PRÁCTICA 4: Expresiones con Variables y Expresiones Regulares (DPR)

EJERCICIO 4.1: Utilizando una variable que contenga el valor entero 365 y otra que guarde el número del día actual del año en curso, realice la misma operación del ejemplo anterior usando cada una de las diversas formas de cálculo comentadas hasta el momento, es decir, utilizando expr, $((...)) y $[...].

```
usuario@usuario-VirtualBox:~$ declare DIAS=365
usuario@usuario-VirtualBox:~$ declare D_ACTUAL=`date +%j`

usuario@usuario-VirtualBox:~$ echo "FALTAN $(( (DIAS - D_ACTUAL) / 7 ))
SEMANAS"
FALTAN 11 SEMANAS

usuario@usuario-VirtualBox:~$ echo "FALTAN $[ (DIAS - D_ACTUAL) / 7 ]
SEMANAS"
FALTAN 11 SEMANAS
```

EJERCICIO 4.2: Realice las siguientes operaciones para conocer el funcionamiento del operador de incremento como sufijo y como prefijo. Razone el resultado obtenido en cada una de ellas:

```
v=1
Asigna a la variable v, el valor 1

echo $v
Valor que muestra: 1
Muestra el valor de la variable introducida

echo $((v++))
Valor que muestra: 1
Primero se muestra el valor de la variable y después la incrementa en 1.

echo $v
Valor que muestra: 2
El valor de la variable ha aumentado en 1 por v++.

echo $((++v))
Valor que muestra: 3
El valor de la variable se incrementa en 1 y después se muestra dicho
valor.

echo $v
Valor que muestra: 3
Como no se ha realizado ningún cambio desde el último incremento, se
mantiene el valor.
```

EJERCICIO 4.3: Utilizando el operador de división, ponga un caso concreto donde se aprecie que la asignación abreviada es equivalente a la asignación completa, es decir, que x/=y equivale a x=x/y. En el resultado del cálculo de expresiones aritméticas, bash solamente trabaja con números enteros, por lo que si se necesitase calcular un resultado con decimales, habría que utilizar una forma alternativa, como puede ser la ofrecida por la orden bc , cuya opción - l , letra “ele”, permite hacer algunos cálculos matemáticos (admite otras posibilidades que pueden verse mediante man).

```
usuario@usuario-VirtualBox:~$ x=20 && y=5
usuario@usuario-VirtualBox:~$ echo $x
20
usuario@usuario-VirtualBox:~$ echo $y
5
usuario@usuario-VirtualBox:~$ echo $[x=x/y]
4
usuario@usuario-VirtualBox:~$ x=20 && y=5
usuario@usuario-VirtualBox:~$ echo $[x/=y]
4
```

EJERCICIO 4.4: Compruebe qué ocurre si en el ejemplo anterior utiliza comillas dobles o simples para acotar todo lo que sigue a la orden echo. ¿Qué sucede si se acota entre comillas dobles solamente la expresión aritmética que se quiere calcular?, ¿y si se usan comillas simples?

```
usuario@usuario-VirtualBox:~$ echo "6/5|bc -l"
6/5|bc -l

usuario@usuario-VirtualBox:~$ echo '6/5|bc -l'
6/5|bc -l

usuario@usuario-VirtualBox:~$ echo '6/5'|bc -l
1.20000000000000000000

usuario@usuario-VirtualBox:~$ echo "6/5"|bc -l
1.20000000000000000000
```

EJERCICIO 3.5: Calcule con decimales el resultado de la expresión aritmética (3-2)/5. Escriba todas las expresiones que haya probado hasta dar con una respuesta válida. Utilizando una solución válida, compruebe qué sucede cuando la expresión aritmética se acota entre comillas dobles; ¿qué ocurre si se usan comillas simples?, ¿y si se ponen apóstrofos inversos?

```
usuario@usuario-VirtualBox:~$ echo (3-2)/5 | bc -l
bash: error sintáctico cerca del elemento inesperado `3-2'
usuario@usuario-VirtualBox:~$ echo $[(3-2)/5] | bc -l
0
usuario@usuario-VirtualBox:~$ echo "(3-2)/5" | bc -l
.20000000000000000000
usuario@usuario-VirtualBox:~$ echo '(3-2)/5' | bc -l
.20000000000000000000
```

EJERCICIO 4.6: Consulte la sintaxis completa de la orden let utilizando la orden de ayuda para las órdenes empotradas (help let) y tome nota de su sintaxis general.

```
let: let arg [arg ...]
Evalúa expresiones aritméticas.

Evalúa cada ARG como una expresión aritmética.  La evaluación se hace
con enteros de longitud fija, sin revisar desbordamientos, aunque la
la división por 0 se captura y se marca como un error.
```

