PRÁCTICA 5

Ejercicio 5.1: **Escriba un guión que acepte dos argumentos. El primero será el nombre de un directorio y el segundo será un valor entero. El funcionamiento del guión será el siguiente: deberán anotarse en un archivo denominado archivosSizN.txt aquellos archivos del directorio dado como argumento y que cumplan la condición de tener un tamaño menor al valor aportado en el segundo argumento. Se deben tener en cuenta las comprobaciones sobre los argumentos, es decir, debe haber dos argumentos, el primero deberá ser un directorio existente y el segundo un valor entero.**

```
#!/bin/bash

directorio=`test -d $1 && echo true || echo false`
entero=`echo $2 | grep -q "[0-9]\+" && echo true || echo false`

if (( $# == 2 )); then

	if [ $directorio == "true" ] && [ $entero == "true" ]; then

		find . -size -$2 >> archivosSizN.txt
		echo "Aquellos archivos pertenecientes al directorio de trabajo con un tamaño menor a $2 han sido anotados correctamente en el fichero archivosSizN.txt"

	elif [ $directorio == "false" ] && [ $entero == "false" ]; then

		echo "Ambos argumentos han sido introducidos erróneamente. El primero de ellos deberá ser un directorio existente y el segundo un número entero."

	elif [ $directorio == "false" ]; then

		echo "El primer argumento introducido debe ser un directorio existente."

	elif [ $entero == "false" ]; then

		echo "El segundo argumento ha de ser un entero."

	fi

else

	echo "Ha introducido un número incorrecto de argumentos."

fi
```

Ejercicio 5.2: **Escriba un guión que acepte el nombre de un directorio como argumento y muestre como resultado el nombre de todos y cada uno de los archivos del mismo y una leyenda que diga "Directorio", "Enlace" o "Archivo regular", según corresponda. Incluya la comprobación necesaria sobre el argumento, es decir, determine si el nombre aportado se trata de un directorio existente.**

```
#!/bin/bash

es_directorio=`test -d $1 && echo true || echo false`

if (( $# == 1 )); then

	if [ $es_directorio == "true" ]; then

		for archivo in `find $1`

		do

			tipo=`( test -h $archivo && echo "Enlace simbólico" ) || ( test -f $archivo && echo "Archivo regular" ) || ( test -d $archivo && echo "Directorio" )`

		echo "$archivo : $tipo"

		done

	else

		echo "El argumento ha de ser un directorio existente"

	fi

else

	echo "El número de argumentos es incorrecto."

fi
```

Ejercicio 5.3: **Escriba un guión en el que, a partir de la pulsación de una tecla, detecte la zona del teclado donde se encuentre. Las zonas vendrán determinadas por las filas. La fila de los números 1, 2, 3, 4, ... será la fila 1, las teclas donde se encuentra la Q, W, E, R, T, Y,... serán de la fila 2, las teclas de la A, S, D, F, ... serán de la fila 3 y las teclas de la Z, X, C, V, ... serán de la fila 4. La captura de la tecla se realizará mediante la orden read.**

```
#!/bin/bash
```

```
echo "Pulse una tecla."
read tecla

case $tecla in

        [º0-9\'¡]) echo "La tecla pulsada, $tecla , se encuentra en la primera fila";;
        [qwertyuiop\`+]) echo "La tecla pulsada, $tecla , se encuentra en la segunda fila";;
        [asdfghjklñ\'ç]) echo "La tecla pulsada, $tecla , se encuentra en la tercera fila";;
        [\<zxvbnm,.-]) echo "La tecla pulsada, $tecla , se encuentra en la cuarta fila";;
        *) echo "La tecla pulsada, $tecla , se encuentra en una fila indeterminada";;

esac
```

Ejercicio 5.4: **Escriba un guión que acepte como argumento un parámetro en el que el usuario indica el mes que quiere ver, ya sea en formato numérico o usando las tres primeras letras del nombre del mes, y muestre el nombre completo del mes introducido. Si el número no está comprendido entre 1 y 12 o las letras no son significativas del nombre de un mes, el guión deberá mostrar el correspondiente mensaje de error.**

```
#!/bin/bash

echo "Introduzca las tres primeras letras de un mes : "
read mes

case $mes in

        ene) echo "Enero";;
        feb) echo "Febrero";;
        mar) echo "Marzo";;
        abr) echo "Abril";;
        may) echo "Mayo";;
        jun) echo "Junio";;
        jul) echo "Julio";;
        ago) echo "Agosto";;
        sep) echo "Septiembre";;
        oct) echo "Octubre";;
        nov) echo "Noviembre";;
        dic) echo "Diciembre";;
        *) echo "Introduzca de forma correcta el dato pedido, es decir, las tres primeras letras significativas de un mes.";;

esac
```

Ejercicio 5.5: **Escriba un guión que solicite un número hasta que su valor esté comprendido entre 1 y 10. Deberá usar la orden while y, para la captura del número, la orden read.**

```
#!/bin/bash

echo "Introduzca un número comprendido entre 1 y 10 :"
read numero

while (( $numero < 1 || $numero > 10 ))

        do

                echo "El dato introducido no se encuentra en el rango especificado."
                echo "Vuelva a introducir el número pedido."
                read numero

        done

echo "El número introducido es $numero y se encuentra dentro del rango especificado."
```

Ejercicio 5.6: **Copie este ejercicio y pruébelo en su sistema para ver su funcionamiento. ¿Qué podemos modificar para que el giro se vea más rápido o más lento? ¿Qué hace la opción -e de las órdenes echo del guión?**

Para que el giro se vea más rápido tendremos que modificar el valor de la variable INTERVAL por otro menor. Cuánto más pequeño sea el valor que le asignemos, más rápido será el giro.

