

Computación en Física

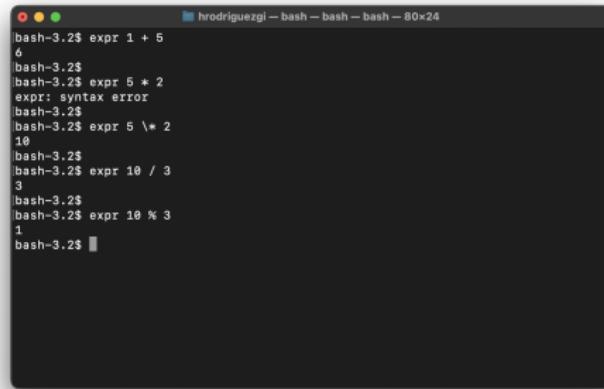
Harvey Rodriguez Gil

Universidad EIA

6 de Agosto de 2024

Expresiones matemáticas

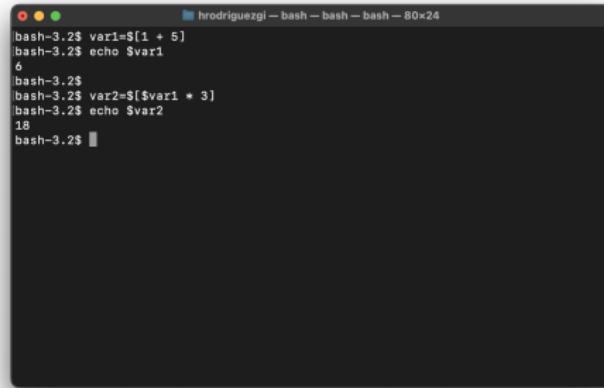
El intérprete de Linux Bash cuenta con la capacidad de realizar operaciones matemáticas, una de las formas es utilizando el comando `expr`:



```
hrodriguezpi - bash - bash - bash - 80x24
bash-3.2$ expr 1 + 5
6
bash-3.2$ expr 5 * 2
expr: syntax error
bash-3.2$ expr 5 \* 2
10
bash-3.2$ expr 10 / 3
3
bash-3.2$ expr 10 % 3
1
bash-3.2$
```

Expresiones matemáticas

Otra alternativa es utilizar los corchetes [...] para escribir las expresiones matemáticas:

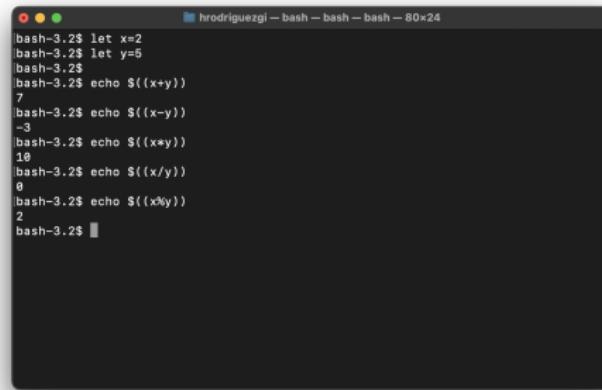


A screenshot of a Linux terminal window titled "hrodriguezgi — bash — bash — bash — 80x24". The terminal shows the following command-line interaction:

```
hrodriguezgi — bash — bash — bash — bash — 80x24
bash-3.2$ vari=$((1 + 6)
bash-3.2$ echo $vari
6
bash-3.2$ bash-3.2$ var2=$((vari * 3)
bash-3.2$ echo $var2
18
bash-3.2$
```

