Ejercicios

José Antonio de la Rosa Cubero

Práctica 4

Ejercicio 1

```
$dias=365
$dia_actual=`date +%j`
echo "Faltan $[ $(expr $dias - $dia_actual)/7 ] semanas hasta fin de año"
```

Ejercicio 2

```
$ v=1
$ echo $v
1
$ echo $((v++))
1
$ echo $v
2
$((++v))
3
$ echo $v
3
```

Ejercicio 3

```
var1=12
var2=4
$(( var1/=var2 ))
echo $var1
3
var1=12
$(( var1=var1/var2 ))
echo $var1
3
```

Ejercicio 4

```
$echo 6/5|bc -l
1.20000000000000000000
$echo "6/5|bc -l"
6/5|bc -l
$echo '6/5|bc -l'
6/5|bc -l
$echo "6/5"|bc -l
1.20000000000000000000
$echo '6/5'|bc -l
1.20000000000000000000
```

Ejercicio 5

Tanto las comillas dobles como las simples permiten calcular la expresión aritmética, mientras que los apóstrofos inversos dan error.

```
$echo (3-2)/5|bc -l
bash: syntax error near unexpected token `3-2'
$echo "(3-2)/5"|bc -l
.20000000000000000000
$echo '(3-2)/5'|bc -l
.20000000000000000000
$echo `(3-2)/5`|bc -l
bash: command substitution: line 1: syntax error near unexpected token `/5'
bash: command substitution: line 1: `(3-2)/5'
```

Ejercicio 6

```
$help let
let: let arg [arg ...]
    Evaluate arithmetic expressions.

    Evaluate each ARG as an arithmetic expression.  Evaluation is done in
    fixed-width integers with no check for overflow, though division by 0
    is trapped and flagged as an error.  The following list of operators is
    grouped into levels of equal-precedence operators.  The levels are listed
    in order of decreasing precedence.

        id++, id--  variable post-increment, post-decrement
        ++id, --id  variable pre-increment, pre-decrement
        -, +        unary minus, plus
        !, ~        logical and bitwise negation
        **      exponentiation
        *, /, %     multiplication, division, remainder
        +, -        addition, subtraction
        <<, >>      left and right bitwise shifts
        <=, >=, <, >    comparison
        ==, !=      equality, inequality
        &       bitwise AND
        ^       bitwise XOR
        |       bitwise OR
        &&      logical AND
        ||      logical OR
        expr ? expr : expr
                conditional operator
```

```
        =, *=, /=, %=,
        +=, -=, <<=, >>=,
        &=, ^=, |=  assignment

    Shell variables are allowed as operands.  The name of the variable
    is replaced by its value (coerced to a fixed-width integer) within
    an expression.  The variable need not have its integer attribute
    turned on to be used in an expression.

    Operators are evaluated in order of precedence.  Sub-expressions in
    parentheses are evaluated first and may override the precedence
    rules above.

    Exit Status:
    If the last ARG evaluates to 0, let returns 1; let returns 0 otherwise.
```

Ejercicio 7

```
$let "a=7, b=5"
$echo $a
7
$echo $b
5
```

Ejercicio 8

```
ejemplo1
no
6

ejemplo2
no
5

ejemplo3
ejercicios4a.md
si
5

ejemplo4
no
6

ejemplo5
no
5

ejemplo6
ejercicios4a.md
si
5

ejemplo7
0
0
```

```
ejemplo8
1

ejemplo9
3 no es mayor que 5
```

Ejercicio 9

```
#!/bin/bash


echo Veamos si $1 es mayor que $2
echo $[ $1 > $2 ]
echo Veamos si $1 es menor que $2
echo $[ $1 < $2 ]
echo Veamos si $1 es igual a $2
echo $[ $1 == $2 ]
```

