U4 ADMINISTRACIÓN Y ASEGURAMIENTO DE LA INFORMACIÓN

ACTIVIDAD PRÁCTICA 14: Automatización (SCRIPTS)

5 DE MAYO DE 2023

SISTEMAS INFORMÁTICOS

Sergio Cobo García

Contenido

Ejercicio 14

14	Escriba un script de shell que, a partir de la ruta de un directorio determinado, si existe, acceda a ese directorio. • Si no existe, deberá crear ese directorio y acceder a él. • En caso de no poder crear el directorio, mostrará por pantalla el mensaje "No se ha podido crear el directorio" y quedarse en la ruta original. Resuelva este ejercicio sin utilizar los operadores && y ||, gestionando la casuística con estructuras alternativas. Para comprobar que su script de shell se comporta de la forma esperada, muestre por pantalla el directorio de trabajo actual. ¿Dónde queda su terminal? ¿Se había realizado la ejecución en el nuevo directorio o en el directorio original?

```
#!/bin/bash

# Verificar si el número de parámetros es correcto
if [ "$#" -ne 1 ];
then
  echo "Número de parámetros incorrecto"
  exit 1
fi

#Ruta directorio
ruta=$1

if (test -d $ruta)
then
    echo "El directorio existe"
    cd ./$ruta;
else
    echo "La ruta no existe";
    echo "Se ha creado el directorio $ruta";
    mkdir $ruta || echo "No se ha podido crear";
    cd $ruta;
fi

exit 0;
```

```
sergio@sergio-VirtualBox:~/Escritorio/PRACTICA 14_2$ ./ejercicio14.sh
hellou
La ruta no existe
Se ha creado el directorio hellou
```

Ejercicio 15

15	Escribe un shellscript que compruebe con qué usuario se ha ejecutado. Si se ha ejecutado como superusuario, mostrará el mensaje "Soy el amo del mundo". Si no se ha ejecutado inicialmente como superusuario, se volverá a ejecutar a sí mismo como superusuario. En ningún momento del shellscript puede aparecer el nombre del propio shellscript, deberás utilizar algún otro mecanismo o parámetro para poder ejecutarlo como superusuario. Ten en cuenta que si se ha ejecutado algún sudo desde el terminal recientemente el resultado puede resultar confuso: cierra y abre el terminal para asegurarte del resultado.

```
#!/bin/bash

# Verificar si el número de parámetros es correcto
if `test $# -ne 0`
then
  echo "Número de parámetros incorrecto"
  exit 1
fi

# Compruebo quien es el usuario
usuario=`whoami`;

echo "Se ha ejecutado con el usuario $usuario";

# Compruebo si es el usuario root, sino pido la contraseña para que
# ejecute el archivo como root
if (test $usuario = "root")
then
    echo "Soy el dueño del mundo";
else
    echo "Intenta como superusuario";
    sudo $0;
fi;

exit 0;
```

```
sergio@sergio-VirtualBox:~/Escritorio/PRACTICA 14_2$ ./ejercicio15.sh
Se ha ejecutado con el usuario sergio
Intenta como superusuario
[sudo] contraseña para sergio:
Se ha ejecutado con el usuario root
Soy el dueño del mundo
```

Ejercicio 16

16	Escribe un shellscript que se mate a sí mismo (como proceso). Para comprobar que funciona correctamente, coloca alguna instrucción debajo del final del proceso y asegúrate de que no se ejecute.

```bash
#!/bin/bash

# Verificar si el número de parámetros es correcto
if `test $# -ne 0`
then
    echo "Número de parámetros incorrecto"
    exit 1
fi

# PID del archivo
echo "PID del script: $$"

# Comprobar si se ha matado el proceso
if (test -e $0)
then
    echo "El proceso $$ se ha matado correctamente"
    kill $$;
else
    echo "No se ha podido matar el proceso $$"
fi;

exit 0;
```

```
sergio@sergio-VirtualBox:~/Escritorio/PRACTICA 14_2$ ./ejercicio16.sh
PID del script: 26397
El proceso 26397 se ha matado correctamente
Terminado
```