EJERCICIO 4.7: Al realizar el ejercicio anterior habrá observado que la orden let admite asignaciones múltiples y operadores que nosotros no hemos mencionado anteriormente. Ponga un ejemplo de asignación múltiple y, por otra parte, copie en un archivo el orden en el que se evalúan los operadores que admite.

```
id++, id--   post-incremento, post-decremento de variable
++id, --id   pre-incremento, pre-decremento de variable
-,
+      menos, más unario
!, ~      negación lógica y basada en bits
**     exponenciación
*, /, %      multiplicación, división, residuo
+, -      adición, sustracción
<<, >>    desplazamientos de bits izquierdo y derecho
<=, >=, <, > comparación
==, !=    equivalencia, inequivalencia
&      AND de bits
^      XOR de bits
|      OR de bits
&&     AND lógico
||     OR lógico
expr ? expr : expr
operador condicional
=, *=, /=, %=,
+=, -=, <<=, >>=,
&=, ^=, |
=  asignación
```

EJERCICIO 4.8: Probad los siguientes ejemplos y escribir los resultados obtenidos con la evaluación de expresiones

```
usuario@usuario-VirtualBox:~$ echo ejemplo1
ejemplo1
usuario@usuario-VirtualBox:~$ valor=6
usuario@usuario-VirtualBox:~$ if [ $valor = 3 ]; then echo si; else echo
no; fi
no
usuario@usuario-VirtualBox:~$ echo $valor
6

usuario@usuario-VirtualBox:~$ echo ejemplo2
ejemplo2
usuario@usuario-VirtualBox:~$ valor=5
usuario@usuario-VirtualBox:~$ if [ $valor = 3 ] && ls; then echo si;
else echo no; fi
no
```

```
usuario@usuario-VirtualBox:~$ echo $valor
5

usuario@usuario-VirtualBox:~$ echo ejemplo3
ejemplo3
usuario@usuario-VirtualBox:~$ valor=5
usuario@usuario-VirtualBox:~$ if [ $valor = 5 ] && ls; then echo si;
else echo no; fi
Descargas  Documentos         Imágenes  numE.sh     practica2  Vídeos
Desktop    ejer3.6_script.sh  Música    Plantillas  Público
si
usuario@usuario-VirtualBox:~$ echo $valor
5

usuario@usuario-VirtualBox:~$ echo ejemplo4
ejemplo4
usuario@usuario-VirtualBox:~$ valor=6
usuario@usuario-VirtualBox:~$ if ((valor==3)); then echo si; else echo
no; fi
no
usuario@usuario-VirtualBox:~$ echo $valor
6

usuario@usuario-VirtualBox:~$ echo ejemplo5
ejemplo5
usuario@usuario-VirtualBox:~$ valor=5
usuario@usuario-VirtualBox:~$ if ((valor==3)) && ls; then echo si; else
echo no; fi
no
usuario@usuario-VirtualBox:~$ echo $valor
5

usuario@usuario-VirtualBox:~$ echo ejemplo6
ejemplo6
usuario@usuario-VirtualBox:~$ valor=5
usuario@usuario-VirtualBox:~$ if ((valor==5)) && ls; then echo si; else
echo no; fi
Descargas  Documentos         Imágenes  numE.sh     practica2  Vídeos
Desktop    ejer3.6_script.sh  Música    Plantillas  Público
si
usuario@usuario-VirtualBox:~$ echo $valor
5

usuario@usuario-VirtualBox:~$ echo ejemplo7
ejemplo7
usuario@usuario-VirtualBox:~$ echo $((3>5))
0
usuario@usuario-VirtualBox:~$ echo $?
0

usuario@usuario-VirtualBox:~$ echo ejemplo8
ejemplo8
usuario@usuario-VirtualBox:~$ ((3>5))
usuario@usuario-VirtualBox:~$ echo $?
1
```

```
usuario@usuario-VirtualBox:~$ echo ejemplo9
ejemplo9
usuario@usuario-VirtualBox:~$ if ((3>5)); then echo 3 es mayor que 5;
else echo 3 no es mayor que 5; fi
3 no es mayor que 5
```