La opción -e de las órdenes echo es para que se interpreten las barras invertidas como caracteres.

Ejercicio 5.7: **Escriba un guión que admita como argumento el nombre de un tipo de shell (por ejemplo, csh, sh, bash, tcsh, etc.) y nos dé un listado ordenado alfabéticamente de los usuarios que tienen dicho tipo de shell por defecto cuando abren un terminal. Dicha información del tipo de shell asignado a un usuario se puede encontrar en el archivo /etc/passwd, cuyo contenido está delimitado por ':'. Cada información situada entre esos delimitadores representa un campo y precisamente el campo que nos interesa se encuentra situado en primer lugar. En definitiva, para quedarnos con lo que aparece justo antes del primer delimitador será útil la orden siguiente:**

```
cut -d':' -f1 /etc/passwd
```

Donde la opción –d indica cuál es el delimitador utilizado y la opción –f1 representa a la secuencia de caracteres del primer campo.

Realice, utilizando el mecanismo de cauces, el ejercicio pero usando la orden cat para mostrar el contenido de un archivo y encauzado con la orden cut para filtrar la información que aparece justo antes del delimitador ':'. Realice también la comprobación de la validez del tipo de Shell que se introduce como argumento. Use para ello la información que encontrará en el archivo /etc/shells donde encontrará los tipos de Shell que se pueden utilizar en el sistema.

```
#!/bin/bash

es_shell="false"

for linea in `cat /etc/shells | cut -d'/' -f3`

        do

                if [ $1 == `echo $linea` ]; then

                        es_shell="true"

                fi

        done

if [ $es_shell == "true" ]; then

        for linea in `cat /etc/passwd`

                do

                        usuario=`echo $linea | cut -d':' -f1`
                        shell=`echo $linea | cut -d':' -f7`
                        valor="/bin/$1"

                        if [[ `echo $shell` == $valor ]]; then

                                echo $usuario >> tmp.txt

                        fi

                done

        cat tmp.txt 2> /dev/null | sort
        rm tmp.txt 2> /dev/null
```

```
else
        echo "El argumento introducido no es un shell."
fi
```

Ejercicio 5.8: **Dos órdenes frecuentes de Unix son tar y gzip. La orden tar permite almacenar/extraer varios archivos de otro archivo. Por ejemplo, podemos almacenar el contenido de un directorio en un archivo con**

tar -cvf archivo.tar directorio

(la opción -x extrae los archivos de un archivo .tar).

La orden gzip permite comprimir el contenido de un archivo para que ocupe menos espacio. Por ejemplo, gzip archivo comprime archivo y lo sustituye por otro con el mismo nombre y con la extensión .gz. La orden para descomprimir un archivo .gz o .zip es gunzip.

Dadas estas órdenes construya un guión, denominado cpback, que dado un directorio o lista de archivos como argumento(s) los archive y comprima en un archivo con nombre copiaYYMMDD, donde YY corresponde al año, la MM al mes y la DD al día, dentro de un directorio denominado CopiasSeguridad. El guión debe realizar las comprobaciones oportunas: los argumentos existen, el directorio de destino existe y si no, lo crea.

```
#!/bin/bash
destino="CopiasSeguridad"
if ! test -d $destino; then
        mkdir $destino
fi
if (( $# >= 1 )); then
        errores=0
        for i in $@
                do
```

```
                if test -d $i; then

                        errores=$[$errores + 1]

                fi

        done

    if (( $errores == 0 )); then

                tar cfv ./$destino/copia`date +"%Y%m%d"`.tar $@
                gzip ./$destino/copia`date +"%Y%m%d"`.tar

    else

        echo "Alguno de los parámetros introducidos no existe."

    fi

else

    echo "El número de parámetros introducidos es incorrecto."

fi
```

Ejercicio 5.9: **Hacer un script en Bash denominado newdirfiles con los siguientes tres argumentos:**

- **<dirname> Nombre del directorio que, en caso de no existir, se debe crear para alojar en él los archivos que se han de crear.**
- **<num_files> Número de archivos que se han de crear.**
- **<basefilename> Será una cadena de caracteres que represente el nombre base de los archivos.**

Ese guión debe realizar lo siguiente:

- **Comprobar que el número de argumentos es el correcto y que el segundo argumento tenga un valor comprendido entre 1 y 99.**
- **Crear, en caso de no existir, el directorio dado en el primer argumento a partir del directorio donde se esté situado y que posea permisos de lectura y escritura para el usuario $USER.**
- **Dentro del directorio dado en el primer argumento, crear archivos cuyos contenidos estarán vacíos y cuyos nombres lo formarán el nombre dado como tercer argumento y un número que irá desde 01 hasta el número dado en el segundo argumento.**

```
#!/bin/bash

dir_name=$1
num_files=$2
base_filename=$3

if (( $# == 3 )); then

	if ! test -d $dir_name; then

		mkdir $dir_name

	fi

	chmod u+rw $dir_name

	while (( $num_files < 1 || $num_files > 99 ))

		do

			echo "El segundo argumento no es válido, pues ha de ser un número comprendido entre 1 y 99."
			echo "Vuelva a introducir dicho dato."
			read num_files

		done

	for i in `seq 1 1 $num_files`

		do

			if (( $i < 10 )); then

				touch ./$dir_name/$base_filename"0"$i

			else

				touch ./$dir_name/$base_filename$i

			fi

		done

	echo "El archivo se ha creado correctamente."
```

```
else
        echo "Se ha introducido un número incorrecto de parámetros."
fi
```

Realizado por Alejandro Becerra Burgos - 1º DGIIM Grupo 1 de FS