Expresiones matemáticas

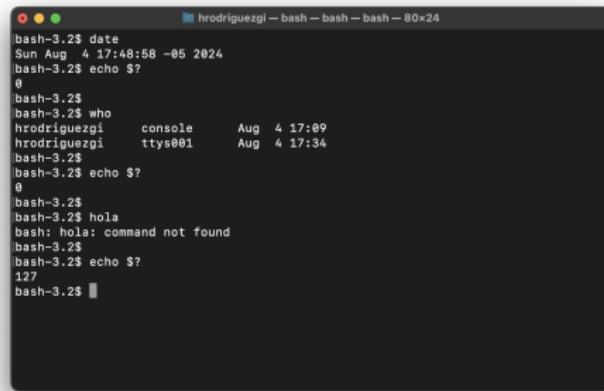
Otra alternativa es utilizar el comando let y utilizar los dobles parentesis:



```
hrodriguezgi - bash - bash - bash - 80x24
bash-3.2$ let x=2
bash-3.2$ let y=5
bash-3.2$ echo $((x+y))
7
bash-3.2$ echo $((x-y))
-3
bash-3.2$ echo $((x*y))
10
bash-3.2$ echo $((x/y))
0
bash-3.2$ echo $((x%y))
2
bash-3.2$
```

Códigos de Salida

Siempre que se ejecuta un comando en bash, es retornado un código de salida indicando si el proceso ha finalizado correctamente o no. Para validar el código de salida del comando anterior se utiliza `$?`. Por convención, se define como 0 si el comando se ha completado correctamente. Si se ha generado un error, se devolverá un valor entero positivo



```
hrodriguezgi — bash — bash — bash — 80x24
bash-3.2$ date
Sun Aug 4 17:48:58 -05 2024
bash-3.2$ echo $?
0
bash-3.2$ who
hrodriguezgi      console      Aug 4 17:09
hrodriguezgi      ttys001     Aug 4 17:34
bash-3.2$ echo $?
0
bash-3.2$ hola
bash: hola: command not found
bash-3.2$ echo $?
127
bash-3.2$
```

Códigos de Salida

Los scripts de bash utilizan esta misma convención de valores de códigos de salida, sin embargo por medio del comando `exit` se puede personalizar el valor que retornará nuestro script:



A screenshot of a terminal window titled "hrodriguezgi - bash - bash - bash - 80x24". The terminal shows the following interaction:

```
bash-3.2$ cat test3
#!/bin/bash

var1=10
var2=20
var3=`expr $var2 / $var1`

echo El resultado es $var3
exit 5

bash-3.2$ bash test3
El resultado es 2
bash-3.2$ echo $?
5
bash-3.2$
```

Códigos de Salida

Algunos de los valores más comunes en los códigos de salida son:

Código	Descripción
0	Ejecución exitosa
1	Error general desconocido
2	Mal uso de comando
126	Comando no puede ejecutarse
127	Comando no encontrado
128	Argumento de salida invalido
130	Comando terminado con Ctrl-C
255	Estado de salida fuera de rango

Condicionales

Bash nos permite hacer uso de condicionales en los script para hacer más compleja y completa nuestras tareas en ellos. Se debe tener presente que los condicionales `if` evalúan el código de salida del comando para ejecutarse.



```
hrodriguezg1:bash:bash:bash-80x24
bash-3.2$ cat test4
#!/bin/bash

testuser=baduser

if grep $testuser /etc/passwd
then
    echo Los archivos del usuario $testuser son:
    ls -a /home/$testuser
else
    echo El usuario no existe en el sistema
fi
bash-3.2$
bash-3.2$ bash test4
El usuario no existe en el sistema
bash-3.2$
```

Condicionales

La sintaxis para escribir más de un condicional será la siguiente:

```
if condicion_1
then
    comandos
elif condicion_2
then
    comandos
fi
```

Comando test

El comando test provee una forma de complementar los condicionales, para que adicional de esperar el código de salida, pueda realizar operaciones de Falso/Verdadero. La sintaxis de este comando es:

```
if test condicion
then
    comandos
fi
```