EJERCICIO 4.9: Haciendo uso de las órdenes conocidas hasta el momento, construya un guion que admita dos parámetros, que compare por separado si el primer parámetro que se le pasa es igual al segundo, o es menor, o es mayor, y que informe tanto del valor de cada uno de los parámetros como del resultado de cada una de las evaluaciones mostrando un 0 o un 1 según corresponda.

```
#!/bin/bash
#Titulo:  Evaluate
#Fecha:   15/10/2018
#Autor:   Daniel Pérez Ruiz
#Version: 1.0
#Descripcion:   El guión admite dos parámetros, compara el primero con
#               el segundo si es igual, mayor o menor y devuelve 0 o 1.
#Opciones:       Ninguna
#Uso:            Evaluate num1 num2

num1=$1 && num2=$2
echo 'El primer numero introducido es:'  $num1;
echo 'EL segundo numero introducido es:'  $num2;

echo Evaluate $num1 '<' $num2
if (( $num1<$num2 ));
then echo value 1; else echo value 0; fi;

echo Evaluate $num1 '>' $num2
if (( $num1>$num2 ));
then echo value 1; else echo value 0; fi;

echo Evaluate $num1 '=' $num2
if (( $num1==$num2 ));
then echo value 1; else echo value 0; fi;
```

EJERCICIO 4.10: Usando test , construya un guion que admita como parámetro un nombre de archivo y realice las siguientes acciones: asignar a una variable el resultado de comprobar si el archivo dado como parámetro es plano y tiene permiso de ejecución sobre él; asignar a otra variable el resultado de comprobar si el archivo es un enlace simbólico; mostrar el valor de las dos variables anteriores con un mensaje que aclare su significado. Pruebe el guion ejecutándolo con /bin/cat y también con /bin/rnano.

```
#!/bin/bash
#Titulo:  checktest
#Fecha:   15/10/2018
#Autor:   Daniel Pérez Ruiz
#Version: 1.0
#Descripcion:   Comprueba si el archivo es plano es plano y se tiene
#               permiso de ejecucion o si es un enlace simbolico
#Opciones:     Ninguna
#Uso:          checktest <archivo>

planoexec=`((test -x $1) && (test -f $1)) && echo "es" || echo "no es"`;
enlacesimb=`(test -h $1) && echo "es" || echo "no es"`;

echo "El archivo $1 $planoexec un archivo plano ejecutable";
echo "El archivo $1 $enlacesimb un enlace simbolico";
```

EJERCICIO 4.11: Ejecute help test y anote qué otros operadores se pueden utilizar con la orden test y para qué sirven. Ponga un ejemplo de uso de la orden test comparando dos expresiones aritméticas y otro comparando dos cadenas de caracteres.

```
String operators:
      -z STRING      True if string is empty.
      -n STRING
         STRING      True if string is not empty.
      STRING1 = STRING2
                     True if the strings are equal.
      STRING1 != STRING2
                     True if the strings are not equal.
      STRING1 < STRING2
                     True if STRING1 sorts before STRING2
lexicographically.
      STRING1 > STRING2
                     True if STRING1 sorts after STRING2
lexicographically.
    arg1 OP arg2   Arithmetic tests.  OP is one of -eq, -ne,
                          -lt, -le, -gt, or -ge.

Compración de cadenas:
$ echo `[ "hola" = "hola" ] && echo "hola y hola son cadenas idénticas"`
 hola y hola son cadenas idénticas

Comparaciones aritméticas
$ echo `[ 3 -ge 2 ] && echo "3 es mayor que 2"`
3 es mayor que 2
```

EJERCICIO 4.12:

APARTADO A: Razone qué hace la siguiente orden:

```
Esta orden imprime por pantalla que el archivo sesion5.pdf del
directorio donde es ejecutada la orden existe, si este, además de
existir, es un archivo plano (-f)
```

APARTADO B: Añada los cambios necesarios en la orden anterior para que también muestre un mensaje de aviso en caso de no existir el archivo. (Recuerde que, para escribir de forma legible una orden que ocupe más de una línea, puede utilizar el carácter " \ " como final de cada línea que no sea la última.)

```
$ if [ -f ./sesion5.pdf ]; then printf "El archivo ./sesion5.pdf
existe\n"; \else printf "\n\nEl archivo ./sesion5.pdf no existe"; fi
```

APARTADO C: Sobre la solución anterior, añada un bloque elif para que, cuando no exista el archivo ./sesion5.pdf, compruebe si el archivo /bin es un directorio. Ponga los mensajes adecuados para conocer el resultado en cada caso posible.

```
$ if [ -f ./sesion5.pdf ]; then printf "El archivo ./sesion5.pdf
existe\n"; \elif [ -d /bin ]; then printf "/bin es un directorio\n";
\else printf "\n\nEl archivo ./sesion5.pdf no existe y /bin no es un \
directorio"; fi
/bin es un directorio
```