Ejercicio 10

```
#!/bin/bash


test -f $1
var1=$?
test -x $1
var1=$var1$?
test -h $1
var2=$[ 1 - $?]

if [ $var1 == 00 ] || [ $var1 == 01 ]
then
    printf "El archivo %s es de texto, " $1
else
    printf "El archivo %s no es de texto, " $1
fi

if [ $var2 == 00 ] || [ $var1 == 10 ]
then
    printf "tiene permisos de ejecución y " $1
else
    printf "no tiene permisos de ejecución y " $1
fi


if [ $var2 == 0 ]
then
    printf "no es un enlace simbólico.\n" $1
else
    printf "es un enlace simbólico.\n" $1
fi
```

Y la salida:

```
$ ./ejercicio10.sh /bin/cat
El archivo /bin/cat es de texto, no tiene permisos de ejecución y no es un
enlace simbólico.
$ ./ejercicio10.sh /bin/rnano
El archivo /bin/rnano es de texto, no tiene permisos de ejecución y es un enlace
simbólico.
```

Ejercicio 11

```
#!/bin/bash

test $1 -eq -$2 || test $1 = $2
echo $?
```

Ejecuciones:

```
$ ./ejercicio11.sh 4 4
0
$ ./ejercicio11.sh 4 6
1
$ ./ejercicio11.sh hola feo
1
$ ./ejercicio11.sh hola hola
0
```

Ejercicio 12

a

Imprime el mensaje "El archivo ./sesion5.pdf existe\n" si sesion5.pdf es un archivo regular que esta en el mismo directorio.

b

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

c

```
if test -f ./sesion5.pdf ; then printf "El archivo ./sesion5.pdf existe\n"; elif
test -d /bin; then printf "El archivo ./sesion5.pdf no existe, pero /bin es un
directorio\n"; else "Ni archivo ./sesion5.pdf existe ni /bin es un directorio";
fi
```

d

```
#!/bin/bash

var1=$PWD/$1
if test -f "$var1";\
 then printf "El archivo $1 existe\n";\
 elif test -d $2;\
 then printf "El archivo $1 no existe, pero $2 es un directorio\n";\
 else "Ni archivo $1 existe ni $2 es un directorio";\
fi
```

Ejercicio 13

```
#!/bin/bash

existe=true

if test -O $1
then
    echo "El archivo existe y es de tu propiedad"
else
    echo "El archivo no es de tu propiedad"
    existe=false
fi

if test -r $1
then
    echo "Tienes permiso de lectura del archivo"
elif $existe
then
    echo "No tienes permiso de lectura del archivo"
else
    echo "No tienes permiso de lectura del archivo o puede que el archivo no
exista"
fi
```

Ejercicio 14

```
#!/bin/bash

if [ $# == 0 ]
then
    dias=365
    dia_actual=`date +%j`
    dias_hasta_fin=$(expr $dias - $dia_actual)

    var1mod5=$[ $dias_hasta_fin%5 ]


    if [ $var1mod5 == 0 ]
    then
        echo "El número de días hasta fin de año es múltiplo de 5"
    else
        echo "El número de días hasta fin de año no es múltiplo de 5"
    fi
elif [ $1 == "-h" ]
```

```
then
    echo "Ejecute el guión para saber cuantos días faltan hasta fin de año"
fi
```

Ejercicio 15

Si borramos la última línea, tenemos la siguiente salida:

```
$ ./ejercicio15.sh temporal
rm: cannot remove 'temporal': No such file or directory
ejercicio15.sh:8 no es posible borrar archivo – código de finalizacion 177
```

Ejercicio 16

```
#!/bin/bash

bas=`echo $1 | grep "\<\([a-z]\|[A-Z]\)\>"`
var=$?
if [ $var == 0 ]
then
    echo "Es una letra"
else
    echo "No es una letra"
fi
```

Ejercicio 17

```
$ find . -regex '\(.*e.*[0-9].*\)\|\(.*[0-9].*e.*\)'
$ find . -regex '.*e.*' -and -not -regex '.*[01].*'
```

Ejercicio 18

```
$ echo `grep -c -E ^ ejercicio10.sh` ejercicio10.sh
```