuberías

Pero ejecutar estás dos órdenes seguidas hace que el proceso sea más lento y que cuando hayamos terminado nos tengamos que acordar de borrar el fichero de trabajo, `nombres.txt`, que hemos creado y ya no lo necesitamos.

Para evitar estos problemas se pueden utilizar las tuberías (`|`). Las tuberías conectan directamente la salida de una orden a la entrada de otra orden. Por ejemplo podemos conseguir lo mismo tecleando:

```
$ ls  |  wc -w
```

La salida de la orden `ls` es la entrada de la orden `wc`, consiguiendo como resultado el número de ficheros y directorios que cuelgan del directorio de trabajo actual. En este caso no se cuenta el fichero `nombres.txt` ya que no se ha creado.

3. Permisos de acceso en el sistema de ficheros

Cuando ha tecleado:

```
$ ls -l
```

habrá observado que para cada fichero o directorio que se lista, la primera información que se obtiene es algo como:

```
-rwxrw-r--
```

Es una cadena de 10 símbolos que pueden tomar los siguientes valores: `d`, `r`, `w`, `x`, `-`
El primer símbolo indica si es un fichero o directorio:

- `d`: indica que es un directorio
- `-`: indica que es un fichero

Los nueve siguientes símbolos indican los permisos de acceso y se agrupan en tres grupos:

- El primer grupo de 3 símbolos indican los permisos para el usuario propietario del fichero o directorio. En el ejemplo `rwx`
- El segundo grupo de 3 símbolos indican los permisos para el grupo al que pertenece el propietario del fichero o directorio. En el ejemplo `rw-`
- El último grupo de 3 símbolos indican los permisos para el resto de usuarios. En el ejemplo `r--`

Los símbolos `r`, `w`, `x` tienen diferentes valores en función de que sea un fichero o directorio.

Si es un fichero los permisos de acceso indican:

- `r`: permiso de lectura, permiso para leer o copiar el fichero. Si en su lugar hay un `-` indica que no tiene permiso de lectura
- `w`: permiso de escritura, permiso para cambiar el fichero. Si en su lugar hay un `-` indica que no tiene permiso de escritura
- `x`: permiso de ejecución, permiso para ejecutar el fichero. Si en su lugar hay un `-` indica que no tiene permiso de ejecución

Si es un directorio los permisos de acceso indican:

- `r`: permiso a los usuarios para visualizar los ficheros en el directorio. Si en su lugar hay un `-` indica que no tiene permiso para visualizar los ficheros
- `w`: permiso para borrar, crear o mover ficheros en el directorio. Si en su lugar hay un `-` indica que no tiene permiso para borrar, crear o mover los ficheros
- `x`: permiso de acceso a los ficheros del directorio. Si en su lugar hay un `-` indica que no tiene permiso de acceso

En el ejemplo anterior `-rwxrw-r--` significa:

- El primer `-` indica que es un fichero
- `rwx` indica que el propietario del fichero puede leerlo, escribirlo y ejecutarlo
- `rw-` indica que el grupo del propietario del fichero, puede leerlo y escribir sobre él, pero no ejecutarlo
- `r--` indica que el resto de usuarios sólo pueden leerlo

3.1. Cambiar los permisos de acceso en el sistema de ficheros

Con la orden **`chmod`** (**change mode**) se pueden cambiar los permisos de un fichero. Los permisos de un fichero únicamente pueden ser cambiados por el propietario de dicho fichero. La orden `chmod` tiene dos parámetros:

- Tres dígitos en octal que indican los nuevos permisos que se le dan al fichero. El primer dígito corresponde a los permisos del usuario, el segundo dígito a los permisos del grupo y el último dígito a los permisos del resto de usuarios
- El nombre del fichero cuyos permisos se cambian

Por ejemplo: `chmod 764 <fichero>`

Cada dígito en octal, pasado a binario se representa con tres dígitos binarios (0 o 1):

- El primer dígito binario indica si tiene (1) o no (0) permisos de lectura
- El segundo dígito binario indica si tiene (1) o no (0) permisos de escritura
- El tercer dígito binario indica si tiene (1) o no (0) permisos de ejecución

Por lo tanto, los 3 dígitos en octal que hemos puesto a la derecha de `chmod`, `764`, pasados a binario serían, 111 110 100. El `7` (111) indica que el propietario del fichero puede leerlo, escribirlo y ejecutarlo. El `6` (110) indica que el grupo puede leerlo, escribirlo pero no ejecutarlo. El `4` (100) indica que el resto de usuarios sólo pueden leerlo.

