PRÁCTICA 6

Ejercicio 6.1. **Indique cuál ha sido el error introducido en el guión anterior y cómo se corregiría.**

El error de este guión se encuentra en el operador, donde debería haber un doble igual sin espacios, es decir, “if ["$2"=="after"] ; then”.

También nos serviría la opción de simplemente añadir doble corchete :
if [["$2" = "after"]] ; then

Nótese que para ejecutar este guión , el cual modifica la variable PATH, se ha de usar el comando "source", ya que si no se pone la variable PATH cambiará en el shell hijo, no en el padre, y no notaremos cambio alguno.

~/Escritorio/Practica6$ source pathmas tryingpath after
~/Escritorio/Practica6$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
:tryingpath

Ejercicio 6.2. **Aplicar las herramientas de depuración vistas en la sección 2 para la detección de errores durante el desarrollo de los guiones propuestos como ejercicios en la práctica 5.**

Utilizando los métodos de depuración aprendidos en esta sección, he podido comprobar que los guiones de la práctica anterior no tienen ningún error y funcionan correctamente.

Algunos ejemplos de depuración que he realizado :

bash -x

~/Escritorio/Practica5$ bash -x cpback
+ destino=CopiasSeguridad
+ test -d CopiasSeguridad
+ ((0 >= 1))
+ echo 'El número de parámetros introducidos es incorrecto.'
El número de parámetros introducidos es incorrecto.

~/Escritorio/Practica5$ bash -x newdirfiles
+ dir_name=
+ num_files=
+ base_filename=
+ ((0 == 3))
+ echo 'Se ha introducido un número incorrecto de parámetros.'
Se ha introducido un número incorrecto de parámetros.

bash -n

~/Escritorio/Practica5$ bash -n cpback
~/Escritorio/Practica5$ bash -n 5.7
~/Escritorio/Practica5$ bash -n 5.6

bash -v

~/Escritorio/Practica5$ bash -v 5.5
#!/bin/bash

echo "Introduzca un número comprendido entre 1 y 10 :"
Introduzca un número comprendido entre 1 y 10 :
read numero
11

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
El dato introducido no se encuentra en el rango especificado.
Vuelva a introducir el número pedido.
8

echo "El número introducido es $numero y se encuentra dentro del rango especificado."
El número introducido es 8 y se encuentra dentro del rango especificado.

trap 'echo Fin del guión $0' EXIT

~/Escritorio/Practica5$ ./5.7 bash
bece
root
Fin del guión ./5.7

Ejercicio 6.3. **Escribir un guión que nos dé el nombre del proceso del sistema que consume más memoria.**

```
#!/bin/bash

#Explicación : Copiamos la información de toda una iteración en un archivo. Posteriormente, guardamos en una variable el número de líneas del archivo, para de esta forma crear otro variable que nos almacene el número de líneas sin contar las 8 primeras que son inutilizables. A continuación, cogemos mediante la orden tail todas las líneas de nuestro archivo menos esas 8 primeras y las ordenamos a partir del campo que indica la memoria consumida, de tal forma que se ordenan de las que menos consumen a las que más. Una vez hecho esto, nos quedamos con la última línea, pues es la que indica el programa que mayor memoria está consumiendo, y mediante la orden 'tr -s' hacemos que los espacios se conviertan en uno solo, de tal forma que podremos utilizar el espacio como un delimitador de campo para así poder sacar el nombre del programa mediante un cut.

top -bn 1 > archivo_temporal.txt

numero_lineas=`cat archivo_temporal.txt | wc -l`
numero_lineas_validas=`expr $numero_lineas - 8`

tail -n $numero_lineas_validas archivo_temporal.txt | sort -k 10 | tail -n 1 | tr -s " " " " | cut -d " " -f12 | cat

rm archivo_temporal.txt
```

Ejercicio 6.4. **Escribir un guión que escriba números desde el 1 en adelante en intervalos de un segundo ¿Cómo se podría, desde otro terminal, detener la ejecución de dicho proceso, reanudarlo y terminar definitivamente su ejecución?**

Realizamos el siguiente guión :

```
#!/bin/bash

numero=0

for (( contador=1; contador>0; contador++ ))

do
        sleep 1
        numero=`expr $numero + 1`
        echo $numero

done
```

Para poder matar este proceso tras ejecutarlo desde otra terminal tendremos que saber su identificador. Para ello usaremos, en la nueva terminal, la orden ps -a y buscaremos el nombre de nuestro proceso y miraremos su PID.

Para detener la ejecución del proceso usaremos la orden:

kill -STOP PID

Para reanudarlo:

kill -CONT PID

Y para finalizarlo definitivamente:

kill -TERM PID

Nótese que también podríamos buscar el PID de nuestro proceso mediante las órdenes ps -ax ó ps -axu.

Ejercicio 6.5. **¿Se puede matar un proceso que se encuentra suspendido? En su caso, ¿cómo?**

Sí, mediante la orden kill -9 PID, donde PID es el identificador del proceso o mediante la orden kill -9 %n, donde n es el número de trabajo correspondiente al proceso.

Nótese que para saber el identificador de los procesos o sus números de trabajo hacemos uso de la orden jobs -l, donde [n] será el número de trabajo y un número tal como 10217 será su identificador.

Ejercicio 6.6. ***¿Qué debemos hacer a la orden top para que nos muestre sólo los procesos nuestros?***

Añadimos la opción -u junto con nuestro nombre de usuario:
top -u $USER

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