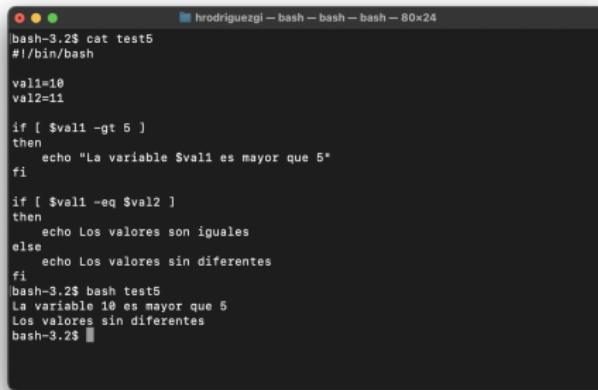
Comando test

Comparaciones numéricas:

Comparación	Descripción
n1 -eq n2	Valida si n1 es igual a n2
n1 -ge n2	Valida si n1 es mayor o igual que n2
n1 -gt n2	Valida si n1 es mayor que n2
n1 -le n2	Valida si n1 es menor o igual que n2
n1 -lt n2	Valida si n1 es menor que n2
n1 -ne n2	Valida si n1 no es igual a n2

Comando test

Comparaciones numéricas:



```
hrodriguezgi@hrodriguezgi: bash - bash - bash - 80x24
bash-3.2$ cat test5
#!/bin/bash

val1=10
val2=11

if [ $val1 -gt 6 ]
then
    echo "La variable $val1 es mayor que 5"
fi

if [ $val1 -eq $val2 ]
then
    echo Los valores son iguales
else
    echo Los valores son diferentes
fi
bash-3.2$ bash test5
La variable 10 es mayor que 5
Los valores son diferentes
bash-3.2$
```

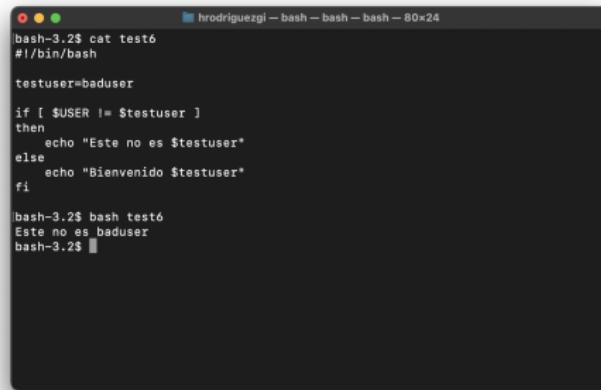
Comando test

Comparaciones de cadena:

Comparación	Descripción
<code>str1 = str2</code>	Valida si str1 es la misma que str2
<code>str1 != str2</code>	Valida si str1 no es la misma que str2
<code>str1 < str2</code>	Valida si str1 es menor que str2
<code>str1 > str2</code>	Valida si str1 es mayor que str2
<code>-n str1</code>	Valida si str1 tiene una longitud mayor a cero
<code>-z str1</code>	Valida si str1 tiene una longitud de cero

Comando test

Comparaciones de cadena:



A screenshot of a Linux terminal window titled "hrodriguezgi - bash - bash - bash - 80x24". The terminal shows the following content:

```
bash-3.2$ cat test6
#!/bin/bash

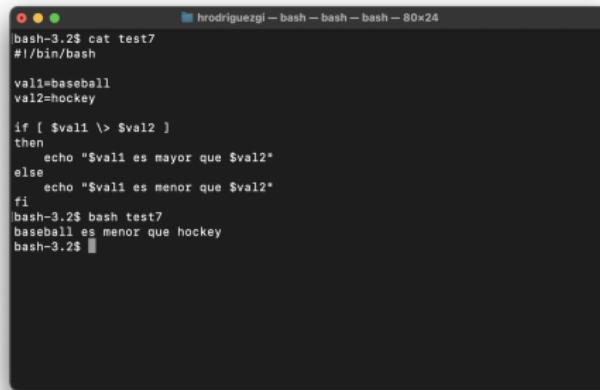
testuser=baduser

if [ $USER != $testuser ]
then
    echo "Este no es $testuser"
else
    echo "Bienvenido $testuser"
fi

bash-3.2$ bash test6
Este no es baduser
bash-3.2$
```

