


Tema 1

Nombre y apellido:

DNI:

1. Al ejecutar el siguiente comando vemos esta salida:

```
 Bash
$ ls -l
-rw-r--r-- 1 user user 99 Jun 5 19:57 bonjour
-rw-r--r-- 1 user user 99 Jun 5 19:46 ciao
-rw-r--r-- 2 user user 99 Jun 5 19:43 hello
-rw-r--r-- 2 user user 99 Jun 5 19:43 hola
-rw-r--r-- 1 user user 82 Jun 5 19:55 ola
```

- a) Con la información disponible, ¿podemos afirmar si 2 archivos son iguales? ¿Por qué?
- b) Escriba que comando utilizaría para verificar si el contenido de dos archivos es igual.
2. Dado el siguiente script que categoriza archivos, interprete cómo se usaría y complete la función `get_extension()`:

```
#!/bin/bash

mkdir -p txt
mkdir -p doc
mkdir -p ppt
mkdir -p unknown

DIR=$1

get_extension () {
    # Completame!
}

for FILE in $DIR/*
do
    EXT=$(get_extension $FILE)
    mv $FILE $EXT
done
```

3. Se ejecuta el siguiente comando sobre el archivo **file.txt** (asumiendo que el archivo existe):

 Bash

```
$ chmod 540 file.txt
```

- a) Escriba como quedarían los permisos.
- b) Si los permisos iniciales del archivo eran los que se muestran a continuación, escriba como haría el mismo cambio en los permisos pero en vez de usar la forma numérica (chmod 540), use caracteres especiales: [ugoa][+ -=][rwx]:

 Bash

```
$ ls -lh file.txt  
-rw-r--r-- 1 user user 82 Jun 5 20:06 file.txt
```

4. A partir del archivo 'compras.csv' que tiene el siguiente formato:

 Bash

```
$ cat compras.csv  
DNI,Apellido,Nombre,Producto,Precio  
12345678,Alvarez,Augusto,Manzana,500  
45678912,D'Alessandro,Ariel,Zapallo,300  
12345678,Alvarez,Augusto,Zanahoria,800  
78912345,Di Marco,Nicolás,Melón,750  
32165498,Gonzalez,Hernán,Sandía,500  
78912345,Di Marco,Nicolás,Kiwi,250  
...
```

NOTA:

- \* La columna DNI son números de 8 cifras.
- \* La columna Precio son números de 3 cifras.

- a) Escriba un comando que devuelva la cantidad de productos que compró **Augusto (DNI: 12345678)**.
- b) Escriba un comando que devuelva el menor precio entre los productos que compró **Nicolás (DNI: 78912345)**.

5. El legajo de un estudiante tiene el formato L-NNNN/N, donde:

\* L=Letra: primer letra del apellido del estudiante

\* N=numero: un dígito.

a) Escriba una expresión regular que valide legajos de estudiantes.

b) Tengo un script llamado **verificador\_legajo.sh** que toma por parámetro el legajo que necesito verificar y me devuelve si es correcto o no. Escriba el contenido del mismo.

La llamada al script se realizará de la siguiente manera:

 Bash

```
$ ./verificador_legajo.sh "L-NNNN/N" # (donde L son letras y N son números)
```

