007. Concatenar archivos

7.1 El comando echo y clear

En MSDos la gente usaba el comando cls. Aún sigue pudiéndose usar en sistemas operativos privativos como Windows desde el interprete de comandos que cmd que este incorpora.

Cls en GNU/Linux es clear y sirve para hacer limpieza de lo que estamos viendo en el terminal.

```
$ clear
```

Sin darnos cuenta vamos a usar este comando mucho. Cuando queremos una salida limpia sin que interfiera a la vista.

El comando echo nos servirá para mostrar en pantalla algo. Ese algo puede ser una variable de sistema como por ejemplo $PATH

```
$ echo $PATH
```

O podemos usarlo para mostrar un texto como por ejemplo “hola”

```
$ echo hola
```

Veremos que se mostrará hola en pantalla. Si queremos mostrar hola mundo podemos hacerlo poniendo el texto entre comillas.

```
$ echo “hola mundo”
```

Dependiendo de la versión de echo que estemos usando esto puede variar. No obstante meter entre comillas los textos con palabras separadas es buena cosa.

Podemos mostrar un texto y luego el contenido de una variable:

```
$ echo “la version de bash es $BASH_VERSION”
```

La salida será algo como esto: **la version de bash es 4.2.37(1)-release**

Esto da juego a crear nuestras propias variables. Podemos poner en una variable que nos inventemos un contenido, por ejemplo en una variable llamada nombre meteremos nuestro nombre.

```
$ nombre="Fanta"
```

en otras variables vamos a meter nuestro apellido.

```
$ apellido1="De"

$ apellido2="Naranja"
```

Ahora con echo vamos a mostrar nuestro nombre completo

```
$ echo “Mi nombre es $nombre y mis apellidos son $apellido1 $apellido2”
```

La salida será algo como esto: **Mi nombre es Fanta y mis apellidos son De Naranja**

Podemos borrar/limpiar con **clear** la pantalla para tenerla bien limpia y lista para meter más comandos.

Aparte de texto echo puede servirnos para emitir un pitido. Lo que suele llamarse un beep por el altavoz interno de nuestro ordenador.

Estamos en un servidor de modo que no vamos a escuchar el pitido no obstante es bueno saberlo.

Esto se realiza usando “backslash escapes” que son unas barritas como \n para nueva linea \a para emitir un pitido \t para un tabulado horizontal, etc...

Para activarlo se usa el argumento -e y con eso indicamos que queremos que **echo** interprete.

```
$ echo “$nombre\n$apellido1\n$apellido2”
```

Mostrará el nombre y apellidos cada uno en una linea diferente:

Fanta
De
Naranja

Con saber esto nos basta y sobra.

Posiblemente ahora no encontremos muchos usos a **echo** pero para eso estamos. Ahora después veremos algunos.

7.2 El famoso comando cat

Hasta este momento hemos estado utilizando un editor de textos para introducir texto dentro de archivos. Es una forma muy cómoda de hacerlo pero hemos de saber que no es la única.

Con un **echo** vamos a poder introducir texto dentro de un archivo y por tanto vamos a recordar algunos comandos que hemos usado.

tocuh – Nos estaba sirviendo y sigue sirviendo para crear archivos sin nada en su interior.
echo – Nos sirve para mostrar texto en pantalla. La pantalla es la salida standard.

Cuando ejecutamos un comando y pulsamos intro este va a escupir la salida por lo que se llama la salida standard. Esta salida es la pantalla y por tanto si por ejemplo realizamos un ls -ls para ver los archivos estos saldrán impresos en pantalla.

Podemos variar por donde queremos que salga escupida la salida.

Al ejecutar **echo “hola mundo”** nos salia el texto por la pantalla pero podemos hacer que en vez de salir por pantalla este se guarde en un fichero.