Ejercicio 17

17	Escriba un shellscript que, a partir de un nombre de archivo ($1) y un número de partes ($2), divida el archivo $1 en $2 partes. Para hacerlo, utilizará el comando que divide archivos y alguna operación matemática. Pruebe a dividir el archivo en varios números de partes y asegúrese de que siempre se genere el número de archivos indicado exactamente.

```bash
#!/bin/bash

# Verificar si el número de parámetros es correcto
if `test $# -ne 2`
then
    echo "Número de parámetros incorrecto"
    exit 1
fi

archivo=$1;
partesArchivo=$2;

# Comprobar si el archivo existe
if (test -f $archivo)
then
    echo "El archivo $archivo existe"
else
    echo "El archivo $archivo no existe"
fi

# Contar el número de líneas del archivo
numeroLineas=`wc -l 2> /dev/null < $archivo`;

# Calcula el número de líneas por parte
porLineas=`expr $numeroLineas / $partesArchivo 2> /dev/null`;

# Divide el archivo por partes
`split -l $porLineas $archivo 2> /dev/null`;

# Comprobar que la cantidad de archivos creados sea correcto
cantidadArchivos=`ls -1 x* 2> /dev/null | wc -l`;
if (test $cantidadArchivos -ne $partesArchivo)
then
    echo "Error: el número de lineas no es divisible o no existe el
    archivo $archivo";
    `rm x* 2> /dev/null`;
else
    echo "El archivo $archivo se ha dividido correctamente en $partesArchivo";
fi

exit 0;
```

```
alumati@laD20215ubt2204:/media/alumati/KGT USB/IES/1r DAW/Sistemes Informàtics/SCRIPTS/PRACTIC
A 14_2$ ./ejercicio17.sh ejemplo.txt 2
El archivo ejemplo.txt existe
El archivo ejemplo.txt se ha dividido correctamente en 2
```

Ejercicio 18

18	Escribe un shellscript que, dado un archivo (con su ruta absoluta) y el nombre de un paquete, nos indique si el archivo pertenece al paquete o no (es decir, si fue instalado o requerido por el paquete).

```bash
#!/bin/bash

# Verificar si el número de parámetros es correcto
if `test $# -ne 2`
then
    echo "Número de parámetros incorrecto"
    exit 1
fi

ruta_archivo=$1;
paquete=$2;

# Compruebo si el archivo pertenece al paquete
if `dpkg -S $ruta_archivo | grep -q $paquete`
then
    echo "$ruta_archivo pertenece al paquete $paquete";
else
    echo "$ruta_archivo no pertenece al paquete $paquete";
fi

exit 0;
```

```
alumati@laD20203ubt2204:/media/alumati/KGT USB/IES/1r DAW/Sistemes Informàtics/S
CRIPTS/PRACTICA 14_2$ ./ejercicio18.sh vim /usr/bin/vim
vim pertenece al paquete /usr/bin/vim
```

Ejercicio 19

19	Tienes un archivo de texto en el que cada línea contiene dos nombres de archivo separados por un guión "-". Escribe un shellscript que procese todas las líneas de este archivo realizando la siguiente operación: si el primer archivo existe, lo copiará sobre el segundo; si no existe, copiará el contenido del archivo /etc/group sobre el segundo.

```bash
#!/bin/bash

# Verificar si el número de parámetros es correcto
if `test $# -ne 1`
then
    echo "Número de parámetros incorrecto"
    exit 1
fi

# Almaceno el contenido de $archivo1
a=`cat ./$archivo1 2> /dev/null`;

# Compruebo si $archivo1 existe
while read linea
do
    # Guardo el nombre de cada archivo
    archivo1=`echo $linea | cut -d- -f1`;
    archivo2=`echo $linea | cut -d- -f2`;

    if `test -e $archivo1`
    then
        cp $archivo1 $archivo2;
        echo "El contenido de $archivo1 se ha copiado en $archivo2";
    else
        cp /etc/group $archivo2;
        echo "El contenido de $archivo1 no se ha copiado, se ha copiado /etc/group en $archivo2";
    fi;
done < $1;

exit 0;
```

---

ejemplo2.txt

```
ejemplo5.txt-ejemplo4.txt
ejemplo7.txt-ejemplo6.txt
```

```
sergio@sergio-VirtualBox:~/Escritorio/PRACTICA 14_2$ ./ejercicio19.sh ejemplo2.t
xt
El contenido de ejemplo5.txt se ha copiado en ejemplo4.txt
El contenido de ejemplo7.txt no se ha copiado, se ha copiado /etc/group en ejemp
lo6.txt
```

Ejercicio 20

20	Indique justificadamente cuál es la función del siguiente shellscript, indicando cuál es el significado más lógico de sus parámetros. ¿Qué efecto tendría sustituir la línea en azul por: if grep ttt $i?

```
#!/bin/bash

j=0

for i in $*
do
    if grep ttt $i > /dev/null 2> /dev/null
    then
        j=`expr $j + 1`
    fi
done

echo $# $j
```

El bucle itera todos los parámetros, dentro de él se busca en un condicional la cadena de texto “ttt” en el valor de i, si la encuentra no la imprime o si da algún error lo descarta e incrementa 1 la variable j. Finalmente imprime el número de parámetros y los que tengan alguna coincidencia ttt.

