



Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

bc: la calculadora en la consola (36692 lectures)

Per **Carles Pina i Estany**, [cpina](http://pinux.info) (<http://pinux.info>)

Creado el 08/06/2004 12:34 modificado el 08/06/2004 12:34

A veces hay conjuntos de herramientas que nos suenan y no las aprovechamos al máximo. Quizás una de esas sea "bc". La calculadora que podemos usar desde la consola, que aparte de sumar y restar puede realizar cálculos con una precisión de varios decimales, cambiar de base numérica o programarla, etc. veremos como...

Introducción

bc es el nombre mismo de un ejecutable, en Debian está en el paquete "bc". Siempre lo he visto instalado por defecto, así que seguramente ya lo tendremos.

Si ejecutamos bc veremos:

```
carles@pinux:~$ bc
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
```

Y aquí se queda esperando que trabajemos con él:

```
4+2
6
```

Sabe sumar (y restar, dividir, multiplicar; sin problemas)

Trabajar con variables

Si queremos que trabaje con cierto número de decimales, asignaremos un valor a la variable `scale`.

```
scale=3
2/3
.666
```

También podríamos haber hecho:

```
scale=9;2/3
.666666666
```

Al ser `scale` una variable más, podemos consultar su valor sencillamente escribiéndola:

```
scale
3
#o también usando print
print scale
3
```

Otro ejemplo de usos de variables podría ser:



```
scale=5
variable=3
100/variable
33.33333
```

Hay una variable especial llamada `last`, almacena el resultado de la última operación:

```
100/3
33.33333
last*2
66.66666
```

Cambios de base

Hay dos variables especiales que son `ibase` y `obase` que definen la base de entrada y de salida de los números, respectivamente.

Si queremos calcular el número 5 de base 10 (decimal) a base 2 (binario), haremos:

```
obase=2
5
101
```

Si queremos pasar de binario a hexadecimal:

```
obase=16
ibase=2
11111111
FF
```

Otras operaciones matemáticas

Podemos usar también `sqrt`, `2^3`, etc.

Si queremos tener operaciones matemáticas más complejas tenemos que ejecutar el `bc` con la opción `-l` (`bc -l`) para cargar la librería matemática, y entonces tendremos:

```
s (x) Seno de x, en radianes
c (x) Coseno de x
a (x) Arcotangente de x
l (x) Logaritmo neperiano de x
e (x) Exponencial de x
j (n,x) Función de Bessel de un entero de orden n de x
```

Haciendo un pequeño programa

Si queremos pedir que el usuario teclee algo por teclado, la función es `read`. Por tanto.

```
variable=read()
```

esperará que tecleemos algo y lo pondrá en `variable`.

También podemos hacer un programa en un fichero y ejecutarlo. Por ejemplo, si hacemos un fichero que contenga eso:

```
print "Hola\n"
print "Escribe tu año de nacimiento\n"
nacimiento=read()
edad=2003-nacimiento
print edad
print " años\n"
halt
```



Y lo ejecutamos con `bc -q fichero.bc` (el `-q` es para que no muestre el Copyright) estaremos preguntando el año nacimiento y mostrando la edad.

Ahora un ejemplo definiendo una función, pasándole un parámetro y retornándolo. Veremos que es totalmente intuitivo:

```
define mayor_edad(edad) {
    if (edad <18)
        return 0
    if (edad >= 18)
        return 1
}

print "Dí tus edad\n"
edad=read()

if (mayor_edad(edad))
    print "Eres mayor de edad\n"

if (!mayor_edad(edad))
    print "Eres menor de edad\n"

halt
```

Un último ejemplo, con un for:

```
for (i=0;i
```

Observemos en la `,` para separar la variable y la cadena en el print.

Para salir

```
halt
```

Más información

En man `bc`

También en el directorio `/usr/share/doc/bc/examples` (hay unos ejemplos que están bien para ver la sintaxis).

Recordar que se puede usar desde scripts de forma muy cómoda, por ejemplo:

```
resultado=$(echo "scale=3;2/3" | bc)
```

E-mail del autor: carles_ARROBA_pinux.info

Podrás encontrar este artículo e información adicional en: <http://bulma.net/body.phtml?nIdNoticia=2045>