Comando test

Comparaciones de cadena:



```
hrodriguezgi - bash - bash - bash - 80x24
bash-3.2$ cat test7
#!/bin/bash

val1=baseball
val2=hockey

if [ $val1 \> $val2 ]
then
    echo "$val1 es mayor que $val2"
else
    echo "$val1 es menor que $val2"
fi
bash-3.2$ bash test7
baseball es menor que hockey
bash-3.2$
```

Comando test

Comparaciones de archivo:

Comparación	Descripción
-d file	Valida si file existe y es un directorio
-e file	Valida si file existe
-f file	Valida si file existe y es un archivo
-r file	Valida si file existe y se puede leer
-s file	Valida si file existe y no está vacío
-w file	Valida si file existe y se puede escribir
-x file	Valida si file existe y se puede ejecutar

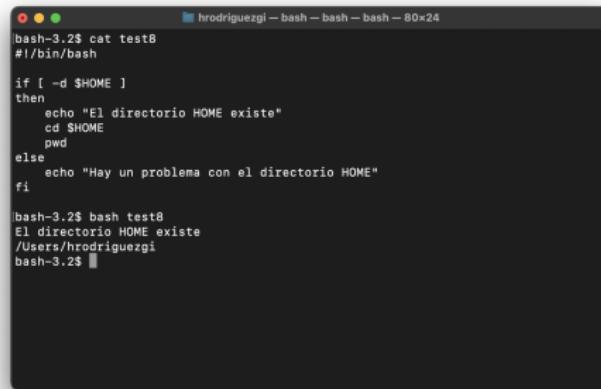
Comando test

Comparaciones de archivo:

Comparación	Descripción
-O file	Valida si file existe y el usuario actual es el propietario
-G file	Valida si file existe y su grupo es el mismo del usuario actual
file1 -nt file2	Valida si file1 es más reciente que file2
file1 -ot file2	Valida si file1 es más antiguo que file2

Comando test

Comparaciones de archivo:



```
hrodriguezgi - bash - bash - bash - 80x24
bash-3.2$ cat test8
#!/bin/bash

if [ -d $HOME ]
then
    echo "El directorio HOME existe"
    cd $HOME
    pwd
else
    echo "Hay un problema con el directorio HOME"
fi

bash-3.2$ bash test8
El directorio HOME existe
/Users/hrodriguezgi
bash-3.2$
```

Comando test

Comparaciones de archivo:

```
hrodriguezgi$ bash -bash - bash - 80x24
bash-3.2$ cat test9
#!/bin/bash

if [ -e $HOME ]
then
    echo "OK en el directorio, ahora vamos a validar el archivo"
    if [ -e $HOME/testing ]
    then
        echo "Adicionando informacion al archivo"
        date >> $HOME/testing
    else
        echo "Creando el archivo"
        date > $HOME/testing
    fi
else
    echo "Hay un problema con el directorio HOME"
fi

bash-3.2$ bash test9
OK en el directorio, ahora vamos a validar el archivo
Creando el archivo
bash-3.2$ cat /Users/hrodriguezgi/testing
Sun Aug 4 19:17:28 -05 2024
bash-3.2$
```

Múltiples condiciones

Los condicionales nos permiten combinar multiples comandos test de la siguiente forma:

- ▶ [condicion_1] && [condicion_2]
- ▶ [condicion_1] || [condicion_2]

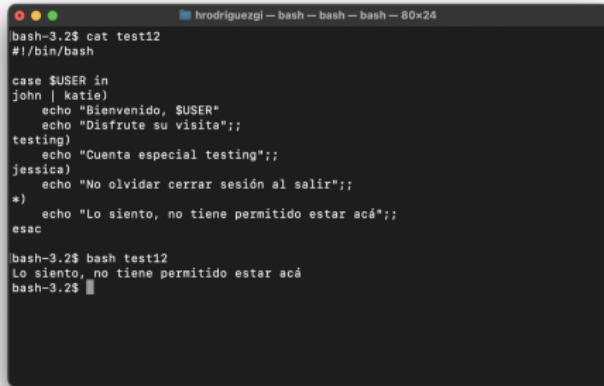
El primero realiza una operación de AND lógico, que significa que ambas condiciones se deben cumplir. El segundo es un OR lógico y significa que una de las dos se debe cumplir