Vamos a crear un archivo llamado **texto.txt** con **touch**.

```
$ touch texto.txt
```

Ahora vamos a usar **cat** para ver que contiene en su interior.

```
$ cat texto.txt
```

La salida por pantalla no va a contener nada pero si usamos echo para volcar el contenido en ese archivo si que vamos a ver algo cuando luego usemos cat.

```
$ echo “hola mundo” > texto.txt
```

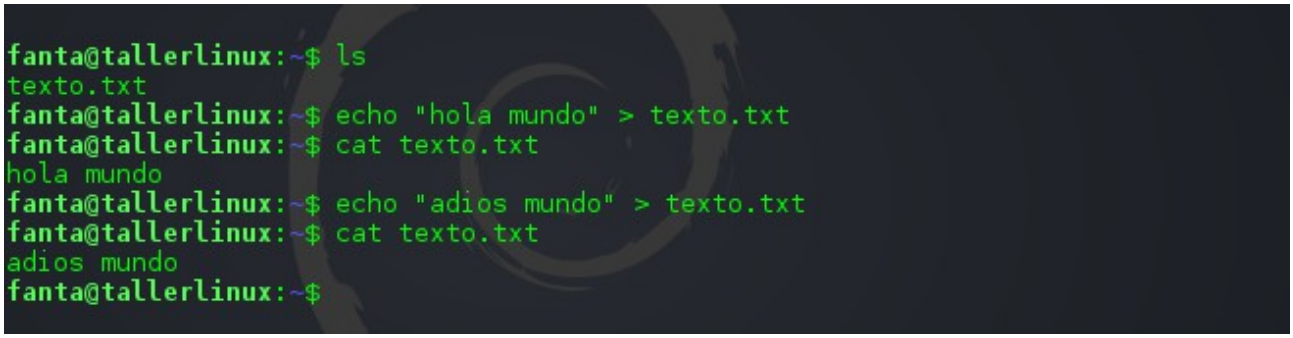
Lo que acabamos de hacer es redireccionar la salida del comando echo *“hola mundo”* para que se escriba en el archivo **texto.txt** sobrescribiendo cualquier cosa que tenga en su interior. En nuestro caso no es un problema ya que estaba sin nada dentro.

```
$ cat texto.txt
```

Con el comando cat vamos a ver que contiene y veremos que efectivamente el contenido se ha volcado con éxito.

Si ahora volcamos de nuevo con un echo otra frase como por ejemplo *“adios mundo”* veremos que se sobrescribe el contenido del archivo por el nuevo.

```
$ echo “adios mundo” > texto.txt
```



En esa captura puede verse con claridad.

Lo cierto es que **no era necesario usar touch para crear el archivo primero**.

Si ahora queremos guardar nuestro nombre en otro fichero llamado nombre.txt podemos hacerlo sin necesidad de crearlo primero.

```
$ echo “fanta” > nombre.txt
```

Se creará si no existe. En caso de existir se sobreescribe.

Si deseamos no sobrescribir el archivo y simplemente añadirle más lineas usaremos >> en vez de >

```
$ echo “benito” >> nombre.txt
```

Si hacemos un **cat** al archivo nombre.txt veremos que ahora existen 2 lineas y 2 nombres.

Pero cat no es solamente para ver un archivo.

El comando cat nos permite concatenar el contenido de diferentes archivos o visualizar el de ambos.

```
$ cat nombre.txt texto.txt
```

De esta forma podemos ver el contenido de ambos ficheros del tirón.

Fanta
benito
adios mundo

Eso será la salida que se mostrará.

Podemos volcarla como ya sabemos a un nuevo fichero llamado *juntos.txt*

```
$ cat nombre.txt texto.txt > juntos.txt
```

Si ahora hacemos cat al fichero *juntos.txt* vamos a ver el contenido junto.
Hemos unido por tanto 2 ficheros. Podemos hacer esto usando los comodines así:

```
$ cat *.txt > juntos.txt
```

Se sobrescribirá el fichero *juntos.txt* con el contenido de todos los archivos con extensión .txt que tengamos en el directorio.

Y **cat** tiene muchas más opciones de las que por ejemplo nos puede interesar ver la salida de un archivo con las lineas numeradas.

```
$ cat -n juntos.txt
```

Veremos algo así como esto:

1 fanta
2 benito
3 adios mundo

Lo cual nos sirve para entre otras cosas saber en que linea esta algo sin necesidad de usar un editor de textos como nano.