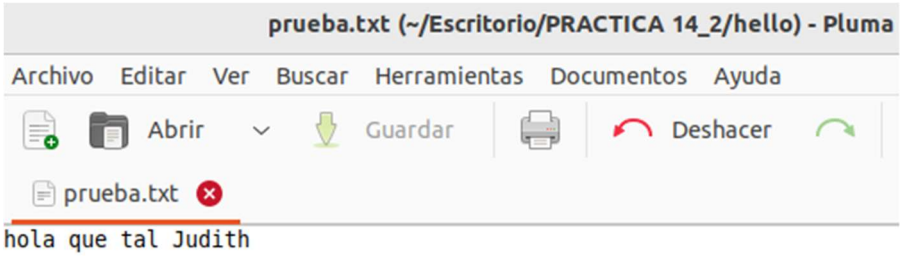
La diferencia es que sin los /dev/null las coincidencias con “ttt” sí se imprimirán por pantalla.

Ejercicio 21

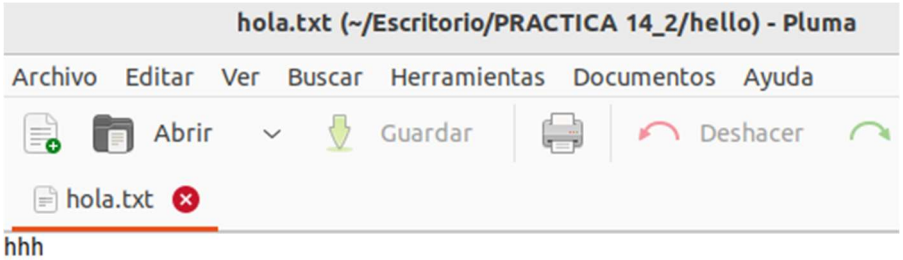
21	Escriba un shellscript que, a partir de un directorio y una palabra, pasados como parámetros, compruebe que el primer parámetro sea un directorio: a) En caso de que no lo sea, mostrará un mensaje de error. b) En caso de que sí lo sea, buscará en todos los archivos del directorio indicado (sólo el directorio, no sus subdirectorios) la palabra pasada como parámetro e indicará, para cada archivo, el número de ocurrencias de la palabra indicada. Resuelva este ejercicio utilizando un bucle for. No se permite el uso del comando find.

```bash
#!/bin/bash

# Verificar si el número de parámetros es correcto
if `test $# -ne 2`
then
    echo "Número de parámetros incorrecto"
    exit 1
fi

directorio=$1;
palabra=$2;
j=0;

# Compruebo si es un directorio
if `test ! -d $directorio`
then
    echo "Error: $directorio no es un directorio"
fi;

# Iterar cada archivo del directorio
for archivo in $directorio/*
do
    # Compruebo si es un archivo
    if `test -f $archivo`
    then
        # Buscar y contar ocurrencias
        ocurrencias=`cat $archivo | grep -o $palabra $archivo | wc -l`;
        echo "El archivo $archivo tiene $ocurrencias ocurrencias de la palabra $palabra";
        j=`expr $j + $ocurrencias`;
    fi
done

echo "Hay $j ocurrencias de la palabra $palabra en los archivos del directorio $directorio"

exit 0;
```

```
sergio@sergio-VirtualBox:~/Escritorio/PRACTICA 14_2$ ./ejercicio21.sh hello hola
El archivo hello/hola.txt tiene 0 ocurrencias de la palabra hola
El archivo hello/prueba.txt tiene 1 ocurrencias de la palabra hola
Hay 1 ocurrencias de la palabra hola en los archivos del directorio hello
```




```
hola que tal Judith
```




```
hhh
```