6. Lea el siguiente script, encuentre los errores y explique lo indicado.

En total hay 3 errores para corregir y 2 verificaciones que explicar. Vea los comentarios en las líneas 8, 13, 19, 27.

```
1 !/bin/bash

2 # Declaro Variables
3 SOURCE_DIR="$1"
4 BACKUP_DIR="$2"
5 LOG_FILE="/var/log/backup.log"
6 DATE=$(date +%Y-%m-%d_%H-%M-%S)
7 BACKUP_NAME="backup_$(date +%Y-%m-%d_%H-%M-%S).tar.gz"

8 # Cada vez que llamo a log(), le paso por parametro una variable
  MENSAJE (local) y luego graba al final del archivo LOGFILE. Señalar
  el error y corregirlo.
9 log() {
10     local MENSAJE="$1"
11     echo "$(date +%Y-%m-%d\ %H:%M:%S) : $MENSAJE" > "$LOG_FILE"
12 }

13 # Hago un chequeo si el directorio existe o no, llama a la
  funcion log e imprime por stdout un mensaje. Señalar el error y
  corregirlo.
14 if [ -d "$SOURCE_DIR" ]; then
```

```
15     log "ERROR: El directorio $SOURCE_DIR no existe."
16     echo "ERROR: El directorio $SOURCE_DIR no existe."
17     exit 1
18 fi

19 # ¿Qué hace esta verificación?
20 if [ ! -d "$BACKUP_DIR" ]; then
21     mkdir -p "$BACKUP_DIR"
22     log "Se ha creado el directorio $BACKUP_DIR."
23 fi

24 # Acá no es necesario explicar ni corregir nada. Graba un mensaje
  en el log y realiza el backup.
25 log "Comienza el backup de $SOURCE_DIR."
26 tar -czf "$BACKUP_DIR/$BACKUP_NAME" -C "$SOURCE_DIR" .

27 # Encontrar y corregir el error y explicar que se esta intentando
  chequear con el if.
28 if [ $? -eq 0 ]; then
29     log "El Backup de $SOURCE_DIR se completó exitosamente. La
ubicación del backup es: $BACKUP_DIR/$BACKUP_NAME."
30     echo "El Backup de $SOURCE_DIR se completó exitosamente. La
ubicación del backup es: $BACKUP_DIR/$BACKUP_NAME."
31 else
32     log "ERROR: El backup de $SOURCE_DIR fallo"
33     echo "ERROR: El backup de $SOURCE_DIR fallo"
34     exit 1
35 fi
```


7. Determine si las siguientes afirmaciones son verdaderas o falsas. Justifique con sus palabras (\*\*muy brevemente\*\*) la opción elegida.

a) El comando

 Bash
\$ kill -s SIGTSTP 1954

manda una señal para detener la ejecución del proceso 1954

b) El siguiente comando


```
 Bash
$ which python && echo "Python esta en la versión $(which python)"
```

devuelve la versión de Python que tengo instalada.


c) El siguiente comando ejecuta el navegador chromium y me bloquea la terminal:

```
 Bash
$ chromium &
```


d) Si ejecuto en la terminal lo siguiente:

```
 Bash
$ chromium &
$ firefox &
```

luego el siguiente comando

```
 Bash
$ jobs
```

Obtengo la siguiente salida

```
 Bash
[1] - running chromium
[2] + running firefox
```

si luego ejecuto,

```
 Bash
$ disown firefox
```

Esto me que permitirá que el programa firefox siga corriendo una vez que se cierre la sesión de shell actual, mientras que chromium se cerrará junto con la sesión de shell

actual.

8. Interprete el resultado de la siguiente línea:

 Bash

```
$ find /home/user/Documentos -type f -size -100M -exec rm -f {} \;
```

Te podés ayudar con las siguientes salidas del manual del comando

find:

-type c

Buscar archivos de tipo c:

- b archivo especial de bloque
- c archivo especial de carácter
- d directorio
- p FIFO
- f archivo regular
- l enlace simbólico
- s socket (socket)

-size n[cwbkMG]

Buscar archivos de tamaño n (en unidades especificadas). El sufijo puede ser:

- c bytes
- w palabras de 2 bytes
- b bloques de 512 bytes (por compatibilidad con otros programas)
- k kilobytes (1024 bytes)
- M megabytes (1024\*1024 bytes)
- G gigabytes (1024\*1024\*1024 bytes)

El tamaño se puede especificar con prefijos:

- +n mayor que n unidades
- n menor que n unidades
- n exactamente n unidades

`-exec comando {} \;`

Ejecuta el comando especificado, sustituyendo "{}" por el nombre del archivo encontrado. El comando se termina con un punto y coma (;). Se necesita escapar el punto y coma con una barra invertida (\;) para que no sea interpretado por el shell.

Tambien te puede ayudar esta parte del comando rm

#### SINOPSIS

`rm [OPCIÓN]... [ARCHIVO]...`

#### OPCIONES

`-f, --force` Ignorar archivos inexistentes y argumentos, nunca solicitar la confirmación.

`-i` Solicitar confirmación antes de eliminar cualquier archivo.

`-I` Solicitar confirmación una vez antes de eliminar más de tres archivos, o al eliminar recursivamente. Menos intrusivo que `-i`, mientras se protege contra la mayoría de los errores.