APARTADO D: Usando como base la solución del apartado anterior, construya un guion que sea capaz de hacer lo mismo pero admitiendo como parámetros la ruta relativa del primer archivo a buscar y la ruta absoluta del segundo. Pruébelo con los dos archivos del apartado anterior.

```
#!/bin/bash
# Titulo:       existe
# Fecha:        18/10/2018
# Autor:        Daniel Pérez Ruiz
# Version:      1.0
# Descripción:  Comprueba si el primer argumento en un archivo plano
#               y existe o si el segundo es un directorio
# Opciones: ninguna
# Uso: existe <ruta_relativa_posible_archivo>
<ruta_absoluta_posible_directorio>

if [ -f $1 ]; then
    printf "El archivo $1 existe\n";
elif [ -d $2 ]; then
    printf "El archivo $1 no existe pero $2 es un directorio\n";
else
    printf "El archivo $1 no existe ni $2 es un directorio\n";
fi
```

EJERCICIO 4.13: Construya un guion que admita como argumento el nombre de un archivo o directorio y que permita saber si somos el propietario del archivo y si tenemos permiso de lectura sobre él.

```
#!/bin/bash
# Titulo:       lecturaypropio
# Fecha:        18/10/2018
# Autor:        Daniel Pérez Ruiz
# Version:      1.0
# Descripción:  Comprueba si un archivo pertenece al usuario y
#               si este tiene permisos de lectura sobre él.
# Opciones: ninguna
# Uso: lecturaypropio <archivo>

if [ -O $1 ]; then
    echo "Eres el propietario del archivo $1";
else
    echo "No eres el propietario del archivo $1";
fi

if [ -r $1 ]; then
    echo "Tienes permisos de lectura sobre el archivo $1";
else
    echo "No tienes permisos de lectura sobre el archivo $1";
fi
```

EJERCICIO 4.14: Escriba un guion que calcule si el número de días que faltan hasta fin de año es múltiplo de cinco o no, y que comunique el resultado de la evaluación. Modifique el guion anterior para que admita la opción -h de manera que, al ejecutarlo con esa opción, muestre información de para qué sirve el guion y cómo debe ejecutarse.

```
#!/bin/bash
# Titulo:       diasmultiplo
# Fecha:        19/10/2017
# Autor:        Ricardo Ruiz
# Version:      1.0
# Descripción:  Comprueba si el número de días restantes para fin de año
#               el múltiplo de 5
# Opciones: ninguna
# Uso: diasmultiplo [-h]

if [ "$1" == "-h" ]; then
    echo "Este programa comprueba si el número de días restantes para \
fin de año es múltiplo de 5."
    echo "Para ejecutarlo simplemente ejecute ./diasmultiplo";
else
    dias_restantes=$[365 -$(date +%j)]
    echo "Quedan $dias_restantes días para el fin de año."

    if [ $[dias_restantes % 5] == 0 ]; then
        echo "Y $dias_restantes es múltiplo de 5!";
    else
        echo "Pero $dias_restantes no es múltiplo de 5";
    fi

fi
```

EJERCICIO 4.15: ¿Qué pasa en el ejemplo anterior si eliminamos la redirección de la orden if?

```
Lo que ocurre al eliminar la redirección del if es que, en caso de
error, se mostrará tanto nuestro mensaje propio como el error de la
propia orden rm.
```

EJERCICIO 4.16: Haciendo uso del mecanismo de cauces y de la orden echo , construya un guion que admita un argumento y que informe si el argumento dado es una única letra, en mayúsculas o en minúsculas, o es algo distinto de una única letra.

```
#!/bin/bash
# Titulo:       unicaletra
# Fecha:        19/10/2017
# Autor:        Ricardo Ruiz
# Version:      1.0
# Descripción:  Informa si el argumento dado es una única letra
#               en mayúsculas o en minúsculas o es algo distinto
#               de una única letra
# Opciones: ninguna
# Uso: unicaletra <algo>

if  echo $1 | grep '^[a-Z]\{1\}$' 2> /dev/null ; then
    echo "Es una única letra";
else
    echo "Es algo distinto de una única letra";
fi 2> /dev/null
```

EJERCICIO 4.17: Haciendo uso de expresiones regulares, escriba una orden que permita buscar en el árbol de directorios los nombres de los archivos que contengan al menos un dígito y la letra e. ¿Cómo sería la orden si quisiéramos obtener los nombres de los archivos que tengan la letra e y no contengan ni el 0 ni el 1?

```
usuario@usuario-virtualbox:$ find / || [1,2,3,4,5,6,7,8,9] e*
```

EJERCICIO 4.18: Utilizando la orden grep , exprese una forma alternativa de realizar lo mismo que con wc -l.

```
usuario@usuario-virtualbox:#$ grep wc -l
```