Ejercicio 22

22	Escribe un shellscript que reciba como parámetro el nombre de un directorio y obtenga como resultado un único archivo, a partir de la compresión y empaquetamiento del directorio y todos sus subdirectorios. a) Haced las comprobaciones necesarias para que el shellscript compruebe si el directorio a comprimir existe. Haced también las comprobaciones necesarias para que solo admita directorios (no archivos, por lo tanto). b) Haz que el nombre del archivo termine con la fecha actual en el formato AAAA_MM_DD. Por ejemplo, la copia de seguridad del 9 de marzo de 2011 sería backup_2011_03_09.tar.gz. Debe almacenarse en la ruta que llama el shellscript. Si se muestran mensajes de error al comprimir (típicamente relacionados con falta de permisos), deberán silenciarse las salidas.

```
#!/bin/bash

# Verificar si el número de parámetros es correcto
if `test $# -ne 1`
then
    echo "Número de parámetros incorrecto"
    exit 1
fi

# Compruebo si es un directorio
if `test ! -d $directorio`
then
    echo "Error: $directorio no es un directorio o no existe";
fi;

fecha_actual=(date +%Y_%m_%d);
archivo=$1_$fecha_actual.tar.gz;
tar -czf $archivo $1 > /dev/null;

# Compruebo si se ha comprimido
if `test $? -ne 0`
then
    echo "Error: se produjo un error al comprimir el directorio";
fi;

echo "Copia de seguridad realizada correctamente en el archivo $archivo";

exit 0;
```

```
sergio@sergio-VirtualBox:~/Escritorio/PRACTICA 14_2$ ./ejercicio22.sh hello
Copia de seguridad realizada correctamente en el archivo hello_date.tar.gz
```

Ejercicio 23

23	Escribid un shellscript que reciba como parámetro un valor alfanumérico V. Este shellscript deberá poner el sistema en el runlevel indicado. Antes de ponerlo en este runlevel, deberá comprobar que efectivamente este es un runlevel legal. En caso de que este runlevel no exista, el sistema mostrará

	un mensaje por pantalla indicando que no se ha podido ejecutar la acción. En caso de que exista, el sistema cambiará el runlevel al indicado. Tengan en cuenta el tipo de usuario necesario para ejecutar este shellscript y las comprobaciones necesarias para poder ejecutarlo.

```
#!/bin/bash

# Verificar si el número de parámetros es correcto
if `test $# -ne 1`
then
    echo "Número de parámetros incorrecto"
    exit 1
fi

# Compruebo si se usa como root
if `id -u != 0`
then
    echo "Debe ejecutarse como root"
fi;

# Comprobar si el runlevel es válido en el rango
if `$1 -lt 0 || $1 -gt 6`
then
    echo "Error: el runlevel $1 no es válido, introduce uno del 0 al 6"
fi;

init $1;

exit 0;
```

```
sergio@sergio-VirtualBox:~/Escritorio/PRACTICA 14_2$ ./ejercicio23.sh 6
```

Ejercicio 24

24	Indique justificadamente cuál es la función del siguiente shellscript, indicando cuál es el significado más lógico de los parámetros.

```
#!/bin/bash

for j in ../directori/*
do
   if tail $j | grep $1 > /dev/null
   then
      mv $j .
   fi
done
```

El bucle itera sobre cada archivo del directorio padre del directorio actual, dentro de él se busca las últimas 10 líneas del archivo y en ellas busca si alguna contiene el primer parámetro. Finalmente, si la encuentra no la imprime y mueve el archivo al directorio actual.

Ejercicio 25

25	Escriba un shellscript que, a partir de una ruta de directorio (absoluta o relativa) que será el primer parámetro de este ($1), examine todos los archivos con extensión *.c del directorio indicado y cuente el número de líneas no blancas. La salida del shellscript deberá ser un archivo "linies.txt" que contendrá los nombres de cada archivo, junto con el número de líneas no vacías de este, ordenado de forma decreciente.

```
$ exercici ../practiquesC
$ cat linies.txt
practica3.c 1530
practica2.c 720
practica1.c 378
```

```bash
#!/bin/bash

# Verificar si el número de parámetros es correcto
if `test $# -ne 1`
then
    echo "Número de parámetros incorrecto"
    exit 1
fi

ruta=$1;

# Compruebo si existe el directorio
if `test ! -d $ruta`
then
    echo "El directorio $ruta no existe"
fi;

# Recorrer los archivos con extensión .c
for archivo in $ruta*.c
do
    total_lineas_noVacias=`grep -v "^$" $archivo | wc -l`;
    echo $archivo $total_lineas_noVacias >> linies.txt;
done

# Ordenar por número de líneas no vacías
sort -rn -k2 linies.txt -o linies.txt;

exit 0;
```

archivo1.c

```
1  hola
2
3  que
4
5
```

archivo2.c

```
1  hola
2
3  que
4  tal
5
```

archivo3.c

```
1  hola
2
```

ejercicio25.sh

```
../archivo2.c 3
../archivo1.c 2
../archivo3.c 1
```

linies.txt

```
sergio@sergio-VirtualBox:~/Escritorio/PRACTICA 14_2$ ./ejercicio25.sh ../
```