Shell Tips!

How To Do Advanced Math Calculation Using Bc?

Last updated: 2020-09-26

On This Page

Shell Tips!

The Linux **bc** command line allows you to perform arithmetic and algebra in a shell like bash by using mathematical functions like ***sine***, ***cosine***, ***tangent*** and so on.

My previous post ***Performing Math Calculation in Bash*** was an introduction to elementary arithmetic operations (addition, subtraction, division, multiplication) in a `bash` shell or by using `bc`. In this new *Advanced Math Calculation on Linux* post, we cover how to use the ***GNU bc*** command line tool and how to write your own mathematical functions using `bc`.

What is bc?

`bc` stand for **b**asic **c**alculator, it was preceded by *dc*, a cross-platform reverse-polish **d**esk **c**alculator one of the oldest Unix utilities. **bc** is part of the POSIX standard.

> bc, for basic calculator, is "an arbitrary precision calculator language" with syntax similar to the C programming language. bc is typically used as either a mathematical scripting language or as an interactive mathematical shell.

All the standard mathematical operators are available in `bc` and you can also use relational expressions and boolean expressions.

like `if`, `print`, `while`, and `for`.

How to use bc's Math Library Functions?

In order to use `bc` advanced math libraries (*mathlib*) you need to use the `-l` option, i.e. `bc -l`. This will load the Math library and set the default value of scale to *20*. Below is the list of predefined functions that comes with the *bc* math library.

`s (*x*)`	The sine of *x*, *x* is in radians.
`c (*x*)`	The cosine of *x*, *x* is in radians.
`a (*x*)`	The arctangent of *x*, arctangent returns radians.
`l (*x*)`	The natural logarithm of *x*.
`e (*x*)`	The exponential function of raising *e* to the value *x*.
`j (*n*,*x*)`	The bessel function of integer order *n* of *x*.

```
[me@linux ~]$ bc -l <<< "l(3)"
1.09861228866810969139
```

What are the bc Special Variables?

The `bc` command line provide four special variables with specific meaning and behavior on the arithmetic expression to be evaluated.

`ibase`	Defines the conversion base for input numbers. Default is to use base 10.
`obase`	Defines the conversion base four output numbers. Default is to use base 10.
`last`	Contains the value of the last printed number. It is a GNU bc extension.

What are the bc Special Expressions (standard functions)?

GNU bc provide few special expressions, i.e. standard functions, that allows to perform common operations easily and make the language richer.

`length ( expression )`

The value of the length function is the number of significant digits in the expression.

`read ( )`

The `read` function (an extension) will read a number from the standard input, regardless of where the function occurs. Beware, this can cause problems with the mixing of data and program in the standard input. The best use for this function is in a previously written program that needs input from the user, but never allows program code to be input from the user. The value of the `read` function is the number read from the standard input using the current value of the variable *ibase* for the conversion base.

`scale ( expression )`

The value of the `scale` function is the number of digits after

How to write a user-defined function in bc?

`bc` allows you to define your own user-defined functions which makes the language very powerful as you can create all the mathematical functions that you may need.

```
define name ( parameters ) { newline
    auto_list   statement_list }
```

GNU bc Examples

How to calculate pi in shell using bc?

The number Pi `π` is always equal to the circumference divided by the diameter of a circle. So, you can use the `bc` math library with the `arctangent` function of 1 and multiply it by 4 to get the Pi value.

```
[me@linux ~]$ pi=$(echo "scale=10; 4*a(1)" | bc -
l)
[me@linux ~]$ echo $pi
3.1415926532
```

How to find factorial of a number in a shell script using bc?

You can easily define a recursive factorial function in `bc` using the `define` keyword and a `if` statement. The factorial

```
f(1)
1
f(3)
6
f(9)
362880
```

A Simple Arithmetic Calculator using bc

By using some of GNU `bc` advanced features, you can prompt user for entries, create some infinite loops and use condtional statements. Below example is a simple calculator in `bc` to use in your bash shell using a floating-point precision of two decimal digits after the decimal point.

```
scale=2
print "\nA Simple Arithmetic Calculator using
bc\n"
print "  Enter x and y value then select an
operation.\n\n"

while (1) {
  print "x=? "; x = read()
  print "y=? "; y = read()
  print "Choose an operation: addition (1),
subtraction (2), multiplication (3), division (4)
"; op = read()
  if (op == 1) print "Addition: ", x, "+", y, "=",
x+y;
  if (op == 2) print "Subtraction: ", x, "-", y,
"=", x-y;
  if (op == 3) print "Multiplication: ", x, "*",
```

```
[me@linux ~]$ bc -q bash_simple_bc_calculator.bc
A Simple Arithmetic Calculator using bc
  Enter x and y value then select an operation.

x=? 3.78
y=? 2.6
Choose an operation: addition (1),  subtraction
(2), multiplication (3), division (4) 3
Multiplication: 3.78*2.6=9.82

x=? 1.5
y=? 18.32
Choose an operation: addition (1),  subtraction
(2), multiplication (3), division (4) 4
Division: 1.5/18.32=.08
```

More bc resources and examples

There is a large list of existing open source projects which extend standard bc capabilities with a lot of user-defined functions. You can check the *X-BC* resources.

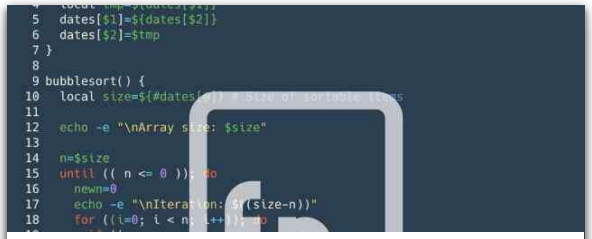
- extensions.bc: contains functions of trigonometry, exponential functions, functions of number theory and some mathematical constants.
- scientific_constants.bc: contains particle masses, basic constants, such as speed of light in the vacuum and the gravitational constant.

Another amazing resource is the *GNU bc FAQ* on phodd.net. You will find a large amount of functions in the file funcs.bc

Shell Tips!

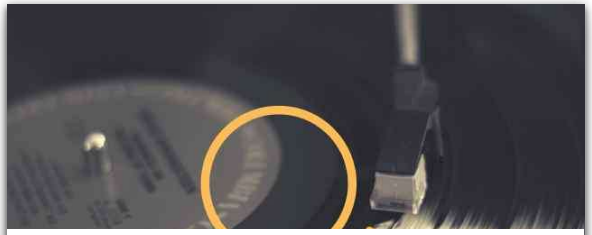
Email SIGN UP

AND **FOLLOW ME** ON



The Complete How To Guide of Bash Functions

How To Create Simple Menu with the Shell Select Loop?



What is the Right Way to do Bash Loops?

5 Mistakes To Avoid For Writing High-Quality Bash Comments

Shell Tips!

DISCLAIMERS & CONTACTS