Comando case

El comando case es una alternativa a los condicionales if. Este nos permite validar multiples valores de una única variable de una forma más ordenada. La sintaxis de este comando es de la siguiente forma:

```
case variable in
patron1 | patron2) comandos;;
patron3) comandos;;
*) comandos por defecto;;
esac
```

Comando case



```
hrodriguezgi - bash - bash - bash - 80x24
bash-3.2$ cat test12
#!/bin/bash

case $USER in
john | katie)
    echo "Bienvenido, $USER"
    echo "Disfrute su visita";;
testing)
    echo "Cuenta especial testing";;
jessica)
    echo "No olvidar cerrar sesión al salir";;
*)
    echo "Lo siento, no tiene permitido estar acá";;
esac

bash-3.2$ bash test12
Lo siento, no tiene permitido estar acá
bash-3.2$
```

Argumentos de entrada

El intérprete de linux no solo nos permite utilizar variables en el interior de un script, si no que también nos permiten utilizar las que son enviadas junto con el script mismo a ejecutar. La sintaxis de un script de bash que recibe parámetros es algo como:

```
script arg1 arg2... argn
```

Para utilizar los argumentos en el script utilizaremos \$1, \$2, ... , \$n. Si queremos tomarlos todos al tiempo utilizaremos \$@

Argumentos de entrada



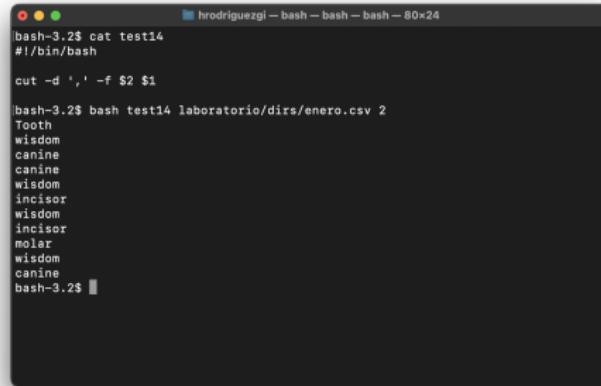
```
hrodriguezgi - bash - bash - bash - 80x24
bash-3.2$ cat test13
#!/bin/bash

usuario=$1

if grep $usuario /etc/passwd
then
    echo "Usuario $usuario registrado en el sistema"
else
    echo "El usuario $usuario no se encuentra registrado"
fi

bash-3.2$ bash test13 pedro
El usuario pedro no se encuentra registrado
bash-3.2$
```

Argumentos de entrada



A screenshot of a Linux terminal window titled "hrodriguezgi - bash - bash - bash - 80x24". The terminal shows the following command sequence and output:

```
bash-3.2$ cat test14
#!/bin/bash

cut -d ',' -f $2 $1

bash-3.2$ bash test14 laboratorio/dirs/enero.csv 2
Tooth
wisdom
canine
canine
wisdom
incisor
wisdom
incisor
molar
wisdom
canine
bash-3.2$
```

Vamos a practicar

- ▶ Crear un script que permita obtener una linea y una columna específica de un archivo csv
- ▶ Crear un script que a partir del directorio dirs imprima una salida similar a la siguiente, el cual es a partir de identificar el archivo con menor cantidad de lineas y luego el de mayor cantidad de lineas:

Archivo dirs/abril.csv con 11 lineas

Archivo dirs/junio.csv con 26 lineas

- ▶ Crear un script que reciba 3 argumentos, num1, num2 y operación: suma, resta, multiplicación o división. La salida deberá ser el resultado de la operación