Ejecute:

```
$ chmod  440  frutas.txt
$ ls -l
$ nano frutas.txt
```

e intente modificar el nombre de una fruta

Con la primera orden hemos cambiado los permisos al fichero `frutas` de forma que el propietario y el grupo sólo pueden leerlo y el resto de usuarios no pueden hacer nada con el fichero

Con la segunda orden comprobará el cambio de dichos permisos y con la tercera orden comprobará cómo dicho fichero no puede ser modificado

Si mira las características de `chmod` en el manual con la orden `man`, observará que hay una alternativa a la hora de indicar los permisos que se cambian a un fichero. Esta alternativa consiste en utilizar símbolos como + (añadir), − (quitar), `r`, `w`, `x`,.. en lugar de los tres dígitos en octal.

4. Entrega de resultados, fichero de texto con la historia

Grabe la historia de todas las órdenes que ha ejecutado hasta ahora:

```
$ history  -w  Grupo-Apell1-Apell2-L3-1.txt
```

Compruebe con `less` el contenido del fichero generado y súbalo a Moodle durante la sesión presencial de laboratorio. Asegúrese de que el fichero que sube es el correcto (que no esté vacío o tenga otros contenidos).

Aquellos ficheros que se suban a Moodle y no respeten estrictamente el formato del nombre, se ignorarán y por tanto no se evaluará la entrega correspondiente.

5. Ejercicio a realizar sobre JavaScript

Se trata en este ejercicio de escribir un programa JavaScript que **genere y muestre por consola la tabla completa de unidades binarias** que van desde Kibi hasta Yobi. El programa deberá utilizar un bucle `while`. Hará uso de la función `unidad(e)` definida en la práctica 2 anterior y la función `dos_a(n)` de la transparencia 44 de JavaScript del Tema 1. El programa generará para cada fila el siguiente contenido:

```
"1 Kibibyte = 2^10 Bytes = 1024 Bytes"
"1 Mebibyte = 2^20 Bytes = 1048576 Bytes"
……
```

El bucle generará una fila de la tabla en cada iteración. Tendrá además un índice que se incrementará en cada iteración indicando qué unidad sacar. El bucle deberá finalizar cuando imprima la última fila de la tabla:

```
"1 Yobibyte = 2^80 Bytes = 1.2089258196146292e+24 Bytes"
```

El programa también deberá mostrar en la consola, como cabecera de la tabla, **su nombre, apellidos y grupo** al que pertenece.

5.1. Entrega del programa fuente en JavaScript

Antes de la sesión de laboratorio presencial, debe haber escrito el programa JavaScript con las especificaciones descritas en el apartado anterior para llevar a la sesión de laboratorio el fichero con el programa fuente ya escrito y probado.

Suba a Moodle el fichero con el programa JavaScript al que habrá puesto el siguiente nombre:

`Grupo-Apell1-Apell2-L3.js`

6. Resumen

Orden	**Uso**
orden > fichero	Redirecciona la salida estándar a un fichero
orden >> fichero	Añade la salida estándar a un fichero
orden < fichero	Redirecciona a la entrada estándar el contenido de un fichero
orden1 | orden2	Conecta la salida de la orden1 a la entrada de la orden2
cat fich1 fich2 > fich0	Concatena fich1 y fich2 al fich0
sort sort –o	Ordena alfabética o numéricamente los datos Guarda el resultado en otro fichero
chmod [3díg] fichero	Cambia los permisos de acceso del fichero